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

1 ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

3 ; ----------------------------------------------------------------------------

4 ; Last Update: 31/12/2017

5 ; ----------------------------------------------------------------------------

6 ; Beginning: 04/01/2016

7 ; ----------------------------------------------------------------------------

8 ; Assembler: NASM version 2.11 (trdos386.s)

9 ; ----------------------------------------------------------------------------

10 ; Turkish Rational DOS

11 ; Operating System Project v2.0 by ERDOGAN TAN (Beginning: 04/01/2016)

12 ;

13 ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan

14 ; unix386.s (03/01/2016)

15 ;

16 ; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan

17 ; TRDOS2.ASM (09/11/2011)

18 ;

19 ; Derived from 'IBM PC-XT-286' BIOS source code (1986)

20 ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

21 ; nasm trdos386.s -l trdos386.txt -o TRDOS386.SYS

22

23

24 KLOAD equ 10000h ; Kernel loading address

25 ; NOTE: Retro UNIX 8086 v1 /boot code loads kernel at 1000h:0000h

26 KCODE equ 08h ; Code segment descriptor (ring 0)

27 KDATA equ 10h ; Data segment descriptor (ring 0)

28 ; 19/03/2015

29 UCODE equ 1Bh ; 18h + 3h (ring 3)

30 UDATA equ 23h ; 20h + 3h (ring 3)

31 ; 24/03/2015

32 TSS equ 28h ; Task state segment descriptor (ring 0)

33 ; 19/03/2015

34 CORE equ 400000h ; Start of USER's virtual/linear address space

35 ; (at the end of the 1st 4MB)

36 ECORE equ 0FFC00000h ; End of USER's virtual address space (4GB - 4MB)

37 ; ULIMIT = (ECORE/4096) - 1 = 0FFBFFh (in GDT)

38

39 ;; 27/12/2013

40 ;KEND equ KLOAD + 65536 ; (28/12/2013) (end of kernel space)

41 ; 04/07/2016

42 KEND equ KERNELFSIZE + KLOAD

43

44

45 ; IBM PC/AT BIOS ----- 10/06/85 (postequ.inc)

46 ;--------- CMOS TABLE LOCATION ADDRESS'S -------------------------------------

47 CMOS\_SECONDS EQU 00H ; SECONDS (BCD)

48 CMOS\_SEC\_ALARM EQU 01H ; SECONDS ALARM (BCD)

49 CMOS\_MINUTES EQU 02H ; MINUTES (BCD)

50 CMOS\_MIN\_ALARM EQU 03H ; MINUTES ALARM (BCD)

51 CMOS\_HOURS EQU 04H ; HOURS (BCD

52 CMOS\_HR\_ALARM EQU 005H ; HOURS ALARM (BCD)

53 CMOS\_DAY\_WEEK EQU 06H ; DAY OF THE WEEK (BCD)

54 CMOS\_DAY\_MONTH EQU 07H ; DAY OF THE MONTH (BCD)

55 CMOS\_MONTH EQU 08H ; MONTH (BCD)

56 CMOS\_YEAR EQU 09H ; YEAR (TWO DIGITS) (BCD)

57 CMOS\_CENTURY EQU 32H ; DATE CENTURY BYTE (BCD)

58 CMOS\_REG\_A EQU 0AH ; STATUS REGISTER A

59 CMOS\_REG\_B EQU 00BH ; STATUS REGISTER B ALARM

60 CMOS\_REG\_C EQU 00CH ; STATUS REGISTER C FLAGS

61 CMOS\_REG\_D EQU 0DH ; STATUS REGISTER D BATTERY

62 CMOS\_SHUT\_DOWN EQU 0FH ; SHUTDOWN STATUS COMMAND BYTE

63 ;----------------------------------------

64 ; CMOS EQUATES FOR THIS SYSTEM ;

65 ;-----------------------------------------------------------------------------

66 CMOS\_PORT EQU 070H ; I/O ADDRESS OF CMOS ADDRESS PORT

67 CMOS\_DATA EQU 071H ; I/O ADDRESS OF CMOS DATA PORT

68 NMI EQU 10000000B ; DISABLE NMI INTERRUPTS MASK -

69 ; HIGH BIT OF CMOS LOCATION ADDRESS

70

71 ; Memory Allocation Table Address

72 ; 05/11/2014

73 ; 31/10/2014

74 MEM\_ALLOC\_TBL equ 100000h ; Memory Allocation Table at the end of

75 ; the 1st 1 MB memory space.

76 ; (This address must be aligned

77 ; on 128 KB boundary, if it will be

78 ; changed later.)

79 ; ((lower 17 bits of 32 bit M.A.T.

80 ; address must be ZERO)).

81 ; ((((Reason: 32 bit allocation

82 ; instructions, dword steps)))

83 ; (((byte >> 12 --> page >> 5)))

84 ;04/11/2014

85 PDE\_A\_PRESENT equ 1 ; Present flag for PDE

86 PDE\_A\_WRITE equ 2 ; Writable (write permission) flag

87 PDE\_A\_USER equ 4 ; User (non-system/kernel) page flag

88 ;

89 PTE\_A\_PRESENT equ 1 ; Present flag for PTE (bit 0)

90 PTE\_A\_WRITE equ 2 ; Writable (write permission) flag (bit 1)

91 PTE\_A\_USER equ 4 ; User (non-system/kernel) page flag (bit 2)

92 PTE\_A\_ACCESS equ 32 ; Accessed flag (bit 5) ; 09/03/2015

93

94 ; 17/02/2015 (unix386.s)

95 ; 10/12/2014 - 30/12/2014 (0B000h -> 9000h) (dsectrm2.s)

96 DPT\_SEGM equ 09000h ; FDPT segment (EDD v1.1, EDD v3)

97 ;

98 HD0\_DPT equ 0 ; Disk parameter table address for hd0

99 HD1\_DPT equ 32 ; Disk parameter table address for hd1

100 HD2\_DPT equ 64 ; Disk parameter table address for hd2

101 HD3\_DPT equ 96 ; Disk parameter table address for hd3

102

103

104 ; FDPT (Phoenix, Enhanced Disk Drive Specification v1.1, v3.0)

105 ; (HDPT: Programmer's Guide to the AMIBIOS, 1993)

106 ;

107 FDPT\_CYLS equ 0 ; 1 word, number of cylinders

108 FDPT\_HDS equ 2 ; 1 byte, number of heads

109 FDPT\_TT equ 3 ; 1 byte, A0h = translated FDPT with logical values

110 ; otherwise it is standard FDPT with physical values

111 FDPT\_PCMP equ 5 ; 1 word, starting write precompensation cylinder

112 ; (obsolete for IDE/ATA drives)

113 FDPT\_CB equ 8 ; 1 byte, drive control byte

114 ; Bits 7-6 : Enable or disable retries (00h = enable)

115 ; Bit 5 : 1 = Defect map is located at last cyl. + 1

116 ; Bit 4 : Reserved. Always 0

117 ; Bit 3 : Set to 1 if more than 8 heads

118 ; Bit 2-0 : Reserved. Alsways 0

119 FDPT\_LZ equ 12 ; 1 word, landing zone (obsolete for IDE/ATA drives)

120 FDPT\_SPT equ 14 ; 1 byte, sectors per track

121

122 ; Floppy Drive Parameters Table (Programmer's Guide to the AMIBIOS, 1993)

123 ; (11 bytes long) will be used by diskette handler/bios

124 ; which is derived from IBM PC-AT BIOS (DISKETTE.ASM, 21/04/1986).

125

126 ; 01/02/2016

127 Logical\_DOSDisks equ 90000h + 100h ; 26\*256 = 6656 bytes

128 Directory\_Buffer equ 80000h ; max = 64K Bytes

129 FAT\_Buffer equ 91C00h ; 1536 bytes (3 sectors)

130 ; 15/02/2016

131 Cluster\_Buffer equ 70000h ; max = 64K Bytes ; buffer for file read & write

132 ; 11/04/2016

133 Env\_Page: equ 93000h ; 512 bytes (4096 bytes)

134 Env\_Page\_Size equ 512 ; (4096 bytes)

135 ; 30/07/2016

136 Video\_Pg\_Backup equ 98000h ; Mode 3h, video page backup (32K, 8 pages)

137

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

139

140 [ORG 0]

141 ; 12/11/2014

142 ; Save boot drive number (that is default root drive)

143 00000000 8816[F25C] mov [boot\_drv], dl ; physical drv number

144

145 ; Determine installed memory

146 ; 31/10/2014

147 ;

148 00000004 B801E8 mov ax, 0E801h ; Get memory size

149 00000007 CD15 int 15h ; for large configurations

150 00000009 7308 jnc short chk\_ms

151 0000000B B488 mov ah, 88h ; Get extended memory size

152 0000000D CD15 int 15h

153 ;

154 ;mov al, 17h ; Extended memory (1K blocks) low byte

155 ;out 70h, al ; select CMOS register

156 ;in al, 71h ; read data (1 byte)

157 ;mov cl, al

158 ;mov al, 18h ; Extended memory (1K blocks) high byte

159 ;out 70h, al ; select CMOS register

160 ;in al, 71h ; read data (1 byte)

161 ;mov ch, al

162 ;

163 0000000F 89C1 mov cx, ax

164 00000011 31D2 xor dx, dx

165 chk\_ms:

166 00000013 890E[EE5C] mov [mem\_1m\_1k], cx

167 00000017 8916[F05C] mov [mem\_16m\_64k], dx

168 ; 05/11/2014

169 ;and dx, dx

170 ;jz short L2

171 0000001B 81F90004 cmp cx, 1024

172 0000001F 7351 jnb short L0

173 ; insufficient memory\_error

174 ; Minimum 2 MB memory is needed...

175 ; 05/11/2014

176 ; (real mode error printing)

177 00000021 FB sti

178 00000022 BE[3600] mov si, msg\_out\_of\_memory

179 00000025 BB0700 mov bx, 7

180 00000028 B40E mov ah, 0Eh ; write tty

181 oom\_1:

182 0000002A AC lodsb

183 0000002B 08C0 or al, al

184 0000002D 7404 jz short oom\_2

185 0000002F CD10 int 10h

186 00000031 EBF7 jmp short oom\_1

187 oom\_2:

188 00000033 F4 hlt

189 00000034 EBFD jmp short oom\_2

190

191 ; 20/02/2017

192 ; 05/11/2014

193 msg\_out\_of\_memory:

194 00000036 070D0A db 07h, 0Dh, 0Ah

195 00000039 496E73756666696369- db 'Insufficient memory !'

195 00000042 656E74206D656D6F72-

195 0000004B 792021

196 0000004E 0D0A db 0Dh, 0Ah

197 \_int13h\_48h\_buffer: ; 07/07/2016

198 00000050 284D696E696D756D20- db '(Minimum 2MB memory is needed.)'

198 00000059 324D42206D656D6F72-

198 00000062 79206973206E656564-

198 0000006B 65642E29

199 0000006F 0D0A00 db 0Dh, 0Ah, 0

200 ;

201

202 L0:

203 %include 'diskinit.s' ; 07/03/2015

1 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2 <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0 - diskinit.s

3 <1> ; ----------------------------------------------------------------------------

4 <1> ; Last Update: 09/07/2016

5 <1> ; ----------------------------------------------------------------------------

6 <1> ; Beginning: 24/01/2016

7 <1> ; ----------------------------------------------------------------------------

8 <1> ; Assembler: NASM version 2.11 (trdos386.s)

9 <1> ; ----------------------------------------------------------------------------

10 <1> ; Turkish Rational DOS

11 <1> ; Operating System Project v2.0 by ERDOGAN TAN (Beginning: 04/01/2016)

12 <1> ;

13 <1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan

14 <1> ; diskinit.inc (10/07/2015)

15 <1> ;

16 <1> ; Derived from 'IBM PC-XT-286' BIOS source code (1986)

17 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

18 <1>

19 <1> ; Retro UNIX 386 v1 Kernel - DISKINIT.INC

20 <1> ; Last Modification: 10/07/2015

21 <1>

22 <1> ; DISK I/O SYSTEM INITIALIZATION - Erdogan Tan (Retro UNIX 386 v1 project)

23 <1>

24 <1> ; ///////// DISK I/O SYSTEM STRUCTURE INITIALIZATION ///////////////

25 <1>

26 <1> ; 10/12/2014 - 02/02/2015 - dsectrm2.s

27 <1> ;L0:

28 <1> ; 12/11/2014 (Retro UNIX 386 v1 - beginning)

29 <1> ; Detecting disk drives... (by help of ROM-BIOS)

30 00000072 BA7F00 <1> mov dx, 7Fh

31 <1> L1:

32 00000075 FEC2 <1> inc dl

33 00000077 B441 <1> mov ah, 41h ; Check extensions present

34 <1> ; Phoenix EDD v1.1 - EDD v3

35 00000079 BBAA55 <1> mov bx, 55AAh

36 0000007C CD13 <1> int 13h

37 0000007E 721A <1> jc short L2

38 <1>

39 00000080 81FB55AA <1> cmp bx, 0AA55h

40 00000084 7514 <1> jne short L2

41 00000086 FE06[F55C] <1> inc byte [hdc] ; count of hard disks (EDD present)

42 0000008A 8816[F45C] <1> mov [last\_drv], dl ; last hard disk number

43 0000008E BB[785C] <1> mov bx, hd0\_type - 80h

44 00000091 01D3 <1> add bx, dx

45 00000093 880F <1> mov [bx], cl ; Interface support bit map in CX

46 <1> ; Bit 0 - 1, Fixed disk access subset ready

47 <1> ; Bit 1 - 1, Drv locking and ejecting ready

48 <1> ; Bit 2 - 1, Enhanced Disk Drive Support

49 <1> ; (EDD) ready (DPTE ready)

50 <1> ; Bit 3 - 1, 64bit extensions are present

51 <1> ; (EDD-3)

52 <1> ; Bit 4 to 15 - 0, Reserved

53 00000095 80FA83 <1> cmp dl, 83h ; drive number < 83h

54 00000098 72DB <1> jb short L1

55 <1> L2:

56 <1> ; 23/11/2014

57 <1> ; 19/11/2014

58 0000009A 30D2 <1> xor dl, dl ; 0

59 <1> ; 04/02/2016 (esi -> si)

60 0000009C BE[F65C] <1> mov si, fd0\_type

61 <1> L3:

62 <1> ; 14/01/2015

63 0000009F 8816[F35C] <1> mov [drv], dl

64 <1> ;

65 000000A3 B408 <1> mov ah, 08h ; Return drive parameters

66 000000A5 CD13 <1> int 13h

67 000000A7 7210 <1> jc short L4

68 <1> ; BL = drive type (for floppy drives)

69 <1> ; DL = number of floppy drives

70 <1> ;

71 <1> ; ES:DI = Address of DPT from BIOS

72 <1> ;

73 000000A9 881C <1> mov [si], bl ; Drive type

74 <1> ; 4 = 1.44 MB, 80 track, 3 1/2"

75 <1> ; 14/01/2015

76 000000AB E8BC01 <1> call set\_disk\_parms

77 <1> ; 10/12/2014

78 000000AE 81FE[F65C] <1> cmp si, fd0\_type

79 000000B2 7705 <1> ja short L4

80 000000B4 46 <1> inc si ; fd1\_type

81 000000B5 B201 <1> mov dl, 1

82 000000B7 EBE6 <1> jmp short L3

83 <1> L4:

84 <1> ; Older BIOS (INT 13h, AH = 48h is not available)

85 000000B9 B27F <1> mov dl, 7Fh

86 <1> ; 24/12/2014 (Temporary)

87 000000BB 803E[F55C]00 <1> cmp byte [hdc], 0 ; EDD present or not ?

88 000000C0 0F879000 <1> ja L10 ; yes, all fixed disk operations

89 <1> ; will be performed according to

90 <1> ; present EDD specification

91 <1> L6:

92 000000C4 FEC2 <1> inc dl

93 000000C6 8816[F35C] <1> mov [drv], dl

94 000000CA 8816[F45C] <1> mov [last\_drv], dl ; 14/01/2015

95 000000CE B408 <1> mov ah, 08h ; Return drive parameters

96 000000D0 CD13 <1> int 13h ; (conventional function)

97 000000D2 0F828601 <1> jc L13 ; fixed disk drive not ready

98 000000D6 8816[F55C] <1> mov [hdc], dl ; number of drives

99 <1> ;; 14/01/2013

100 <1> ;;push cx

101 000000DA E88D01 <1> call set\_disk\_parms

102 <1> ;;pop cx

103 <1> ;

104 <1> ;;and cl, 3Fh ; sectors per track (bits 0-6)

105 000000DD 8A16[F35C] <1> mov dl, [drv]

106 000000E1 BB0401 <1> mov bx, 65\*4 ; hd0 parameters table (INT 41h)

107 000000E4 80FA80 <1> cmp dl, 80h

108 000000E7 7603 <1> jna short L7

109 000000E9 83C314 <1> add bx, 5\*4 ; hd1 parameters table (INT 46h)

110 <1> L7:

111 000000EC 31C0 <1> xor ax, ax

112 000000EE 8ED8 <1> mov ds, ax

113 000000F0 8B37 <1> mov si, [bx]

114 000000F2 8B4702 <1> mov ax, [bx+2]

115 000000F5 8ED8 <1> mov ds, ax

116 000000F7 3A4C0E <1> cmp cl, [si+FDPT\_SPT] ; sectors per track

117 000000FA 0F855A01 <1> jne L12 ; invalid FDPT

118 000000FE BF0000 <1> mov di, HD0\_DPT

119 00000101 80FA80 <1> cmp dl, 80h

120 00000104 7603 <1> jna short L8

121 00000106 BF2000 <1> mov di, HD1\_DPT

122 <1> L8:

123 <1> ; 30/12/2014

124 00000109 B80090 <1> mov ax, DPT\_SEGM

125 0000010C 8EC0 <1> mov es, ax

126 <1> ; 24/12/2014

127 0000010E B90800 <1> mov cx, 8

128 00000111 F3A5 <1> rep movsw ; copy 16 bytes to the kernel's DPT location

129 00000113 8CC8 <1> mov ax, cs

130 00000115 8ED8 <1> mov ds, ax

131 <1> ; 02/02/2015

132 00000117 8A0E[F35C] <1> mov cl, [drv]

133 0000011B 88CB <1> mov bl, cl

134 0000011D B8F001 <1> mov ax, 1F0h

135 00000120 80E301 <1> and bl, 1

136 00000123 7406 <1> jz short L9

137 00000125 C0E304 <1> shl bl, 4

138 00000128 2D8000 <1> sub ax, 1F0h-170h

139 <1> L9:

140 0000012B AB <1> stosw ; I/O PORT Base Address (1F0h, 170h)

141 0000012C 050602 <1> add ax, 206h

142 0000012F AB <1> stosw ; CONTROL PORT Address (3F6h, 376h)

143 00000130 88D8 <1> mov al, bl

144 00000132 04A0 <1> add al, 0A0h

145 00000134 AA <1> stosb ; Device/Head Register upper nibble

146 <1> ;

147 00000135 FE06[F35C] <1> inc byte [drv]

148 00000139 BB[785C] <1> mov bx, hd0\_type - 80h

149 0000013C 01CB <1> add bx, cx

150 0000013E 800F80 <1> or byte [bx], 80h ; present sign (when lower nibble is 0)

151 00000141 A0[F55C] <1> mov al, [hdc]

152 00000144 FEC8 <1> dec al

153 00000146 0F841201 <1> jz L13

154 0000014A 80FA80 <1> cmp dl, 80h

155 0000014D 0F8673FF <1> jna L6

156 00000151 E90801 <1> jmp L13

157 <1> L10:

158 00000154 FEC2 <1> inc dl

159 <1> ; 25/12/2014

160 00000156 8816[F35C] <1> mov [drv], dl

161 0000015A B408 <1> mov ah, 08h ; Return drive parameters

162 0000015C CD13 <1> int 13h ; (conventional function)

163 0000015E 0F82FA00 <1> jc L13

164 <1> ; 14/01/2015

165 00000162 8A16[F35C] <1> mov dl, [drv]

166 00000166 52 <1> push dx

167 00000167 51 <1> push cx

168 00000168 E8FF00 <1> call set\_disk\_parms

169 0000016B 59 <1> pop cx

170 0000016C 5A <1> pop dx

171 <1> ; 06/07/2016 (BugFix for >64K kernel files)

172 <1> ; 04/02/2016 (esi -> si)

173 <1> ;mov si, \_end ; 30 byte temporary buffer address

174 <1> ; ; at the '\_end' of kernel.

175 <1> ;mov word [si], 30

176 <1> ; 06/07/2016

177 0000016D BE[5000] <1> mov si, \_int13h\_48h\_buffer

178 <1> ; 09/07/2016

179 00000170 B81E00 <1> mov ax, 001Eh

180 00000173 8824 <1> mov [si], ah ; 0

181 00000175 46 <1> inc si

182 00000176 8904 <1> mov word [si], ax

183 <1> ; word [si] = 30

184 <1> ;

185 00000178 B448 <1> mov ah, 48h ; Get drive parameters (EDD function)

186 0000017A CD13 <1> int 13h

187 0000017C 0F82DC00 <1> jc L13

188 <1> ; 04/02/2016 (ebx -> bx)

189 <1> ; 14/01/2015

190 00000180 28FF <1> sub bh, bh

191 00000182 88D3 <1> mov bl, dl

192 00000184 80EB80 <1> sub bl, 80h

193 00000187 81C3[F85C] <1> add bx, hd0\_type

194 0000018B 8A07 <1> mov al, [bx]

195 0000018D 0C80 <1> or al, 80h

196 0000018F 8807 <1> mov [bx], al

197 00000191 81EB[F65C] <1> sub bx, hd0\_type - 2 ; 15/01/2015

198 00000195 81C3[425D] <1> add bx, drv.status

199 00000199 8807 <1> mov [bx], al

200 <1> ; 04/02/2016 (eax -> ax)

201 0000019B 8B4410 <1> mov ax, [si+16]

202 0000019E 854412 <1> test ax, [si+18]

203 000001A1 7412 <1> jz short L10\_A0h

204 <1> ; 'CHS only' disks on EDD system

205 <1> ; are reported with ZERO disk size

206 000001A3 81EB[425D] <1> sub bx, drv.status

207 000001A7 C1E302 <1> shl bx, 2

208 000001AA 81C3[265D] <1> add bx, drv.size ; disk size (in sectors)

209 000001AE 8907 <1> mov [bx], ax

210 000001B0 8B4412 <1> mov ax, [si+18]

211 000001B3 8907 <1> mov [bx], ax

212 <1>

213 <1> L10\_A0h: ; Jump here to fix a ZERO (LBA) disk size problem

214 <1> ; for CHS disks (28/02/2015)

215 <1> ; 30/12/2014

216 000001B5 BF0000 <1> mov di, HD0\_DPT

217 000001B8 88D0 <1> mov al, dl

218 000001BA 83E003 <1> and ax, 3

219 000001BD C0E005 <1> shl al, 5 ; \*32

220 000001C0 01C7 <1> add di, ax

221 000001C2 B80090 <1> mov ax, DPT\_SEGM

222 000001C5 8EC0 <1> mov es, ax

223 <1> ;

224 000001C7 88E8 <1> mov al, ch ; max. cylinder number (bits 0-7)

225 000001C9 88CC <1> mov ah, cl

226 000001CB C0EC06 <1> shr ah, 6 ; max. cylinder number (bits 8-9)

227 000001CE 40 <1> inc ax ; logical cylinders (limit 1024)

228 000001CF AB <1> stosw

229 000001D0 88F0 <1> mov al, dh ; max. head number

230 000001D2 FEC0 <1> inc al

231 000001D4 AA <1> stosb ; logical heads (limits 256)

232 000001D5 B0A0 <1> mov al, 0A0h ; Indicates translated table

233 000001D7 AA <1> stosb

234 000001D8 8A440C <1> mov al, [si+12]

235 000001DB AA <1> stosb ; physical sectors per track

236 000001DC 31C0 <1> xor ax, ax

237 <1> ;dec ax ; 02/01/2015

238 000001DE AB <1> stosw ; precompensation (obsolete)

239 <1> ;xor al, al ; 02/01/2015

240 000001DF AA <1> stosb ; reserved

241 000001E0 B008 <1> mov al, 8 ; drive control byte

242 <1> ; (do not disable retries,

243 <1> ; more than 8 heads)

244 000001E2 AA <1> stosb

245 000001E3 8B4404 <1> mov ax, [si+4]

246 000001E6 AB <1> stosw ; physical number of cylinders

247 <1> ;push ax ; 02/01/2015

248 000001E7 8A4408 <1> mov al, [si+8]

249 000001EA AA <1> stosb ; physical num. of heads (limit 16)

250 000001EB 29C0 <1> sub ax, ax

251 <1> ;pop ax ; 02/01/2015

252 000001ED AB <1> stosw ; landing zone (obsolete)

253 000001EE 88C8 <1> mov al, cl ; logical sectors per track (limit 63)

254 000001F0 243F <1> and al, 3Fh

255 000001F2 AA <1> stosb

256 <1> ;sub al, al ; checksum

257 <1> ;stosb

258 <1> ;

259 000001F3 83C61A <1> add si, 26 ; (BIOS) DPTE address pointer

260 000001F6 AD <1> lodsw

261 000001F7 50 <1> push ax ; (BIOS) DPTE offset

262 000001F8 AD <1> lodsw

263 000001F9 50 <1> push ax ; (BIOS) DPTE segment

264 <1> ;

265 <1> ; checksum calculation

266 000001FA 89FE <1> mov si, di

267 000001FC 06 <1> push es

268 000001FD 1F <1> pop ds

269 <1> ;mov cx, 16

270 000001FE B90F00 <1> mov cx, 15

271 00000201 29CE <1> sub si, cx

272 00000203 30E4 <1> xor ah, ah

273 <1> ;del cl

274 <1> L11:

275 00000205 AC <1> lodsb

276 00000206 00C4 <1> add ah, al

277 00000208 E2FB <1> loop L11

278 <1> ;

279 0000020A 88E0 <1> mov al, ah

280 0000020C F6D8 <1> neg al ; -x+x = 0

281 0000020E AA <1> stosb ; put checksum in byte 15 of the tbl

282 <1> ;

283 0000020F 1F <1> pop ds ; (BIOS) DPTE segment

284 00000210 5E <1> pop si ; (BIOS) DPTE offset

285 <1> ;

286 <1> ; 23/02/2015

287 00000211 57 <1> push di

288 <1> ; ES:DI points to DPTE (FDPTE) location

289 <1> ;mov cx, 8

290 00000212 B108 <1> mov cl, 8

291 00000214 F3A5 <1> rep movsw

292 <1> ;

293 <1> ; 23/02/2015

294 <1> ; (P)ATA drive and LBA validation

295 <1> ; (invalidating SATA drives and setting

296 <1> ; CHS type I/O for old type fixed disks)

297 00000216 5B <1> pop bx

298 00000217 8CC8 <1> mov ax, cs

299 00000219 8ED8 <1> mov ds, ax

300 0000021B 268B07 <1> mov ax, [es:bx]

301 0000021E 3DF001 <1> cmp ax, 1F0h

302 00000221 7418 <1> je short L11a

303 00000223 3D7001 <1> cmp ax, 170h

304 00000226 7413 <1> je short L11a

305 <1> ; invalidation

306 <1> ; (because base port address is not 1F0h or 170h)

307 00000228 30FF <1> xor bh, bh

308 0000022A 88D3 <1> mov bl, dl

309 0000022C 80EB80 <1> sub bl, 80h

310 0000022F C687[F85C]00 <1> mov byte [bx+hd0\_type], 0 ; not a valid disk drive !

311 00000234 808F[445D]F0 <1> or byte [bx+drv.status+2], 0F0h ; (failure sign)

312 00000239 EB14 <1> jmp short L11b

313 <1> L11a:

314 <1> ; LBA validation

315 0000023B 268A4704 <1> mov al, [es:bx+4] ; Head register upper nibble

316 0000023F A840 <1> test al, 40h ; LBA bit (bit 6)

317 00000241 750C <1> jnz short L11b ; LBA type I/O is OK! (E0h or F0h)

318 <1> ; force CHS type I/O for this drive (A0h or B0h)

319 00000243 28FF <1> sub bh, bh

320 00000245 88D3 <1> mov bl, dl

321 00000247 80EB80 <1> sub bl, 80h ; 26/02/2015

322 0000024A 80A7[445D]FE <1> and byte [bx+drv.status+2], 0FEh ; clear bit 0

323 <1> ; bit 0 = LBA ready bit

324 <1> ; 'diskio' procedure will check this bit !

325 <1> L11b:

326 0000024F 3A16[F45C] <1> cmp dl, [last\_drv] ; 25/12/2014

327 00000253 7307 <1> jnb short L13

328 00000255 E9FCFE <1> jmp L10

329 <1> L12:

330 <1> ; Restore data registers

331 00000258 8CC8 <1> mov ax, cs

332 0000025A 8ED8 <1> mov ds, ax

333 <1> L13:

334 <1> ; 13/12/2014

335 0000025C 0E <1> push cs

336 0000025D 07 <1> pop es

337 <1> L14:

338 0000025E B411 <1> mov ah, 11h

339 00000260 CD16 <1> int 16h

340 00000262 7466 <1> jz short L16 ; no keys in keyboard buffer

341 00000264 B010 <1> mov al, 10h

342 00000266 CD16 <1> int 16h

343 00000268 EBF4 <1> jmp short L14

344 <1>

345 <1> set\_disk\_parms:

346 <1> ; 04/02/2016 (ebx -> bx)

347 <1> ; 10/07/2015

348 <1> ; 14/01/2015

349 <1> ;push bx

350 0000026A 28FF <1> sub bh, bh

351 0000026C 8A1E[F35C] <1> mov bl, [drv]

352 00000270 80FB80 <1> cmp bl, 80h

353 00000273 7203 <1> jb short sdp0

354 00000275 80EB7E <1> sub bl, 7Eh

355 <1> sdp0:

356 00000278 81C3[425D] <1> add bx, drv.status

357 0000027C C60780 <1> mov byte [bx], 80h ; 'Present' flag

358 <1> ;

359 0000027F 88E8 <1> mov al, ch ; last cylinder (bits 0-7)

360 00000281 88CC <1> mov ah, cl ;

361 00000283 C0EC06 <1> shr ah, 6 ; last cylinder (bits 8-9)

362 00000286 81EB[425D] <1> sub bx, drv.status

363 0000028A D0E3 <1> shl bl, 1

364 0000028C 81C3[FC5C] <1> add bx, drv.cylinders

365 00000290 40 <1> inc ax ; convert max. cyl number to cyl count

366 00000291 8907 <1> mov [bx], ax

367 00000293 50 <1> push ax ; \*\* cylinders

368 00000294 81EB[FC5C] <1> sub bx, drv.cylinders

369 00000298 81C3[0A5D] <1> add bx, drv.heads

370 0000029C 30E4 <1> xor ah, ah

371 0000029E 88F0 <1> mov al, dh ; heads

372 000002A0 40 <1> inc ax

373 000002A1 8907 <1> mov [bx], ax

374 000002A3 81EB[0A5D] <1> sub bx, drv.heads

375 000002A7 81C3[185D] <1> add bx, drv.spt

376 000002AB 30ED <1> xor ch, ch

377 000002AD 80E13F <1> and cl, 3Fh ; sectors (bits 0-6)

378 000002B0 890F <1> mov [bx], cx

379 000002B2 81EB[185D] <1> sub bx, drv.spt

380 000002B6 D1E3 <1> shl bx, 1

381 000002B8 81C3[265D] <1> add bx, drv.size ; disk size (in sectors)

382 <1> ; LBA size = cylinders \* heads \* secpertrack

383 000002BC F7E1 <1> mul cx

384 000002BE 89C2 <1> mov dx, ax ; heads\*spt

385 000002C0 58 <1> pop ax ; \*\* cylinders

386 000002C1 48 <1> dec ax ; 1 cylinder reserved (!?)

387 000002C2 F7E2 <1> mul dx ; cylinders \* (heads\*spt)

388 000002C4 8907 <1> mov [bx], ax

389 000002C6 895702 <1> mov [bx+2], dx

390 <1> ;

391 <1> ;pop bx

392 000002C9 C3 <1> retn

393 <1>

394 <1> L16: ; 28/05/2016

204

205 ; 10/11/2014

206 000002CA FA cli ; Disable interrupts (clear interrupt flag)

207 ; Reset Interrupt MASK Registers (Master&Slave)

208 ;mov al, 0FFh ; mask off all interrupts

209 ;out 21h, al ; on master PIC (8259)

210 ;jmp $+2 ; (delay)

211 ;out 0A1h, al ; on slave PIC (8259)

212 ;

213 ; Disable NMI

214 000002CB B080 mov al, 80h

215 000002CD E670 out 70h, al ; set bit 7 to 1 for disabling NMI

216 ;23/02/2015

217 ;nop ;

218 ;in al, 71h ; read in 71h just after writing out to 70h

219 ; for preventing unknown state (!?)

220 ;

221 ; 20/08/2014

222 ; Moving the kernel 64 KB back (to physical address 0)

223 ; DS = CS = 1000h

224 ; 05/11/2014

225 000002CF 31C0 xor ax, ax

226 000002D1 8EC0 mov es, ax ; ES = 0

227 ;

228 ; 04/07/2016 - TRDOS 386 (64K - 128K kernel)

229 000002D3 31F6 xor si, si

230 000002D5 31FF xor di, di

231 000002D7 B90040 mov cx, 16384

232 000002DA F366A5 rep movsd

233 ;

234 000002DD 06 push es ; 0

235 000002DE 68[E202] push L17

236 000002E1 CB retf

237 L17:

238 000002E2 B90010 mov cx, 1000h

239 000002E5 8EC1 mov es, cx ; 1000h

240 000002E7 01C9 add cx, cx

241 000002E9 8ED9 mov ds, cx ; 2000h

242 000002EB 29F6 sub si, si

243 000002ED 29FF sub di, di

244 000002EF B90040 mov cx, 16384

245 000002F2 F366A5 rep movsd

246

247 ; Turn off the floppy drive motor

248 000002F5 BAF203 mov dx, 3F2h

249 000002F8 EE out dx, al ; 0 ; 31/12/2013

250

251 ; Enable access to memory above one megabyte

252 L18:

253 000002F9 E464 in al, 64h

254 000002FB A802 test al, 2

255 000002FD 75FA jnz short L18

256 000002FF B0D1 mov al, 0D1h ; Write output port

257 00000301 E664 out 64h, al

258 L19:

259 00000303 E464 in al, 64h

260 00000305 A802 test al, 2

261 00000307 75FA jnz short L19

262 00000309 B0DF mov al, 0DFh ; Enable A20 line

263 0000030B E660 out 60h, al

264 ;L20:

265 ;

266 ; Load global descriptor table register

267

268 ;mov ax, cs

269 ;mov ds, ax

270

271 0000030D 2E0F0116[605C] lgdt [cs:gdtd]

272

273 00000313 0F20C0 mov eax, cr0

274 ; or eax, 1

275 00000316 40 inc ax

276 00000317 0F22C0 mov cr0, eax

277

278 ; Jump to 32 bit code

279

280 0000031A 66 db 66h ; Prefix for 32-bit

281 0000031B EA db 0EAh ; Opcode for far jump

282 0000031C [22030000] dd StartPM ; Offset to start, 32-bit

283 ; (1000h:StartPM = StartPM + 10000h)

284 00000320 0800 dw KCODE ; This is the selector for CODE32\_DESCRIPTOR,

285 ; assuming that StartPM resides in code32

286

287 ; 20/02/2017

288

289

290 [BITS 32]

291

292 StartPM:

293 ; Kernel Base Address = 0 ; 30/12/2013

294 00000322 66B81000 mov ax, KDATA ; Save data segment identifier

295 00000326 8ED8 mov ds, ax ; Move a valid data segment into DS register

296 00000328 8EC0 mov es, ax ; Move data segment into ES register

297 0000032A 8EE0 mov fs, ax ; Move data segment into FS register

298 0000032C 8EE8 mov gs, ax ; Move data segment into GS register

299 0000032E 8ED0 mov ss, ax ; Move data segment into SS register

300 00000330 BC00000900 mov esp, 90000h ; Move the stack pointer to 090000h

301

302 clear\_bss: ; Clear uninitialized data area

303 ; 11/03/2015

304 00000335 31C0 xor eax, eax ; 0

305 00000337 B9136F0000 mov ecx, (bss\_end - bss\_start)/4

306 ;shr ecx, 2 ; bss section is already aligned for double words

307 0000033C BF[4E550100] mov edi, bss\_start

308 00000341 F3AB rep stosd

309

310 memory\_init:

311 ; Initialize memory allocation table and page tables

312 ; 16/11/2014

313 ; 15/11/2014

314 ; 07/11/2014

315 ; 06/11/2014

316 ; 05/11/2014

317 ; 04/11/2014

318 ; 31/10/2014 (Retro UNIX 386 v1 - Beginning)

319 ;

320 ; xor eax, eax

321 ; xor ecx, ecx

322 00000343 B108 mov cl, 8

323 00000345 BF00001000 mov edi, MEM\_ALLOC\_TBL

324 0000034A F3AB rep stosd ; clear Memory Allocation Table

325 ; for the first 1 MB memory

326 ;

327 0000034C 668B0D[EE5C0000] mov cx, [mem\_1m\_1k] ; Number of contiguous KB between

328 ; 1 and 16 MB, max. 3C00h = 15 MB.

329 00000353 66C1E902 shr cx, 2 ; convert 1 KB count to 4 KB count

330 00000357 890D[40580100] mov [free\_pages], ecx

331 0000035D 668B15[F05C0000] mov dx, [mem\_16m\_64k] ; Number of contiguous 64 KB blocks

332 ; between 16 MB and 4 GB.

333 00000364 6609D2 or dx, dx

334 00000367 7413 jz short mi\_0

335 ;

336 00000369 6689D0 mov ax, dx

337 0000036C C1E004 shl eax, 4 ; 64 KB -> 4 KB (page count)

338 0000036F 0105[40580100] add [free\_pages], eax

339 00000375 0500100000 add eax, 4096 ; 16 MB = 4096 pages

340 0000037A EB07 jmp short mi\_1

341 mi\_0:

342 0000037C 6689C8 mov ax, cx

343 0000037F 66050001 add ax, 256 ; add 256 pages for the first 1 MB

344 mi\_1:

345 00000383 A3[3C580100] mov [memory\_size], eax ; Total available memory in pages

346 ; 1 alloc. tbl. bit = 1 memory page

347 ; 32 allocation bits = 32 mem. pages

348 ;

349 00000388 05FF7F0000 add eax, 32767 ; 32768 memory pages per 1 M.A.T. page

350 0000038D C1E80F shr eax, 15 ; ((32768 \* x) + y) pages (y < 32768)

351 ; --> x + 1 M.A.T. pages, if y > 0

352 ; --> x M.A.T. pages, if y = 0

353 00000390 66A3[50580100] mov [mat\_size], ax ; Memory Alloc. Table Size in pages

354 00000396 C1E00C shl eax, 12 ; 1 M.A.T. page = 4096 bytes

355 ; ; Max. 32 M.A.T. pages (4 GB memory)

356 00000399 89C3 mov ebx, eax ; M.A.T. size in bytes

357 ; Set/Calculate Kernel's Page Directory Address

358 0000039B 81C300001000 add ebx, MEM\_ALLOC\_TBL

359 000003A1 891D[38580100] mov [k\_page\_dir], ebx ; Kernel's Page Directory address

360 ; just after the last M.A.T. page

361 ;

362 000003A7 83E804 sub eax, 4 ; convert M.A.T. size to offset value

363 000003AA A3[48580100] mov [last\_page], eax ; last page ofset in the M.A.T.

364 ; ; (allocation status search must be

365 ; stopped after here)

366 000003AF 31C0 xor eax, eax

367 000003B1 48 dec eax ; FFFFFFFFh (set all bits to 1)

368 000003B2 6651 push cx

369 000003B4 C1E905 shr ecx, 5 ; convert 1 - 16 MB page count to

370 ; count of 32 allocation bits

371 000003B7 F3AB rep stosd

372 000003B9 6659 pop cx

373 000003BB 40 inc eax ; 0

374 000003BC 80E11F and cl, 31 ; remain bits

375 000003BF 7412 jz short mi\_4

376 000003C1 8907 mov [edi], eax ; reset

377 mi\_2:

378 000003C3 0FAB07 bts [edi], eax ; 06/11/2014

379 000003C6 FEC9 dec cl

380 000003C8 7404 jz short mi\_3

381 000003CA FEC0 inc al

382 000003CC EBF5 jmp short mi\_2

383 mi\_3:

384 000003CE 28C0 sub al, al ; 0

385 000003D0 83C704 add edi, 4 ; 15/11/2014

386 mi\_4:

387 000003D3 6609D2 or dx, dx ; check 16M to 4G memory space

388 000003D6 7421 jz short mi\_6 ; max. 16 MB memory, no more...

389 ;

390 000003D8 B900021000 mov ecx, MEM\_ALLOC\_TBL + 512 ; End of first 16 MB memory

391 ;

392 000003DD 29F9 sub ecx, edi ; displacement (to end of 16 MB)

393 000003DF 7406 jz short mi\_5 ; jump if EDI points to

394 ; end of first 16 MB

395 000003E1 D1E9 shr ecx, 1 ; convert to dword count

396 000003E3 D1E9 shr ecx, 1 ; (shift 2 bits right)

397 000003E5 F3AB rep stosd ; reset all bits for reserved pages

398 ; (memory hole under 16 MB)

399 mi\_5:

400 000003E7 6689D1 mov cx, dx ; count of 64 KB memory blocks

401 000003EA D1E9 shr ecx, 1 ; 1 alloc. dword per 128 KB memory

402 000003EC 9C pushf ; 16/11/2014

403 000003ED 48 dec eax ; FFFFFFFFh (set all bits to 1)

404 000003EE F3AB rep stosd

405 000003F0 40 inc eax ; 0

406 000003F1 9D popf ; 16/11/2014

407 000003F2 7305 jnc short mi\_6

408 000003F4 6648 dec ax ; eax = 0000FFFFh

409 000003F6 AB stosd

410 000003F7 6640 inc ax ; 0

411 mi\_6:

412 000003F9 39DF cmp edi, ebx ; check if EDI points to

413 000003FB 730A jnb short mi\_7 ; end of memory allocation table

414 ; ; (>= MEM\_ALLOC\_TBL + 4906)

415 000003FD 89D9 mov ecx, ebx ; end of memory allocation table

416 000003FF 29F9 sub ecx, edi ; convert displacement/offset

417 00000401 D1E9 shr ecx, 1 ; to dword count

418 00000403 D1E9 shr ecx, 1 ; (shift 2 bits right)

419 00000405 F3AB rep stosd ; reset all remain M.A.T. bits

420 mi\_7:

421 ; Reset M.A.T. bits in M.A.T. (allocate M.A.T. pages)

422 00000407 BA00001000 mov edx, MEM\_ALLOC\_TBL

423 ;sub ebx, edx ; Mem. Alloc. Tbl. size in bytes

424 ;shr ebx, 12 ; Mem. Alloc. Tbl. size in pages

425 0000040C 668B0D[50580100] mov cx, [mat\_size] ; Mem. Alloc. Tbl. size in pages

426 00000413 89D7 mov edi, edx

427 00000415 C1EF0F shr edi, 15 ; convert M.A.T. address to

428 ; byte offset in M.A.T.

429 ; (1 M.A.T. byte points to

430 ; 32768 bytes)

431 ; Note: MEM\_ALLOC\_TBL address

432 ; must be aligned on 128 KB

433 ; boundary!

434 00000418 01D7 add edi, edx ; points to M.A.T.'s itself

435 ; eax = 0

436 0000041A 290D[40580100] sub [free\_pages], ecx ; 07/11/2014

437 mi\_8:

438 00000420 0FB307 btr [edi], eax ; clear bit 0 to bit x (1 to 31)

439 ;dec bl

440 00000423 FEC9 dec cl

441 00000425 7404 jz short mi\_9

442 00000427 FEC0 inc al

443 00000429 EBF5 jmp short mi\_8

444 mi\_9:

445 ;

446 ; Reset Kernel's Page Dir. and Page Table bits in M.A.T.

447 ; (allocate pages for system page tables)

448

449 ; edx = MEM\_ALLOC\_TBL

450 0000042B 8B0D[3C580100] mov ecx, [memory\_size] ; memory size in pages (PTEs)

451 00000431 81C1FF030000 add ecx, 1023 ; round up (1024 PTEs per table)

452 00000437 C1E90A shr ecx, 10 ; convert memory page count to

453 ; page table count (PDE count)

454 ;

455 0000043A 51 push ecx ; (\*\*) PDE count (<= 1024)

456 ;

457 0000043B 41 inc ecx ; +1 for kernel page directory

458 ;

459 0000043C 290D[40580100] sub [free\_pages], ecx ; 07/11/2014

460 ;

461 00000442 8B35[38580100] mov esi, [k\_page\_dir] ; Kernel's Page Directory address

462 00000448 C1EE0C shr esi, 12 ; convert to page number

463 mi\_10:

464 0000044B 89F0 mov eax, esi ; allocation bit offset

465 0000044D 89C3 mov ebx, eax

466 0000044F C1EB03 shr ebx, 3 ; convert to alloc. byte offset

467 00000452 80E3FC and bl, 0FCh ; clear bit 0 and bit 1

468 ; to align on dword boundary

469 00000455 83E01F and eax, 31 ; set allocation bit position

470 ; (bit 0 to bit 31)

471 ;

472 00000458 01D3 add ebx, edx ; offset in M.A.T. + M.A.T. address

473 ;

474 0000045A 0FB303 btr [ebx], eax ; reset relevant bit (0 to 31)

475 ;

476 0000045D 46 inc esi ; next page table

477 0000045E E2EB loop mi\_10 ; allocate next kernel page table

478 ; (ecx = page table count + 1)

479 ;

480 00000460 59 pop ecx ; (\*\*) PDE count (= pg. tbl. count)

481 ;

482 ; Initialize Kernel Page Directory and Kernel Page Tables

483 ;

484 ; Initialize Kernel's Page Directory

485 00000461 8B3D[38580100] mov edi, [k\_page\_dir]

486 00000467 89F8 mov eax, edi

487 00000469 0C03 or al, PDE\_A\_PRESENT + PDE\_A\_WRITE

488 ; supervisor + read&write + present

489 0000046B 89CA mov edx, ecx ; (\*\*) PDE count (= pg. tbl. count)

490 mi\_11:

491 0000046D 0500100000 add eax, 4096 ; Add page size (PGSZ)

492 ; EAX points to next page table

493 00000472 AB stosd

494 00000473 E2F8 loop mi\_11

495 00000475 29C0 sub eax, eax ; Empty PDE

496 00000477 66B90004 mov cx, 1024 ; Entry count (PGSZ/4)

497 0000047B 29D1 sub ecx, edx

498 0000047D 7402 jz short mi\_12

499 0000047F F3AB rep stosd ; clear remain (empty) PDEs

500 ;

501 ; Initialization of Kernel's Page Directory is OK, here.

502 mi\_12:

503 ; Initialize Kernel's Page Tables

504 ;

505 ; (EDI points to address of page table 0)

506 ; eax = 0

507 00000481 8B0D[3C580100] mov ecx, [memory\_size] ; memory size in pages

508 00000487 89CA mov edx, ecx ; (\*\*\*)

509 00000489 B003 mov al, PTE\_A\_PRESENT + PTE\_A\_WRITE

510 ; supervisor + read&write + present

511 mi\_13:

512 0000048B AB stosd

513 0000048C 0500100000 add eax, 4096

514 00000491 E2F8 loop mi\_13

515 00000493 6681E2FF03 and dx, 1023 ; (\*\*\*)

516 00000498 740B jz short mi\_14

517 0000049A 66B90004 mov cx, 1024

518 0000049E 6629D1 sub cx, dx ; from dx (<= 1023) to 1024

519 000004A1 31C0 xor eax, eax

520 000004A3 F3AB rep stosd ; clear remain (empty) PTEs

521 ; of the last page table

522 mi\_14:

523 ; Initialization of Kernel's Page Tables is OK, here.

524 ;

525 000004A5 89F8 mov eax, edi ; end of the last page table page

526 ; (beginging of user space pages)

527 000004A7 C1E80F shr eax, 15 ; convert to M.A.T. byte offset

528 000004AA 24FC and al, 0FCh ; clear bit 0 and bit 1 for

529 ; aligning on dword boundary

530

531 000004AC A3[4C580100] mov [first\_page], eax

532 000004B1 A3[44580100] mov [next\_page], eax ; The first free page pointer

533 ; for user programs

534 ; (Offset in Mem. Alloc. Tbl.)

535 ;

536 ; Linear/FLAT (1 to 1) memory paging for the kernel is OK, here.

537 ;

538

539 ; Enable paging

540 ;

541 000004B6 A1[38580100] mov eax, [k\_page\_dir]

542 000004BB 0F22D8 mov cr3, eax

543 000004BE 0F20C0 mov eax, cr0

544 000004C1 0D00000080 or eax, 80000000h ; set paging bit (bit 31)

545 000004C6 0F22C0 mov cr0, eax

546 ;jmp KCODE:StartPMP

547

548 000004C9 EA db 0EAh ; Opcode for far jump

549 000004CA [D0040000] dd StartPMP ; 32 bit offset

550 000004CE 0800 dw KCODE ; kernel code segment descriptor

551

552

553 StartPMP:

554 ; 06/11//2014

555 ; Clear video page 0

556 ;

557 ; Temporary Code

558 ;

559 000004D0 B9E8030000 mov ecx, 80\*25/2

560 000004D5 BF00800B00 mov edi, 0B8000h

561 ; 30/01/2016

562 ;xor eax, eax ; black background, black fore color

563 000004DA B800070007 mov eax, 07000700h ; black background, light gray fore color

564 000004DF F3AB rep stosd

565

566 ; 19/08/2014

567 ; Kernel Base Address = 0

568 ; It is mapped to (physically) 0 in the page table.

569 ; So, here is exactly 'StartPMP' address.

570

571 ; 29/01/2016 (TRDOS 386 = TRDOS v2.0)

572 000004E1 BE[8D190100] mov esi, starting\_msg

573 ;; 14/08/2015 (kernel version message will appear

574 ;; when protected mode and paging is enabled)

575 000004E6 BF00800B00 mov edi, 0B8000h ; 27/08/2014

576 000004EB B40A mov ah, 0Ah ; Black background, light green forecolor

577 ; 20/08/2014

578 000004ED E88F010000 call printk

579

580 ; 'UNIX v7/x86' source code by Robert Nordier (1999)

581 ; // Set IRQ offsets

582 ;

583 ; Linux (v0.12) source code by Linus Torvalds (1991)

584 ;

585 ;; ICW1

586 000004F2 B011 mov al, 11h ; Initialization sequence

587 000004F4 E620 out 20h, al ; 8259A-1

588 ; jmp $+2

589 000004F6 E6A0 out 0A0h, al ; 8259A-2

590 ;; ICW2

591 000004F8 B020 mov al, 20h ; Start of hardware ints (20h)

592 000004FA E621 out 21h, al ; for 8259A-1

593 ; jmp $+2

594 000004FC B028 mov al, 28h ; Start of hardware ints (28h)

595 000004FE E6A1 out 0A1h, al ; for 8259A-2

596 ;

597 00000500 B004 mov al, 04h ;; ICW3

598 00000502 E621 out 21h, al ; IRQ2 of 8259A-1 (master)

599 ; jmp $+2

600 00000504 B002 mov al, 02h ; is 8259A-2 (slave)

601 00000506 E6A1 out 0A1h, al ;

602 ;; ICW4

603 00000508 B001 mov al, 01h ;

604 0000050A E621 out 21h, al ; 8086 mode, normal EOI

605 ; jmp $+2

606 0000050C E6A1 out 0A1h, al ; for both chips.

607

608 ;mov al, 0FFh ; mask off all interrupts for now

609 ;out 21h, al

610 ;; jmp $+2

611 ;out 0A1h, al

612

613 ; 02/04/2015

614 ; 26/03/2015 System call (INT 30h) modification

615 ; DPL = 3 (Interrupt service routine can be called from user mode)

616 ;

617 ;; Linux (v0.12) source code by Linus Torvalds (1991)

618 ; setup\_idt:

619 ;

620 ;; 16/02/2015

621 ;;mov dword [DISKETTE\_INT], fdc\_int ; IRQ 6 handler

622 ; 21/08/2014 (timer\_int)

623 0000050E BE[50160100] mov esi, ilist

624 00000513 8D3D[50550100] lea edi, [idt]

625 ; 26/03/2015

626 00000519 B930000000 mov ecx, 48 ; 48 hardware interrupts (INT 0 to INT 2Fh)

627 ; 02/04/2015

628 0000051E BB00000800 mov ebx, 80000h

629 rp\_sidt1:

630 00000523 AD lodsd

631 00000524 89C2 mov edx, eax

632 00000526 66BA008E mov dx, 8E00h

633 0000052A 6689C3 mov bx, ax

634 0000052D 89D8 mov eax, ebx ; /\* selector = 0x0008 = cs \*/

635 ; /\* interrupt gate - dpl=0, present \*/

636 0000052F AB stosd ; selector & offset bits 0-15

637 00000530 89D0 mov eax, edx

638 00000532 AB stosd ; attributes & offset bits 16-23

639 00000533 E2EE loop rp\_sidt1

640 ; 15/04/2016

641 ; TRDOS 386 (TRDOS v2.0) /// 32 sofware interrupts ///

642 ;mov cl, 16 ; 16 software interrupts (INT 30h to INT 3Fh)

643 00000535 B120 mov cl, 32 ; 32 software interrupts (INT 30h to INT 4Fh)

644 rp\_sidt2:

645 00000537 AD lodsd

646 00000538 21C0 and eax, eax

647 0000053A 7413 jz short rp\_sidt3

648 0000053C 89C2 mov edx, eax

649 0000053E 66BA00EE mov dx, 0EE00h ; P=1b/DPL=11b/01110b

650 00000542 6689C3 mov bx, ax

651 00000545 89D8 mov eax, ebx ; selector & offset bits 0-15

652 00000547 AB stosd

653 00000548 89D0 mov eax, edx

654 0000054A AB stosd

655 0000054B E2EA loop rp\_sidt2

656 0000054D EB16 jmp short sidt\_OK

657 rp\_sidt3:

658 0000054F B8[AA0A0000] mov eax, ignore\_int

659 00000554 89C2 mov edx, eax

660 00000556 66BA00EE mov dx, 0EE00h ; P=1b/DPL=11b/01110b

661 0000055A 6689C3 mov bx, ax

662 0000055D 89D8 mov eax, ebx ; selector & offset bits 0-15

663 rp\_sidt4:

664 0000055F AB stosd

665 00000560 92 xchg eax, edx

666 00000561 AB stosd

667 00000562 92 xchg edx, eax

668 00000563 E2FA loop rp\_sidt4

669 sidt\_OK:

670 00000565 0F011D[665C0000] lidt [idtd]

671 ;

672 ; TSS descriptor setup ; 24/03/2015

673 0000056C B8[D0570100] mov eax, task\_state\_segment

674 00000571 66A3[5A5C0000] mov [gdt\_tss0], ax

675 00000577 C1C010 rol eax, 16

676 0000057A A2[5C5C0000] mov [gdt\_tss1], al

677 0000057F 8825[5F5C0000] mov [gdt\_tss2], ah

678 00000585 66C705[36580100]68- mov word [tss.IOPB], tss\_end - task\_state\_segment

678 0000058D 00

679 ;

680 ; IO Map Base address (When this address points

681 ; to end of the TSS, CPU does not use IO port

682 ; permission bit map for RING 3 IO permissions,

683 ; access to any IO ports in ring 3 will be forbidden.)

684 ;

685 ;mov [tss.esp0], esp ; TSS offset 4

686 ;mov word [tss.ss0], KDATA ; TSS offset 8 (SS)

687 0000058E 66B82800 mov ax, TSS ; It is needed when an interrupt

688 ; occurs (or a system call -software INT- is requested)

689 ; while cpu running in ring 3 (in user mode).

690 ; (Kernel stack pointer and segment will be loaded

691 ; from offset 4 and 8 of the TSS, by the CPU.)

692 00000592 0F00D8 ltr ax ; Load task register

693 ;

694 esp0\_set0:

695 ; 30/07/2015

696 00000595 8B0D[3C580100] mov ecx, [memory\_size] ; memory size in pages

697 0000059B C1E10C shl ecx, 12 ; convert page count to byte count

698 0000059E 81F900004000 cmp ecx, CORE ; beginning of user's memory space (400000h)

699 ; (kernel mode virtual address)

700 000005A4 7605 jna short esp0\_set1

701 ;

702 ; If available memory > CORE (end of the 1st 4 MB)

703 ; set stack pointer to CORE

704 ;(Because, PDE 0 is reserved for kernel space in user's page directory)

705 ;(PDE 0 points to page table of the 1st 4 MB virtual address space)

706 000005A6 B900004000 mov ecx, CORE

707 esp0\_set1:

708 000005AB 89CC mov esp, ecx ; top of kernel stack (\*\*tss.esp0\*\*)

709 esp0\_set\_ok:

710 ; 30/07/2015 (\*\*tss.esp0\*\*)

711 000005AD 8925[D4570100] mov [tss.esp0], esp

712 000005B3 66C705[D8570100]10- mov word [tss.ss0], KDATA

712 000005BB 00

713 ; 14/08/2015

714 ; 10/11/2014 (Retro UNIX 386 v1 - Erdogan Tan)

715 ;

716 ;cli ; Disable interrupts (for CPU)

717 ; (CPU will not handle hardware interrupts, except NMI!)

718 ;

719 000005BC 30C0 xor al, al ; Enable all hardware interrupts!

720 000005BE E621 out 21h, al ; (IBM PC-AT compatibility)

721 000005C0 EB00 jmp $+2 ; (All conventional PC-AT hardware

722 000005C2 E6A1 out 0A1h, al ; interrupts will be in use.)

723 ; (Even if related hardware component

724 ; does not exist!)

725 ; Enable NMI

726 000005C4 B07F mov al, 7Fh ; Clear bit 7 to enable NMI (again)

727 000005C6 E670 out 70h, al

728 ; 23/02/2015

729 000005C8 90 nop

730 000005C9 E471 in al, 71h ; read in 71h just after writing out to 70h

731 ; for preventing unknown state (!?)

732 ;

733 ; Only a NMI can occur here... (Before a 'STI' instruction)

734 ;

735 ; 02/09/2014

736 000005CB 6631DB xor bx, bx

737 000005CE 66BA0002 mov dx, 0200h ; Row 2, column 0 ; 07/03/2015

738 000005D2 E871170000 call \_set\_cpos ; 24/01/2016

739 ;

740 ; 06/11/2014

741 000005D7 E8782C0000 call memory\_info

742 ; 14/08/2015

743 ;call getch ; 28/02/2015

744 drv\_init:

745 000005DC FB sti ; Enable Interrupts

746 ; 06/02/2015

747 000005DD 8B15[F85C0000] mov edx, [hd0\_type] ; hd0, hd1, hd2, hd3

748 000005E3 668B1D[F65C0000] mov bx, [fd0\_type] ; fd0, fd1

749 ; 22/02/2015

750 000005EA 6621DB and bx, bx

751 000005ED 751C jnz short di1

752 ;

753 000005EF 09D2 or edx, edx

754 000005F1 752A jnz short di2

755 ;

756 setup\_error:

757 000005F3 BE[56190100] mov esi, setup\_error\_msg

758 psem:

759 000005F8 AC lodsb

760 000005F9 08C0 or al, al

761 ;jz short haltx ; 22/02/2015

762 000005FB 7427 jz short di3

763 000005FD 56 push esi

764 ; 13/05/2016

765 000005FE BB07000000 mov ebx, 7 ; Black background,

766 ; light gray forecolor

767 ; Video page 0 (BH=0)

768 00000603 E8AA160000 call \_write\_tty

769 00000608 5E pop esi

770 00000609 EBED jmp short psem

771

772 di1:

773 ; supress 'jmp short T6'

774 ; (activate fdc motor control code)

775 0000060B 66C705[EB060000]90- mov word [T5], 9090h ; nop

775 00000613 90

776 ;

777 ;mov ax, int\_0Eh ; IRQ 6 handler

778 ;mov di, 0Eh\*4 ; IRQ 6 vector

779 ;stosw

780 ;mov ax, cs

781 ;stosw

782 ;; 16/02/2015

783 ;;mov dword [DISKETTE\_INT], fdc\_int ; IRQ 6 handler

784 ;

785 00000614 E8AF3B0000 CALL DSKETTE\_SETUP ; Initialize Floppy Disks

786 ;

787 00000619 09D2 or edx, edx

788 0000061B 7407 jz short di3

789 di2:

790 0000061D E8EC3B0000 call DISK\_SETUP ; Initialize Fixed Disks

791 00000622 72CF jc short setup\_error

792 di3:

793 00000624 E8FF2B0000 call setup\_rtc\_int ; 22/05/2015 (dsectrpm.s)

794 ;

795 00000629 E8BE110100 call display\_disks ; 07/03/2015 (Temporary)

796 ;haltx:

797 ; 14/08/2015

798 ;call getch ; 22/02/2015

799 ;sti ; Enable interrupts (for CPU)

800 ; ; 29/01/2016

801 ; sub ah, ah ; read time count

802 ; call int1Ah

803 ; mov edx, ecx ; 18.2 \* seconds

804 ;md\_info\_msg\_wait1:

805 ; ; 29/01/2016

806 ; mov ah, 1

807 ; call int16h

808 ; jz short md\_info\_msg\_wait2

809 ; xor ah, ah ; 0

810 ; call int16h

811 ; jmp short md\_info\_msg\_ok

812 ;md\_info\_msg\_wait2:

813 ; sub ah, ah ; read time count

814 ; call int1Ah

815 ; cmp edx, ecx ; ; 18.2 \* seconds

816 ; jna short md\_info\_msg\_wait3

817 ; xchg edx, ecx

818 ;md\_info\_msg\_wait3:

819 ; sub ecx, edx

820 ; cmp ecx, 127 ; 7 seconds (18.2 \* 7)

821 ; jb short md\_info\_msg\_wait1

822 ;md\_info\_msg\_ok:

823 ; 08/09/2016

824 0000062E 0F20C0 mov eax, cr0

825 00000631 A810 test al, 10h ; Bit 4, ET (Extension Type)

826 00000633 7408 jz short sysinit

827 ; 27/02/2017

828 00000635 FE05[F8650100] inc byte [fpready]

829 ; 80387 (FPU) is ready

830 0000063B DBE3 fninit ; Initialize Floating-Point Unit

831 sysinit:

832 ; 30/06/2015

833 0000063D E80C5C0000 call sys\_init

834 ;

835 ;jmp cpu\_reset ; 22/02/2015

836 hang:

837 ; 23/02/2015

838 ;sti ; Enable interrupts

839 00000642 F4 hlt

840 ;

841 ;nop

842 ;; 03/12/2014

843 ;; 28/08/2014

844 ;mov ah, 11h

845 ;call getc

846 ;jz \_c8

847 ;

848 ; 23/02/2015

849 ; 06/02/2015

850 ; 07/09/2014

851 00000643 31DB xor ebx, ebx

852 00000645 8A1D[66580100] mov bl, [ptty] ; active\_page

853 0000064B 89DE mov esi, ebx

854 0000064D 66D1E6 shl si, 1

855 00000650 81C6[68580100] add esi, ttychr

856 00000656 668B06 mov ax, [esi]

857 00000659 6621C0 and ax, ax

858 ;jz short \_c8

859 0000065C 74E4 jz short hang

860 0000065E 66C7060000 mov word [esi], 0

861 00000663 80FB03 cmp bl, 3 ; Video page 3

862 ;jb short \_c8

863 00000666 72DA jb short hang

864 ;

865 ; 13/05/2016

866 ; 07/09/2014

867 nxtl:

868 00000668 6653 push bx

869 0000066A 66BB0E00 mov bx, 0Eh ; Yellow character

870 ; on black background

871 ; bh = 0 (video page 0)

872 ; Retro UNIX 386 v1 - Video Mode 0

873 ; (PC/AT Video Mode 3 - 80x25 Alpha.)

874 0000066E 6650 push ax

875 00000670 E83D160000 call \_write\_tty

876 00000675 6658 pop ax

877 00000677 665B pop bx

878 00000679 3C0D cmp al, 0Dh ; carriage return (enter)

879 ;jne short \_c8

880 0000067B 75C5 jne short hang

881 0000067D B00A mov al, 0Ah ; next line

882 0000067F EBE7 jmp short nxtl

883

884 ;\_c8:

885 ; ; 25/08/2014

886 ; cli ; Disable interrupts

887 ; mov al, [scounter + 1]

888 ; and al, al

889 ; jnz hang

890 ; call rtc\_p

891 ; jmp hang

892

893

894 ; 27/08/2014

895 ; 20/08/2014

896 printk:

897 ;mov edi, [scr\_row]

898 pkl:

899 00000681 AC lodsb

900 00000682 08C0 or al, al

901 00000684 7404 jz short pkr

902 00000686 66AB stosw

903 00000688 EBF7 jmp short pkl

904 pkr:

905 0000068A C3 retn

906

907 ; 28/02/2017

908 ; 22/01/2017

909 ; 15/01/2017

910 ; 14/01/2017

911 ; 02/01/2017

912 ; 25/12/2016

913 ; 19/12/2016

914 ; 10/12/2016 (callback)

915 ; 06/06/2016

916 ; 23/05/2016

917 ; 22/05/2016 - TRDOS 386 (TRDOS v2.0) Timer Event Modifications

918 ; 25/07/2015

919 ; 14/05/2015 (multi tasking -time sharing- 'clock', x\_timer)

920 ; 17/02/2015

921 ; 06/02/2015 (unix386.s)

922 ; 11/12/2014 - 22/12/2014 (dsectrm2.s)

923 ;

924 ; IBM PC-XT Model 286 Source Code - BIOS2.ASM (06/10/85)

925 ;

926 ;-- HARDWARE INT 08 H - ( IRQ LEVEL 0 ) ---------------------------------------

927 ; THIS ROUTINE HANDLES THE TIMER INTERRUPT FROM FROM CHANNEL 0 OF :

928 ; THE 8254 TIMER. INPUT FREQUENCY IS 1.19318 MHZ AND THE DIVISOR :

929 ; IS 65536, RESULTING IN APPROXIMATELY 18.2 INTERRUPTS EVERY SECOND. :

930 ; :

931 ; THE INTERRUPT HANDLER MAINTAINS A COUNT (40:6C) OF INTERRUPTS SINCE :

932 ; POWER ON TIME, WHICH MAY BE USED TO ESTABLISH TIME OF DAY. :

933 ; THE INTERRUPT HANDLER ALSO DECREMENTS THE MOTOR CONTROL COUNT (40:40) :

934 ; OF THE DISKETTE, AND WHEN IT EXPIRES, WILL TURN OFF THE :

935 ; DISKETTE MOTOR(s), AND RESET THE MOTOR RUNNING FLAGS. :

936 ; THE INTERRUPT HANDLER WILL ALSO INVOKE A USER ROUTINE THROUGH :

937 ; INTERRUPT 1CH AT EVERY TIME TICK. THE USER MUST CODE A :

938 ; ROUTINE AND PLACE THE CORRECT ADDRESS IN THE VECTOR TABLE. :

939 ;-------------------------------------------------------------------------------

940 ;

941

942 timer\_int: ; IRQ 0

943 ;int\_08h: ; Timer

944 ; 14/10/2015

945 ; Here, we are simulating system call entry (for task switch)

946 ; (If multitasking is enabled,

947 ; 'clock' procedure may jump to 'sysrelease')

948

949 0000068B 1E push ds

950 0000068C 06 push es

951 0000068D 0FA0 push fs

952 0000068F 0FA8 push gs

953

954 00000691 60 pushad ; eax, ecx, edx, ebx, esp -before pushad-, ebp, esi, edi

955 00000692 66B91000 mov cx, KDATA

956 00000696 8ED9 mov ds, cx

957 00000698 8EC1 mov es, cx

958 0000069A 8EE1 mov fs, cx

959 0000069C 8EE9 mov gs, cx

960

961 0000069E 0F20D9 mov ecx, cr3

962 000006A1 890D[5C040300] mov [cr3reg], ecx ; save current cr3 register value/content

963

964 ; 14/01/2017

965 000006A7 3B0D[38580100] cmp ecx, [k\_page\_dir]

966 000006AD 7409 je short T3

967

968 000006AF 8B0D[38580100] mov ecx, [k\_page\_dir]

969 000006B5 0F22D9 mov cr3, ecx

970 T3:

971 ;sti ; INTERRUPTS BACK ON

972 000006B8 66FF05[B8580100] INC word [TIMER\_LOW] ; INCREMENT TIME

973 000006BF 7507 JNZ short T4 ; GO TO TEST\_DAY

974 000006C1 66FF05[BA580100] INC word [TIMER\_HIGH] ; INCREMENT HIGH WORD OF TIME

975 T4: ; TEST\_DAY

976 000006C8 66833D[BA580100]18 CMP word [TIMER\_HIGH],018H ; TEST FOR COUNT EQUALING 24 HOURS

977 000006D0 7519 JNZ short T5 ; GO TO DISKETTE\_CTL

978 000006D2 66813D[B8580100]B0- CMP word [TIMER\_LOW],0B0H

978 000006DA 00

979 000006DB 750E JNZ short T5 ; GO TO DISKETTE\_CTL

980

981 ;----- TIMER HAS GONE 24 HOURS

982 ;;SUB AX,AX

983 ;MOV [TIMER\_HIGH],AX

984 ;MOV [TIMER\_LOW],AX

985 000006DD 29C0 sub eax, eax

986 000006DF A3[B8580100] mov [TIMER\_LH], eax

987 ;

988 000006E4 C605[BC580100]01 MOV byte [TIMER\_OFL],1

989

990 ;----- TEST FOR DISKETTE TIME OUT

991

992 T5:

993 ; 23/12/2014

994 000006EB EB1D jmp short T6 ; will be replaced with nop, nop

995 ; (9090h) if a floppy disk

996 ; is detected.

997 ;mov al,[CS:MOTOR\_COUNT]

998 000006ED A0[BF580100] mov al, [MOTOR\_COUNT]

999 000006F2 FEC8 dec al

1000 ;mov [CS:MOTOR\_COUNT], al ; DECREMENT DISKETTE MOTOR CONTROL

1001 000006F4 A2[BF580100] mov [MOTOR\_COUNT], al

1002 ;mov [ORG\_MOTOR\_COUNT], al

1003 000006F9 750F JNZ short T6 ; RETURN IF COUNT NOT OUT

1004 000006FB B0F0 mov al,0F0h

1005 ;AND [CS:MOTOR\_STATUS],al ; TURN OFF MOTOR RUNNING BITS

1006 000006FD 2005[BE580100] and [MOTOR\_STATUS], al

1007 ;and [ORG\_MOTOR\_STATUS], al

1008 00000703 B00C MOV AL,0CH ; bit 3 = enable IRQ & DMA,

1009 ; bit 2 = enable controller

1010 ; 1 = normal operation

1011 ; 0 = reset

1012 ; bit 0, 1 = drive select

1013 ; bit 4-7 = motor running bits

1014 00000705 66BAF203 MOV DX,03F2H ; FDC CTL PORT

1015 00000709 EE OUT DX,AL ; TURN OFF THE MOTOR

1016 T6:

1017 ;inc word [CS:wait\_count] ; 22/12/2014 (byte -> word)

1018 ; TIMER TICK INTERRUPT

1019 ;;inc word [wait\_count] ;;27/02/2015

1020 ;INT 1CH ; TRANSFER CONTROL TO A USER ROUTINE

1021 ;cli

1022 0000070A E857040000 call u\_timer ; TRANSFER CONTROL TO A USER ROUTINE

1023 ; 23/05/2016

1024 0000070F E823F20000 call clock ; Multi Tasking control procedure

1025 T7:

1026 ; 14/10/2015

1027 00000714 B020 MOV AL,EOI ; GET END OF INTERRUPT MASK

1028 00000716 FA CLI ; DISABLE INTERRUPTS TILL STACK CLEARED

1029 00000717 E620 OUT INTA00,AL ; END OF INTERRUPT TO 8259 - 1

1030 ;

1031 rtc\_int\_2:

1032 ; 26/12/2016

1033 ;mov ecx, [cr3reg]

1034 ; 13/01/2017

1035 00000719 803D[D4030300]00 cmp byte [u.t\_lock], 0 ; T\_LOCK

1036 00000720 7730 ja short timer\_int\_return ; Timer Lock : 'sysrele' is needed !

1037 ; 28/02/2017

1038 ; We need to exit if the user's IRQ callback service is in progress!

1039 ; (To prevent a conflict!)

1040 00000722 803D[D8030300]00 cmp byte [u.r\_lock], 0 ; R\_LOCK, IRQ callback service lock !

1041 00000729 7727 ja short timer\_int\_return ; Timer Lock : 'sysrele' is needed !

1042 ; 15/01/2017

1043 0000072B 803D[CC650100]02 cmp byte [priority], 2

1044 00000732 733A jnb short T8 ; current process has a timer event (15/01/2017)

1045 ; 22/05/2016

1046 00000734 803D[CD650100]00 cmp byte [p\_change], 0 ; in 'set\_run\_sequence', in 'rtc\_p'

1047 0000073B 7615 jna short timer\_int\_return ; 23/05/2016

1048

1049 ; 15/01/2017

1050

1051 ; present process must be changed with high priority process

1052 ;xor al, al

1053 0000073D 31C0 xor eax, eax ; 26/12/2016

1054 0000073F A2[CD650100] mov [p\_change], al ; 0

1055 ;mov byte [priority], 2 ; 15/01/2017 (there is a timer event)

1056

1057 00000744 803D[5B030300]FF cmp byte [sysflg], 0FFh ; user or system space ?

1058 0000074B 7416 je short rtc\_int\_3 ; user space ([sysflg]= 0FFh)

1059

1060 ; system space, wait for 'sysret'

1061 ; to change running process

1062 ; with high priority (event) process

1063

1064 0000074D A2[A8030300] mov [u.quant], al ; 0

1065

1066 timer\_int\_return: ; 23/05/2016 - jump from 'rtc\_int' ('rtc\_int\_2')

1067 00000752 8B0D[5C040300] mov ecx, [cr3reg] ; previous value/content of cr3 register

1068 00000758 0F22D9 mov cr3, ecx ; restore cr3 register content

1069 ;

1070 0000075B 61 popad ; edi, esi, ebp, temp (icrement esp by 4), ebx, edx, ecx, eax

1071 ;

1072 0000075C 0FA9 pop gs

1073 0000075E 0FA1 pop fs

1074 00000760 07 pop es

1075 00000761 1F pop ds

1076 ;

1077 00000762 CF iretd ; return from interrupt

1078

1079 rtc\_int\_3:

1080 00000763 FE05[5B030300] inc byte [sysflg] ; now, we are in system space

1081 ;

1082 00000769 E990BF0000 jmp sysrelease ; change running process immediatelly

1083

1084 T8:

1085 ; 13/01/2017 (eax -> ebx)

1086 ; callback checking... (19/12/2016)

1087 0000076E 31DB xor ebx, ebx

1088 00000770 871D[D0030300] xchg ebx, [u.tcb] ; callback address (0 = normal return)

1089 00000776 09DB or ebx, ebx

1090 00000778 74D8 jz short timer\_int\_return

1091

1092 ; Set user's callback routine as return address from this interrupt

1093 ; and set normal return address as return address from callback

1094 ; routine!!! (19/12/2016)

1095

1096 ; 14/01/2017

1097 ; 13/01/2017 - Timer Lock (T\_LOCK)

1098 0000077A FE05[D4030300] inc byte [u.t\_lock]

1099 00000780 8A0D[5B030300] mov cl, [sysflg]

1100 00000786 880D[D5030300] mov [u.t\_mode], cl

1101

1102 0000078C 8B2D[D4570100] mov ebp, [tss.esp0] ; kernel stack address (for ring 0)

1103 00000792 83ED14 sub ebp, 20 ; eip, cs, eflags, esp, ss

1104 00000795 892D[5C030300] mov [u.sp], ebp

1105 0000079B 8925[60030300] mov [u.usp], esp

1106

1107 ;or word [ebp+8], 200h ; 22/01/2017, force enabling interrupts

1108

1109 000007A1 8B44241C mov eax, [esp+28] ; pushed eax

1110 000007A5 A3[64030300] mov [u.r0], eax

1111

1112 000007AA E87EDE0000 call wswap ; save user's registers & status

1113

1114 ; software int is in ring 0 but timer int must return to ring 3

1115 ; so, ring 3 return address and stack registers

1116 ; (eip, cs, eflags, esp, ss)

1117 ; must be copied to timer int return

1118 ; eip will be replaced by callback service routine address

1119

1120 000007AF C605[5B030300]FF mov byte [sysflg], 0FFh ; user mode

1121

1122 ; system mode (system call)

1123 ;mov ebp, [u.sp] ; EIP (u), CS (UCODE), EFLAGS (u),

1124 ; ESP (u), SS (UDATA)

1125

1126 000007B6 8B4510 mov eax, [ebp+16] ; SS (UDATA

1127 000007B9 89E6 mov esi, esp

1128 000007BB 50 push eax

1129 000007BC 50 push eax

1130 000007BD 89E7 mov edi, esp

1131 000007BF 893D[60030300] mov [u.usp], edi

1132 000007C5 B908000000 mov ecx, ((ESPACE/4) - 4) ; except DS, ES, FS, GS

1133 000007CA F3A5 rep movsd

1134 000007CC B104 mov cl, 4

1135 000007CE F3AB rep stosd

1136 000007D0 893D[5C030300] mov [u.sp], edi

1137 000007D6 89EE mov esi, ebp

1138 000007D8 B105 mov cl, 5 ; EIP (u), CS (UCODE), EFLAGS (u), ESP (u), SS (UDATA)

1139 000007DA F3A5 rep movsd

1140

1141 000007DC 8B0D[B8030300] mov ecx, [u.pgdir]

1142 000007E2 890D[5C040300] mov [cr3reg], ecx

1143

1144 ; 13/01/207 (eax -> ebx)

1145 ; EBX = callback routine address (virtual, not physical address!)

1146

1147 ; 09/01/2017

1148 ; !!! CALLBACK ROUTINE MUST BE ENDED/RETURNED WITH 'sysrele'

1149 ; system call !!!

1150 ; 25/12/2016

1151 ; Callback Note: (19/12/2016)

1152 ; !!! CALLBACK ROUTINE MUST BE ENDED/RETURNED WITH 'RETN' !!!

1153 ; pushf ; save flags

1154 ; <callback service code>

1155 ; popf ; restore flags

1156 ; retn ; return to normal running address

1157 ;

1158

1159 ; 15/01/2017

1160 ; 14/01/2017

1161 ; 13/01/2017 (eax -> ebx)

1162 ; 10/01/2017

1163 set\_callback\_addr:

1164 ; 09/01/2017 (\*\*)

1165 ; 02/01/2017 (\*)

1166 ; 25/12/2016 (\*)

1167 ; 19/12/2016 (TRDOS 386 feature only!)

1168 ;

1169 ; This routine sets return address

1170 ; to start of user's interrupt

1171 ; service (callback) address

1172 ;; and sets callback 'retn' address to normal

1173 ;; return address of user's running code!

1174 ;

1175 ; INPUT:

1176 ; EBX = callback routine/service address

1177 ; (virtual, not physical address!)

1178 ; [u.sp] = kernel stack, points to

1179 ; user's EIP,CS,EFLAGS,ESP,SS

1180 ; registers.

1181 ; OUTPUT:

1182 ; EIP (user) = callback (service) address

1183 ; CS (user) = UCODE

1184 ; EFLAGS (user) = flags before callback

1185 ; ESP (user) = ESP-4 (user, before callback)

1186 ; [ESP](user) = EIP (user) before callback

1187 ;

1188 ; Note: If CPU was in user mode while entering

1189 ; the timer interrupt service routine,

1190 ; 'IRET' will get return to callback routine

1191 ; immediately. If CPU was in system/kernel mode

1192 ; 'iret' will get return to system call and

1193 ; then, callback routine will be return address

1194 ; from system call. (User's callback/service code

1195 ; will be able to return to normal return address

1196 ; via an 'retn' at the end.)

1197 ;

1198 ; Note(\*\*): User's callback service code must be ended

1199 ; with a 'sysrele' sytstem call ! (09/01/2017)

1200 ;

1201 ; For example:

1202 ;

1203 ; timer\_callback:

1204 ; ...

1205 ; inc dword [time\_counter]

1206 ; ...

1207 ; mov eax, 39 ; 'sysrele'

1208 ; int 40h ; TRDOS 386 system call (interrupt)

1209 ;

1210 ;

1211 ;; Note(\*): User's callback service code must preserve cpu

1212 ;; flags if it has any instructions which changes

1213 ;; flags in the service code. (25/12/2016)

1214 ;;

1215 ;; For example:

1216 ;;

1217 ;; timer\_callback:

1218 ;; pushf ; save flags

1219 ;; ; this instruction changes zero flag

1220 ;; inc dword [time\_counter]

1221 ;; popf ; restore flags

1222 ;; retn ; return to normal user code

1223 ;; (which is interrupted by the

1224 ;; timer interput)

1225 ;;

1226

1227 ; 15/01/2017

1228 000007E8 8B2D[5C030300] mov ebp, [u.sp]; kernel's stack, points to EIP (user)

1229 000007EE 895D00 mov [ebp], ebx

1230 000007F1 E95CFFFFFF jmp timer\_int\_return

1231

1232 ; 15/01/2017

1233 ; 13/01/2017

1234 ; 19/12/2016

1235 ; 06/06/2016

1236 ; 23/05/2016

1237 ; 22/05/2016

1238 ; 19/05/2016 - TRDOS 386 (TRDOS v2.0)

1239 ; 26/02/2015

1240 ; 07/09/2014

1241 ; 25/08/2014

1242 rtc\_int: ; Real Time Clock Interrupt (IRQ 8)

1243 ; 22/05/2016

1244 000007F6 1E push ds ; \*\* ; 23/05/2016

1245 000007F7 50 push eax ; \*

1246 000007F8 66B81000 mov ax, KDATA

1247 000007FC 8ED8 mov ds, ax

1248 ;

1249 000007FE 8A25[B6580100] mov ah, [RTC\_2Hz] ; 2 Hz interrupt to 1 Hz function

1250 00000804 80F401 xor ah, 1

1251 00000807 8825[B6580100] mov [RTC\_2Hz], ah ; 1 = 0.5 second, 0 = 1 second

1252 0000080D 753B jnz short rtc\_int\_return ; half second

1253 ; 1 second

1254 rtc\_int\_0:

1255 ; 22/05/2016

1256 0000080F 58 pop eax ; \*

1257 ;

1258 ; 14/10/2015 ('timer\_int')

1259 ; Here, we are simulating system call entry (for task switch)

1260 ; (If multitasking is enabled,

1261 ; 'clock' procedure may jump to 'sysrelease')

1262 ;push ds ; \*\* ; 23/05/2016

1263 00000810 06 push es

1264 00000811 0FA0 push fs

1265 00000813 0FA8 push gs

1266 00000815 60 pushad ; eax, ecx, edx, ebx, esp -before pushad-, ebp, esi, edi

1267 00000816 66B91000 mov cx, KDATA

1268 ;mov ds, cx ; 06/06/2016

1269 0000081A 8EC1 mov es, cx

1270 0000081C 8EE1 mov fs, cx

1271 0000081E 8EE9 mov gs, cx

1272 ;

1273 00000820 0F20D9 mov ecx, cr3

1274 00000823 890D[5C040300] mov [cr3reg], ecx ; save current cr3 register value/content

1275 ;

1276 00000829 803D[D4030300]00 cmp byte [u.t\_lock], 0 ; timer lock (callback) status ?

1277 00000830 7711 ja short rtc\_int\_1 ; yes

1278

1279 ; 15/01/2017

1280 00000832 3B0D[38580100] cmp ecx, [k\_page\_dir]

1281 00000838 7409 je short rtc\_int\_1

1282

1283 0000083A 8B0D[38580100] mov ecx, [k\_page\_dir]

1284 00000840 0F22D9 mov cr3, ecx

1285 rtc\_int\_1:

1286 ; Timer event (kernel) functions must be performed with

1287 ; 1 second intervals - TRDOS 386 (TRDOS v2.0) feature ! -

1288 ;

1289 ; 25/08/2014

1290 00000843 E81A030000 call rtc\_p ; 19/05/2016 - major modification

1291

1292 ; 23/05/2016

1293 00000848 28E4 sub ah, ah ; 0

1294 ; 22/05/2016 - TRDOS 386 timer event modifications

1295 rtc\_int\_return: ; 19/05/2016

1296 ; 22/02/2015 - dsectpm.s

1297 ; [ source: http://wiki.osdev.org/RTC ]

1298 ; read status register C to complete procedure

1299 ;(it is needed to get a next IRQ 8)

1300 0000084A B00C mov al, 0Ch ;

1301 0000084C E670 out 70h, al ; select register C

1302 0000084E 90 nop

1303 0000084F E471 in al, 71h ; just throw away contents

1304 ; 22/02/2015

1305 00000851 B020 MOV AL,EOI ; END OF INTERRUPT

1306 ;CLI ; DISABLE INTERRUPTS TILL STACK CLEARED

1307 00000853 E6A0 OUT INTB00,AL ; FOR CONTROLLER #2

1308

1309 ; 23/05/2016

1310 00000855 B020 MOV AL,EOI ; GET END OF INTERRUPT MASK

1311 00000857 FA CLI ; DISABLE INTERRUPTS TILL STACK CLEARED

1312 00000858 E620 OUT INTA00,AL ; END OF INTERRUPT TO 8259 - 1

1313 ;

1314 ; 23/05/2016

1315 0000085A 20E4 and ah, ah

1316 0000085C 0F84B7FEFFFF jz rtc\_int\_2

1317

1318 ; ah = 1 (half second)

1319 00000862 58 pop eax ; \*

1320 00000863 1F pop ds ; \*\*

1321 00000864 CF iretd

1322

1323 ; ////////////////

1324

1325 ; 28/08/2014

1326 irq0:

1327 00000865 6A00 push dword 0

1328 00000867 EB48 jmp short which\_irq

1329 irq1:

1330 00000869 6A01 push dword 1

1331 0000086B EB44 jmp short which\_irq

1332 irq2:

1333 0000086D 6A02 push dword 2

1334 0000086F EB40 jmp short which\_irq

1335 irq3:

1336 ; 20/11/2015

1337 ; 24/10/2015

1338 00000871 2EFF15[F5FB0000] call dword [cs:com2\_irq3]

1339 00000878 6A03 push dword 3

1340 0000087A EB35 jmp short which\_irq

1341 irq4:

1342 ; 20/11/2015

1343 ; 24/10/2015

1344 0000087C 2EFF15[F1FB0000] call dword [cs:com1\_irq4]

1345 00000883 6A04 push dword 4

1346 00000885 EB2A jmp short which\_irq

1347 irq5:

1348 00000887 6A05 push dword 5

1349 00000889 EB26 jmp short which\_irq

1350 irq6:

1351 0000088B 6A06 push dword 6

1352 0000088D EB22 jmp short which\_irq

1353 irq7:

1354 0000088F 6A07 push dword 7

1355 00000891 EB1E jmp short which\_irq

1356 irq8:

1357 00000893 6A08 push dword 8

1358 00000895 EB1A jmp short which\_irq

1359 irq9:

1360 00000897 6A09 push dword 9

1361 00000899 EB16 jmp short which\_irq

1362 irq10:

1363 0000089B 6A0A push dword 10

1364 0000089D EB12 jmp short which\_irq

1365 irq11:

1366 0000089F 6A0B push dword 11

1367 000008A1 EB0E jmp short which\_irq

1368 irq12:

1369 000008A3 6A0C push dword 12

1370 000008A5 EB0A jmp short which\_irq

1371 irq13:

1372 000008A7 6A0D push dword 13

1373 000008A9 EB06 jmp short which\_irq

1374 irq14:

1375 000008AB 6A0E push dword 14

1376 000008AD EB02 jmp short which\_irq

1377 irq15:

1378 000008AF 6A0F push dword 15

1379 ;jmp short which\_irq

1380

1381 ; 22/01/2017

1382 ; 19/10/2015

1383 ; 29/08/2014

1384 ; 21/08/2014

1385 which\_irq:

1386 000008B1 870424 xchg eax, [esp] ; 28/08/2014

1387 000008B4 53 push ebx

1388 000008B5 56 push esi

1389 000008B6 57 push edi

1390 000008B7 1E push ds

1391 000008B8 06 push es

1392 ;

1393 000008B9 88C3 mov bl, al

1394 ;

1395 000008BB B810000000 mov eax, KDATA

1396 000008C0 8ED8 mov ds, ax

1397 000008C2 8EC0 mov es, ax

1398 ; 19/10/2015

1399 000008C4 FC cld

1400 ; 27/08/2014

1401 000008C5 8105[48160100]A000- add dword [scr\_row], 0A0h

1401 000008CD 0000

1402 ;

1403 000008CF B417 mov ah, 17h ; blue (1) background,

1404 ; light gray (7) forecolor

1405 000008D1 8B3D[48160100] mov edi, [scr\_row]

1406 000008D7 B049 mov al, 'I'

1407 000008D9 66AB stosw

1408 000008DB B052 mov al, 'R'

1409 000008DD 66AB stosw

1410 000008DF B051 mov al, 'Q'

1411 000008E1 66AB stosw

1412 000008E3 B020 mov al, ' '

1413 000008E5 66AB stosw

1414 000008E7 88D8 mov al, bl

1415 000008E9 3C0A cmp al, 10

1416 000008EB 7208 jb short ii1

1417 000008ED B031 mov al, '1'

1418 000008EF 66AB stosw

1419 000008F1 88D8 mov al, bl

1420 000008F3 2C0A sub al, 10

1421 ii1:

1422 000008F5 0430 add al, '0'

1423 000008F7 66AB stosw

1424 000008F9 B020 mov al, ' '

1425 000008FB 66AB stosw

1426 000008FD B021 mov al, '!'

1427 000008FF 66AB stosw

1428 00000901 B020 mov al, ' '

1429 00000903 66AB stosw

1430 ; 23/02/2015

1431 00000905 80FB07 cmp bl, 7 ; check for IRQ 8 to IRQ 15

1432 00000908 7604 jna ii2

1433 ; 22/01/2017

1434 0000090A B020 mov al, 20h ; END OF INTERRUPT COMMAND TO

1435 0000090C E6A0 out 0A0h, al ; the 2nd 8259

1436 ii2:

1437 0000090E B020 mov al, 20h ; END OF INTERRUPT COMMAND TO

1438 00000910 E620 out 20h, al ; the 2nd 8259

1439 00000912 E9CD010000 jmp iiret

1440 ;

1441 ; 22/08/2014

1442 ;mov al, 20h ; END OF INTERRUPT COMMAND TO 8259

1443 ;out 20h, al ; 8259 PORT

1444 ;

1445 ;pop es

1446 ;pop ds

1447 ;pop edi

1448 ;pop esi

1449 ;pop ebx

1450 ;pop eax

1451 ;iret

1452

1453 ; 02/04/2015

1454 ; 25/08/2014

1455 exc0:

1456 00000917 6A00 push dword 0

1457 00000919 E990000000 jmp cpu\_except

1458 exc1:

1459 0000091E 6A01 push dword 1

1460 00000920 E989000000 jmp cpu\_except

1461 exc2:

1462 00000925 6A02 push dword 2

1463 00000927 E982000000 jmp cpu\_except

1464 exc3:

1465 0000092C 6A03 push dword 3

1466 0000092E EB7E jmp cpu\_except

1467 exc4:

1468 00000930 6A04 push dword 4

1469 00000932 EB7A jmp cpu\_except

1470 exc5:

1471 00000934 6A05 push dword 5

1472 00000936 EB76 jmp cpu\_except

1473 exc6:

1474 00000938 6A06 push dword 6

1475 0000093A EB72 jmp cpu\_except

1476 exc7:

1477 0000093C 6A07 push dword 7

1478 0000093E EB6E jmp cpu\_except

1479 exc8:

1480 ; [esp] = Error code

1481 00000940 6A08 push dword 8

1482 00000942 EB5C jmp cpu\_except\_en

1483 exc9:

1484 00000944 6A09 push dword 9

1485 00000946 EB66 jmp cpu\_except

1486 exc10:

1487 ; [esp] = Error code

1488 00000948 6A0A push dword 10

1489 0000094A EB54 jmp cpu\_except\_en

1490 exc11:

1491 ; [esp] = Error code

1492 0000094C 6A0B push dword 11

1493 0000094E EB50 jmp cpu\_except\_en

1494 exc12:

1495 ; [esp] = Error code

1496 00000950 6A0C push dword 12

1497 00000952 EB4C jmp cpu\_except\_en

1498 exc13:

1499 ; [esp] = Error code

1500 00000954 6A0D push dword 13

1501 00000956 EB48 jmp cpu\_except\_en

1502 exc14:

1503 ; [esp] = Error code

1504 00000958 6A0E push dword 14

1505 0000095A EB44 jmp short cpu\_except\_en

1506 exc15:

1507 0000095C 6A0F push dword 15

1508 0000095E EB4E jmp cpu\_except

1509 exc16:

1510 00000960 6A10 push dword 16

1511 00000962 EB4A jmp cpu\_except

1512 exc17:

1513 ; [esp] = Error code

1514 00000964 6A11 push dword 17

1515 00000966 EB38 jmp short cpu\_except\_en

1516 exc18:

1517 00000968 6A12 push dword 18

1518 0000096A EB42 jmp short cpu\_except

1519 exc19:

1520 0000096C 6A13 push dword 19

1521 0000096E EB3E jmp short cpu\_except

1522 exc20:

1523 00000970 6A14 push dword 20

1524 00000972 EB3A jmp short cpu\_except

1525 exc21:

1526 00000974 6A15 push dword 21

1527 00000976 EB36 jmp short cpu\_except

1528 exc22:

1529 00000978 6A16 push dword 22

1530 0000097A EB32 jmp short cpu\_except

1531 exc23:

1532 0000097C 6A17 push dword 23

1533 0000097E EB2E jmp short cpu\_except

1534 exc24:

1535 00000980 6A18 push dword 24

1536 00000982 EB2A jmp short cpu\_except

1537 exc25:

1538 00000984 6A19 push dword 25

1539 00000986 EB26 jmp short cpu\_except

1540 exc26:

1541 00000988 6A1A push dword 26

1542 0000098A EB22 jmp short cpu\_except

1543 exc27:

1544 0000098C 6A1B push dword 27

1545 0000098E EB1E jmp short cpu\_except

1546 exc28:

1547 00000990 6A1C push dword 28

1548 00000992 EB1A jmp short cpu\_except

1549 exc29:

1550 00000994 6A1D push dword 29

1551 00000996 EB16 jmp short cpu\_except

1552 exc30:

1553 00000998 6A1E push dword 30

1554 0000099A EB04 jmp short cpu\_except\_en

1555 exc31:

1556 0000099C 6A1F push dword 31

1557 0000099E EB0E jmp short cpu\_except

1558

1559 ; 19/10/2015

1560 ; 19/09/2015

1561 ; 01/09/2015

1562 ; 28/08/2015

1563 ; 28/08/2014

1564 cpu\_except\_en:

1565 000009A0 87442404 xchg eax, [esp+4] ; Error code

1566 000009A4 36A3[78050300] mov [ss:error\_code], eax

1567 000009AA 58 pop eax ; Exception number

1568 000009AB 870424 xchg eax, [esp]

1569 ; eax = eax before exception

1570 ; [esp] -> exception number

1571 ; [esp+4] -> EIP to return

1572 ; 22/01/2017

1573 ; 19/10/2015

1574 ; 19/09/2015

1575 ; 01/09/2015

1576 ; 28/08/2015

1577 ; 29/08/2014

1578 ; 28/08/2014

1579 ; 25/08/2014

1580 ; 21/08/2014

1581 cpu\_except: ; CPU Exceptions

1582 000009AE FC cld

1583 000009AF 870424 xchg eax, [esp]

1584 ; eax = Exception number

1585 ; [esp] = eax (before exception)

1586 000009B2 53 push ebx

1587 000009B3 56 push esi

1588 000009B4 57 push edi

1589 000009B5 1E push ds

1590 000009B6 06 push es

1591 ; 28/08/2015

1592 000009B7 66BB1000 mov bx, KDATA

1593 000009BB 8EDB mov ds, bx

1594 000009BD 8EC3 mov es, bx

1595 000009BF 0F20DB mov ebx, cr3

1596 000009C2 53 push ebx ; (\*) page directory

1597 ; 19/10/2015

1598 000009C3 FC cld

1599 ; 25/03/2015

1600 000009C4 8B1D[38580100] mov ebx, [k\_page\_dir]

1601 000009CA 0F22DB mov cr3, ebx

1602 ; 28/08/2015

1603 000009CD 83F80E cmp eax, 0Eh ; 14, PAGE FAULT

1604 000009D0 750F jne short cpu\_except\_nfp

1605 000009D2 E87B440000 call page\_fault\_handler

1606 000009D7 21C0 and eax, eax

1607 000009D9 0F8401010000 jz iiretp ; 01/09/2015

1608 000009DF B00E mov al, 0Eh ; 14

1609 cpu\_except\_nfp:

1610 ; 23/08/2016

1611 000009E1 803D[C25E0000]03 cmp byte [CRT\_MODE], 3

1612 000009E8 7409 je short cpu\_except\_mode\_3

1613 000009EA 50 push eax

1614 000009EB B003 mov al, 3

1615 000009ED E8730B0000 call \_set\_mode

1616 000009F2 58 pop eax

1617 cpu\_except\_mode\_3:

1618 ; 02/04/2015

1619 000009F3 BB[42060000] mov ebx, hang

1620 000009F8 875C241C xchg ebx, [esp+28]

1621 ; EIP (points to instruction which faults)

1622 ; New EIP (hang)

1623 000009FC 891D[7C050300] mov [FaultOffset], ebx

1624 00000A02 C744242008000000 mov dword [esp+32], KCODE ; kernel's code segment

1625 00000A0A 814C242400020000 or dword [esp+36], 200h ; enable interrupts (set IF)

1626 ;

1627 00000A12 88C4 mov ah, al

1628 00000A14 240F and al, 0Fh

1629 00000A16 3C09 cmp al, 9

1630 00000A18 7602 jna short h1ok

1631 00000A1A 0407 add al, 'A'-':'

1632 h1ok:

1633 00000A1C C0EC04 shr ah, 4

1634 00000A1F 80FC09 cmp ah, 9

1635 00000A22 7603 jna short h2ok

1636 00000A24 80C407 add ah, 'A'-':'

1637 h2ok:

1638 00000A27 86E0 xchg ah, al

1639 00000A29 66053030 add ax, '00'

1640 00000A2D 66A3[A0180100] mov [excnstr], ax

1641 ;

1642 ; 29/08/2014

1643 00000A33 A1[7C050300] mov eax, [FaultOffset]

1644 00000A38 51 push ecx

1645 00000A39 52 push edx

1646 00000A3A 89E3 mov ebx, esp

1647 ; 28/08/2015

1648 00000A3C B910000000 mov ecx, 16 ; divisor value to convert binary number

1649 ; to hexadecimal string

1650 ;mov ecx, 10 ; divisor to convert

1651 ; binary number to decimal string

1652 b2d1:

1653 00000A41 31D2 xor edx, edx

1654 00000A43 F7F1 div ecx

1655 00000A45 6652 push dx

1656 00000A47 39C8 cmp eax, ecx

1657 00000A49 73F6 jnb short b2d1

1658 00000A4B BF[AB180100] mov edi, EIPstr ; EIP value

1659 ; points to instruction which faults

1660 ; 28/08/2015

1661 00000A50 89C2 mov edx, eax

1662 b2d2:

1663 ;add al, '0'

1664 00000A52 8A82[1B330000] mov al, [edx+hexchrs]

1665 00000A58 AA stosb ; write hexadecimal digit to its place

1666 00000A59 39E3 cmp ebx, esp

1667 00000A5B 7606 jna short b2d3

1668 00000A5D 6658 pop ax

1669 00000A5F 88C2 mov dl, al

1670 00000A61 EBEF jmp short b2d2

1671 b2d3:

1672 00000A63 B068 mov al, 'h' ; 28/08/2015

1673 00000A65 AA stosb

1674 00000A66 B020 mov al, 20h ; space

1675 00000A68 AA stosb

1676 00000A69 30C0 xor al, al ; to do it an ASCIIZ string

1677 00000A6B AA stosb

1678 ;

1679 00000A6C 5A pop edx

1680 00000A6D 59 pop ecx

1681 ;

1682 00000A6E B44F mov ah, 4Fh ; red (4) background,

1683 ; white (F) forecolor

1684 00000A70 BE[90180100] mov esi, exc\_msg ; message offset

1685 ;

1686 ; 20/01/2017 (!cpu exception!)

1687 ;

1688 00000A75 8105[48160100]A000- add dword [scr\_row], 0A0h

1688 00000A7D 0000

1689 00000A7F 8B3D[48160100] mov edi, [scr\_row]

1690 ;

1691 00000A85 C605[5B030300]00 mov byte [sysflg], 0 ; system mode

1692 00000A8C FB sti

1693 ;

1694 00000A8D E8EFFBFFFF call printk

1695 ;

1696 00000A92 B410 mov ah, 10h

1697 00000A94 E87D010000 call int16h ; getc

1698 ;

1699 00000A99 B003 mov al, 3

1700 00000A9B E8C50A0000 call \_set\_mode

1701 ;

1702 00000AA0 B801000000 mov eax, 1

1703 00000AA5 E9BBBD0000 jmp sysexit ; terminate process !!!

1704

1705 ; 22/01/2017

1706 ; 18/04/2016

1707 ; 28/08/2015

1708 ; 23/02/2015

1709 ; 20/08/2014

1710 ignore\_int:

1711 00000AAA 50 push eax

1712 00000AAB 53 push ebx ; 23/02/2015

1713 00000AAC 56 push esi

1714 00000AAD 57 push edi

1715 00000AAE 1E push ds

1716 00000AAF 06 push es

1717 ; 18/04/2016

1718 00000AB0 66B81000 mov ax, KDATA

1719 00000AB4 8ED8 mov ds, ax

1720 00000AB6 8EC0 mov es, ax

1721 ; 28/08/2015

1722 00000AB8 0F20D8 mov eax, cr3

1723 00000ABB 50 push eax ; (\*) page directory

1724 ;

1725 00000ABC B467 mov ah, 67h ; brown (6) background,

1726 ; light gray (7) forecolor

1727 00000ABE BE[58170100] mov esi, int\_msg ; message offset

1728 piemsg:

1729 ; 27/08/2014

1730 00000AC3 8105[48160100]A000- add dword [scr\_row], 0A0h

1730 00000ACB 0000

1731 00000ACD 8B3D[48160100] mov edi, [scr\_row]

1732 ;

1733 00000AD3 E8A9FBFFFF call printk

1734 ;

1735 ; 23/02/2015

1736 00000AD8 B020 mov al, 20h ; END OF INTERRUPT COMMAND TO

1737 00000ADA E6A0 out 0A0h, al ; the 2nd 8259

1738 ; 22/08/2014

1739 00000ADC B020 mov al, 20h ; END OF INTERRUPT COMMAND TO 8259

1740 00000ADE E620 out 20h, al ; 8259 PORT

1741 iiretp:

1742 ; 22/01/2017

1743 ; 01/09/2015

1744 ; 28/08/2015

1745 00000AE0 58 pop eax ; (\*) page directory

1746 00000AE1 0F22D8 mov cr3, eax

1747 iiret:

1748 00000AE4 07 pop es

1749 00000AE5 1F pop ds

1750 00000AE6 5F pop edi

1751 00000AE7 5E pop esi

1752 00000AE8 5B pop ebx ; 29/08/2014

1753 00000AE9 58 pop eax

1754 00000AEA CF iretd

1755

1756 ; 23/05/2016

1757 ; 22/08/2014

1758 ; IBM PC/AT BIOS source code ----- 10/06/85 (bios.asm)

1759 ; (INT 1Ah)

1760 ;; Linux (v0.12) source code (main.c) by Linus Torvalds (1991)

1761 time\_of\_day:

1762 00000AEB E8EE500000 call UPD\_IPR ; WAIT TILL UPDATE NOT IN PROGRESS

1763 00000AF0 726F jc short time\_of\_day\_retn ; 23/05/2016

1764 00000AF2 B000 mov al, CMOS\_SECONDS

1765 00000AF4 E800510000 call CMOS\_READ

1766 00000AF9 A2[A8580100] mov [time\_seconds], al

1767 00000AFE B002 mov al, CMOS\_MINUTES

1768 00000B00 E8F4500000 call CMOS\_READ

1769 00000B05 A2[A9580100] mov [time\_minutes], al

1770 00000B0A B004 mov al, CMOS\_HOURS

1771 00000B0C E8E8500000 call CMOS\_READ

1772 00000B11 A2[AA580100] mov [time\_hours], al

1773 00000B16 B006 mov al, CMOS\_DAY\_WEEK

1774 00000B18 E8DC500000 call CMOS\_READ

1775 00000B1D A2[AB580100] mov [date\_wday], al

1776 00000B22 B007 mov al, CMOS\_DAY\_MONTH

1777 00000B24 E8D0500000 call CMOS\_READ

1778 00000B29 A2[AC580100] mov [date\_day], al

1779 00000B2E B008 mov al, CMOS\_MONTH

1780 00000B30 E8C4500000 call CMOS\_READ

1781 00000B35 A2[AD580100] mov [date\_month], al

1782 00000B3A B009 mov al, CMOS\_YEAR

1783 00000B3C E8B8500000 call CMOS\_READ

1784 00000B41 A2[AE580100] mov [date\_year], al

1785 00000B46 B032 mov al, CMOS\_CENTURY

1786 00000B48 E8AC500000 call CMOS\_READ

1787 00000B4D A2[AF580100] mov [date\_century], al

1788 ;

1789 00000B52 B000 mov al, CMOS\_SECONDS

1790 00000B54 E8A0500000 call CMOS\_READ

1791 00000B59 3A05[A8580100] cmp al, [time\_seconds]

1792 00000B5F 758A jne short time\_of\_day

1793

1794 time\_of\_day\_retn:

1795 00000B61 C3 retn

1796

1797 ; 15/01/2017

1798 ; 10/06/2016

1799 ; 07/06/2016

1800 ; 06/06/2016

1801 ; 23/05/2016

1802 rtc\_p:

1803 00000B62 B101 mov cl, 1 ; 15/01/2017

1804 00000B64 EB02 jmp short rtc\_p0

1805 u\_timer:

1806 ; Timer Events with 18.2 Hz Timer Ticks

1807 ; (and also timer events with RTC seconds)

1808 00000B66 28C9 sub cl, cl ; mov cl, 0 ; 15/01/2017

1809 rtc\_p0:

1810 ; 19/05/2016 - TRDOS 386 (TRDOS v2.0)

1811 ; Major Modification:

1812 ; Check and Perform Timer Events (for RTC)

1813 ; 25/08/2014 - 07/09/2014

1814 ; Retro UNIX 386 v1:

1815 ; Print Real Time Clock content

1816

1817 ; 15/01/2017

1818 00000B68 880D[CC650100] mov byte [priority], cl ; 0 or 1 (not 2)

1819 00000B6E 8A2D[CF650100] mov ch, [timer\_events]

1820 00000B74 20ED and ch, ch

1821 00000B76 7420 jz short rtc\_p3

1822

1823 00000B78 BE[60040300] mov esi, timer\_set ; beginning address of

1824 ; timer events space

1825 rtc\_p1:

1826 00000B7D 8B06 mov eax, [esi]

1827 00000B7F 20C0 and al, al ; 0 = free, >0 = process no.

1828 00000B81 7416 jz short rtc\_p4

1829 ;

1830 00000B83 C1C810 ror eax, 16

1831 ; ah = response value, al = interrupt type

1832 ; 15/01/2017

1833 ; cl = interrupt source

1834 ; 1 = RTC, 0 = PIT

1835 00000B86 38C8 cmp al, cl

1836 00000B88 750A jne short rtc\_p2 ; not as requested or undefined !

1837 00000B8A 3C01 cmp al, 1 ; 1 ; RTC interrupt ?

1838 00000B8C 7410 je short rtc\_p5 ; yes, check for response

1839 ; 06/06/2016 - 18.2 Hz Timer Ticks

1840 00000B8E 836E080A sub dword [esi+8], 10 ; 1 tick = 10

1841 00000B92 7613 jna short rtc\_p6 ; continue for responding

1842 rtc\_p2:

1843 ; 15/01/2017 (cl -> ch)

1844 ; 07/06/2016

1845 00000B94 FECD dec ch ; remain count of timer events

1846 00000B96 7501 jnz short rtc\_p4

1847 rtc\_p3:

1848 00000B98 C3 retn

1849 rtc\_p4:

1850 ;cmp esi, timer\_set + 240 ; 15\*16 (last event)

1851 ;jnb short rtc\_p3 ; end of timer event space

1852 00000B99 83C610 add esi, 16 ; next timer event

1853 00000B9C EBDF jmp short rtc\_p1

1854 rtc\_p5:

1855 ; current timer count ; 06/06/2016 (182)

1856 00000B9E 816E08B6000000 sub dword [esi+8], 182 ; 1 second (10\*18.2)

1857 00000BA5 77ED ja short rtc\_p2 ; check for the next

1858 rtc\_p6:

1859 ; it is the time of response!

1860 00000BA7 8B5E04 mov ebx, [esi+4] ; set (count limit) value

1861 00000BAA 895E08 mov [esi+8], ebx ; reset count down value

1862 ; to count limit

1863 ; 19/12/2016

1864 ; 10/12/2016 - timer callback modification

1865 00000BAD 8B7E0C mov edi, [esi+12] ; response (or callback) address

1866 00000BB0 807E0100 cmp byte [esi+1], 0 ; >0 = callback

1867 00000BB4 762A jna short rtc\_p8

1868

1869 ; timer callback !

1870 00000BB6 0FB61E movzx ebx, byte [esi] ; process number (>0)

1871 00000BB9 89D8 mov eax, ebx

1872 00000BBB C0E302 shl bl, 2 ; \*4

1873 00000BBE 89BB[0C010300] mov [ebx+p.tcb-4], edi ; user's callback service addr

1874 00000BC4 3A05[B3030300] cmp al, [u.uno]

1875 00000BCA 7521 jne short rtc\_p9

1876 00000BCC 893D[D0030300] mov [u.tcb], edi

1877 rtc\_p7:

1878 ; 15/01/2017

1879 00000BD2 B002 mov al, 2

1880 00000BD4 A2[CC650100] mov [priority], al ; 2

1881 ; 10/01/2017

1882 ;mov byte [u.pri], 2

1883 00000BD9 A2[A9030300] mov [u.pri], al ; 2

1884 00000BDE EBB4 jmp short rtc\_p2

1885 rtc\_p8:

1886 ; response address is physical address of

1887 ; the program's response (signal return) byte

1888 ; 06/06/2016

1889 ;mov edi, [esi+12] ; response address

1890 00000BE0 8827 mov [edi], ah ; response value

1891 ;

1892 00000BE2 C1C010 rol eax, 16

1893 ; 15/01/2017

1894 00000BE5 3A05[B3030300] cmp al, [u.uno] ; running process ?

1895 00000BEB 74E5 je short rtc\_p7

1896 rtc\_p9:

1897 ; al = process number ; 10/06/2016

1898 00000BED B202 mov dl, 2 ; priority, 2 = event (high)

1899 00000BEF E8F7EC0000 call set\_run\_sequence ; 19/05/2016

1900 00000BF4 EB9E jmp short rtc\_p2 ; 10/06/2016

1901

1902

1903 ; Default IRQ 7 handler against spurious IRQs (from master PIC)

1904 ; 25/02/2015 (source: http://wiki.osdev.org/8259\_PIC)

1905 default\_irq7:

1906 00000BF6 6650 push ax

1907 00000BF8 B00B mov al, 0Bh ; In-Service register

1908 00000BFA E620 out 20h, al

1909 00000BFC EB00 jmp short $+2

1910 00000BFE EB00 jmp short $+2

1911 00000C00 E420 in al, 20h

1912 00000C02 2480 and al, 80h ; bit 7 (is it real IRQ 7 or fake?)

1913 00000C04 7404 jz short irq7\_iret ; Fake (spurious) IRQ, do not send EOI

1914 00000C06 B020 mov al, 20h ; EOI

1915 00000C08 E620 out 20h, al

1916 irq7\_iret:

1917 00000C0A 6658 pop ax

1918 00000C0C CF iretd

1919

1920 bcd\_to\_ascii:

1921 ; 25/08/2014

1922 ; INPUT ->

1923 ; al = Packed BCD number

1924 ; OUTPUT ->

1925 ; ax = ASCII word/number

1926 ;

1927 ; Erdogan Tan - 1998 (proc\_hex) - TRDOS.ASM (2004-2011)

1928 ;

1929 00000C0D D410 db 0D4h,10h ; Undocumented inst. AAM

1930 ; AH = AL / 10h

1931 ; AL = AL MOD 10h

1932 00000C0F 660D3030 or ax,'00' ; Make it ASCII based

1933

1934 00000C13 86E0 xchg ah, al

1935

1936 00000C15 C3 retn

1937

1938

1939 %include 'keyboard.s' ; 07/03/2015

1 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2 <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0 - keyboard.s

3 <1> ; ----------------------------------------------------------------------------

4 <1> ; Last Update: 15/01/2017

5 <1> ; ----------------------------------------------------------------------------

6 <1> ; Beginning: 17/01/2016

7 <1> ; ----------------------------------------------------------------------------

8 <1> ; Assembler: NASM version 2.11 (trdos386.s)

9 <1> ; ----------------------------------------------------------------------------

10 <1> ; Turkish Rational DOS

11 <1> ; Operating System Project v2.0 by ERDOGAN TAN (Beginning: 04/01/2016)

12 <1> ;

13 <1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan

14 <1> ; keyboard.inc (17/10/2015)

15 <1> ;

16 <1> ; Derived from 'IBM PC-XT-286' BIOS source code (1986)

17 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

18 <1>

19 <1> ; Retro UNIX 386 v1 Kernel - KEYBOARD.INC

20 <1> ; Last Modification: 17/10/2015

21 <1> ; (Keyboard Data is in 'KYBDATA.INC')

22 <1> ;

23 <1> ; ///////// KEYBOARD FUNCTIONS (PROCEDURES) ///////////////

24 <1>

25 <1> ; 17/01/2016 (TRDOS 386 = TRDOS v2.0)

26 <1>

27 <1> ; 03/12/2014

28 <1> ; 26/08/2014

29 <1> ; KEYBOARD I/O

30 <1> ; (INT\_16h - Retro UNIX 8086 v1 - U9.ASM, 30/06/2014)

31 <1>

32 <1> ;NOTE: 'k0' to 'k7' are name of OPMASK registers.

33 <1> ; (The reason of using '\_k' labels!!!) (27/08/2014)

34 <1> ;NOTE: 'NOT' keyword is '~' unary operator in NASM.

35 <1> ; ('NOT LC\_HC' --> '~LC\_HC') (bit reversing operator)

36 <1>

37 <1> int16h: ; 30/06/2015

38 <1> ;getc:

39 00000C16 9C <1> pushfd ; 28/08/2014

40 00000C17 0E <1> push cs

41 00000C18 E801000000 <1> call KEYBOARD\_IO\_1 ; getc\_int

42 00000C1D C3 <1> retn

43 <1>

44 <1> getc\_int:

45 <1> ; 28/02/2015

46 <1> ; 03/12/2014 (derivation from pc-xt-286 bios source code -1986-,

47 <1> ; instead of pc-at bios - 1985-)

48 <1> ; 28/08/2014 (\_k1d)

49 <1> ; 30/06/2014

50 <1> ; 03/03/2014

51 <1> ; 28/02/2014

52 <1> ; Derived from "KEYBOARD\_IO\_1" procedure of IBM "pc-xt-286"

53 <1> ; rombios source code (21/04/1986)

54 <1> ; 'keybd.asm', INT 16H, KEYBOARD\_IO

55 <1> ;

56 <1> ; KYBD --- 03/06/86 KEYBOARD BIOS

57 <1> ;

58 <1> ;--- INT 16 H -----------------------------------------------------------------

59 <1> ; KEYBOARD I/O :

60 <1> ; THESE ROUTINES PROVIDE READ KEYBOARD SUPPORT :

61 <1> ; INPUT :

62 <1> ; (AH)= 00H READ THE NEXT ASCII CHARACTER ENTERED FROM THE KEYBOARD, :

63 <1> ; RETURN THE RESULT IN (AL), SCAN CODE IN (AH). :

64 <1> ; THIS IS THE COMPATIBLE READ INTERFACE, EQUIVALENT TO THE :

65 <1> ; STANDARD PC OR PCAT KEYBOARD :

66 <1> ;-----------------------------------------------------------------------------:

67 <1> ; (AH)= 01H SET THE ZERO FLAG TO INDICATE IF AN ASCII CHARACTER IS :

68 <1> ; AVAILABLE TO BE READ FROM THE KEYBOARD BUFFER. :

69 <1> ; (ZF)= 1 -- NO CODE AVAILABLE :

70 <1> ; (ZF)= 0 -- CODE IS AVAILABLE (AX)= CHARACTER :

71 <1> ; IF (ZF)= 0, THE NEXT CHARACTER IN THE BUFFER TO BE READ IS :

72 <1> ; IN (AX), AND THE ENTRY REMAINS IN THE BUFFER. :

73 <1> ; THIS WILL RETURN ONLY PC/PCAT KEYBOARD COMPATIBLE CODES :

74 <1> ;-----------------------------------------------------------------------------:

75 <1> ; (AH)= 02H RETURN THE CURRENT SHIFT STATUS IN AL REGISTER :

76 <1> ; THE BIT SETTINGS FOR THIS CODE ARE INDICATED IN THE :

77 <1> ; EQUATES FOR @KB\_FLAG :

78 <1> ;-----------------------------------------------------------------------------:

79 <1> ; (AH)= 03H SET TYPAMATIC RATE AND DELAY :

80 <1> ; (AL) = 05H :

81 <1> ; (BL) = TYPAMATIC RATE (BITS 5 - 7 MUST BE RESET TO 0) :

82 <1> ; :

83 <1> ; REGISTER RATE REGISTER RATE :

84 <1> ; VALUE SELECTED VALUE SELECTED :

85 <1> ; -------------------------------------------- :

86 <1> ; 00H 30.0 10H 7.5 :

87 <1> ; 01H 26.7 11H 6.7 :

88 <1> ; 02H 24.0 12H 6.0 :

89 <1> ; 03H 21.8 13H 5.5 :

90 <1> ; 04H 20.0 14H 5.0 :

91 <1> ; 05H 18.5 15H 4.6 :

92 <1> ; 06H 17.1 16H 4.3 :

93 <1> ; 07H 16.0 17H 4.0 :

94 <1> ; 08H 15.0 18H 3.7 :

95 <1> ; 09H 13.3 19H 3.3 :

96 <1> ; 0AH 12.0 1AH 3.0 :

97 <1> ; 0BH 10.9 1BH 2.7 :

98 <1> ; 0CH 10.0 1CH 2.5 :

99 <1> ; 0DH 9.2 1DH 2.3 :

100 <1> ; 0EH 8.6 1EH 2.1 :

101 <1> ; 0FH 8.0 1FH 2.0 :

102 <1> ; :

103 <1> ; (BH) = TYPAMATIC DELAY (BITS 2 - 7 MUST BE RESET TO 0) :

104 <1> ; :

105 <1> ; REGISTER DELAY :

106 <1> ; VALUE VALUE :

107 <1> ; ------------------ :

108 <1> ; 00H 250 ms :

109 <1> ; 01H 500 ms :

110 <1> ; 02H 750 ms :

111 <1> ; 03H 1000 ms :

112 <1> ;-----------------------------------------------------------------------------:

113 <1> ; (AH)= 05H PLACE ASCII CHARACTER/SCAN CODE COMBINATION IN KEYBOARD :

114 <1> ; BUFFER AS IF STRUCK FROM KEYBOARD :

115 <1> ; ENTRY: (CL) = ASCII CHARACTER :

116 <1> ; (CH) = SCAN CODE :

117 <1> ; EXIT: (AH) = 00H = SUCCESSFUL OPERATION :

118 <1> ; (AL) = 01H = UNSUCCESSFUL - BUFFER FULL :

119 <1> ; FLAGS: CARRY IF ERROR :

120 <1> ;-----------------------------------------------------------------------------:

121 <1> ; (AH)= 10H EXTENDED READ INTERFACE FOR THE ENHANCED KEYBOARD, :

122 <1> ; OTHERWISE SAME AS FUNCTION AH=0 :

123 <1> ;-----------------------------------------------------------------------------:

124 <1> ; (AH)= 11H EXTENDED ASCII STATUS FOR THE ENHANCED KEYBOARD, :

125 <1> ; OTHERWISE SAME AS FUNCTION AH=1 :

126 <1> ;-----------------------------------------------------------------------------:

127 <1> ; (AH)= 12H RETURN THE EXTENDED SHIFT STATUS IN AX REGISTER :

128 <1> ; AL = BITS FROM KB\_FLAG, AH = BITS FOR LEFT AND RIGHT :

129 <1> ; CTL AND ALT KEYS FROM KB\_FLAG\_1 AND KB\_FLAG\_3 :

130 <1> ; OUTPUT :

131 <1> ; AS NOTED ABOVE, ONLY (AX) AND FLAGS CHANGED :

132 <1> ; ALL REGISTERS RETAINED :

133 <1> ;------------------------------------------------------------------------------

134 <1>

135 <1> ; 15/01/2017

136 <1> ; 14/01/2017

137 <1> ; 02/01/2017

138 <1> ; 29/05/2016

139 <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)

140 <1> int32h: ; Keyboard BIOS

141 <1>

142 <1> KEYBOARD\_IO\_1:

143 <1> ;sti ; INTERRUPTS BACK ON

144 <1> ; 29/05/2016

145 00000C1E 80642408BE <1> and byte [esp+8], 10111110b ; clear zero flag and cary flag

146 <1> ;

147 00000C23 1E <1> push ds ; SAVE CURRENT DS

148 00000C24 53 <1> push ebx ; SAVE BX TEMPORARILY

149 <1> ;push ecx ; SAVE CX TEMPORARILY

150 00000C25 66BB1000 <1> mov bx, KDATA

151 00000C29 8EDB <1> mov ds, bx ; PUT SEGMENT VALUE OF DATA AREA INTO DS

152 <1>

153 <1> ; 14/01/2017

154 00000C2B 8B1C24 <1> mov ebx, [esp]

155 <1> ;; 15/01/2017

156 <1> ; 02/01/2017

157 <1> ;;mov byte [intflg], 32h ; keyboard interrupt

158 00000C2E FB <1> sti

159 <1> ;

160 <1>

161 00000C2F 08E4 <1> or ah, ah ; CHECK FOR (AH)= 00H

162 00000C31 743A <1> jz short \_K1 ; ASCII\_READ

163 00000C33 FECC <1> dec ah ; CHECK FOR (AH)= 01H

164 00000C35 7453 <1> jz short \_K2 ; ASCII\_STATUS

165 00000C37 FECC <1> dec ah ; CHECK FOR (AH)= 02H

166 00000C39 0F8494000000 <1> jz \_K3 ; SHIFT STATUS

167 00000C3F FECC <1> dec ah ; CHECK FOR (AH)= 03H

168 00000C41 0F8493000000 <1> jz \_K300 ; SET TYPAMATIC RATE/DELAY

169 00000C47 80EC02 <1> sub ah, 2 ; CHECK FOR (AH)= 05H

170 00000C4A 0F84BC000000 <1> jz \_K500 ; KEYBOARD WRITE

171 <1> \_KIO1:

172 00000C50 80EC0B <1> sub ah, 11 ; AH = 10H

173 00000C53 740C <1> jz short \_K1E ; EXTENDED ASCII READ

174 00000C55 FECC <1> dec ah ; CHECK FOR (AH)= 11H

175 00000C57 7422 <1> jz short \_K2E ; EXTENDED\_ASCII\_STATUS

176 00000C59 FECC <1> dec ah ; CHECK FOR (AH)= 12H

177 00000C5B 7458 <1> jz short \_K3E ; EXTENDED\_SHIFT\_STATUS

178 <1> \_KIO\_EXIT:

179 <1> ; 02/01/2017

180 00000C5D FA <1> cli

181 <1> ;;mov byte [intflg], 0 ;; 15/01/2017

182 <1> ;

183 <1> ;pop ecx ; RECOVER REGISTER

184 00000C5E 5B <1> pop ebx ; RECOVER REGISTER

185 00000C5F 1F <1> pop ds ; RECOVER SEGMENT

186 00000C60 CF <1> iretd ; INVALID COMMAND, EXIT

187 <1>

188 <1> ;----- ASCII CHARACTER

189 <1> \_K1E:

190 00000C61 E8D3000000 <1> call \_K1S ; GET A CHARACTER FROM THE BUFFER (EXTENDED)

191 00000C66 E848010000 <1> call \_KIO\_E\_XLAT ; ROUTINE TO XLATE FOR EXTENDED CALLS

192 00000C6B EBF0 <1> jmp short \_KIO\_EXIT ; GIVE IT TO THE CALLER

193 <1> \_K1:

194 00000C6D E8C7000000 <1> call \_K1S ; GET A CHARACTER FROM THE BUFFER

195 00000C72 E847010000 <1> call \_KIO\_S\_XLAT ; ROUTINE TO XLATE FOR STANDARD CALLS

196 00000C77 72F4 <1> jc short \_K1 ; CARRY SET MEANS TROW CODE AWAY

197 <1> \_K1A:

198 00000C79 EBE2 <1> jmp short \_KIO\_EXIT ; RETURN TO CALLER

199 <1>

200 <1> ;----- ASCII STATUS

201 <1> \_K2E:

202 00000C7B E804010000 <1> call \_K2S ; TEST FOR CHARACTER IN BUFFER (EXTENDED)

203 00000C80 7420 <1> jz short \_K2B ; RETURN IF BUFFER EMPTY

204 00000C82 9C <1> pushf ; SAVE ZF FROM TEST

205 00000C83 E82B010000 <1> call \_KIO\_E\_XLAT ; ROUTINE TO XLATE FOR EXTENDED CALLS

206 00000C88 EB17 <1> jmp short \_K2A ; GIVE IT TO THE CALLER

207 <1> \_K2:

208 00000C8A E8F5000000 <1> call \_K2S ; TEST FOR CHARACTER IN BUFFER

209 00000C8F 7411 <1> jz short \_K2B ; RETURN IF BUFFER EMPTY

210 00000C91 9C <1> pushf ; SAVE ZF FROM TEST

211 00000C92 E827010000 <1> call \_KIO\_S\_XLAT ; ROUTINE TO XLATE FOR STANDARD CALLS

212 00000C97 7308 <1> jnc short \_K2A ; CARRY CLEAR MEANS PASS VALID CODE

213 00000C99 9D <1> popf ; INVALID CODE FOR THIS TYPE OF CALL

214 00000C9A E89A000000 <1> call \_K1S ; THROW THE CHARACTER AWAY

215 00000C9F EBE9 <1> jmp short \_K2 ; GO LOOK FOR NEXT CHAR, IF ANY

216 <1> \_K2A:

217 00000CA1 9D <1> popf ; RESTORE ZF FROM TEST

218 <1> \_K2B:

219 <1> ; 02/01/2017

220 00000CA2 FA <1> cli

221 <1> ;; mov byte [intflg], 0 ;; 15/01/2017

222 <1> ;

223 <1> ;pop ecx ; RECOVER REGISTER

224 00000CA3 5B <1> pop ebx ; RECOVER REGISTER

225 00000CA4 1F <1> pop ds ; RECOVER SEGMENT

226 <1> ; (\*) 29/05/2016

227 <1> ; (\*) retf 4 ; THROW AWAY (e)FLAGS

228 00000CA5 7208 <1> jc short \_k2d

229 00000CA7 7505 <1> jnz short \_k2c

230 00000CA9 804C240840 <1> or byte [esp+8], 01000000b ; set zero flag bit of eflags register

231 <1> \_k2c:

232 00000CAE CF <1> iretd

233 <1> \_k2d:

234 <1> ; 29/05/2016 -set carry flag on stack-

235 <1> ; [esp] = EIP

236 <1> ; [esp+4] = CS

237 <1> ; [esp+8] = E-FLAGS

238 00000CAF 804C240801 <1> or byte [esp+8], 1 ; set carry bit of eflags register

239 <1> ; [esp+12] = ESP (user)

240 <1> ; [esp+16] = SS (User)

241 00000CB4 CF <1> iretd

242 <1>

243 <1>

244 <1> ; (\*) 29/05/2016 - 'ref 4' intruction causes to stack fault

245 <1> ; (OUTER-PRIVILEGE-LEVEL)

246 <1> ; INTEL 80386 PROGRAMMER'S REFERENCE MANUAL 1986

247 <1> ; // RETF instruction:

248 <1> ;

249 <1> ; IF OperandMode=32 THEN

250 <1> ; Load CS:EIP from stack;

251 <1> ; Set CS RPL to CPL;

252 <1> ; Increment eSP by 8 plus the immediate offset if it exists;

253 <1> ; Load SS:eSP from stack;

254 <1> ; ELSE (\* OperandMode=16 \*)

255 <1> ; Load CS:IP from stack;

256 <1> ; Set CS RPL to CPL;

257 <1> ; Increment eSP by 4 plus the immediate offset if it exists;

258 <1> ; Load SS:eSP from stack;

259 <1> ; FI;

260 <1> ;

261 <1> ; //

262 <1>

263 <1> ;----- SHIFT STATUS

264 <1> \_K3E: ; GET THE EXTENDED SHIFT STATUS FLAGS

265 00000CB5 8A25[8E5E0000] <1> mov ah, [KB\_FLAG\_1] ; GET SYSTEM SHIFT KEY STATUS

266 00000CBB 80E404 <1> and ah, SYS\_SHIFT ; MASK ALL BUT SYS KEY BIT

267 <1> ;mov cl, 5 ; SHIFT THEW SYSTEMKEY BIT OVER TO

268 <1> ;shl ah, cl ; BIT 7 POSITION

269 00000CBE C0E405 <1> shl ah, 5

270 00000CC1 A0[8E5E0000] <1> mov al, [KB\_FLAG\_1] ; GET SYSTEM SHIFT STATES BACK

271 00000CC6 2473 <1> and al, 01110011b ; ELIMINATE SYS SHIFT, HOLD\_STATE AND INS\_SHIFT

272 00000CC8 08C4 <1> or ah, al ; MERGE REMAINING BITS INTO AH

273 00000CCA A0[905E0000] <1> mov al, [KB\_FLAG\_3] ; GET RIGHT CTL AND ALT

274 00000CCF 240C <1> and al, 00001100b ; ELIMINATE LC\_E0 AND LC\_E1

275 00000CD1 08C4 <1> or ah, al ; OR THE SHIFT FLAGS TOGETHER

276 <1> \_K3:

277 00000CD3 A0[8D5E0000] <1> mov al, [KB\_FLAG] ; GET THE SHIFT STATUS FLAGS

278 <1> ;jmp short \_KIO\_EXIT ; RETURN TO CALLER

279 00000CD8 EB83 <1> jmp \_KIO\_EXIT

280 <1>

281 <1> ;----- SET TYPAMATIC RATE AND DELAY

282 <1> \_K300:

283 00000CDA 3C05 <1> cmp al, 5 ; CORRECT FUNCTION CALL?

284 <1> ;jne short \_KIO\_EXIT ; NO, RETURN

285 00000CDC 0F857BFFFFFF <1> jne \_KIO\_EXIT

286 00000CE2 F6C3E0 <1> test bl, 0E0h ; TEST FOR OUT-OF-RANGE RATE

287 00000CE5 0F8572FFFFFF <1> jnz \_KIO\_EXIT ; RETURN IF SO

288 00000CEB F6C7FC <1> test BH, 0FCh ; TEST FOR OUT-OF-RANGE DELAY

289 00000CEE 0F8569FFFFFF <1> jnz \_KIO\_EXIT ; RETURN IF SO

290 00000CF4 B0F3 <1> mov al, KB\_TYPA\_RD ; COMMAND FOR TYPAMATIC RATE/DELAY

291 00000CF6 E8DA060000 <1> call SND\_DATA ; SEND TO KEYBOARD

292 <1> ;mov cx, 5 ; SHIFT COUNT

293 <1> ;shl bh, cl ; SHIFT DELAY OVER

294 00000CFB C0E705 <1> shl bh, 5

295 00000CFE 88D8 <1> mov al, bl ; PUT IN RATE

296 00000D00 08F8 <1> or al, bh ; AND DELAY

297 00000D02 E8CE060000 <1> call SND\_DATA ; SEND TO KEYBOARD

298 00000D07 E951FFFFFF <1> jmp \_KIO\_EXIT ; RETURN TO CALLER

299 <1>

300 <1> ;----- WRITE TO KEYBOARD BUFFER

301 <1> \_K500:

302 00000D0C 56 <1> push esi ; SAVE SI (esi)

303 00000D0D FA <1> cli ;

304 00000D0E 8B1D[9E5E0000] <1> mov ebx, [BUFFER\_TAIL] ; GET THE 'IN TO' POINTER TO THE BUFFER

305 00000D14 89DE <1> mov esi, ebx ; SAVE A COPY IN CASE BUFFER NOT FULL

306 00000D16 E8D3000000 <1> call \_K4 ; BUMP THE POINTER TO SEE IF BUFFER IS FULL

307 00000D1B 3B1D[9A5E0000] <1> cmp ebx, [BUFFER\_HEAD] ; WILL THE BUFFER OVERRUN IF WE STORE THIS?

308 00000D21 740D <1> je short \_K502 ; YES - INFORM CALLER OF ERROR

309 00000D23 66890E <1> mov [esi], cx ; NO - PUT ASCII/SCAN CODE INTO BUFFER

310 00000D26 891D[9E5E0000] <1> mov [BUFFER\_TAIL], ebx ; ADJUST 'IN TO' POINTER TO REFLECT CHANGE

311 00000D2C 28C0 <1> sub al, al ; TELL CALLER THAT OPERATION WAS SUCCESSFUL

312 00000D2E EB02 <1> jmp short \_K504 ; SUB INSTRUCTION ALSO RESETS CARRY FLAG

313 <1> \_K502:

314 00000D30 B001 <1> mov al, 01h ; BUFFER FULL INDICATION

315 <1> \_K504:

316 00000D32 FB <1> sti

317 00000D33 5E <1> pop esi ; RECOVER SI (esi)

318 00000D34 E924FFFFFF <1> jmp \_KIO\_EXIT ; RETURN TO CALLER WITH STATUS IN AL

319 <1>

320 <1> ;----- READ THE KEY TO FIGURE OUT WHAT TO DO -----

321 <1> \_K1S:

322 00000D39 FA <1> cli ; 03/12/2014

323 00000D3A 8B1D[9A5E0000] <1> mov ebx, [BUFFER\_HEAD] ; GET POINTER TO HEAD OF BUFFER

324 00000D40 3B1D[9E5E0000] <1> cmp ebx, [BUFFER\_TAIL] ; TEST END OF BUFFER

325 <1> ;jne short \_K1U ; IF ANYTHING IN BUFFER SKIP INTERRUPT

326 00000D46 750F <1> jne short \_k1x ; 03/12/2014

327 <1> ;

328 <1> ; 03/12/2014

329 <1> ; 28/08/2014

330 <1> ; PERFORM OTHER FUNCTION ?? here !

331 <1> ;; MOV AX, 9002h ; MOVE IN WAIT CODE & TYPE

332 <1> ;; INT 15H ; PERFORM OTHER FUNCTION

333 <1> \_K1T: ; ASCII READ

334 00000D48 FB <1> sti ; INTERRUPTS BACK ON DURING LOOP

335 00000D49 90 <1> nop ; ALLOW AN INTERRUPT TO OCCUR

336 <1> \_K1U:

337 00000D4A FA <1> cli ; INTERRUPTS BACK OFF

338 00000D4B 8B1D[9A5E0000] <1> mov ebx, [BUFFER\_HEAD] ; GET POINTER TO HEAD OF BUFFER

339 00000D51 3B1D[9E5E0000] <1> cmp ebx, [BUFFER\_TAIL] ; TEST END OF BUFFER

340 <1> \_k1x:

341 00000D57 53 <1> push ebx ; SAVE ADDRESS

342 00000D58 9C <1> pushf ; SAVE FLAGS

343 00000D59 E82F070000 <1> call MAKE\_LED ; GO GET MODE INDICATOR DATA BYTE

344 00000D5E 8A1D[8F5E0000] <1> mov bl, [KB\_FLAG\_2] ; GET PREVIOUS BITS

345 00000D64 30C3 <1> xor bl, al ; SEE IF ANY DIFFERENT

346 00000D66 80E307 <1> and bl, 07h ; KB\_LEDS ; ISOLATE INDICATOR BITS

347 00000D69 7406 <1> jz short \_K1V ; IF NO CHANGE BYPASS UPDATE

348 00000D6B E8C9060000 <1> call SND\_LED1

349 00000D70 FA <1> cli ; DISABLE INTERRUPTS

350 <1> \_K1V:

351 00000D71 9D <1> popf ; RESTORE FLAGS

352 00000D72 5B <1> pop ebx ; RESTORE ADDRESS

353 00000D73 74D3 <1> je short \_K1T ; LOOP UNTIL SOMETHING IN BUFFER

354 <1> ;

355 00000D75 668B03 <1> mov ax, [ebx] ; GET SCAN CODE AND ASCII CODE

356 00000D78 E871000000 <1> call \_K4 ; MOVE POINTER TO NEXT POSITION

357 00000D7D 891D[9A5E0000] <1> mov [BUFFER\_HEAD], ebx ; STORE VALUE IN VARIABLE

358 00000D83 C3 <1> retn ; RETURN

359 <1>

360 <1> ;----- READ THE KEY TO SEE IF ONE IS PRESENT -----

361 <1> \_K2S:

362 00000D84 FA <1> cli ; INTERRUPTS OFF

363 00000D85 8B1D[9A5E0000] <1> mov ebx, [BUFFER\_HEAD] ; GET HEAD POINTER

364 00000D8B 3B1D[9E5E0000] <1> cmp ebx, [BUFFER\_TAIL] ; IF EQUAL (Z=1) THEN NOTHING THERE

365 00000D91 668B03 <1> mov ax, [ebx]

366 00000D94 9C <1> pushf ; SAVE FLAGS

367 00000D95 6650 <1> push ax ; SAVE CODE

368 00000D97 E8F1060000 <1> call MAKE\_LED ; GO GET MODE INDICATOR DATA BYTE

369 00000D9C 8A1D[8F5E0000] <1> mov bl, [KB\_FLAG\_2] ; GET PREVIOUS BITS

370 00000DA2 30C3 <1> xor bl, al ; SEE IF ANY DIFFERENT

371 00000DA4 80E307 <1> and bl, 07h ; KB\_LEDS ; ISOLATE INDICATOR BITS

372 00000DA7 7405 <1> jz short \_K2T ; IF NO CHANGE BYPASS UPDATE

373 00000DA9 E874060000 <1> call SND\_LED ; GO TURN ON MODE INDICATORS

374 <1> \_K2T:

375 00000DAE 6658 <1> pop ax ; RESTORE CODE

376 00000DB0 9D <1> popf ; RESTORE FLAGS

377 00000DB1 FB <1> sti ; INTERRUPTS BACK ON

378 00000DB2 C3 <1> retn ; RETURN

379 <1>

380 <1> ;----- ROUTINE TO TRANSLATE SCAN CODE PAIRS FOR EXTENDED CALLS -----

381 <1> \_KIO\_E\_XLAT:

382 00000DB3 3CF0 <1> cmp al, 0F0h ; IS IT ONE OF THE FILL-INs?

383 00000DB5 7506 <1> jne short \_KIO\_E\_RET ; NO, PASS IT ON

384 00000DB7 08E4 <1> or ah, ah ; AH = 0 IS SPECIAL CASE

385 00000DB9 7402 <1> jz short \_KIO\_E\_RET ; PASS THIS ON UNCHANGED

386 00000DBB 30C0 <1> xor al, al ; OTHERWISE SET AL = 0

387 <1> \_KIO\_E\_RET:

388 00000DBD C3 <1> retn ; GO BACK

389 <1>

390 <1> ;----- ROUTINE TO TRANSLATE SCAN CODE PAIRS FOR STANDARD CALLS -----

391 <1> \_KIO\_S\_XLAT:

392 00000DBE 80FCE0 <1> cmp ah, 0E0h ; IS IT KEYPAD ENTER OR / ?

393 00000DC1 750F <1> jne short \_KIO\_S2 ; NO, CONTINUE

394 00000DC3 3C0D <1> cmp al, 0Dh ; KEYPAD ENTER CODE?

395 00000DC5 7408 <1> je short \_KIO\_S1 ; YES, MASSAGE A BIT

396 00000DC7 3C0A <1> cmp al, 0Ah ; CTRL KEYPAD ENTER CODE?

397 00000DC9 7404 <1> je short \_KIO\_S1 ; YES, MASSAGE THE SAME

398 00000DCB B435 <1> mov ah, 35h ; NO, MUST BE KEYPAD /

399 <1> \_kio\_ret: ; 03/12/2014

400 00000DCD F8 <1> clc

401 00000DCE C3 <1> retn

402 <1> ;jmp short \_KIO\_USE ; GIVE TO CALLER

403 <1> \_KIO\_S1:

404 00000DCF B41C <1> mov ah, 1Ch ; CONVERT TO COMPATIBLE OUTPUT

405 <1> ;jmp short \_KIO\_USE ; GIVE TO CALLER

406 00000DD1 C3 <1> retn

407 <1> \_KIO\_S2:

408 00000DD2 80FC84 <1> cmp ah, 84h ; IS IT ONE OF EXTENDED ONES?

409 00000DD5 7715 <1> ja short \_KIO\_DIS ; YES, THROW AWAY AND GET ANOTHER CHAR

410 00000DD7 3CF0 <1> cmp al, 0F0h ; IS IT ONE OF THE FILL-INs?

411 00000DD9 7506 <1> jne short \_KIO\_S3 ; NO, TRY LAST TEST

412 00000DDB 08E4 <1> or ah, ah ; AH = 0 IS SPECIAL CASE

413 00000DDD 740C <1> jz short \_KIO\_USE ; PASS THIS ON UNCHANGED

414 00000DDF EB0B <1> jmp short \_KIO\_DIS ; THROW AWAY THE REST

415 <1> \_KIO\_S3:

416 00000DE1 3CE0 <1> cmp al, 0E0h ; IS IT AN EXTENSION OF A PREVIOUS ONE?

417 <1> ;jne short \_KIO\_USE ; NO, MUST BE A STANDARD CODE

418 00000DE3 75E8 <1> jne short \_kio\_ret

419 00000DE5 08E4 <1> or ah, ah ; AH = 0 IS SPECIAL CASE

420 00000DE7 7402 <1> jz short \_KIO\_USE ; JUMP IF AH = 0

421 00000DE9 30C0 <1> xor al, al ; CONVERT TO COMPATIBLE OUTPUT

422 <1> ;jmp short \_KIO\_USE ; PASS IT ON TO CALLER

423 <1> \_KIO\_USE:

424 <1> ;clc ; CLEAR CARRY TO INDICATE GOOD CODE

425 00000DEB C3 <1> retn ; RETURN

426 <1> \_KIO\_DIS:

427 00000DEC F9 <1> stc ; SET CARRY TO INDICATE DISCARD CODE

428 00000DED C3 <1> retn ; RETURN

429 <1>

430 <1> ;----- INCREMENT BUFFER POINTER ROUTINE -----

431 <1> \_K4:

432 00000DEE 43 <1> inc ebx

433 00000DEF 43 <1> inc ebx ; MOVE TO NEXT WORD IN LIST

434 00000DF0 3B1D[965E0000] <1> cmp ebx, [BUFFER\_END] ; AT END OF BUFFER?

435 <1> ;jne short \_K5 ; NO, CONTINUE

436 00000DF6 7206 <1> jb short \_K5

437 00000DF8 8B1D[925E0000] <1> mov ebx, [BUFFER\_START] ; YES, RESET TO BUFFER BEGINNING

438 <1> \_K5:

439 00000DFE C3 <1> retn

440 <1>

441 <1> ; 20/02/2015

442 <1> ; 05/12/2014

443 <1> ; 26/08/2014

444 <1> ; KEYBOARD (HARDWARE) INTERRUPT - IRQ LEVEL 1

445 <1> ; (INT\_09h - Retro UNIX 8086 v1 - U9.ASM, 07/03/2014)

446 <1> ;

447 <1> ; Derived from "KB\_INT\_1" procedure of IBM "pc-at"

448 <1> ; rombios source code (06/10/1985)

449 <1> ; 'keybd.asm', HARDWARE INT 09h - (IRQ Level 1)

450 <1>

451 <1> ; EQUATES (IBM PC-XT-286 BIOS, 1986, 'POSQEQU.INC')

452 <1>

453 <1> ;--------- 8042 COMMANDS -------------------------------------------------------

454 <1> ENA\_KBD equ 0AEh ; ENABLE KEYBOARD COMMAND

455 <1> DIS\_KBD equ 0ADh ; DISABLE KEYBOARD COMMAND

456 <1> SHUT\_CMD equ 0FEh ; CAUSE A SHUTDOWN COMMAND

457 <1> ;--------- 8042 KEYBOARD INTERFACE AND DIAGNOSTIC CONTROL REGISTERS ------------

458 <1> STATUS\_PORT equ 064h ; 8042 STATUS PORT

459 <1> INPT\_BUF\_FULL equ 00000010b ; 1 = +INPUT BUFFER FULL

460 <1> PORT\_A equ 060h ; 8042 KEYBOARD SCAN CODE/CONTROL PORT

461 <1> ;---------- 8042 KEYBOARD RESPONSE ---------------------------------------------

462 <1> KB\_ACK equ 0FAh ; ACKNOWLEDGE PROM TRANSMISSION

463 <1> KB\_RESEND equ 0FEh ; RESEND REQUEST

464 <1> KB\_OVER\_RUN equ 0FFh ; OVER RUN SCAN CODE

465 <1> ;---------- KEYBOARD/LED COMMANDS ----------------------------------------------

466 <1> KB\_ENABLE equ 0F4h ; KEYBOARD ENABLE

467 <1> LED\_CMD equ 0EDh ; LED WRITE COMMAND

468 <1> KB\_TYPA\_RD equ 0F3h ; TYPAMATIC RATE/DELAY COMMAND

469 <1> ;---------- KEYBOARD SCAN CODES ------------------------------------------------

470 <1> NUM\_KEY equ 69 ; SCAN CODE FOR NUMBER LOCK KEY

471 <1> SCROLL\_KEY equ 70 ; SCAN CODE FOR SCROLL LOCK KEY

472 <1> ALT\_KEY equ 56 ; SCAN CODE FOR ALTERNATE SHIFT KEY

473 <1> CTL\_KEY equ 29 ; SCAN CODE FOR CONTROL KEY

474 <1> CAPS\_KEY equ 58 ; SCAN CODE FOR SHIFT LOCK KEY

475 <1> DEL\_KEY equ 83 ; SCAN CODE FOR DELETE KEY

476 <1> INS\_KEY equ 82 ; SCAN CODE FOR INSERT KEY

477 <1> LEFT\_KEY equ 42 ; SCAN CODE FOR LEFT SHIFT

478 <1> RIGHT\_KEY equ 54 ; SCAN CODE FOR RIGHT SHIFT

479 <1> SYS\_KEY equ 84 ; SCAN CODE FOR SYSTEM KEY

480 <1> ;---------- ENHANCED KEYBOARD SCAN CODES ---------------------------------------

481 <1> ID\_1 equ 0ABh ; 1ST ID CHARACTER FOR KBX

482 <1> ID\_2 equ 041h ; 2ND ID CHARACTER FOR KBX

483 <1> ID\_2A equ 054h ; ALTERNATE 2ND ID CHARACTER FOR KBX

484 <1> F11\_M equ 87 ; F11 KEY MAKE

485 <1> F12\_M equ 88 ; F12 KEY MAKE

486 <1> MC\_E0 equ 224 ; GENERAL MARKER CODE

487 <1> MC\_E1 equ 225 ; PAUSE KEY MARKER CODE

488 <1> ;---------- FLAG EQUATES WITHIN @KB\_FLAG----------------------------------------

489 <1> RIGHT\_SHIFT equ 00000001b ; RIGHT SHIFT KEY DEPRESSED

490 <1> LEFT\_SHIFT equ 00000010b ; LEFT SHIFT KEY DEPRESSED

491 <1> CTL\_SHIFT equ 00000100b ; CONTROL SHIFT KEY DEPRESSED

492 <1> ALT\_SHIFT equ 00001000b ; ALTERNATE SHIFT KEY DEPRESSED

493 <1> SCROLL\_STATE equ 00010000b ; SCROLL LOCK STATE IS ACTIVE

494 <1> NUM\_STATE equ 00100000b ; NUM LOCK STATE IS ACTIVE

495 <1> CAPS\_STATE equ 01000000b ; CAPS LOCK STATE IS ACTIVE

496 <1> INS\_STATE equ 10000000b ; INSERT STATE IS ACTIVE

497 <1> ;---------- FLAG EQUATES WITHIN @KB\_FLAG\_1 -------------------------------------

498 <1> L\_CTL\_SHIFT equ 00000001b ; LEFT CTL KEY DOWN

499 <1> L\_ALT\_SHIFT equ 00000010b ; LEFT ALT KEY DOWN

500 <1> SYS\_SHIFT equ 00000100b ; SYSTEM KEY DEPRESSED AND HELD

501 <1> HOLD\_STATE equ 00001000b ; SUSPEND KEY HAS BEEN TOGGLED

502 <1> SCROLL\_SHIFT equ 00010000b ; SCROLL LOCK KEY IS DEPRESSED

503 <1> NUM\_SHIFT equ 00100000b ; NUM LOCK KEY IS DEPRESSED

504 <1> CAPS\_SHIFT equ 01000000b ; CAPS LOCK KEY IS DEPRE55ED

505 <1> INS\_SHIFT equ 10000000b ; INSERT KEY IS DEPRESSED

506 <1> ;---------- FLAGS EQUATES WITHIN @KB\_FLAG\_2 -----------------------------------

507 <1> KB\_LEDS equ 00000111b ; KEYBOARD LED STATE BITS

508 <1> ; equ 00000001b ; SCROLL LOCK INDICATOR

509 <1> ; equ 00000010b ; NUM LOCK INDICATOR

510 <1> ; equ 00000100b ; CAPS LOCK INDICATOR

511 <1> ; equ 00001000b ; RESERVED (MUST BE ZERO)

512 <1> KB\_FA equ 00010000b ; ACKNOWLEDGMENT RECEIVED

513 <1> KB\_FE equ 00100000b ; RESEND RECEIVED FLAG

514 <1> KB\_PR\_LED equ 01000000b ; MODE INDICATOR UPDATE

515 <1> KB\_ERR equ 10000000b ; KEYBOARD TRANSMIT ERROR FLAG

516 <1> ;----------- FLAGS EQUATES WITHIN @KB\_FLAG\_3 -----------------------------------

517 <1> LC\_E1 equ 00000001b ; LAST CODE WAS THE E1 HIDDEN CODE

518 <1> LC\_E0 equ 00000010b ; LAST CODE WAS THE E0 HIDDEN CODE

519 <1> R\_CTL\_SHIFT equ 00000100b ; RIGHT CTL KEY DOWN

520 <1> R\_ALT\_SHIFT equ 00001000b ; RIGHT ALT KEY DOWN

521 <1> GRAPH\_ON equ 00001000b ; ALT GRAPHICS KEY DOWN (WT ONLY)

522 <1> KBX equ 00010000b ; ENHANCED KEYBOARD INSTALLED

523 <1> SET\_NUM\_LK equ 00100000b ; FORCE NUM LOCK IF READ ID AND KBX

524 <1> LC\_AB equ 01000000b ; LAST CHARACTER WAS FIRST ID CHARACTER

525 <1> RD\_ID equ 10000000b ; DOING A READ ID (MUST BE BIT0)

526 <1> ;

527 <1> ;----------- INTERRUPT EQUATES -------------------------------------------------

528 <1> EOI equ 020h ; END OF INTERRUPT COMMAND TO 8259

529 <1> INTA00 equ 020h ; 8259 PORT

530 <1>

531 <1>

532 <1> kb\_int:

533 <1>

534 <1> ; 17/10/2015 ('ctrlbrk')

535 <1> ; 05/12/2014

536 <1> ; 04/12/2014 (derived from pc-xt-286 bios source code -1986-)

537 <1> ; 26/08/2014

538 <1> ;

539 <1> ; 03/06/86 KEYBOARD BIOS

540 <1> ;

541 <1> ;--- HARDWARE INT 09H -- (IRQ LEVEL 1) ------------------------------------------

542 <1> ; ;

543 <1> ; KEYBOARD INTERRUPT ROUTINE ;

544 <1> ; ;

545 <1> ;--------------------------------------------------------------------------------

546 <1>

547 <1> KB\_INT\_1:

548 00000DFF FB <1> sti ; ENABLE INTERRUPTS

549 <1> ;push ebp

550 00000E00 50 <1> push eax

551 00000E01 53 <1> push ebx

552 00000E02 51 <1> push ecx

553 00000E03 52 <1> push edx

554 00000E04 56 <1> push esi

555 00000E05 57 <1> push edi

556 00000E06 1E <1> push ds

557 00000E07 06 <1> push es

558 00000E08 FC <1> cld ; FORWARD DIRECTION

559 00000E09 66B81000 <1> mov ax, KDATA

560 00000E0D 8ED8 <1> mov ds, ax

561 00000E0F 8EC0 <1> mov es, ax

562 <1> ;

563 <1> ;----- WAIT FOR KEYBOARD DISABLE COMMAND TO BE ACCEPTED

564 00000E11 B0AD <1> mov al, DIS\_KBD ; DISABLE THE KEYBOARD COMMAND

565 00000E13 E8A9050000 <1> call SHIP\_IT ; EXECUTE DISABLE

566 00000E18 FA <1> cli ; DISABLE INTERRUPTS

567 00000E19 B900000100 <1> mov ecx, 10000h ; SET MAXIMUM TIMEOUT

568 <1> KB\_INT\_01:

569 00000E1E E464 <1> in al, STATUS\_PORT ; READ ADAPTER STATUS

570 00000E20 A802 <1> test al, INPT\_BUF\_FULL ; CHECK INPUT BUFFER FULL STATUS BIT

571 00000E22 E0FA <1> loopnz KB\_INT\_01 ; WAIT FOR COMMAND TO BE ACCEPTED

572 <1> ;

573 <1> ;----- READ CHARACTER FROM KEYBOARD INTERFACE

574 00000E24 E460 <1> in al, PORT\_A ; READ IN THE CHARACTER

575 <1> ;

576 <1> ;----- SYSTEM HOOK INT 15H - FUNCTION 4FH (ON HARDWARE INT LEVEL 9H)

577 <1> ;MOV AH, 04FH ; SYSTEM INTERCEPT - KEY CODE FUNCTION

578 <1> ;STC ; SET CY=1 (IN CASE OF IRET)

579 <1> ;INT 15H ; CASETTE CALL (AL)=KEY SCAN CODE

580 <1> ; ; RETURNS CY=1 FOR INVALID FUNCTION

581 <1> ;JC KB\_INT\_02 ; CONTINUE IF CARRY FLAG SET ((AL)=CODE)

582 <1> ;JMP K26 ; EXIT IF SYSTEM HANDLES SCAN CODE

583 <1> ; ; EXİT HANDLES HARDWARE EOI AND ENABLE

584 <1> ;

585 <1> ;----- CHECK FOR A RESEND COMMAND TO KEYBOARD

586 <1> KB\_INT\_02: ; (AL)= SCAN CODE

587 00000E26 FB <1> sti ; ENABLE INTERRUPTS AGAIN

588 00000E27 3CFE <1> cmp al, KB\_RESEND ; IS THE INPUT A RESEND

589 00000E29 7411 <1> je short KB\_INT\_4 ; GO IF RESEND

590 <1> ;

591 <1> ;----- CHECK FOR RESPONSE TO A COMMAND TO KEYBOARD

592 00000E2B 3CFA <1> cmp al, KB\_ACK ; IS THE INPUT AN ACKNOWLEDGE

593 00000E2D 751A <1> jne short KB\_INT\_2 ; GO IF NOT

594 <1> ;

595 <1> ;----- A COMMAND TO THE KEYBOARD WAS ISSUED

596 00000E2F FA <1> cli ; DISABLE INTERRUPTS

597 00000E30 800D[8F5E0000]10 <1> or byte [KB\_FLAG\_2], KB\_FA ; INDICATE ACK RECEIVED

598 00000E37 E97A020000 <1> jmp K26 ; RETURN IF NOT (ACK RETURNED FOR DATA)

599 <1> ;

600 <1> ;----- RESEND THE LAST BYTE

601 <1> KB\_INT\_4:

602 00000E3C FA <1> cli ; DISABLE INTERRUPTS

603 00000E3D 800D[8F5E0000]20 <1> or byte [KB\_FLAG\_2], KB\_FE ; INDICATE RESEND RECEIVED

604 00000E44 E96D020000 <1> jmp K26 ; RETURN IF NOT ACK RETURNED FOR DATA)

605 <1> ;

606 <1> ;----- UPDATE MODE INDICATORS IF CHANGE IN STATE

607 <1> KB\_INT\_2:

608 00000E49 6650 <1> push ax ; SAVE DATA IN

609 00000E4B E83D060000 <1> call MAKE\_LED ; GO GET MODE INDICATOR DATA BYTE

610 00000E50 8A1D[8F5E0000] <1> mov bl, [KB\_FLAG\_2] ; GET PREVIOUS BITS

611 00000E56 30C3 <1> xor bl, al ; SEE IF ANY DIFFERENT

612 00000E58 80E307 <1> and bl, KB\_LEDS ; ISOLATE INDICATOR BITS

613 00000E5B 7405 <1> jz short UP0 ; IF NO CHANGE BYPASS UPDATE

614 00000E5D E8C0050000 <1> call SND\_LED ; GO TURN ON MODE INDICATORS

615 <1> UP0:

616 00000E62 6658 <1> pop ax ; RESTORE DATA IN

617 <1> ;------------------------------------------------------------------------

618 <1> ; START OF KEY PROCESSING ;

619 <1> ;------------------------------------------------------------------------

620 00000E64 88C4 <1> mov ah, al ; SAVE SCAN CODE IN AH ALSO

621 <1> ;

622 <1> ;----- TEST FOR OVERRUN SCAN CODE FROM KEYBOARD

623 00000E66 3CFF <1> cmp al, KB\_OVER\_RUN ; IS THIS AN OVERRUN CHAR

624 00000E68 0F843F050000 <1> je K62 ; BUFFER\_FULL\_BEEP

625 <1> ;

626 <1> K16:

627 00000E6E 8A3D[905E0000] <1> mov bh, [KB\_FLAG\_3] ; LOAD FLAGS FOR TESTING

628 <1> ;

629 <1> ;----- TEST TO SEE IF A READ\_ID IS IN PROGRESS

630 00000E74 F6C7C0 <1> test bh, RD\_ID+LC\_AB ; ARE WE DOING A READ ID?

631 00000E77 7449 <1> jz short NOT\_ID ; CONTINUE IF NOT

632 00000E79 7917 <1> jns short TST\_ID\_2 ; IS THE RD\_ID FLAG ON?

633 00000E7B 3CAB <1> cmp al, ID\_1 ; IS THIS THE 1ST ID CHARACTER?

634 00000E7D 7507 <1> jne short RST\_RD\_ID

635 00000E7F 800D[905E0000]40 <1> or byte [KB\_FLAG\_3], LC\_AB ; INDICATE 1ST ID WAS OK

636 <1> RST\_RD\_ID:

637 00000E86 8025[905E0000]7F <1> and byte [KB\_FLAG\_3], ~RD\_ID ; RESET THE READ ID FLAG

638 <1> ;jmp short ID\_EX ; AND EXIT

639 00000E8D E924020000 <1> jmp K26

640 <1> ;

641 <1> TST\_ID\_2:

642 00000E92 8025[905E0000]BF <1> and byte [KB\_FLAG\_3], ~LC\_AB ; RESET FLAG

643 00000E99 3C54 <1> cmp al, ID\_2A ; IS THIS THE 2ND ID CHARACTER?

644 00000E9B 7419 <1> je short KX\_BIT ; JUMP IF SO

645 00000E9D 3C41 <1> cmp al, ID\_2 ; IS THIS THE 2ND ID CHARACTER?

646 <1> ;jne short ID\_EX ; LEAVE IF NOT

647 00000E9F 0F8511020000 <1> jne K26

648 <1> ;

649 <1> ;----- A READ ID SAID THAT IT WAS ENHANCED KEYBOARD

650 00000EA5 F6C720 <1> test bh, SET\_NUM\_LK ; SHOULD WE SET NUM LOCK?

651 00000EA8 740C <1> jz short KX\_BIT ; EXIT IF NOT

652 00000EAA 800D[8D5E0000]20 <1> or byte [KB\_FLAG], NUM\_STATE ; FORCE NUM LOCK ON

653 00000EB1 E86C050000 <1> call SND\_LED ; GO SET THE NUM LOCK INDICATOR

654 <1> KX\_BIT:

655 00000EB6 800D[905E0000]10 <1> or byte [KB\_FLAG\_3], KBX ; INDICATE ENHANCED KEYBOARD WAS FOUND

656 00000EBD E9F4010000 <1> ID\_EX: jmp K26 ; EXIT

657 <1> ;

658 <1> NOT\_ID:

659 00000EC2 3CE0 <1> cmp al, MC\_E0 ; IS THIS THE GENERAL MARKER CODE?

660 00000EC4 750C <1> jne short TEST\_E1

661 00000EC6 800D[905E0000]12 <1> or byte [KB\_FLAG\_3], LC\_E0+KBX ; SET FLAG BIT, SET KBX, AND

662 <1> ;jmp short EXIT ; THROW AWAY THIS CODE

663 00000ECD E9EB010000 <1> jmp K26A

664 <1> TEST\_E1:

665 00000ED2 3CE1 <1> cmp al, MC\_E1 ; IS THIS THE PAUSE KEY?

666 00000ED4 750C <1> jne short NOT\_HC

667 00000ED6 800D[905E0000]11 <1> or byte [KB\_FLAG\_3], LC\_E1+KBX ; SET FLAG BIT, SET KBX, AND

668 00000EDD E9DB010000 <1> EXIT: jmp K26A ; THROW AWAY THIS CODE

669 <1> ;

670 <1> NOT\_HC:

671 00000EE2 247F <1> and al, 07Fh ; TURN OFF THE BREAK BIT

672 00000EE4 F6C702 <1> test bh, LC\_E0 ; LAST CODE THE E0 MARKER CODE

673 00000EE7 7414 <1> jz short NOT\_LC\_E0 ; JUMP IF NOT

674 <1> ;

675 00000EE9 BF[7A5D0000] <1> mov edi, \_K6+6 ; IS THIS A SHIFT KEY?

676 00000EEE AE <1> scasb

677 00000EEF 0F84C1010000 <1> je K26 ; K16B ; YES, THROW AWAY & RESET FLAG

678 00000EF5 AE <1> scasb

679 00000EF6 757C <1> jne short K16A ; NO, CONTINUE KEY PROCESSING

680 <1> ;jmp short K16B ; YES, THROW AWAY & RESET FLAG

681 00000EF8 E9B9010000 <1> jmp K26

682 <1> ;

683 <1> NOT\_LC\_E0:

684 00000EFD F6C701 <1> test bh, LC\_E1 ; LAST CODE THE E1 MARKER CODE?

685 00000F00 7435 <1> jz short T\_SYS\_KEY ; JUMP IF NOT

686 00000F02 B904000000 <1> mov ecx, 4 ; LENGHT OF SEARCH

687 00000F07 BF[785D0000] <1> mov edi, \_K6+4 ; IS THIS AN ALT, CTL, OR SHIFT?

688 00000F0C F2AE <1> repne scasb ; CHECK IT

689 <1> ;je short EXIT ; THROW AWAY IF SO

690 00000F0E 0F84A9010000 <1> je K26A

691 <1> ;

692 00000F14 3C45 <1> cmp al, NUM\_KEY ; IS IT THE PAUSE KEY?

693 <1> ;jne short K16B ; NO, THROW AWAY & RESET FLAG

694 00000F16 0F859A010000 <1> jne K26

695 00000F1C F6C480 <1> test ah, 80h ; YES, IS IT THE BREAK OF THE KEY?

696 <1> ;jnz short K16B ; YES, THROW THIS AWAY, TOO

697 00000F1F 0F8591010000 <1> jnz K26

698 <1> ; 20/02/2015

699 00000F25 F605[8E5E0000]08 <1> test byte [KB\_FLAG\_1],HOLD\_STATE ; NO, ARE WE PAUSED ALREADY?

700 <1> ;jnz short K16B ; YES, THROW AWAY

701 00000F2C 0F8584010000 <1> jnz K26

702 00000F32 E9E1020000 <1> jmp K39P ; NO, THIS IS THE REAL PAUSE STATE

703 <1> ;

704 <1> ;----- TEST FOR SYSTEM KEY

705 <1> T\_SYS\_KEY:

706 00000F37 3C54 <1> cmp al, SYS\_KEY ; IS IT THE SYSTEM KEY?

707 00000F39 7539 <1> jnz short K16A ; CONTINUE IF NOT

708 <1> ;

709 00000F3B F6C480 <1> test ah, 80h ; CHECK IF THIS A BREAK CODE

710 00000F3E 7524 <1> jnz short K16C ; DO NOT TOUCH SYSTEM INDICATOR IF TRUE

711 <1> ;

712 00000F40 F605[8E5E0000]04 <1> test byte [KB\_FLAG\_1], SYS\_SHIFT ; SEE IF IN SYSTEM KEY HELD DOWN

713 <1> ;jnz short K16B ; IF YES, DO NOT PROCESS SYSTEM INDICATOR

714 00000F47 0F8569010000 <1> jnz K26

715 <1> ;

716 00000F4D 800D[8E5E0000]04 <1> or byte [KB\_FLAG\_1], SYS\_SHIFT ; INDICATE SYSTEM KEY DEPRESSED

717 00000F54 B020 <1> mov al, EOI ; END OF INTERRUPT COMMAND

718 00000F56 E620 <1> out 20h, al ;out INTA00, al ; SEND COMMAND TO INTERRUPT CONTROL PORT

719 <1> ; INTERRUPT-RETURN-NO-EOI

720 00000F58 B0AE <1> mov al, ENA\_KBD ; INSURE KEYBOARD IS ENABLED

721 00000F5A E862040000 <1> call SHIP\_IT ; EXECUTE ENABLE

722 <1> ; !!! SYSREQ !!! function/system call (INTERRUPT) must be here !!!

723 <1> ;MOV AL, 8500H ; FUNCTION VALUE FOR MAKE OF SYSTEM KEY

724 <1> ;STI ; MAKE SURE INTERRUPTS ENABLED

725 <1> ;INT 15H ; USER INTERRUPT

726 00000F5F E965010000 <1> jmp K27A ; END PROCESSING

727 <1> ;

728 <1> ;K16B: jmp K26 ; IGNORE SYSTEM KEY

729 <1> ;

730 <1> K16C:

731 00000F64 8025[8E5E0000]FB <1> and byte [KB\_FLAG\_1], ~SYS\_SHIFT ; TURN OFF SHIFT KEY HELD DOWN

732 00000F6B B020 <1> mov al, EOI ; END OF INTERRUPT COMMAND

733 00000F6D E620 <1> out 20h, al ;out INTA00, al ; SEND COMMAND TO INTERRUPT CONTROL PORT

734 <1> ; INTERRUPT-RETURN-NO-EOI

735 <1> ;MOV AL, ENA\_KBD ; INSURE KEYBOARD IS ENABLED

736 <1> ;CALL SHIP\_IT ; EXECUTE ENABLE

737 <1> ;

738 <1> ;MOV AX, 8501H ; FUNCTION VALUE FOR BREAK OF SYSTEM KEY

739 <1> ;STI ; MAKE SURE INTERRUPTS ENABLED

740 <1> ;INT 15H ; USER INTERRUPT

741 <1> ;JMP K27A ; INGONRE SYSTEM KEY

742 <1> ;

743 00000F6F E94E010000 <1> jmp K27 ; IGNORE SYSTEM KEY

744 <1> ;

745 <1> ;----- TEST FOR SHIFT KEYS

746 <1> K16A:

747 00000F74 8A1D[8D5E0000] <1> mov bl, [KB\_FLAG] ; PUT STATE FLAGS IN BL

748 00000F7A BF[745D0000] <1> mov edi, \_K6 ; SHIFT KEY TABLE offset

749 00000F7F B908000000 <1> mov ecx, \_K6L ; LENGTH

750 00000F84 F2AE <1> repne scasb ; LOOK THROUGH THE TABLE FOR A MATCH

751 00000F86 88E0 <1> mov al, ah ; RECOVER SCAN CODE

752 00000F88 0F8510010000 <1> jne K25 ; IF NO MATCH, THEN SHIFT NOT FOUND

753 <1> ;

754 <1> ;------ SHIFT KEY FOUND

755 <1> K17:

756 00000F8E 81EF[755D0000] <1> sub edi, \_K6+1 ; ADJUST PTR TO SCAN CODE MATCH

757 00000F94 8AA7[7C5D0000] <1> mov ah, [edi+\_K7] ; GET MASK INTO AH

758 00000F9A B102 <1> mov cl, 2 ; SETUP COUNT FOR FLAG SHIFTS

759 00000F9C A880 <1> test al, 80h ; TEST FOR BREAK KEY

760 00000F9E 0F8596000000 <1> jnz K23 ; JUMP OF BREAK

761 <1> ;

762 <1> ;----- SHIFT MAKE FOUND, DETERMINE SET OR TOGGLE

763 <1> K17C:

764 00000FA4 80FC10 <1> cmp ah, SCROLL\_SHIFT

765 00000FA7 732B <1> jae short K18 ; IF SCROLL SHIFT OR ABOVE, TOGGLE KEY

766 <1> ;

767 <1> ;----- PLAIN SHIFT KEY, SET SHIFT ON

768 00000FA9 0825[8D5E0000] <1> or [KB\_FLAG], ah ; TURN ON SHIFT BIT

769 00000FAF A80C <1> test al, CTL\_SHIFT+ALT\_SHIFT ; IS IT ALT OR CTRL?

770 <1> ;jnz short K17D ; YES, MORE FLAGS TO SET

771 00000FB1 0F84FF000000 <1> jz K26 ; NO, INTERRUPT RETURN

772 <1> K17D:

773 00000FB7 F6C702 <1> test bh, LC\_E0 ; IS THIS ONE OF NEW KEYS?

774 00000FBA 740B <1> jz short K17E ; NO, JUMP

775 00000FBC 0825[905E0000] <1> or [KB\_FLAG\_3], ah ; SET BITS FOR RIGHT CTRL, ALT

776 00000FC2 E9EF000000 <1> jmp K26 ; INTERRUPT RETURN

777 <1> K17E:

778 00000FC7 D2EC <1> shr ah, cl ; MOVE FLAG BITS TWO POSITIONS

779 00000FC9 0825[8E5E0000] <1> or [KB\_FLAG\_1], ah ; SET BITS FOR LEFT CTRL, ALT

780 00000FCF E9E2000000 <1> jmp K26

781 <1> ;

782 <1> ;----- TOGGLED SHIFT KEY, TEST FOR 1ST MAKE OR NOT

783 <1> K18: ; SHIFT-TOGGLE

784 00000FD4 F6C304 <1> test bl, CTL\_SHIFT ; CHECK CTL SHIFT STATE

785 <1> ;jz short K18A ; JUMP IF NOT CTL STATE

786 00000FD7 0F85C1000000 <1> jnz K25 ; JUMP IF CTL STATE

787 <1> K18A:

788 00000FDD 3C52 <1> cmp al, INS\_KEY ; CHECK FOR INSERT KEY

789 00000FDF 7524 <1> jne short K22 ; JUMP IF NOT INSERT KEY

790 00000FE1 F6C308 <1> test bl, ALT\_SHIFT ; CHECK FOR ALTERNATE SHIFT

791 <1> ;jz short K18B ; JUMP IF NOT ALTERNATE SHIFT

792 00000FE4 0F85B4000000 <1> jnz K25 ; JUMP IF ALTERNATE SHIFT

793 <1> K18B:

794 00000FEA F6C702 <1> test bh, LC\_E0 ;20/02/2015 ; IS THIS NEW INSERT KEY?

795 00000FED 7516 <1> jnz short K22 ; YES, THIS ONE'S NEVER A '0'

796 <1> K19:

797 00000FEF F6C320 <1> test bl, NUM\_STATE ; CHECK FOR BASE STATE

798 00000FF2 750C <1> jnz short K21 ; JUMP IF NUM LOCK IS ON

799 00000FF4 F6C303 <1> test bl, LEFT\_SHIFT+RIGHT\_SHIFT ; TEST FOR SHIFT STATE

800 00000FF7 740C <1> jz short K22 ; JUMP IF BASE STATE

801 <1> K20: ; NUMERIC ZERO, NOT INSERT KEY

802 00000FF9 88C4 <1> mov ah, al ; PUT SCAN CODE BACK IN AH

803 00000FFB E99E000000 <1> jmp K25 ; NUMERAL '0', STNDRD. PROCESSING

804 <1> K21: ; MIGHT BE NUMERIC

805 00001000 F6C303 <1> test bl, LEFT\_SHIFT+RIGHT\_SHIFT

806 00001003 74F4 <1> jz short K20 ; IS NUMERIC, STD. PROC.

807 <1> ;

808 <1> K22: ; SHIFT TOGGLE KEY HIT; PROCESS IT

809 00001005 8425[8E5E0000] <1> test ah, [KB\_FLAG\_1] ; IS KEY ALREADY DEPRESSED

810 0000100B 0F85A5000000 <1> jnz K26 ; JUMP IF KEY ALREADY DEPRESSED

811 <1> K22A:

812 00001011 0825[8E5E0000] <1> or [KB\_FLAG\_1], ah ; INDICATE THAT THE KEY IS DEPRESSED

813 00001017 3025[8D5E0000] <1> xor [KB\_FLAG], ah ; TOGGLE THE SHIFT STATE

814 <1> ;

815 <1> ;----- TOGGLE LED IF CAPS, NUM OR SCROLL KEY DEPRESSED

816 0000101D F6C470 <1> test ah, CAPS\_SHIFT+NUM\_SHIFT+SCROLL\_SHIFT ; SHIFT TOGGLE?

817 00001020 7409 <1> jz short K22B ; GO IF NOT

818 <1> ;

819 00001022 6650 <1> push ax ; SAVE SCAN CODE AND SHIFT MASK

820 00001024 E8F9030000 <1> call SND\_LED ; GO TURN MODE INDICATORS ON

821 00001029 6658 <1> pop ax ; RESTORE SCAN CODE

822 <1> K22B:

823 0000102B 3C52 <1> cmp al, INS\_KEY ; TEST FOR 1ST MAKE OF INSERT KEY

824 0000102D 0F8583000000 <1> jne K26 ; JUMP IF NOT INSERT KEY

825 00001033 88C4 <1> mov ah, al ; SCAN CODE IN BOTH HALVES OF AX

826 00001035 E999000000 <1> jmp K28 ; FLAGS UPDATED, PROC. FOR BUFFER

827 <1> ;

828 <1> ;----- BREAK SHIFT FOUND

829 <1> K23: ; BREAK-SHIFT-FOUND

830 0000103A 80FC10 <1> cmp ah, SCROLL\_SHIFT ; IS THIS A TOGGLE KEY

831 0000103D F6D4 <1> not ah ; INVERT MASK

832 0000103F 7355 <1> jae short K24 ; YES, HANDLE BREAK TOGGLE

833 00001041 2025[8D5E0000] <1> and [KB\_FLAG], ah ; TURN OFF SHIFT BIT

834 00001047 80FCFB <1> cmp ah, ~CTL\_SHIFT ; IS THIS ALT OR CTL?

835 0000104A 7730 <1> ja short K23D ; NO, ALL DONE

836 <1> ;

837 0000104C F6C702 <1> test bh, LC\_E0 ; 2ND ALT OR CTL?

838 0000104F 7408 <1> jz short K23A ; NO, HANSLE NORMALLY

839 00001051 2025[905E0000] <1> and [KB\_FLAG\_3], ah ; RESET BIT FOR RIGHT ALT OR CTL

840 00001057 EB08 <1> jmp short K23B ; CONTINUE

841 <1> K23A:

842 00001059 D2FC <1> sar ah, cl ; MOVE THE MASK BIT TWO POSITIONS

843 0000105B 2025[8E5E0000] <1> and [KB\_FLAG\_1], ah ; RESET BIT FOR LEFT ALT AND CTL

844 <1> K23B:

845 00001061 88C4 <1> mov ah, al ; SAVE SCAN CODE

846 00001063 A0[905E0000] <1> mov al, [KB\_FLAG\_3] ; GET RIGHT ALT & CTRL FLAGS

847 00001068 D2E8 <1> shr al, cl ; MOVE TO BITS 1 & 0

848 0000106A 0A05[8E5E0000] <1> or al, [KB\_FLAG\_1] ; PUT IN LEFT ALŞT & CTL FLAGS

849 00001070 D2E0 <1> shl al, cl ; MOVE BACK TO BITS 3 & 2

850 00001072 240C <1> and al, ALT\_SHIFT+CTL\_SHIFT ; FILTER OUT OTHER GARBAGE

851 00001074 0805[8D5E0000] <1> or [KB\_FLAG], al ; PUT RESULT IN THE REAL FLAGS

852 0000107A 88E0 <1> mov al, ah

853 <1> K23D:

854 0000107C 3CB8 <1> cmp al, ALT\_KEY+80h ; IS THIS ALTERNATE SHIFT RELEASE

855 0000107E 7536 <1> jne short K26 ; INTERRUPT RETURN

856 <1> ;

857 <1> ;----- ALTERNATE SHIFT KEY RELEASED, GET THE VALUE INTO BUFFER

858 00001080 A0[915E0000] <1> mov al, [ALT\_INPUT]

859 00001085 B400 <1> mov ah, 0 ; SCAN CODE OF 0

860 00001087 8825[915E0000] <1> mov [ALT\_INPUT], ah ; ZERO OUT THE FIELD

861 0000108D 3C00 <1> cmp al, 0 ; WAS THE INPUT = 0?

862 0000108F 7425 <1> je short K26 ; INTERRUPT\_RETURN

863 <1> ; 29/01/2016

864 <1> ;jmp K61 ; IT WASN'T, SO PUT IN BUFFER

865 00001091 E9D0020000 <1> jmp \_K60

866 <1> ;

867 <1> K24: ; BREAK-TOGGLE

868 00001096 2025[8E5E0000] <1> and [KB\_FLAG\_1], ah ; INDICATE NO LONGER DEPRESSED

869 0000109C EB18 <1> jmp short K26 ; INTERRUPT\_RETURN

870 <1> ;

871 <1> ;----- TEST FOR HOLD STATE

872 <1> ; AL, AH = SCAN CODE

873 <1> K25: ; NO-SHIFT-FOUND

874 0000109E 3C80 <1> cmp al, 80h ; TEST FOR BREAK KEY

875 000010A0 7314 <1> jae short K26 ; NOTHING FOR BREAK CHARS FROM HERE ON

876 000010A2 F605[8E5E0000]08 <1> test byte [KB\_FLAG\_1], HOLD\_STATE ; ARE WE IN HOLD STATE

877 000010A9 7428 <1> jz short K28 ; BRANCH AROUND TEST IF NOT

878 000010AB 3C45 <1> cmp al, NUM\_KEY

879 000010AD 7407 <1> je short K26 ; CAN'T END HOLD ON NUM\_LOCK

880 000010AF 8025[8E5E0000]F7 <1> and byte [KB\_FLAG\_1], ~HOLD\_STATE ; TURN OFF THE HOLD STATE BIT

881 <1> ;

882 <1> K26:

883 000010B6 8025[905E0000]FC <1> and byte [KB\_FLAG\_3], ~(LC\_E0+LC\_E1) ; RESET LAST CHAR H.C. FLAG

884 <1> K26A: ; INTERRUPT-RETURN

885 000010BD FA <1> cli ; TURN OFF INTERRUPTS

886 000010BE B020 <1> mov al, EOI ; END OF INTERRUPT COMMAND

887 000010C0 E620 <1> out 20h, al ;out INTA00, al ; SEND COMMAND TO INTERRUPT CONTROL PORT

888 <1> K27: ; INTERRUPT-RETURN-NO-EOI

889 000010C2 B0AE <1> mov al, ENA\_KBD ; INSURE KEYBOARD IS ENABLED

890 000010C4 E8F8020000 <1> call SHIP\_IT ; EXECUTE ENABLE

891 <1> K27A:

892 000010C9 FA <1> cli ; DISABLE INTERRUPTS

893 <1> ;;mov byte [intflg], 0 ; 07/01/2017 ;; 15/01/2017

894 000010CA 07 <1> pop es ; RESTORE REGISTERS

895 000010CB 1F <1> pop ds

896 000010CC 5F <1> pop edi

897 000010CD 5E <1> pop esi

898 000010CE 5A <1> pop edx

899 000010CF 59 <1> pop ecx

900 000010D0 5B <1> pop ebx

901 000010D1 58 <1> pop eax

902 <1> ;pop ebp

903 000010D2 CF <1> iretd ; RETURN

904 <1>

905 <1> ;----- NOT IN HOLD STATE

906 <1> K28: ; NO-HOLD-STATE

907 000010D3 3C58 <1> cmp al, 88 ; TEST FOR OUT-OF-RANGE SCAN CODES

908 000010D5 77DF <1> ja short K26 ; IGNORE IF OUT-OF-RANGE

909 <1> ;

910 000010D7 F6C308 <1> test bl, ALT\_SHIFT ; ARE WE IN ALTERNATE SHIFT

911 <1> ;jz short K28A ; IF NOT ALTERNATE

912 000010DA 0F84F1000000 <1> jz K38

913 <1> ;

914 000010E0 F6C710 <1> test bh, KBX ; IS THIS THE ENCHANCED KEYBOARD?

915 000010E3 740D <1> jz short K29 ; NO, ALT STATE IS REAL

916 <1> ;28/02/2015

917 000010E5 F605[8E5E0000]04 <1> test byte [KB\_FLAG\_1], SYS\_SHIFT ; YES, IS SYSREQ KEY DOWN?

918 <1> ;jz short K29 ; NO, ALT STATE IS REAL

919 000010EC 0F85DF000000 <1> jnz K38 ; YES, THIS IS PHONY ALT STATE

920 <1> ; ; DUE TO PRESSING SYSREQ

921 <1> ;K28A: jmp short K38

922 <1> ;

923 <1> ;----- TEST FOR RESET KEY SEQUENCE (CTL ALT DEL)

924 <1> K29: ; TEST-RESET

925 000010F2 F6C304 <1> test bl, CTL\_SHIFT ; ARE WE IN CONTROL SHIFT ALSO?

926 000010F5 740B <1> jz short K31 ; NO\_RESET

927 000010F7 3C53 <1> cmp al, DEL\_KEY ; CTL-ALT STATE, TEST FOR DELETE KEY

928 000010F9 7507 <1> jne short K31 ; NO\_RESET, IGNORE

929 <1> ;

930 <1> ;----- CTL-ALT-DEL HAS BEEN FOUND

931 <1> ; 26/08/2014

932 <1> cpu\_reset:

933 <1> ; IBM PC/AT ROM BIOS source code - 10/06/85 (TEST4.ASM - PROC\_SHUTDOWN)

934 <1> ; Send FEh (system reset command) to the keyboard controller.

935 000010FB B0FE <1> mov al, SHUT\_CMD ; SHUTDOWN COMMAND

936 000010FD E664 <1> out STATUS\_PORT, al ; SEND TO KEYBOARD CONTROL PORT

937 <1> khere:

938 000010FF F4 <1> hlt ; WAIT FOR 80286 RESET

939 00001100 EBFD <1> jmp short khere ; INSURE HALT

940 <1>

941 <1> ;

942 <1> ;----- IN ALTERNATE SHIFT, RESET NOT FOUND

943 <1> K31: ; NO-RESET

944 00001102 3C39 <1> cmp al, 57 ; TEST FOR SPACE KEY

945 00001104 7507 <1> jne short K311 ; NOT THERE

946 00001106 B020 <1> mov al, ' ' ; SET SPACE CHAR

947 00001108 E948020000 <1> jmp K57 ; BUFFER\_FILL

948 <1> K311:

949 0000110D 3C0F <1> cmp al, 15 ; TEST FOR TAB KEY

950 0000110F 7509 <1> jne short K312 ; NOT THERE

951 00001111 66B800A5 <1> mov ax, 0A500h ; SET SPECIAL CODE FOR ALT-TAB

952 00001115 E93B020000 <1> jmp K57 ; BUFFER\_FILL

953 <1> K312:

954 0000111A 3C4A <1> cmp al, 74 ; TEST FOR KEY PAD -

955 0000111C 0F84A2000000 <1> je K37B ; GO PROCESS

956 00001122 3C4E <1> cmp al, 78 ; TEST FOR KEY PAD +

957 00001124 0F849A000000 <1> je K37B ; GO PROCESS

958 <1> ;

959 <1> ;----- LOOK FOR KEY PAD ENTRY

960 <1> K32: ; ALT-KEY-PAD

961 0000112A BF[505D0000] <1> mov edi, K30 ; ALT-INPUT-TABLE offset

962 0000112F B90A000000 <1> mov ecx, 10 ; LOOK FOR ENTRY USING KEYPAD

963 00001134 F2AE <1> repne scasb ; LOOK FOR MATCH

964 00001136 7525 <1> jne short K33 ; NO\_ALT\_KEYPAD

965 00001138 F6C702 <1> test bh, LC\_E0 ; IS THIS ONE OF THE NEW KEYS?

966 0000113B 0F858A000000 <1> jnz K37C ; YES, JUMP, NOT NUMPAD KEY

967 00001141 81EF[515D0000] <1> sub edi, K30+1 ; DI NOW HAS ENTRY VALUE

968 00001147 A0[915E0000] <1> mov al, [ALT\_INPUT] ; GET THE CURRENT BYTE

969 0000114C B40A <1> mov ah, 10 ; MULTIPLY BY 10

970 0000114E F6E4 <1> mul ah

971 00001150 6601F8 <1> add ax, di ; ADD IN THE LATEST ENTRY

972 00001153 A2[915E0000] <1> mov [ALT\_INPUT], al ; STORE IT AWAY

973 <1> ;K32A:

974 00001158 E959FFFFFF <1> jmp K26 ; THROW AWAY THAT KEYSTROKE

975 <1> ;

976 <1> ;----- LOOK FOR SUPERSHIFT ENTRY

977 <1> K33: ; NO-ALT-KEYPAD

978 0000115D C605[915E0000]00 <1> mov byte [ALT\_INPUT], 0 ; ZERO ANY PREVIOUS ENTRY INTO INPUT

979 00001164 B91A000000 <1> mov ecx, 26 ; (DI),(ES) ALREADY POINTING

980 00001169 F2AE <1> repne scasb ; LOOK FOR MATCH IN ALPHABET

981 0000116B 7450 <1> je short K37A ; MATCH FOUND, GO FILLL THE BUFFER

982 <1> ;

983 <1> ;----- LOOK FOR TOP ROW OF ALTERNATE SHIFT

984 <1> K34: ; ALT-TOP-ROW

985 0000116D 3C02 <1> cmp al, 2 ; KEY WITH '1' ON IT

986 0000116F 7253 <1> jb short K37B ; MUST BE ESCAPE

987 00001171 3C0D <1> cmp al, 13 ; IS IT IN THE REGION

988 00001173 7705 <1> ja short K35 ; NO, ALT SOMETHING ELSE

989 00001175 80C476 <1> add ah, 118 ; CONVERT PSEUDO SCAN CODE TO RANGE

990 00001178 EB43 <1> jmp short K37A ; GO FILL THE BUFFER

991 <1> ;

992 <1> ;----- TRANSLATE ALTERNATE SHIFT PSEUDO SCAN CODES

993 <1> K35: ; ALT-FUNCTION

994 0000117A 3C57 <1> cmp al, F11\_M ; IS IT F11?

995 0000117C 7209 <1> jb short K35A ; 20/02/2015 ; NO, BRANCH

996 0000117E 3C58 <1> cmp al, F12\_M ; IS IT F12?

997 00001180 7705 <1> ja short K35A ; 20/02/2015 ; NO, BRANCH

998 00001182 80C434 <1> add ah, 52 ; CONVERT TO PSEUDO SCAN CODE

999 00001185 EB36 <1> jmp short K37A ; GO FILL THE BUFFER

1000 <1> K35A:

1001 00001187 F6C702 <1> test bh, LC\_E0 ; DO WE HAVE ONE OF THE NEW KEYS?

1002 0000118A 7422 <1> jz short K37 ; NO, JUMP

1003 0000118C 3C1C <1> cmp al, 28 ; TEST FOR KEYPAD ENTER

1004 0000118E 7509 <1> jne short K35B ; NOT THERE

1005 00001190 66B800A6 <1> mov ax, 0A600h ; SPECIAL CODE

1006 00001194 E9BC010000 <1> jmp K57 ; BUFFER FILL

1007 <1> K35B:

1008 00001199 3C53 <1> cmp al, 83 ; TEST FOR DELETE KEY

1009 0000119B 742E <1> je short K37C ; HANDLE WITH OTHER EDIT KEYS

1010 0000119D 3C35 <1> cmp al, 53 ; TEST FOR KEYPAD /

1011 <1> ;jne short K32A ; NOT THERE, NO OTHER E0 SPECIALS

1012 0000119F 0F8511FFFFFF <1> jne K26

1013 000011A5 66B800A4 <1> mov ax, 0A400h ; SPECIAL CODE

1014 000011A9 E9A7010000 <1> jmp K57 ; BUFFER FILL

1015 <1> K37:

1016 000011AE 3C3B <1> cmp al, 59 ; TEST FOR FUNCTION KEYS (F1)

1017 000011B0 7212 <1> jb short K37B ; NO FN, HANDLE W/OTHER EXTENDED

1018 000011B2 3C44 <1> cmp al, 68 ; IN KEYPAD REGION?

1019 <1> ;ja short K32A ; IF SO, IGNORE

1020 000011B4 0F87FCFEFFFF <1> ja K26

1021 000011BA 80C42D <1> add ah, 45 ; CONVERT TO PSEUDO SCAN CODE

1022 <1> K37A:

1023 000011BD B000 <1> mov al, 0 ; ASCII CODE OF ZERO

1024 000011BF E991010000 <1> jmp K57 ; PUT IT IN THE BUFFER

1025 <1> K37B:

1026 000011C4 B0F0 <1> mov al, 0F0h ; USE SPECIAL ASCII CODE

1027 000011C6 E98A010000 <1> jmp K57 ; PUT IT IN THE BUFFER

1028 <1> K37C:

1029 000011CB 0450 <1> add al, 80 ; CONVERT SCAN CODE (EDIT KEYS)

1030 000011CD 88C4 <1> mov ah, al ; (SCAN CODE NOT IN AH FOR INSERT)

1031 000011CF EBEC <1> jmp short K37A ; PUT IT IN THE BUFFER

1032 <1> ;

1033 <1> ;----- NOT IN ALTERNATE SHIFT

1034 <1> K38: ; NOT-ALT-SHIFT

1035 <1> ; BL STILL HAS SHIFT FLAGS

1036 000011D1 F6C304 <1> test bl, CTL\_SHIFT ; ARE WE IN CONTROL SHIFT?

1037 <1> ;jnz short K38A ; YES, START PROCESSING

1038 000011D4 0F84B0000000 <1> jz K44 ; NOT-CTL-SHIFT

1039 <1> ;

1040 <1> ;----- CONTROL SHIFT, TEST SPECIAL CHARACTERS

1041 <1> ;----- TEST FOR BREAK

1042 <1> K38A:

1043 000011DA 3C46 <1> cmp al, SCROLL\_KEY ; TEST FOR BREAK

1044 000011DC 7531 <1> jne short K39 ; JUMP, NO-BREAK

1045 000011DE F6C710 <1> test bh, KBX ; IS THIS THE ENHANCED KEYBOARD?

1046 000011E1 7405 <1> jz short K38B ; NO, BREAK IS VALID

1047 000011E3 F6C702 <1> test bh, LC\_E0 ; YES, WAS LAST CODE AN E0?

1048 000011E6 7427 <1> jz short K39 ; NO-BREAK, TEST FOR PAUSE

1049 <1> K38B:

1050 000011E8 8B1D[9A5E0000] <1> mov ebx, [BUFFER\_HEAD] ; RESET BUFFER TO EMPTY

1051 000011EE 891D[9E5E0000] <1> mov [BUFFER\_TAIL], ebx

1052 000011F4 C605[8C5E0000]80 <1> mov byte [BIOS\_BREAK], 80h ; TURN ON BIOS\_BREAK BIT

1053 <1> ;

1054 <1> ;----- ENABLE KEYBOARD

1055 000011FB B0AE <1> mov al, ENA\_KBD ; ENABLE KEYBOARD

1056 000011FD E8BF010000 <1> call SHIP\_IT ; EXECUTE ENABLE

1057 <1> ;

1058 <1> ; CTRL+BREAK code here !!!

1059 <1> ;INT 1BH ; BREAK INTERRUPT VECTOR

1060 <1> ; 17/10/2015

1061 00001202 E8CF510000 <1> call ctrlbrk ; control+break subroutine

1062 <1> ;

1063 00001207 6629C0 <1> sub ax, ax ; PUT OUT DUMMY CHARACTER

1064 0000120A E946010000 <1> jmp K57 ; BUFFER\_FILL

1065 <1> ;

1066 <1> ;----- TEST FOR PAUSE

1067 <1> K39: ; NO\_BREAK

1068 0000120F F6C710 <1> test bh, KBX ; IS THIS THE ENHANCED KEYBOARD?

1069 00001212 7537 <1> jnz short K41 ; YES, THEN THIS CAN'T BE PAUSE

1070 00001214 3C45 <1> cmp al, NUM\_KEY ; LOOK FOR PAUSE KEY

1071 00001216 7533 <1> jne short K41 ; NO-PAUSE

1072 <1> K39P:

1073 00001218 800D[8E5E0000]08 <1> or byte [KB\_FLAG\_1], HOLD\_STATE ; TURN ON THE HOLD FLAG

1074 <1> ;

1075 <1> ;----- ENABLE KEYBOARD

1076 0000121F B0AE <1> mov al, ENA\_KBD ; ENABLE KEYBOARD

1077 00001221 E89B010000 <1> call SHIP\_IT ; EXECUTE ENABLE

1078 <1> K39A:

1079 00001226 B020 <1> mov al, EOI ; END OF INTERRUPT TO CONTROL PORT

1080 00001228 E620 <1> out 20h, al ;out INTA00, al ; ALLOW FURTHER KEYSTROKE INTERRUPTS

1081 <1> ;

1082 <1> ;----- DURING PAUSE INTERVAL, TURN COLOR CRT BACK ON

1083 0000122A 803D[C25E0000]07 <1> cmp byte [CRT\_MODE], 7 ; IS THIS BLACK AND WHITE CARD

1084 00001231 740A <1> je short K40 ; YES, NOTHING TO DO

1085 00001233 66BAD803 <1> mov dx, 03D8h ; PORT FOR COLOR CARD

1086 00001237 A0[C35E0000] <1> mov al, [CRT\_MODE\_SET] ; GET THE VALUE OF THE CURRENT MODE

1087 0000123C EE <1> out dx, al ; SET THE CRT MODE, SO THAT CRT IS ON

1088 <1> ;

1089 <1> K40: ; PAUSE-LOOP

1090 0000123D F605[8E5E0000]08 <1> test byte [KB\_FLAG\_1], HOLD\_STATE ; CHECK HOLD STATE FLAG

1091 00001244 75F7 <1> jnz short K40 ; LOOP UNTIL FLAG TURNED OFF

1092 <1> ;

1093 00001246 E977FEFFFF <1> jmp K27 ; INTERRUPT\_RETURN\_NO\_EOI

1094 <1> ;

1095 <1> ;----- TEST SPECIAL CASE KEY 55

1096 <1> K41: ; NO-PAUSE

1097 0000124B 3C37 <1> cmp al, 55 ; TEST FOR \*/PRTSC KEY

1098 0000124D 7513 <1> jne short K42 ; NOT-KEY-55

1099 0000124F F6C710 <1> test bh, KBX ; IS THIS THE ENHANCED KEYBOARD?

1100 00001252 7405 <1> jz short K41A ; NO, CTL-PRTSC IS VALID

1101 00001254 F6C702 <1> test bh, LC\_E0 ; YES, WAS LAST CODE AN E0?

1102 00001257 7421 <1> jz short K42B ; NO, TRANSLATE TO A FUNCTION

1103 <1> K41A:

1104 00001259 66B80072 <1> mov ax, 114\*256 ; START/STOP PRINTING SWITCH

1105 0000125D E9F3000000 <1> jmp K57 ; BUFFER\_FILL

1106 <1> ;

1107 <1> ;----- SET UP TO TRANSLATE CONTROL SHIFT

1108 <1> K42: ; NOT-KEY-55

1109 00001262 3C0F <1> cmp al, 15 ; IS IT THE TAB KEY?

1110 00001264 7414 <1> je short K42B ; YES, XLATE TO FUNCTION CODE

1111 00001266 3C35 <1> cmp al, 53 ; IS IT THE / KEY?

1112 00001268 750E <1> jne short K42A ; NO, NO MORE SPECIAL CASES

1113 0000126A F6C702 <1> test bh, LC\_E0 ; YES, IS IT FROM THE KEY PAD?

1114 0000126D 7409 <1> jz short K42A ; NO, JUST TRANSLATE

1115 0000126F 66B80095 <1> mov ax, 9500h ; YES, SPECIAL CODE FOR THIS ONE

1116 00001273 E9DD000000 <1> jmp K57 ; BUFFER FILL

1117 <1> K42A:

1118 <1> ;;mov ebx, \_K8 ; SET UP TO TRANSLATE CTL

1119 00001278 3C3B <1> cmp al, 59 ; IS IT IN CHARACTER TABLE?

1120 <1> ;jb short K45F ; YES, GO TRANSLATE CHAR

1121 <1> ;;jb K56 ; 20/02/2015

1122 <1> ;;jmp K64 ; 20/02/2015

1123 <1> K42B:

1124 0000127A BB[845D0000] <1> mov ebx, \_K8 ; SET UP TO TRANSLATE CTL

1125 0000127F 0F82AE000000 <1> jb K56 ;; 20/02/2015

1126 00001285 E9B9000000 <1> jmp K64

1127 <1> ;

1128 <1> ;----- NOT IN CONTROL SHIFT

1129 <1> K44: ; NOT-CTL-SHIFT

1130 0000128A 3C37 <1> cmp al, 55 ; PRINT SCREEN KEY?

1131 0000128C 7528 <1> jne short K45 ; NOT PRINT SCREEN

1132 0000128E F6C710 <1> test bh, KBX ; IS THIS ENHANCED KEYBOARD?

1133 00001291 7407 <1> jz short K44A ; NO, TEST FOR SHIFT STATE

1134 00001293 F6C702 <1> test bh, LC\_E0 ; YES, LAST CODE A MARKER?

1135 00001296 7507 <1> jnz short K44B ; YES, IS PRINT SCREEN

1136 00001298 EB41 <1> jmp short K45C ; NO, TRANSLATE TO '\*' CHARACTER

1137 <1> K44A:

1138 0000129A F6C303 <1> test bl, LEFT\_SHIFT+RIGHT\_SHIFT ; NOT 101 KBD, SHIFT KEY DOWN?

1139 0000129D 743C <1> jz short K45C ; NO, TRANSLATE TO '\*' CHARACTER

1140 <1> ;

1141 <1> ;----- ISSUE INTERRUPT TO INDICATE PRINT SCREEN FUNCTION

1142 <1> K44B:

1143 0000129F B0AE <1> mov al, ENA\_KBD ; INSURE KEYBOARD IS ENABLED

1144 000012A1 E81B010000 <1> call SHIP\_IT ; EXECUTE ENABLE

1145 000012A6 B020 <1> mov al, EOI ; END OF CURRENT INTERRUPT

1146 000012A8 E620 <1> out 20h, al ;out INTA00, al ; SO FURTHER THINGS CAN HAPPEN

1147 <1> ; Print Screen !!! ; ISSUE PRINT SCREEN INTERRUPT (INT 05h)

1148 <1> ;PUSH BP ; SAVE POINTER

1149 <1> ;INT 5H ; ISSUE PRINT SCREEN INTERRUPT

1150 <1> ;POP BP ; RESTORE POINTER

1151 000012AA 8025[905E0000]FC <1> and byte [KB\_FLAG\_3], ~(LC\_E0+LC\_E1) ; ZERO OUT THESE FLAGS

1152 000012B1 E90CFEFFFF <1> jmp K27 ; GO BACK WITHOUT EOI OCCURRING

1153 <1> ;

1154 <1> ;----- HANDLE IN-CORE KEYS

1155 <1> K45: ; NOT-PRINT-SCREEN

1156 000012B6 3C3A <1> cmp al, 58 ; TEST FOR IN-CORE AREA

1157 000012B8 7734 <1> ja short K46 ; JUMP IF NOT

1158 000012BA 3C35 <1> cmp al, 53 ; IS THIS THE '/' KEY?

1159 000012BC 7505 <1> jne short K45A ; NO, JUMP

1160 000012BE F6C702 <1> test bh, LC\_E0 ; WAS THE LAST CODE THE MARKER?

1161 000012C1 7518 <1> jnz short K45C ; YES, TRANSLATE TO CHARACTER

1162 <1> K45A:

1163 000012C3 B91A000000 <1> mov ecx, 26 ; LENGHT OF SEARCH

1164 000012C8 BF[5A5D0000] <1> mov edi, K30+10 ; POINT TO TABLE OF A-Z CHARS

1165 000012CD F2AE <1> repne scasb ; IS THIS A LETTER KEY?

1166 <1> ; 20/02/2015

1167 000012CF 7505 <1> jne short K45B ; NO, SYMBOL KEY

1168 <1> ;

1169 000012D1 F6C340 <1> test bl, CAPS\_STATE ; ARE WE IN CAPS\_LOCK?

1170 000012D4 750C <1> jnz short K45D ; TEST FOR SURE

1171 <1> K45B:

1172 000012D6 F6C303 <1> test bl, LEFT\_SHIFT+RIGHT\_SHIFT ; ARE WE IN SHIFT STATE?

1173 000012D9 750C <1> jnz short K45E ; YES, UPPERCASE

1174 <1> ; NO, LOWERCASE

1175 <1> K45C:

1176 000012DB BB[DC5D0000] <1> mov ebx, K10 ; TRANSLATE TO LOWERCASE LETTERS

1177 000012E0 EB51 <1> jmp short K56

1178 <1> K45D: ; ALMOST-CAPS-STATE

1179 000012E2 F6C303 <1> test bl, LEFT\_SHIFT+RIGHT\_SHIFT ; CL ON. IS SHIFT ON, TOO?

1180 000012E5 75F4 <1> jnz short K45C ; SHIFTED TEMP OUT OF CAPS STATE

1181 <1> K45E:

1182 000012E7 BB[345E0000] <1> mov ebx, K11 ; TRANSLATE TO UPPER CASE LETTERS

1183 000012EC EB45 <1> K45F: jmp short K56

1184 <1> ;

1185 <1> ;----- TEST FOR KEYS F1 - F10

1186 <1> K46: ; NOT IN-CORE AREA

1187 000012EE 3C44 <1> cmp al, 68 ; TEST FOR F1 - F10

1188 <1> ;ja short K47 ; JUMP IF NOT

1189 <1> ;jmp short K53 ; YES, GO DO FN KEY PROCESS

1190 000012F0 7635 <1> jna short K53

1191 <1> ;

1192 <1> ;----- HANDLE THE NUMERIC PAD KEYS

1193 <1> K47: ; NOT F1 - F10

1194 000012F2 3C53 <1> cmp al, 83 ; TEST NUMPAD KEYS

1195 000012F4 772D <1> ja short K52 ; JUMP IF NOT

1196 <1> ;

1197 <1> ;----- KEYPAD KEYS, MUST TEST NUM LOCK FOR DETERMINATION

1198 <1> K48:

1199 000012F6 3C4A <1> cmp al , 74 ; SPECIAL CASE FOR MINUS

1200 000012F8 74ED <1> je short K45E ; GO TRANSLATE

1201 000012FA 3C4E <1> cmp al , 78 ; SPECIAL CASE FOR PLUS

1202 000012FC 74E9 <1> je short K45E ; GO TRANSLATE

1203 000012FE F6C702 <1> test bh, LC\_E0 ; IS THIS ONE OFTHE NEW KEYS?

1204 00001301 750A <1> jnz short K49 ; YES, TRANSLATE TO BASE STATE

1205 <1> ;

1206 00001303 F6C320 <1> test bl, NUM\_STATE ; ARE WE IN NUM LOCK

1207 00001306 7514 <1> jnz short K50 ; TEST FOR SURE

1208 00001308 F6C303 <1> test bl, LEFT\_SHIFT+RIGHT\_SHIFT ; ARE WE IN SHIFT STATE?

1209 <1> ;jnz short K51 ; IF SHIFTED, REALLY NUM STATE

1210 0000130B 75DA <1> jnz short K45E

1211 <1> ;

1212 <1> ;----- BASE CASE FOR KEYPAD

1213 <1> K49:

1214 0000130D 3C4C <1> cmp al, 76 ; SPECIAL CASE FOR BASE STATE 5

1215 0000130F 7504 <1> jne short K49A ; CONTINUE IF NOT KEYPAD 5

1216 00001311 B0F0 <1> mov al, 0F0h ; SPECIAL ASCII CODE

1217 00001313 EB40 <1> jmp short K57 ; BUFFER FILL

1218 <1> K49A:

1219 00001315 BB[DC5D0000] <1> mov ebx, K10 ; BASE CASE TABLE

1220 0000131A EB27 <1> jmp short K64 ; CONVERT TO PSEUDO SCAN

1221 <1> ;

1222 <1> ;----- MIGHT BE NUM LOCK, TEST SHIFT STATUS

1223 <1> K50: ; ALMOST-NUM-STATE

1224 0000131C F6C303 <1> test bl, LEFT\_SHIFT+RIGHT\_SHIFT

1225 0000131F 75EC <1> jnz short K49 ; SHIFTED TEMP OUT OF NUM STATE

1226 00001321 EBC4 <1> K51: jmp short K45E ; REALLY NUM STATE

1227 <1> ;

1228 <1> ;----- TEST FOR THE NEW KEYS ON WT KEYBOARDS

1229 <1> K52: ; NOT A NUMPAD KEY

1230 00001323 3C56 <1> cmp al, 86 ; IS IT THE NEW WT KEY?

1231 <1> ;jne short K53 ; JUMP IF NOT

1232 <1> ;jmp short K45B ; HANDLE WITH REST OF LETTER KEYS

1233 00001325 74AF <1> je short K45B

1234 <1> ;

1235 <1> ;----- MUST BE F11 OR F12

1236 <1> K53: ; F1 - F10 COME HERE, TOO

1237 00001327 F6C303 <1> test bl, LEFT\_SHIFT+RIGHT\_SHIFT ; TEST SHIFT STATE

1238 0000132A 74E1 <1> jz short K49 ; JUMP, LOWER CASE PSEUDO SC'S

1239 <1> ; 20/02/2015

1240 0000132C BB[345E0000] <1> mov ebx, K11 ; UPPER CASE PSEUDO SCAN CODES

1241 00001331 EB10 <1> jmp short K64 ; TRANSLATE SCAN

1242 <1> ;

1243 <1> ;----- TRANSLATE THE CHARACTER

1244 <1> K56: ; TRANSLATE-CHAR

1245 00001333 FEC8 <1> dec al ; CONVERT ORIGIN

1246 00001335 D7 <1> xlat ; CONVERT THE SCAN CODE TO ASCII

1247 00001336 F605[905E0000]02 <1> test byte [KB\_FLAG\_3], LC\_E0 ; IS THIS A NEW KEY?

1248 0000133D 7416 <1> jz short K57 ; NO, GO FILL BUFFER

1249 0000133F B4E0 <1> mov ah, MC\_E0 ; YES, PUT SPECIAL MARKER IN AH

1250 00001341 EB12 <1> jmp short K57 ; PUT IT INTO THE BUFFER

1251 <1> ;

1252 <1> ;----- TRANSLATE SCAN FOR PSEUDO SCAN CODES

1253 <1> K64: ; TRANSLATE-SCAN-ORGD

1254 00001343 FEC8 <1> dec al ; CONVERT ORIGIN

1255 00001345 D7 <1> xlat ; CTL TABLE SCAN

1256 00001346 88C4 <1> mov ah, al ; PUT VALUE INTO AH

1257 00001348 B000 <1> mov al, 0 ; ZERO ASCII CODE

1258 0000134A F605[905E0000]02 <1> test byte [KB\_FLAG\_3], LC\_E0 ; IS THIS A NEW KEY?

1259 00001351 7402 <1> jz short K57 ; NO, GO FILL BUFFER

1260 00001353 B0E0 <1> mov al, MC\_E0 ; YES, PUT SPECIAL MARKER IN AL

1261 <1> ;

1262 <1> ;----- PUT CHARACTER INTO BUFFER

1263 <1> K57: ; BUFFER\_FILL

1264 00001355 3CFF <1> cmp al, -1 ; IS THIS AN IGNORE CHAR

1265 <1> ;je short K59 ; YES, DO NOTHING WITH IT

1266 00001357 0F8459FDFFFF <1> je K26 ; YES, DO NOTHING WITH IT

1267 0000135D 80FCFF <1> cmp ah, -1 ; LOOK FOR -1 PSEUDO SCAN

1268 <1> ;jne short K61 ; NEAR\_INTERRUPT\_RETURN

1269 00001360 0F8450FDFFFF <1> je K26 ; INTERRUPT\_RETURN

1270 <1> ;K59: ; NEAR\_INTERRUPT\_RETURN

1271 <1> ; jmp K26 ; INTERRUPT\_RETURN

1272 <1>

1273 <1> \_K60: ; 29/01/2016

1274 00001366 80FC68 <1> cmp ah, 68h ; ALT + F1 key

1275 00001369 721F <1> jb short K61

1276 0000136B 80FC6F <1> cmp ah, 6Fh ; ALT + F8 key

1277 0000136E 771A <1> ja short K61

1278 <1> ;

1279 00001370 8A1D[66580100] <1> mov bl, [ACTIVE\_PAGE]

1280 00001376 80C368 <1> add bl, 68h

1281 00001379 38E3 <1> cmp bl, ah

1282 0000137B 740D <1> je short K61

1283 0000137D 6650 <1> push ax

1284 0000137F 88E0 <1> mov al, ah

1285 00001381 2C68 <1> sub al, 68h

1286 00001383 E8F4050000 <1> call set\_active\_page

1287 00001388 6658 <1> pop ax

1288 <1> K61: ; NOT-CAPS-STATE

1289 0000138A 8B1D[9E5E0000] <1> mov ebx, [BUFFER\_TAIL] ; GET THE END POINTER TO THE BUFFER

1290 00001390 89DE <1> mov esi, ebx ; SAVE THE VALUE

1291 00001392 E857FAFFFF <1> call \_K4 ; ADVANCE THE TAIL

1292 00001397 3B1D[9A5E0000] <1> cmp ebx, [BUFFER\_HEAD] ; HAS THE BUFFER WRAPPED AROUND

1293 0000139D 740E <1> je short K62 ; BUFFER\_FULL\_BEEP

1294 0000139F 668906 <1> mov [esi], ax ; STORE THE VALUE

1295 000013A2 891D[9E5E0000] <1> mov [BUFFER\_TAIL], ebx ; MOVE THE POINTER UP

1296 000013A8 E909FDFFFF <1> jmp K26

1297 <1> ;;cli ; TURN OFF INTERRUPTS

1298 <1> ;;mov al, EOI ; END OF INTERRUPT COMMAND

1299 <1> ;;out INTA00, al ; SEND COMMAND TO INTERRUPT CONTROL PORT

1300 <1> ;MOV AL, ENA\_KBD ; INSURE KEYBOARD IS ENABLED

1301 <1> ;CALL SHIP\_IT ; EXECUTE ENABLE

1302 <1> ;MOV AX, 9102H ; MOVE IN POST CODE & TYPE

1303 <1> ;INT 15H ; PERFORM OTHER FUNCTION

1304 <1> ;;and byte [KB\_FLAG\_3],~(LC\_E0+LC\_E1) ; RESET LAST CHAR H.C. FLAG

1305 <1> ;JMP K27A ; INTERRUPT\_RETURN

1306 <1> ;;jmp K27

1307 <1> ;

1308 <1> ;----- BUFFER IS FULL SOUND THE BEEPER

1309 <1> K62:

1310 000013AD B020 <1> mov al, EOI ; ENABLE INTERRUPT CONTROLLER CHIP

1311 000013AF E620 <1> out INTA00, al

1312 000013B1 66B9A602 <1> mov cx, 678 ; DIVISOR FOR 1760 HZ

1313 000013B5 B304 <1> mov bl, 4 ; SHORT BEEP COUNT (1/16 + 1/64 DELAY)

1314 000013B7 E8E5090000 <1> call beep ; GO TO COMMON BEEP HANDLER

1315 000013BC E901FDFFFF <1> jmp K27 ; EXIT

1316 <1>

1317 <1> SHIP\_IT:

1318 <1> ;---------------------------------------------------------------------------------

1319 <1> ; SHIP\_IT

1320 <1> ; THIS ROUTINES HANDLES TRANSMISSION OF COMMAND AND DATA BYTES

1321 <1> ; TO THE KEYBOARD CONTROLLER.

1322 <1> ;---------------------------------------------------------------------------------

1323 <1> ;

1324 000013C1 6650 <1> push ax ; SAVE DATA TO SEND

1325 <1>

1326 <1> ;----- WAIT FOR COMMAND TO ACCEPTED

1327 000013C3 FA <1> cli ; DISABLE INTERRUPTS TILL DATA SENT

1328 <1> ; xor ecx, ecx ; CLEAR TIMEOUT COUNTER

1329 000013C4 B900000100 <1> mov ecx, 10000h

1330 <1> S10:

1331 000013C9 E464 <1> in al, STATUS\_PORT ; READ KEYBOARD CONTROLLER STATUS

1332 000013CB A802 <1> test al, INPT\_BUF\_FULL ; CHECK FOR ITS INPUT BUFFER BUSY

1333 000013CD E0FA <1> loopnz S10 ; WAIT FOR COMMAND TO BE ACCEPTED

1334 <1>

1335 000013CF 6658 <1> pop ax ; GET DATA TO SEND

1336 000013D1 E664 <1> out STATUS\_PORT, al ; SEND TO KEYBOARD CONTROLLER

1337 000013D3 FB <1> sti ; ENABLE INTERRUPTS AGAIN

1338 000013D4 C3 <1> retn ; RETURN TO CALLER

1339 <1>

1340 <1> SND\_DATA:

1341 <1> ; ---------------------------------------------------------------------------------

1342 <1> ; SND\_DATA

1343 <1> ; THIS ROUTINES HANDLES TRANSMISSION OF COMMAND AND DATA BYTES

1344 <1> ; TO THE KEYBOARD AND RECEIPT OF ACKNOWLEDGEMENTS. IT ALSO

1345 <1> ; HANDLES ANY RETRIES IF REQUIRED

1346 <1> ; ---------------------------------------------------------------------------------

1347 <1> ;

1348 000013D5 6650 <1> push ax ; SAVE REGISTERS

1349 000013D7 6653 <1> push bx

1350 000013D9 51 <1> push ecx

1351 000013DA 88C7 <1> mov bh, al ; SAVE TRANSMITTED BYTE FOR RETRIES

1352 000013DC B303 <1> mov bl, 3 ; LOAD RETRY COUNT

1353 <1> SD0:

1354 000013DE FA <1> cli ; DISABLE INTERRUPTS

1355 000013DF 8025[8F5E0000]CF <1> and byte [KB\_FLAG\_2], ~(KB\_FE+KB\_FA) ; CLEAR ACK AND RESEND FLAGS

1356 <1> ;

1357 <1> ;----- WAIT FOR COMMAND TO BE ACCEPTED

1358 000013E6 B900000100 <1> mov ecx, 10000h ; MAXIMUM WAIT COUNT

1359 <1> SD5:

1360 000013EB E464 <1> in al, STATUS\_PORT ; READ KEYBOARD PROCESSOR STATUS PORT

1361 000013ED A802 <1> test al, INPT\_BUF\_FULL ; CHECK FOR ANY PENDING COMMAND

1362 000013EF E0FA <1> loopnz SD5 ; WAIT FOR COMMAND TO BE ACCEPTED

1363 <1> ;

1364 000013F1 88F8 <1> mov al, bh ; REESTABLISH BYTE TO TRANSMIT

1365 000013F3 E660 <1> out PORT\_A, al ; SEND BYTE

1366 000013F5 FB <1> sti ; ENABLE INTERRUPTS

1367 <1> ;mov cx, 01A00h ; LOAD COUNT FOR 10 ms+

1368 000013F6 B9FFFF0000 <1> mov ecx, 0FFFFh

1369 <1> SD1:

1370 000013FB F605[8F5E0000]30 <1> test byte [KB\_FLAG\_2], KB\_FE+KB\_FA ; SEE IF EITHER BIT SET

1371 00001402 750F <1> jnz short SD3 ; IF SET, SOMETHING RECEIVED GO PROCESS

1372 00001404 E2F5 <1> loop SD1 ; OTHERWISE WAIT

1373 <1> SD2:

1374 00001406 FECB <1> dec bl ; DECREMENT RETRY COUNT

1375 00001408 75D4 <1> jnz short SD0 ; RETRY TRANSMISSION

1376 0000140A 800D[8F5E0000]80 <1> or byte [KB\_FLAG\_2], KB\_ERR ; TURN ON TRANSMIT ERROR FLAG

1377 00001411 EB09 <1> jmp short SD4 ; RETRIES EXHAUSTED FORGET TRANSMISSION

1378 <1> SD3:

1379 00001413 F605[8F5E0000]10 <1> test byte [KB\_FLAG\_2], KB\_FA ; SEE IF THIS IS AN ACKNOWLEDGE

1380 0000141A 74EA <1> jz short SD2 ; IF NOT, GO RESEND

1381 <1> SD4:

1382 0000141C 59 <1> pop ecx ; RESTORE REGISTERS

1383 0000141D 665B <1> pop bx

1384 0000141F 6658 <1> pop ax

1385 00001421 C3 <1> retn ; RETURN, GOOD TRANSMISSION

1386 <1>

1387 <1> SND\_LED:

1388 <1> ; ---------------------------------------------------------------------------------

1389 <1> ; SND\_LED

1390 <1> ; THIS ROUTINES TURNS ON THE MODE INDICATORS.

1391 <1> ;

1392 <1> ;----------------------------------------------------------------------------------

1393 <1> ;

1394 00001422 FA <1> cli ; TURN OFF INTERRUPTS

1395 00001423 F605[8F5E0000]40 <1> test byte [KB\_FLAG\_2], KB\_PR\_LED ; CHECK FOR MODE INDICATOR UPDATE

1396 0000142A 755F <1> jnz short SL1 ; DON'T UPDATE AGAIN IF UPDATE UNDERWAY

1397 <1> ;

1398 0000142C 800D[8F5E0000]40 <1> or byte [KB\_FLAG\_2], KB\_PR\_LED ; TURN ON UPDATE IN PROCESS

1399 00001433 B020 <1> mov al, EOI ; END OF INTERRUPT COMMAND

1400 00001435 E620 <1> out 20h, al ;out INTA00, al ; SEND COMMAND TO INTERRUPT CONTROL PORT

1401 00001437 EB11 <1> jmp short SL0 ; GO SEND MODE INDICATOR COMMAND

1402 <1> SND\_LED1:

1403 00001439 FA <1> cli ; TURN OFF INTERRUPTS

1404 0000143A F605[8F5E0000]40 <1> test byte [KB\_FLAG\_2], KB\_PR\_LED ; CHECK FOR MODE INDICATOR UPDATE

1405 00001441 7548 <1> jnz short SL1 ; DON'T UPDATE AGAIN IF UPDATE UNDERWAY

1406 <1> ;

1407 00001443 800D[8F5E0000]40 <1> or byte [KB\_FLAG\_2], KB\_PR\_LED ; TURN ON UPDATE IN PROCESS

1408 <1> SL0:

1409 0000144A B0ED <1> mov al, LED\_CMD ; LED CMD BYTE

1410 0000144C E884FFFFFF <1> call SND\_DATA ; SEND DATA TO KEYBOARD

1411 00001451 FA <1> cli

1412 00001452 E836000000 <1> call MAKE\_LED ; GO FORM INDICATOR DATA BYTE

1413 00001457 8025[8F5E0000]F8 <1> and byte [KB\_FLAG\_2], 0F8h ; ~KB\_LEDS ; CLEAR MODE INDICATOR BITS

1414 0000145E 0805[8F5E0000] <1> or [KB\_FLAG\_2], al ; SAVE PRESENT INDICATORS FOR NEXT TIME

1415 00001464 F605[8F5E0000]80 <1> test byte [KB\_FLAG\_2], KB\_ERR ; TRANSMIT ERROR DETECTED

1416 0000146B 750F <1> jnz short SL2 ; IF SO, BYPASS SECOND BYTE TRANSMISSION

1417 <1> ;

1418 0000146D E863FFFFFF <1> call SND\_DATA ; SEND DATA TO KEYBOARD

1419 00001472 FA <1> cli ; TURN OFF INTERRUPTS

1420 00001473 F605[8F5E0000]80 <1> test byte [KB\_FLAG\_2], KB\_ERR ; TRANSMIT ERROR DETECTED

1421 0000147A 7408 <1> jz short SL3 ; IF NOT, DON'T SEND AN ENABLE COMMAND

1422 <1> SL2:

1423 0000147C B0F4 <1> mov al, KB\_ENABLE ; GET KEYBOARD CSA ENABLE COMMAND

1424 0000147E E852FFFFFF <1> call SND\_DATA ; SEND DATA TO KEYBOARD

1425 00001483 FA <1> cli ; TURN OFF INTERRUPTS

1426 <1> SL3:

1427 00001484 8025[8F5E0000]3F <1> and byte [KB\_FLAG\_2], ~(KB\_PR\_LED+KB\_ERR) ; TURN OFF MODE INDICATOR

1428 <1> SL1: ; UPDATE AND TRANSMIT ERROR FLAG

1429 0000148B FB <1> sti ; ENABLE INTERRUPTS

1430 0000148C C3 <1> retn ; RETURN TO CALLER

1431 <1>

1432 <1> MAKE\_LED:

1433 <1> ;---------------------------------------------------------------------------------

1434 <1> ; MAKE\_LED

1435 <1> ; THIS ROUTINES FORMS THE DATA BYTE NECESSARY TO TURN ON/OFF

1436 <1> ; THE MODE INDICATORS.

1437 <1> ;---------------------------------------------------------------------------------

1438 <1> ;

1439 <1> ;push cx ; SAVE CX

1440 0000148D A0[8D5E0000] <1> mov al, [KB\_FLAG] ; GET CAPS & NUM LOCK INDICATORS

1441 00001492 2470 <1> and al, CAPS\_STATE+NUM\_STATE+SCROLL\_STATE ; ISOLATE INDICATORS

1442 <1> ;mov cl, 4 ; SHIFT COUNT

1443 <1> ;rol al, cl ; SHIFT BITS OVER TO TURN ON INDICATORS

1444 00001494 C0C004 <1> rol al, 4 ; 20/02/2015

1445 00001497 2407 <1> and al, 07h ; MAKE SURE ONLY MODE BITS ON

1446 <1> ;pop cx

1447 00001499 C3 <1> retn ; RETURN TO CALLER

1448 <1>

1449 <1> ; % include 'kybdata.s' ; KEYBOARD DATA

1450 <1>

1451 <1>

1452 <1> ; /// End Of KEYBOARD FUNCTIONS ///

1940

1941 %include 'video.s' ; 07/03/2015

1 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2 <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0 - video.s

3 <1> ; ----------------------------------------------------------------------------

4 <1> ; Last Update: 09/12/2017

5 <1> ; ----------------------------------------------------------------------------

6 <1> ; Beginning: 16/01/2016

7 <1> ; ----------------------------------------------------------------------------

8 <1> ; Assembler: NASM version 2.11 (trdos386.s)

9 <1> ; ----------------------------------------------------------------------------

10 <1> ; Turkish Rational DOS

11 <1> ; Operating System Project v2.0 by ERDOGAN TAN (Beginning: 04/01/2016)

12 <1> ;

13 <1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan

14 <1> ; video.inc (13/08/2015)

15 <1> ;

16 <1> ; Derived from 'IBM PC-AT' BIOS source code (1985)

17 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

18 <1>

19 <1> ; Retro UNIX 386 v1 Kernel - VIDEO.INC

20 <1> ; Last Modification: 13/08/2015

21 <1> ; (Video Data is in 'VIDATA.INC')

22 <1> ;

23 <1> ; ///////// VIDEO (CGA) FUNCTIONS ///////////////

24 <1>

25 <1> ; 16/01/2016 (32 bit modifications, TRDOS386 - TRDOS v2.0, video.s)

26 <1> ; INT 31H (TRDOS 386) = INT 10H (IBM PC/AT REAL MODE)

27 <1>

28 <1> ; IBM PC-AT BIOS Source Code

29 <1> ; TITLE VIDEO1 --- 06/10/85 VIDEO DISPLAY BIOS

30 <1>

31 <1> \_int10h:

32 <1> ; 23/03/2016

33 <1> ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)

34 0000149A 9C <1> pushfd

35 0000149B 0E <1> push cs

36 0000149C E851000000 <1> call VIDEO\_IO\_1

37 000014A1 C3 <1> retn

38 <1>

39 <1> ;--- INT 10 H -------------------------------------------------------------------

40 <1> ; VIDEO\_IO :

41 <1> ; THESE ROUTINES PROVIDE THE CRT DISPLAY INTERFACE :

42 <1> ; THE FOLLOWING FUNCTIONS ARE PROVIDED: :

43 <1> ; :

44 <1> ; (AH)= 00H SET MODE (AL) CONTAINS MODE VALUE :

45 <1> ; (AL) = 00H 40X25 BW MODE (POWER ON DEFAULT) :

46 <1> ; (AL) = 01H 40X25 COLOR :

47 <1> ; (AL) = 02H 80X25 BW :

48 <1> ; (AL) = 03H 80X25 COLOR :

49 <1> ; GRAPHICS MODES :

50 <1> ; (AL) = 04H 320X200 COLOR :

51 <1> ; (AL) = 05H 320X200 BW MODE :

52 <1> ; (AL) = 06H 640X200 BW MODE :

53 <1> ; (AL) = 07H 80X25 MONOCHROME (USED INTERNAL TO VIDEO ONLY) :

54 <1> ; \*\*\* NOTES -BW MODES OPERATE SAME AS COLOR MODES, BUT COLOR :

55 <1> ; BURST IS NOT ENABLED :

56 <1> ; -CURSOR IS NOT DISPLAYED IN GRAPHICS MODE :

57 <1> ; (AH)= 01H SET CURSOR TYPE :

58 <1> ; (CH) = BITS 4-0 = START LINE FOR CURSOR :

59 <1> ; \*\* HARDWARE WILL ALWAYS CAUSE BLINK :

60 <1> ; \*\* SETTING BIT 5 OR 6 WILL CAUSE ERRATIC BLINKING :

61 <1> ; OR NO CURSOR AT ALL :

62 <1> ; (CL) = BITS 4-0 = END LINE FOR CURSOR :

63 <1> ; (AH)= 02H SET CURSOR POSITION :

64 <1> ; (DH,DL) = ROW,COLUMN (00H,00H) IS UPPER LEFT :

65 <1> ; (BH) = A PAGE NUMBER (MUST BE 00H FOR GRAPHICS MODES) :

66 <1> ; (AH)= 03H READ CURSOR POSITION :

67 <1> ; (BH) = PAGE NUMBER (MUST BE 00H FOR GRAPHICS MODES) :

68 <1> ; ON EXIT (DH,DL) = ROW,COLUMN OF CURRENT CURSOR :

69 <1> ; (CH,CL) = CURSOR MODE CURRENTLY SET :

70 <1> ; (AH)= 04H READ LIGHT PEN POSITION :

71 <1> ; ON EXIT: :

72 <1> ; (AH) = 00H -- LIGHT PEN SWITCH NOT DOWN/NOT TRIGGERED :

73 <1> ; (AH) = 01H -- VALID LIGHT PEN VALUE IN REGISTERS :

74 <1> ; (DH,DL) = ROW,COLUMN OF CHARACTER LP POSITION :

75 <1> ; (CH) = RASTER LINE (0-199) :

76 <1> ; (BX) = PIXEL COLUMN (0-319,639) :

77 <1> ; (AH)= 05H SELECT ACTIVE DISPLAY PAGE (VALID ONLY FOR ALPHA MODES) :

78 <1> ; (AL) = NEW PAGE VALUE (0-7 FOR MODES 0&1, 0-3 FOR MODES 2&3) :

79 <1> ; (AH)= 06H SCROLL ACTIVE PAGE UP :

80 <1> ; (AL) = NUMBER OF LINES. ( LINES BLANKED AT BOTTOM OF WINDOW ) :

81 <1> ; (AL) = 00H MEANS BLANK ENTIRE WINDOW :

82 <1> ; (CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF SCROLL :

83 <1> ; (DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF SCROLL :

84 <1> ; (BH) = ATTRIBUTE TO BE USED ON BLANK LINE :

85 <1> ; (AH)= 07H SCROLL ACTIVE PAGE DOWN :

86 <1> ; (AL) = NUMBER OF LINES, INPUT LINES BLANKED AT TOP OF WINDOW :

87 <1> ; (AL) = 00H MEANS BLANK ENTIRE WINDOW :

88 <1> ; (CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF SCROLL :

89 <1> ; (DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF SCROLL :

90 <1> ; (BH) = ATTRIBUTE TO BE USED ON BLANK LINE :

91 <1> ; :

92 <1> ; CHARACTER HANDLING ROUTINES :

93 <1> ; :

94 <1> ; (AH)= 08H READ ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION :

95 <1> ; (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY) :

96 <1> ; ON EXIT: :

97 <1> ; (AL) = CHAR READ :

98 <1> ; (AH) = ATTRIBUTE OF CHARACTER READ (ALPHA MODES ONLY) :

99 <1> ; (AH)= 09H WRITE ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION :

100 <1> ; (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY) :

101 <1> ; (CX) = COUNT OF CHARACTERS TO WRITE :

102 <1> ; (AL) = CHAR TO WRITE :

103 <1> ; (BL) = ATTRIBUTE OF CHARACTER (ALPHA)/COLOR OF CHAR (GRAPHICS) :

104 <1> ; SEE NOTE ON WRITE DOT FOR BIT 7 OF BL = 1. :

105 <1> ; (AH) = 0AH WRITE CHARACTER ONLY AT CURRENT CURSOR POSITION :

106 <1> ; (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY) :

107 <1> ; (CX) = COUNT OF CHARACTERS TO WRITE :

108 <1> ; (AL) = CHAR TO WRITE :

109 <1> ; NOTE: USE FUNCTION (AH)= 09H IN GRAPHICS MODES :

110 <1> ; FOR READ/WRITE CHARACTER INTERFACE WHILE IN GRAPHICS MODE, THE :

111 <1> ; CHARACTERS ARE FORMED FROM A CHARACTER GENERATOR IMAGE :

112 <1> ; MAINTAINED IN THE SYSTEM ROM. ONLY THE 1ST 128 CHARS :

113 <1> ; ARE CONTAINED THERE. TO READ/WRITE THE SECOND 128 CHARS, :

114 <1> ; THE USER MUST INITIALIZE THE POINTER AT INTERRUPT 1FH :

115 <1> ; (LOCATION 0007CH) TO POINT TO THE 1K BYTE TABLE CONTAINING :

116 <1> ; THE CODE POINTS FOR THE SECOND 128 CHARS (128-255). :

117 <1> ; FOR WRITE CHARACTER INTERFACE IN GRAPHICS MODE, THE REPLICATION FACTOR :

118 <1> ; CONTAINED IN (CX) ON ENTRY WILL PRODUCE VALID RESULTS ONLY :

119 <1> ; FOR CHARACTERS CONTAINED ON THE SAME ROW. CONTINUATION TO :

120 <1> ; SUCCEEDING LINES WILL NOT PRODUCE CORRECTLY. :

121 <1> ; :

122 <1> ; GRAPHICS INTERFACE :

123 <1> ; (AH)= 0BH SET COLOR PALETTE :

124 <1> ; (BH) = PALETTE COLOR ID BEING SET (0-127) :

125 <1> ; (BL) = COLOR VALUE TO BE USED WITH THAT COLOR ID :

126 <1> ; NOTE: FOR THE CURRENT COLOR CARD, THIS ENTRY POINT HAS :

127 <1> ; MEANING ONLY FOR 320X200 GRAPHICS. :

128 <1> ; COLOR ID = 0 SELECTS THE BACKGROUND COLOR (0-15) :

129 <1> ; COLOR ID = 1 SELECTS THE PALETTE TO BE USED: :

130 <1> ; 0 = GREEN(1)/RED(2)/YELLOW(3) :

131 <1> ; 1 = CYAN(1)/MAGENTA(2)/WHITE(3) :

132 <1> ; IN 40X25 OR 80X25 ALPHA MODES, THE VALUE SET FOR :

133 <1> ; PALETTE COLOR 0 INDICATES THE BORDER COLOR :

134 <1> ; TO BE USED (VALUES 0-31, WHERE 16-31 SELECT :

135 <1> ; THE HIGH INTENSITY BACKGROUND SET. :

136 <1> ; (AH)= 0CH WRITE DOT :

137 <1> ; (DX) = ROW NUMBER :

138 <1> ; (CX) = COLUMN NUMBER :

139 <1> ; (AL) = COLOR VALUE :

140 <1> ; IF BIT 7 OF AL = 1, THEN THE COLOR VALUE IS EXCLUSIVE :

141 <1> ; ORed WITH THE CURRENT CONTENTS OF THE DOT :

142 <1> ; (AH)= ODH READ DOT :

143 <1> ; (DX) = ROW NUMBER :

144 <1> ; (CX) = COLUMN NUMBER :

145 <1> ; (AL) = RETURNS THE DOT READ :

146 <1> ; :

147 <1> ; ASCII TELETYPE ROUTINE FOR OUTPUT :

148 <1> ; :

149 <1> ; (AH)= 0EH WRITE TELETYPE TO ACTIVE PAGE :

150 <1> ; (AL) = CHAR TO WRITE :

151 <1> ; (BL) = FOREGROUND COLOR IN GRAPHICS MODE :

152 <1> ; NOTE -- SCREEN WIDTH IS CONTROLLED BY PREVIOUS MODE SET :

153 <1> ; (AH)= 0FH CURRENT VIDEO STATE :

154 <1> ; RETURNS THE CURRENT VIDEO STATE :

155 <1> ; (AL) = MODE CURRENTLY SET ( SEE (AH)=00H FOR EXPLANATION) :

156 <1> ; (AH) = NUMBER OR CHARACTER COLUMNS ON SCREEN :

157 <1> ; (BH) = CURRENT ACTIVE DISPLAY PAGE :

158 <1> ; (AH)= 10H RESERVED :

159 <1> ; (AH)= 11H RESERVED :

160 <1> ; (AH)= 12H RESERVED :

161 <1> ; (AH)= 13H WRITE STRING :

162 <1> ; ES:BP - POINTER T0 STRING TO BE WRITTEN :

163 <1> ; CX - LENGTH OF CHARACTER STRING TO WRITTEN :

164 <1> ; DX - CURSOR POSITION FOR STRING TO BE WRITTEN :

165 <1> ; BH - PAGE NUMBER :

166 <1> ; (AL)= 00H WRITE CHARACTER STRING :

167 <1> ; BL - ATTRIBUTE :

168 <1> ; STRING IS <CHAR,CHAR, ... ,CHAR> :

169 <1> ; CURSOR NOT MOVED :

170 <1> ; (AL)= 01H WRITE CHARACTER STRING AND MOVE CURSOR :

171 <1> ; BL - ATTRIBUTE :

172 <1> ; STRING IS <CHAR,CHAR, ... ,CHAR> :

173 <1> ; CURSOR MOVED :

174 <1> ; (AL)= 02H WRITE CHARACTER AND ATTRIBUTE STRING :

175 <1> ; (VALID FOR ALPHA MODES ONLY) :

176 <1> ; STRING IS <CHAR,ATTR,CHAR,ATTR .. ,CHAR,ATTR> :

177 <1> ; CURSOR IS NOT MOVED :

178 <1> ; (AL)= 03H WRITE CHARACTER AND ATTRIBUTE STRING AND MOVE CURSOR :

179 <1> ; (VALID FOR ALPHA MODES ONLY) :

180 <1> ; STRING IS <CHAR,ATTR,CHAR,ATTR .. ,CHAR,ATTR> :

181 <1> ; CURSOR IS MOVED :

182 <1> ; NOTE: CARRIAGE RETURN, LINE FEED, BACKSPACE, AND BELL ARE :

183 <1> ; TREATED AS COMMANDS RATHER THAN PRINTABLE CHARACTERS. :

184 <1> ; :

185 <1> ; BX,CX,DX,SI,DI,BP,SP,DS,ES,SS PRESERVED DURING CALLS EXCEPT FOR :

186 <1> ; BX,CX,DX RETURN VALUES ON FUNCTIONS 03H,04H,0DH AND 0FH. ON ALL CALLS :

187 <1> ; AX IS MODIFIED. :

188 <1> ;--------------------------------------------------------------------------------

189 <1>

190 000014A2 [4F150000] <1> M1: dd SET\_MODE ; TABLE OF ROUTINES WITHIN VIDEO I/O

191 000014A6 [B7180000] <1> dd SET\_CTYPE

192 000014AA [EB180000] <1> dd SET\_CPOS

193 000014AE [13190000] <1> dd READ\_CURSOR

194 <1> ;dd VIDEO\_RETURN ; READ\_LPEN

195 000014B2 [38150000] <1> dd set\_mode\_ncm ; Set mode without clearing video memory

196 000014B6 [59190000] <1> dd ACT\_DISP\_PAGE

197 000014BA [F0190000] <1> dd SCROLL\_UP

198 000014BE [141B0000] <1> dd SCROLL\_DOWN

199 000014C2 [951B0000] <1> dd READ\_AC\_CURRENT

200 000014C6 [ED1B0000] <1> dd WRITE\_AC\_CURRENT

201 000014CA [131C0000] <1> dd WRITE\_C\_CURRENT

202 000014CE [39250000] <1> dd SET\_COLOR

203 000014D2 [A4250000] <1> dd WRITE\_DOT

204 000014D6 [6F250000] <1> dd READ\_DOT

205 000014DA [951C0000] <1> dd WRITE\_TTY

206 000014DE [20150000] <1> dd VIDEO\_STATE

207 000014E2 [EF2E0000] <1> dd vga\_pal\_funcs ; 10/08/2016 (TRDOS 386)

208 000014E6 [A52A0000] <1> dd font\_setup ; 10/07/2016 (TRDOS 386)

209 000014EA [54150000] <1> dd VIDEO\_RETURN ; RESERVED

210 000014EE [021E0000] <1> dd WRITE\_STRING ; 23/06/2016 (TRDOS 386)

211 <1> M1L EQU $ - M1

212 <1>

213 <1> ; 14/01/2017

214 <1> ; 02/01/2017

215 <1> ; 04/07/2016

216 <1> ; 12/05/2016

217 <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)

218 <1> int31h: ; Video BIOS

219 <1>

220 <1> ; BH = Video page number

221 <1> ; BL = Color/Attribute

222 <1> ; AH = Function number

223 <1> ; AL = Character

224 <1>

225 <1> VIDEO\_IO\_1:

226 <1> ;sti ; INTERRUPTS BACK ON

227 000014F2 FC <1> cld ; SET DIRECTION FORWARD

228 000014F3 80FC14 <1> cmp ah, M1L/4 ; TEST FOR WITHIN TABLE RANGE

229 000014F6 7327 <1> jnb short M4 ; BRANCH TO EXIT IF NOT A VALID COMMAND

230 <1>

231 000014F8 06 <1> push es

232 000014F9 1E <1> push ds ; SAVE WORK AND PARAMETER REGISTERS

233 000014FA 52 <1> push edx

234 000014FB 51 <1> push ecx

235 000014FC 53 <1> push ebx

236 000014FD 56 <1> push esi

237 000014FE 57 <1> push edi

238 000014FF 55 <1> push ebp

239 <1>

240 00001500 66BE1000 <1> mov si, KDATA ; POINT DS: TO DATA SEGMENT

241 00001504 8EDE <1> mov ds, si

242 00001506 8EC6 <1> mov es, si

243 00001508 BF00800B00 <1> mov edi, 0B8000h ; GET offset FOR COLOR CARD

244 0000150D A3[C4650100] <1> mov [video\_eax], eax ; 12/05/2016

245 <1> ; 23/03/2016

246 00001512 C0E402 <1> shl ah, 2 ; dword ; TIMES 2 FOR WORD TABLE LOOKUP

247 00001515 0FB6F4 <1> movzx esi, ah ; MOVE OFFSET INTO LOOK UP REGISTER (SI)

248 <1> ;mov ah, [CRT\_MODE] ; MOVE CURRENT MODE INTO (AH) REGISTER

249 <1>

250 <1> ;;15/01/2017

251 <1> ; 14/01/2017

252 <1> ; 02/01/2017

253 <1> ;;mov byte [intflg], 31h ; video interrupt

254 00001518 FB <1> sti

255 <1> ;

256 <1>

257 00001519 FFA6[A2140000] <1> JMP dword [esi+M1] ; GO TO SELECTED FUNCTION

258 <1>

259 <1> M4: ; COMMAND NOT VALID

260 0000151F CF <1> iretd ; DO NOTHING IF NOT IN VALID RANGE

261 <1>

262 <1> VIDEO\_STATE:

263 <1> ; 26/06/2016

264 <1> ; 12/05/2016

265 <1> ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)

266 <1>

267 <1> ;---------------------------------------------------

268 <1> ; VIDEO STATE

269 <1> ; RETURNS THE CURRENT VIDEO STATE IN AX

270 <1> ; AH = NUMBER OF COLUMNS ON THE SCREEN

271 <1> ; AL = CURRENT VIDEO MODE

272 <1> ; BH = CURRENT ACTIVE PAGE

273 <1> ;---------------------------------------------------

274 <1>

275 00001520 8A25[C45E0000] <1> mov ah, [CRT\_COLS] ; GET NUMBER OF COLUMNS

276 00001526 A0[C25E0000] <1> mov al, [CRT\_MODE] ; CURRENT MODE

277 <1> ;movzx esi, al

278 <1> ;mov ah, [esi+M6]

279 <1> ; BH = active page

280 0000152B 8A3D[66580100] <1> mov bh, [ACTIVE\_PAGE] ; GET CURRENT ACTIVE PAGE

281 00001531 FA <1> cli ; 02/01/2017

282 00001532 5D <1> pop ebp ; RECOVER REGISTERS

283 00001533 5F <1> pop edi

284 00001534 5E <1> pop esi

285 00001535 59 <1> pop ecx ; DISCARD SAVED BX

286 00001536 EB26 <1> jmp short M15 ; RETURN TO CALLER

287 <1>

288 <1> set\_mode\_ncm:

289 <1> ; 04/07/2016 - TRDOS 386 (TRDOS v2.0)

290 <1> ; set mode without clearing the video memory

291 <1> ; (ony for graphics modes)

292 00001538 3C07 <1> cmp al, 7 ; IBM PC CGA modes

293 0000153A 7613 <1> jna short SET\_MODE ; normal function (clear)

294 <1> ; do not clear memory

295 0000153C A2[D3650100] <1> mov [noclearmem], al ; > 0

296 00001541 E81F000000 <1> call \_set\_mode

297 00001546 C605[D3650100]00 <1> mov byte [noclearmem], 0

298 0000154D EB05 <1> jmp short VIDEO\_RETURN

299 <1>

300 <1> ; 10/08/2016

301 <1> ; 08/08/2016

302 <1> ; 30/07/2016

303 <1> ; 29/07/2016

304 <1> ; 27/07/2016

305 <1> ; 26/07/2016

306 <1> ; 25/07/2016

307 <1> ; 23/07/2016

308 <1> ; 18/07/2016

309 <1> ; 02/07/2016

310 <1> ; 26/06/2016

311 <1> ; 24/06/2016

312 <1> ; 29/05/2016 - TRDOS 386 (TRDOS v2.0)

313 <1> SET\_MODE:

314 <1> ; For 32 bit TRDOS and Retro UNIX 386:

315 <1> ; valid video mode: 03h only!

316 <1> ; (VGA modes will be selected with another routine)

317 <1> ;

318 <1> ; set\_txt\_mode ; 80\*25 (16 fore colors, 8 back colors)

319 <1>

320 <1> ;------------------------------------------------------

321 <1> ; SET MODE :

322 <1> ; THIS ROUTINE INITIALIZES THE ATTACHMENT TO :

323 <1> ; THE SELECTED MODE, THE SCREEN IS BLANKED. :

324 <1> ; INPUT :

325 <1> ; (AL) - MODE SELECTED (RANGE 0-7) :

326 <1> ; OUTPUT :

327 <1> ; NONE :

328 <1> ;------------------------------------------------------

329 <1>

330 0000154F E811000000 <1> call \_set\_mode ; 24/06/2016 (set\_txt\_mode)

331 <1>

332 <1> ; 12/05/2016

333 <1> ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)

334 <1>

335 <1> ;----- NORMAL RETURN FROM ALL VIDEO RETURNS

336 <1>

337 <1> VIDEO\_RETURN:

338 00001554 A1[C4650100] <1> mov eax, [video\_eax] ; 12/05/2016

339 <1> \_video\_return:

340 00001559 FA <1> cli ; 02/01/2017

341 0000155A 5D <1> pop ebp

342 0000155B 5F <1> pop edi

343 0000155C 5E <1> pop esi

344 0000155D 5B <1> pop ebx

345 <1> M15: ; VIDEO\_RETURN\_C

346 <1> ;;15/01/2017

347 <1> ; 02/01/2017

348 <1> ;;mov byte [intflg], 0

349 <1> ;

350 0000155E 59 <1> pop ecx

351 0000155F 5A <1> pop edx

352 00001560 1F <1> pop ds

353 00001561 07 <1> pop es ; RECOVER SEGMENTS

354 00001562 CF <1> iretd ; ALL DONE

355 <1>

356 <1> set\_txt\_mode:

357 <1> ; 29/07/2016

358 <1> ; 27/06/2016

359 00001563 B003 <1> mov al, 3

360 <1>

361 <1> ; 10/08/2016

362 <1> ; 08/08/2016

363 <1> ; 30/07/2016

364 <1> ; 29/07/2016

365 <1> ; 27/07/2016

366 <1> ; 26/07/2016

367 <1> ; 25/07/2016

368 <1> ; 23/07/2016

369 <1> ; 18/07/2016

370 <1> ; 07/07/2016

371 <1> ; 04/07/2016

372 <1> ; 03/07/2016

373 <1> ; 02/07/2016

374 <1> ; 26/06/2016

375 <1> ; 24/06/2016 (set\_txt\_mode -> \_set\_mode)

376 <1> ; 17/06/2016

377 <1> ; 29/05/2016

378 <1> ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)

379 <1> \_set\_mode:

380 <1> ; 24/06/2016

381 00001565 3805[C25E0000] <1> cmp [CRT\_MODE], al ; current mode = requested mode ?

382 0000156B 750D <1> jne short \_sm\_0

383 0000156D 3C03 <1> cmp al, 3 ; text, 80\*25 color, default mode

384 <1> ; for TRDOS 386 MainProg

385 0000156F 755F <1> jne short \_sm\_2 ; multiscreen is only for mode 3

386 <1>

387 <1> ; If '\_set\_mode' procedure is called for video mode 3

388 <1> ; while video mode is 3, video page will be cleared

389 <1> ; and cursor position of video page will be reset.

390 <1>

391 <1> ; 29/07/2016

392 00001571 800D[D1650100]80 <1> or byte [p\_crt\_mode], 80h ; clear page indicator

393 00001578 EB5B <1> jmp short \_sm\_3

394 <1> \_sm\_0:

395 0000157A 803D[C25E0000]03 <1> cmp byte [CRT\_MODE], 3

396 00001581 7534 <1> jne short \_sm\_1

397 <1>

398 <1> ; If '\_set\_mode' procedure is called for a video mode

399 <1> ; except video mode 3, while current video mode

400 <1> ; is 3; all video pages of mode 3 will be copied

401 <1> ; to 98000h address as backup, before mode change.

402 <1>

403 <1> \_sm\_save\_pm:

404 <1> ; 03/07/2016

405 <1> ; save video pages

406 00001583 BE00800B00 <1> mov esi, 0B8000h

407 00001588 BF00800900 <1> mov edi, 98000h ; 30/07/2016

408 0000158D B900200000 <1> mov ecx, (0B8000h-0B0000h)/4

409 00001592 F3A5 <1> rep movsd

410 <1>

411 00001594 C605[D1650100]03 <1> mov byte [p\_crt\_mode], 3 ; previous mode, backup sign

412 <1> ;mov cl, [ACTIVE\_PAGE]

413 <1> ;mov [p\_crt\_page], cl

414 <1>

415 <1> ; save cursor positions

416 0000159B BE[56580100] <1> mov esi, CURSOR\_POSN

417 000015A0 BF[D6650100] <1> mov edi, cursor\_pposn ; cursor positions backup

418 000015A5 B104 <1> mov cl, 4

419 000015A7 F3A5 <1> rep movsd

420 <1>

421 <1> ; 29/07/2016

422 <1> ;mov [ACTIVE\_PAGE], cl ; 0

423 000015A9 860D[66580100] <1> xchg cl, [ACTIVE\_PAGE]

424 000015AF 880D[D2650100] <1> mov [p\_crt\_page], cl ; previous page (for mode 3)

425 <1> ; [ACTIVE\_PAGE] = 0

426 000015B5 EB19 <1> jmp short \_sm\_2

427 <1>

428 <1> \_sm\_1:

429 000015B7 3C03 <1> cmp al, 3 ; text, 80\*25 color, default mode

430 <1> ; for TRDOS 386 MainProg

431 000015B9 7515 <1> jne short \_sm\_2 ; multiscreen is only for mode 3

432 <1>

433 <1> ; If '\_set\_mode' procedure is called for video mode 3

434 <1> ; while video mode is not 3 and if there is video

435 <1> ; page backup for video mode 3, all (of 8) mode 3

436 <1> ; video pages will be restored from 98000h.

437 <1>

438 000015BB 803D[D1650100]03 <1> cmp byte [p\_crt\_mode], 3 ; previous mode, backup sign

439 000015C2 750C <1> jne short \_sm\_2 ; there is no (multiscreen) video pages

440 <1> ; to be restored

441 000015C4 8A0D[D2650100] <1> mov cl, [p\_crt\_page]

442 000015CA 880D[66580100] <1> mov [ACTIVE\_PAGE], cl

443 <1>

444 <1> \_sm\_2:

445 000015D0 A2[C25E0000] <1> mov [CRT\_MODE], al ; save mode in global variable

446 <1> \_sm\_3:

447 <1> ; 30/07/2016

448 <1> ; 26/07/2016

449 <1> ; 25/07/2016

450 <1> ; set\_mode\_vga:

451 <1> ; 18/07/2016

452 <1> ; 14/07/2016

453 <1> ; 09/07/2016

454 <1> ; 04/07/2016

455 <1> ; 03/07/2016 (TRDOS 386 = TRDOS v2.0)

456 <1> ; /// video mode 13h ///

457 <1> ; derived from 'Plex86/Bochs VGABios' source code

458 <1> ; vgabios-0.7a (2011)

459 <1> ; by the LGPL VGABios developers Team (2001-2008)

460 <1> ; 'vgabios.c', 'vgatables.h'

461 <1> ;

462 <1> ; Oracle VirtualBox 5.0.24 VGABios Source Code

463 <1> ; ('vgabios.c', 'vgatables.h', 'vgafonts.h', 'vgarom.asm')

464 <1> ;

465 000015D5 88C4 <1> mov ah, al

466 000015D7 B910000000 <1> mov ecx, vga\_mode\_count

467 000015DC BE[DE5E0000] <1> mov esi, vga\_modes

468 000015E1 31DB <1> xor ebx, ebx

469 <1> \_sm\_4:

470 000015E3 AC <1> lodsb

471 000015E4 38C4 <1> cmp ah, al

472 000015E6 740C <1> je short \_sm\_5

473 000015E8 FEC3 <1> inc bl

474 000015EA E2F7 <1> loop \_sm\_4

475 <1>

476 <1> ; UNIMPLEMENTED VIDEO MODE !

477 000015EC 31C0 <1> xor eax, eax

478 000015EE A3[C4650100] <1> mov [video\_eax], eax ; 0

479 000015F3 C3 <1> retn

480 <1>

481 <1> ;----- eBX POINTS TO CORRECT ROW OF INITIALIZATION TABLE

482 <1>

483 <1> \_sm\_5: ; 25/07/2016

484 000015F4 89DE <1> mov esi, ebx

485 000015F6 81C6[2E5F0000] <1> add esi, vga\_memmodel

486 000015FC 8A06 <1> mov al, [esi]

487 000015FE A2[EA650100] <1> mov [VGA\_MTYPE], al

488 <1>

489 00001603 89DF <1> mov edi, ebx

490 00001605 81C7[3E5F0000] <1> add edi, vga\_dac\_s

491 0000160B C0E302 <1> shl bl, 2 ; byte -> dword

492 0000160E 81C3[EE5E0000] <1> add ebx, vga\_mode\_tbl\_ptr

493 <1>

494 <1> ;mov dword [VGA\_BASE], 0B8000h

495 <1> ;cmp ah, 0Dh ; [CRT\_MODE]

496 <1> ;jb short M9

497 <1> ;mov dword [VGA\_BASE], 0A0000h

498 <1> ;M9:

499 00001614 8B33 <1> mov esi, [ebx]

500 00001616 89F3 <1> mov ebx, esi

501 00001618 83C614 <1> add esi, vga\_p\_cm\_pos ; ebx + 20

502 0000161B 668B06 <1> mov ax, [esi] ; get the cursor mode from the table

503 0000161E 66A3[DB5E0000] <1> mov [CURSOR\_MODE], ax ; save cursor mode (initial value)

504 <1> ; al = 6, ah = 7

505 <1> ; al = 0Dh, ah = 0Eh ; 25/07/2016

506 00001624 E83B020000 <1> call cursor\_shape\_fix

507 <1> ; al = 14, ah = 15 (If [CHAR\_HEIGHT] = 16)

508 00001629 668906 <1> mov [esi], ax

509 <1>

510 0000162C 56 <1> push esi ; \*

511 <1>

512 0000162D 8A25[C95E0000] <1> mov ah, [VGA\_MODESET\_CTL]

513 00001633 80E408 <1> and ah, 8 ; default palette loading ?

514 00001636 7524 <1> jnz short \_sm\_6

515 00001638 66BAC603 <1> mov dx, 3C6h ; VGAREG\_PEL\_MASK (DAC mask register)

516 0000163C B0FF <1> mov al, 0FFh ; PEL mask

517 0000163E EE <1> out dx, al

518 0000163F 8A27 <1> mov ah, [edi] ; DAC model (selection number)

519 00001641 E8ED0F0000 <1> call load\_dac\_palette

520 <1> ; ecx = 0

521 00001646 F605[C95E0000]02 <1> test byte [VGA\_MODESET\_CTL], 2 ; gray scale summing

522 0000164D 740D <1> jz short \_sm\_6

523 0000164F 53 <1> push ebx

524 00001650 29DB <1> sub ebx, ebx ; sub bl, bl

525 00001652 66B90001 <1> mov cx, 256

526 00001656 E82B100000 <1> call gray\_scale\_summing

527 0000165B 5B <1> pop ebx

528 <1> \_sm\_6:

529 <1> ; Reset Attribute Ctl flip-flop

530 0000165C 66BADA03 <1> mov dx, 3DAh ; VGAREG\_ACTL\_RESET

531 00001660 EC <1> in al, dx

532 <1> ; Set Attribute Ctl

533 00001661 89DE <1> mov esi, ebx ; addr of params tbl for selected mode

534 00001663 83C623 <1> add esi, 35 ; actl regs

535 00001666 30E4 <1> xor ah, ah ; 0

536 00001668 66BAC003 <1> mov dx, 3C0h ; VGAREG\_ACTL\_ADDRESS

537 <1> \_sm\_7:

538 0000166C 88E0 <1> mov al, ah

539 0000166E EE <1> out dx, al ; index

540 0000166F AC <1> lodsb

541 <1> ; DX = 3C0h = VGAREG\_ACTL\_WRITE\_DATA

542 00001670 EE <1> out dx, al ; value

543 00001671 FEC4 <1> inc ah

544 00001673 80FC14 <1> cmp ah, 20 ; number of actl registers

545 00001676 72F4 <1> jb short \_sm\_7

546 <1> ;

547 00001678 88E0 <1> mov al, ah ; 20

548 0000167A EE <1> out dx, al ; index

549 0000167B 28C0 <1> sub al, al ; 0

550 0000167D EE <1> out dx, al ; value

551 <1> ;

552 <1> ; Set Sequencer Ctl

553 0000167E 89DE <1> mov esi, ebx ; addr of params tbl for selected mode

554 00001680 83C605 <1> add esi, 5 ; sequ regs

555 <1> ;

556 00001683 66BAC403 <1> mov dx, 3C4h ; VGAREG\_SEQU\_ADDRESS

557 00001687 EE <1> out dx, al ; 0

558 00001688 6642 <1> inc dx ; 3C5h ; VGAREG\_SEQU\_DATA

559 0000168A B003 <1> mov al, 3

560 0000168C EE <1> out dx, al

561 0000168D B401 <1> mov ah, 1

562 <1> \_sm\_8:

563 0000168F 88E0 <1> mov al, ah

564 <1> ;mov dx, 3C4h ; VGAREG\_SEQU\_ADDRESS

565 00001691 664A <1> dec dx

566 00001693 EE <1> out dx, al ; index

567 00001694 AC <1> lodsb

568 00001695 6642 <1> inc dx ; 3C5h ; VGAREG\_SEQU\_DATA

569 00001697 EE <1> out dx, al

570 00001698 80FC04 <1> cmp ah, 4 ; number of sequ regs

571 0000169B 7304 <1> jnb short \_sm\_9

572 0000169D FEC4 <1> inc ah

573 0000169F EBEE <1> jmp short \_sm\_8

574 <1> \_sm\_9:

575 <1> ; Set Grafx Ctl

576 000016A1 89DE <1> mov esi, ebx ; addr of params tbl for selected mode

577 000016A3 83C637 <1> add esi, 55 ; grdc regs

578 000016A6 30E4 <1> xor ah, ah ; 0

579 <1> \_sm\_10:

580 000016A8 88E0 <1> mov al, ah

581 000016AA 66BACE03 <1> mov dx, 3CEh ; VGAREG\_GRDC\_ADDRESS

582 000016AE EE <1> out dx, al

583 000016AF AC <1> lodsb

584 000016B0 6642 <1> inc dx ; 3CFh ; VGAREG\_GRDC\_DATA

585 000016B2 EE <1> out dx, al

586 000016B3 FEC4 <1> inc ah

587 000016B5 80FC09 <1> cmp ah, 9 ; number of grdc regs

588 000016B8 72EE <1> jb short \_sm\_10

589 <1> ;

590 <1> ; Disable CRTC write protection

591 000016BA 66BAD403 <1> mov dx, 3D4h ; VGAREG\_VGA\_CRTC\_ADDRESS

592 <1> ;mov al, 11h

593 <1> ;our dx, al

594 <1> ;inc dx

595 <1> ;sub al, al

596 <1> ;out dx, al

597 000016BE 66B81100 <1> mov ax, 11h

598 000016C2 66EF <1> out dx, ax

599 000016C4 89DE <1> mov esi, ebx ; addr of params tbl for selected mode

600 000016C6 83C60A <1> add esi, 10 ; crtc regs

601 <1> ; ah = 0

602 <1> \_sm\_11:

603 000016C9 88E0 <1> mov al, ah

604 <1> ; dx = 3D4h = VGAREG\_VGA\_CRTC\_ADDRESS

605 000016CB EE <1> out dx, al ; index

606 000016CC AC <1> lodsb

607 000016CD 6642 <1> inc dx ; VGAREG\_VGA\_CRTC\_ADDRESS + 1

608 000016CF EE <1> out dx, al ; value

609 000016D0 80FC18 <1> cmp ah, 24 ; number of crtc registers - 1

610 000016D3 7306 <1> jnb short \_sm\_12

611 000016D5 FEC4 <1> inc ah

612 000016D7 664A <1> dec dx ; 3D4h

613 000016D9 EBEE <1> jmp short \_sm\_11

614 <1> \_sm\_12:

615 <1> ; Set the misc register

616 000016DB 66BACC03 <1> mov dx, 3CCh ; VGAREG\_READ\_MISC\_OUTPUT

617 000016DF 8A4309 <1> mov al, [ebx+9] ; misc reg

618 000016E2 EE <1> out dx, al

619 <1> ;

620 <1> ; Enable video

621 000016E3 66BAC003 <1> mov dx, 3C0h ; VGAREG\_ACTL\_ADDRESS

622 000016E7 B020 <1> mov al, 20h

623 000016E9 EE <1> out dx, al ; set bit 5 to 1

624 000016EA 66BADA03 <1> mov dx, 3DAh ; VGAREG\_ACTL\_RESET

625 000016EE EC <1> in al, dx

626 <1> ;

627 000016EF 803D[D3650100]00 <1> cmp byte [noclearmem], 0

628 000016F6 7740 <1> ja short \_sm\_15

629 <1>

630 <1> ; 29/07/2016

631 000016F8 31C0 <1> xor eax, eax

632 000016FA B900400000 <1> mov ecx, 4000h ; 16K words (32K)

633 000016FF 803D[EA650100]02 <1> cmp byte [VGA\_MTYPE], 2 ; CTEXT, MTEXT, CGA

634 00001706 7715 <1> ja short \_sm\_14 ; no ? (0A0000h)

635 00001708 BF00800B00 <1> mov edi, 0B8000h

636 0000170D 7409 <1> je short \_sm\_13 ; CGA graphics mode

637 <1> ; 08/08/2016

638 0000170F A3[E6650100] <1> mov [VGA\_INT43H], eax ; 0 ; default font

639 00001714 66B82007 <1> mov ax, 0720h ; CGA text mode

640 <1> \_sm\_13:

641 00001718 F366AB <1> rep stosw

642 0000171B EB1B <1> jmp short \_sm\_15

643 <1>

644 <1> \_sm\_14:

645 0000171D BF00000A00 <1> mov edi, 0A0000h

646 <1> ; ecx = 16384 dwords (64K)

647 00001722 66BAC403 <1> mov dx, 3C4h ; VGAREG\_SEQU\_ADDRESS

648 00001726 B002 <1> mov al, 2

649 00001728 EE <1> out dx, al

650 <1> ;mov dx, 3C5h ; VGAREG\_SEQU\_DATA

651 00001729 6642 <1> inc dx

652 0000172B EC <1> in al, dx ; mmask

653 0000172C 6650 <1> push ax

654 0000172E B00F <1> mov al, 0Fh ; all planes

655 00001730 EE <1> out dx, al

656 00001731 30C0 <1> xor al, al ; 0

657 00001733 F3AB <1> rep stosd ; ecx = 163684 (64K)

658 00001735 6658 <1> pop ax

659 00001737 EE <1> out dx, al ; mmask

660 <1> \_sm\_15:

661 <1> ; ebx = addr of params tbl for selected mode

662 <1> ; 10/08/2016

663 00001738 668B03 <1> mov ax, [ebx] ; num of columns, 'twidth'

664 0000173B A2[C45E0000] <1> mov [CRT\_COLS], al

665 <1> ;; 26/07/2016

666 <1> ;; CRTC\_ADDRESS = 3D4h (always)

667 <1> ;mov ah, [ebx+1] ; num of rows, 'theightm1'

668 00001740 FEC4 <1> inc ah ; 09/07/2016

669 00001742 8825[CA5E0000] <1> mov [VGA\_ROWS], ah

670 <1> ; 10/08/2016

671 00001748 8A4302 <1> mov al, [ebx+2]

672 0000174B A2[C65E0000] <1> mov [CHAR\_HEIGHT], al

673 <1> ; 29/07/2016

674 <1> ; length of regen buffer in bytes

675 00001750 668B4B03 <1> mov cx, [ebx+3] ; 'slength\_l'

676 00001754 66890D[D4650100] <1> mov [CRT\_LEN], cx

677 <1> ;

678 <1> ; 27/07/2016

679 0000175B 30E4 <1> xor ah, ah

680 0000175D A0[66580100] <1> mov al, [ACTIVE\_PAGE] ; may be > 0 for mode 3

681 <1> ;mul word [CRT\_LEN] ; 4096 for mode 3

682 00001762 66F7E1 <1> mul cx ; 29/07/2016

683 00001765 66A3[54580100] <1> mov [CRT\_START], ax

684 <1> ;

685 0000176B B060 <1> mov al, 60h

686 0000176D 803D[D3650100]00 <1> cmp byte [noclearmem], 0

687 00001774 7602 <1> jna short \_sm\_16

688 00001776 0480 <1> add al, 80h

689 <1> \_sm\_16:

690 00001778 A2[C75E0000] <1> mov [VGA\_VIDEO\_CTL], al

691 0000177D C605[C85E0000]F9 <1> mov byte [VGA\_SWITCHES], 0F9h

692 00001784 8025[C95E0000]7F <1> and byte [VGA\_MODESET\_CTL], 7Fh

693 <1>

694 0000178B 5E <1> pop esi ; \*

695 <1>

696 <1> ; 26/07/2016

697 <1> ; 07/07/2016

698 0000178C 668B0D[DB5E0000] <1> mov cx, [CURSOR\_MODE] ; restore cursor mode (initial value)

699 00001793 66870E <1> xchg cx, [esi] ; cl = start line, ch = end line

700 <1> ; reset to initial value

701 00001796 86E9 <1> xchg ch, cl ; ch = start line, cl = end line

702 00001798 66890D[DB5E0000] <1> mov [CURSOR\_MODE], cx ; save (fixed) cursor mode

703 <1>

704 <1> ; 27/07/2016

705 0000179F 803D[EA650100]02 <1> cmp byte [VGA\_MTYPE], 2 ; CTEXT, MTEXT

706 000017A6 7317 <1> jnb short \_sm\_17

707 <1>

708 <1> ; Set cursor shape

709 <1> ;mov cx, 0607h

710 <1> ;call \_set\_ctype

711 <1>

712 <1> ; 29/07/2016

713 000017A8 B40A <1> mov ah, 10 ; 6845 register for cursor set

714 000017AA E8C4050000 <1> call m16 ; output cx register

715 <1>

716 <1> ; 25/07/2016

717 000017AF 803D[C25E0000]03 <1> cmp byte [CRT\_MODE], 03h

718 000017B6 7507 <1> jne short \_sm\_17

719 <1> ; 26/07/2016

720 <1>

721 000017B8 A0[66580100] <1> mov al, [ACTIVE\_PAGE]

722 000017BD EB0C <1> jmp short \_sm\_18

723 <1> \_sm\_17:

724 <1> ; Set cursor pos for page 0..7

725 000017BF 6629C0 <1> sub ax, ax ; eax = 0

726 000017C2 BF[56580100] <1> mov edi, CURSOR\_POSN

727 000017C7 AB <1> stosd

728 000017C8 AB <1> stosd

729 000017C9 AB <1> stosd

730 000017CA AB <1> stosd

731 <1> ;; Set active page 0

732 <1> ;mov [ACTIVE\_PAGE], al ; 0

733 <1> \_sm\_18:

734 <1> ; 29/07/2016

735 000017CB 803D[EA650100]02 <1> cmp byte [VGA\_MTYPE], 2 ; CTEXT, MTEXT

736 000017D2 0F8386000000 <1> jnb \_sm\_23

737 <1>

738 <1> ;cmp byte [CHAR\_HEIGHT], 16

739 <1> ;je short \_sm\_19

740 <1>

741 <1> ;; copy and activate 8x16 font

742 <1>

743 <1> ; 26/07/2016

744 000017D8 B004 <1> mov al, 04h

745 <1> ;sub bl, bl

746 <1> ; AX = 1104H ; Load ROM 8x16 Character Set

747 <1> ; (BL = font block to load (EGA: 0-3; VGA: 0-7))

748 000017DA E83A150000 <1> call load\_text\_8\_16\_pat

749 <1>

750 <1> ; video\_func\_1103h:

751 <1> ; biosfn\_set\_text\_block\_specifier:

752 <1> ; BL = font block selector code

753 <1> ; NOTE: TRDOS 386 only uses and sets font block 0

754 <1> ; (It is as BL = 0 for TRDOS 386)

755 000017DF 66BAC403 <1> mov dx, 3C4h ; VGAREG\_SEQU\_ADDRESS

756 <1> ;mov ah, bl

757 000017E3 28E4 <1> sub ah, ah ; 0

758 000017E5 B003 <1> mov al, 03h

759 000017E7 66EF <1> out dx, ax

760 <1> \_sm\_19:

761 <1> ; 29/07/2016

762 <1> ; 26/07/2016

763 <1> ; 24/06/2016

764 <1> ;mov edi, 0B8000h

765 <1> ;mov cx, 4000h ; 16K words (32K)

766 <1> ;

767 000017E9 30C0 <1> xor al, al

768 000017EB 3805[D1650100] <1> cmp byte [p\_crt\_mode], al ; 0

769 000017F1 7707 <1> ja short \_sm\_20 ; 3h, 80h or 83h

770 <1>

771 <1> ; 30/07/2016

772 <1> ; 24/06/2016

773 <1> ; TRDOS 386 (TRDOS v2) 'set mode' modification

774 <1> ; (for multiscreen feature):

775 <1> ; If '\_set\_mode' procedure is called for video mode 3

776 <1> ; while video mode is 3, video page will be cleared

777 <1> ; and cursor position of video page will be reset.

778 <1> ; If '\_set\_mode' procedure is called for a video mode

779 <1> ; except video mode 3, while current video mode

780 <1> ; is 3; all video pages of mode 3 will be copied

781 <1> ; to 98000h address as backup, before mode change.

782 <1> ; If '\_set\_mode' procedure is called for video mode 3

783 <1> ; while video mode is not 3 and if there is video

784 <1> ; page backup for video mode 3, all (of 8) mode 3

785 <1> ; video pages will be restored from 98000h.

786 <1>

787 000017F3 A2[66580100] <1> mov [ACTIVE\_PAGE], al ; 0

788 <1> ;mov ax, 0720h

789 <1> ;;mov cx, 4000h ; 16K words (32K)

790 <1> ;;mov edi, 0B8000h

791 <1> ;rep stosw

792 <1> ;sub al, al

793 000017F8 EB64 <1> jmp short \_sm\_23

794 <1> \_sm\_20:

795 <1> ; Previous video mode is 3

796 <1> ; New video mode is 3 while current video mode is not 3

797 <1> ; (multi screen) video pages will be restored from 0B0000h

798 <1>

799 000017FA 0FB61D[66580100] <1> movzx ebx, byte [ACTIVE\_PAGE]

800 00001801 D0E3 <1> shl bl, 1 ; \* 2

801 00001803 81C3[56580100] <1> add ebx, CURSOR\_POSN

802 <1>

803 <1> ; 29/07/2016

804 00001809 F605[D1650100]7F <1> test byte [p\_crt\_mode], 7Fh ; 83h or 3h

805 00001810 7427 <1> jz short \_sm\_21 ; do not restore video pages

806 <1>

807 <1> ;; restore video pages

808 00001812 BE00800900 <1> mov esi, 98000h ; 30/07/2016

809 00001817 BF00800B00 <1> mov edi, 0B8000h

810 0000181C 66B90020 <1> mov cx, 2000h ; 8K dwords (32K)

811 00001820 F3A5 <1> rep movsd

812 <1>

813 <1> ; restore cursor positions

814 00001822 BE[D6650100] <1> mov esi, cursor\_pposn

815 00001827 BF[56580100] <1> mov edi, CURSOR\_POSN

816 <1> ;mov ecx, 4 ; restore all cursor positions (16 bytes)

817 0000182C B104 <1> mov cl, 4

818 0000182E F3A5 <1> rep movsd

819 <1>

820 00001830 F605[D1650100]80 <1> test byte [p\_crt\_mode], 80h

821 00001837 7420 <1> jz short \_sm\_22 ; do not clear current video pages

822 <1> \_sm\_21:

823 <1> ; clear video page

824 00001839 668B0D[D4650100] <1> mov cx, [CRT\_LEN] ; 4096

825 00001840 66D1E9 <1> shr cx, 1 ; 2072

826 00001843 66B82007 <1> mov ax, 0720h

827 00001847 BF00800B00 <1> mov edi, 0B8000h ; [crt\_base]

828 0000184C 66033D[54580100] <1> add di, [CRT\_START]

829 00001853 F366AB <1> rep stosw ; FILL THE REGEN BUFFER WITH BLANKS

830 <1> ;

831 00001856 66890B <1> mov [ebx], cx ; reset cursor position

832 <1> \_sm\_22:

833 00001859 A2[D1650100] <1> mov [p\_crt\_mode], al ; 0

834 <1> \_sm\_23:

835 <1> ; al = video page number

836 <1> ; [CRT\_LEN] = length of regen buffer in bytes

837 0000185E E81E010000 <1> call \_set\_active\_page

838 <1>

839 <1> ;----- NORMAL RETURN FROM ALL VIDEO RETURNS

840 00001863 C3 <1> retn

841 <1>

842 <1> cursor\_shape\_fix:

843 <1> ; 07/07/2016

844 <1> ; (Cursor start and cursor end line values -6,7-

845 <1> ; will be fixed depending on character height)

846 <1> ;

847 <1> ; derived from 'Plex86/Bochs VGABios' source code

848 <1> ; vgabios-0.7a (2011)

849 <1> ; by the LGPL VGABios developers Team (2001-2008)

850 <1> ; 'vgabios.c', ' biosfn\_set\_cursor\_shape (CH,CL)'

851 <1> ;

852 <1> ; INPUT ->

853 <1> ; AL = cursor start line (=6)

854 <1> ; AH = cursor end line (=7)

855 <1> ; OUTPUT ->

856 <1> ; AL = cursor start line (=14)

857 <1> ; AH = cursor end line (=15)

858 <1> ;

859 <1> ;; if((modeset\_ctl&0x01)&&(cheight>8)&&(CL<8)&&(CH<0x20))

860 <1>

861 <1> ;test byte [VGA\_MODESET\_CTL], 1 ; VGA active

862 <1> ;jz short csf\_3

863 00001864 803D[C65E0000]08 <1> cmp byte [CHAR\_HEIGHT], 8

864 0000186B 7649 <1> jna short csf\_3

865 0000186D 80FC08 <1> cmp ah, 8

866 00001870 7344 <1> jnb short csf\_3

867 00001872 3C20 <1> cmp al, 20h

868 00001874 7340 <1> jnb short csf\_3

869 <1> ;

870 00001876 6650 <1> push ax

871 <1> ; {

872 <1> ; if(CL!=(CH+1))

873 00001878 FEC0 <1> inc al

874 0000187A 38C4 <1> cmp ah, al ; ah != al + 1

875 0000187C 740F <1> je short csf\_1

876 <1> ; CH = ((CH+1) \* cheight / 8) -1;

877 0000187E 8A25[C65E0000] <1> mov ah, [CHAR\_HEIGHT]

878 00001884 F6E4 <1> mul ah

879 00001886 C0E803 <1> shr al, 3 ; / 8

880 00001889 FEC8 <1> dec al ; - 1

881 0000188B EB0E <1> jmp short csf\_2

882 <1> csf\_1:

883 <1> ; }

884 <1> ; else ; ah = al + 1

885 <1> ; {

886 0000188D FEC4 <1> inc ah ; ah = ah + 1

887 <1> ; CH = ((CL+1) \* cheight / 8) - 2;

888 0000188F A0[C65E0000] <1> mov al, [CHAR\_HEIGHT]

889 00001894 F6E4 <1> mul ah

890 00001896 C0E803 <1> shr al, 3 ; / 8

891 00001899 2C02 <1> sub al, 2 ; - 2

892 <1> ; al = 14 (if [CHAR\_HEIGHT] = 16)

893 <1> csf\_2:

894 0000189B 880424 <1> mov [esp], al

895 0000189E 8A642401 <1> mov ah, [esp+1]

896 <1> ; CL = ((CL+1) \* cheight / 8) - 1;

897 000018A2 FEC4 <1> inc ah

898 000018A4 A0[C65E0000] <1> mov al, [CHAR\_HEIGHT]

899 000018A9 F6E4 <1> mul ah

900 000018AB C0E803 <1> shr al, 3 ; / 8

901 000018AE FEC8 <1> dec al ; - 1

902 000018B0 88442401 <1> mov [esp+1], al

903 <1> ; ah = 15 (if [CHAR\_HEIGHT] = 16)

904 <1> ;

905 000018B4 6658 <1> pop ax

906 <1> csf\_3:

907 000018B6 C3 <1> retn

908 <1>

909 <1> SET\_CTYPE:

910 <1> ; 12/09/2016

911 <1> ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)

912 000018B7 803D[C25E0000]07 <1> cmp byte [CRT\_MODE], 7

913 000018BE 0F8790FCFFFF <1> ja VIDEO\_RETURN ; 12/09/2016

914 000018C4 E805000000 <1> call \_set\_ctype

915 000018C9 E986FCFFFF <1> jmp VIDEO\_RETURN

916 <1>

917 <1> \_set\_ctype:

918 <1> ; 02/09/2014 (Retro UNIX 386 v1)

919 <1> ;

920 <1> ; VIDEO.ASM - 06/10/85 VIDEO DISPLAY BIOS

921 <1>

922 <1> ; (CH) = BITS 4-0 = START LINE FOR CURSOR

923 <1> ; \*\* HARDWARE WILL ALWAYS CAUSE BLINK

924 <1> ; \*\* SETTING BIT 5 OR 6 WILL CAUSE ERRATIC BLINKING

925 <1> ; OR NO CURSOR AT ALL

926 <1> ; (CL) = BITS 4-0 = END LINE FOR CURSOR

927 <1>

928 <1> ;------------------------------------------------

929 <1> ; SET\_CTYPE

930 <1> ; THIS ROUTINE SETS THE CURSOR VALUE

931 <1> ; INPUT

932 <1> ; (CX) HAS CURSOR VALUE CH-START LINE, CL-STOP LINE

933 <1> ; OUTPUT

934 <1> ; NONE

935 <1> ;------------------------------------------------

936 <1>

937 <1> ; 07/07/2016

938 <1> ; Fixing cursor start and stop line depending on

939 <1> ; current character height (=16)

940 <1> ; (Note: Default/initial values are 6 and 7.

941 <1> ; If set values are 6 (start) & 7 (stop) and

942 <1> ; [CHAR\_HEIGHT] = 16 :

943 <1> ; After fixing, start line will be 14, stop line

944 <1> ; will be 15.)

945 000018CE 6689C8 <1> mov ax, cx

946 000018D1 86C4 <1> xchg al, ah

947 <1> ; AL = start line, AH = stop line

948 000018D3 E88CFFFFFF <1> call cursor\_shape\_fix

949 <1> ; AL = start line (fixed), AH = stop line (fixed)

950 000018D8 6689C1 <1> mov cx, ax

951 000018DB 86E9 <1> xchg ch, cl

952 <1> ; CH = start line (fixed), CL = stop line (fixed)

953 <1> ;

954 000018DD B40A <1> mov ah, 10 ; 6845 register for cursor set

955 000018DF 66890D[DB5E0000] <1> mov [CURSOR\_MODE], cx ; save in data area

956 <1> ;call m16 ; output cx register

957 <1> ;retn

958 000018E6 E988040000 <1> jmp m16

959 <1>

960 <1> SET\_CPOS:

961 <1> ; 12/09/2016

962 <1> ; 07/07/2016

963 <1> ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)

964 000018EB 80FF07 <1> cmp bh, 7 ; video page > 7 ; 07/07/2016

965 000018EE 0F8760FCFFFF <1> ja VIDEO\_RETURN

966 <1> ;

967 000018F4 803D[C25E0000]07 <1> cmp byte [CRT\_MODE], 7

968 000018FB 770A <1> ja short vga\_set\_cpos ; 12/09/2016

969 000018FD E846040000 <1> call \_set\_cpos

970 00001902 E94DFCFFFF <1> jmp VIDEO\_RETURN

971 <1>

972 <1> vga\_set\_cpos:

973 <1> ; 12/09/2016

974 <1> ; 09/07/2016

975 <1> ; set cursor position

976 <1> ; NOTE: Hardware cursor position will not be set

977 <1> ; in any VGA modes (>7)

978 <1> ; But, cursor position will be saved into

979 <1> ; [CURSOR\_POSN].

980 <1> ; TRDOS 386 (TRDOS v2.0) uses only one page

981 <1> ; (page 0) for all graphics modes.

982 <1>

983 00001907 668915[56580100] <1> mov [CURSOR\_POSN], dx ; save cursor pos for pg 0

984 <1> ; 04/08/2016

985 <1> ;mov bh, [ACTIVE\_PAGE] ; = 0

986 <1> ;call \_set\_cpos

987 0000190E E941FCFFFF <1> jmp VIDEO\_RETURN

988 <1>

989 <1> READ\_CURSOR:

990 <1> ; 12/09/2016

991 <1> ; 07/07/2016

992 <1> ; 12/05/2016

993 <1> ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)

994 <1> ;

995 <1> ; VIDEO.ASM - 06/10/85 VIDEO DISPLAY BIOS

996 <1>

997 <1> ;------------------------------------------------------

998 <1> ; READ\_CURSOR

999 <1> ; THIS ROUTINE READS THE CURRENT CURSOR VALUE FROM THE

1000 <1> ; 845, FORMATS IT, AND SENDS IT BACK TO THE CALLER

1001 <1> ; INPUT

1002 <1> ; BH - PAGE OF CURSOR

1003 <1> ; OUTPUT

1004 <1> ; DX - ROW, COLUMN OF THE CURRENT CURSOR POSITION

1005 <1> ; CX - CURRENT CURSOR MODE

1006 <1> ;------------------------------------------------------

1007 <1>

1008 <1> ; BH = Video page number (0 to 7)

1009 <1>

1010 <1> ; 07/07/2016

1011 00001913 80FF07 <1> cmp bh, 7 ; video page > 7 (invalid)

1012 00001916 7606 <1> jna short read\_cursor\_1

1013 <1> ; invalid video page (input)

1014 00001918 31C9 <1> xor ecx, ecx ; 0

1015 0000191A 31D2 <1> xor edx, edx ; 0

1016 0000191C EB15 <1> jmp short read\_cursor\_2

1017 <1> read\_cursor\_1:

1018 <1> ; 12/09/2016

1019 0000191E 803D[C25E0000]07 <1> cmp byte [CRT\_MODE], 7 ; vga mode

1020 00001925 7727 <1> ja short vga\_get\_cpos

1021 <1> ;

1022 00001927 E815000000 <1> call get\_cpos

1023 0000192C 0FB70D[DB5E0000] <1> movzx ecx, word [CURSOR\_MODE]

1024 <1> read\_cursor\_2:

1025 00001933 5D <1> pop ebp

1026 00001934 5F <1> pop edi

1027 00001935 5E <1> pop esi

1028 00001936 5B <1> pop ebx

1029 00001937 58 <1> pop eax ; DISCARD SAVED CX AND DX

1030 00001938 58 <1> pop eax

1031 00001939 A1[C4650100] <1> mov eax, [video\_eax] ; 12/05/2016

1032 <1> ;;15/01/2017

1033 <1> ;;mov byte [intflg], 0 ; 07/01/2017

1034 0000193E 1F <1> pop ds

1035 0000193F 07 <1> pop es

1036 00001940 CF <1> iretd

1037 <1>

1038 <1> get\_cpos:

1039 <1> ; 12/05/2016

1040 <1> ; 16/01/2016

1041 <1> ; BH = Video page number (0 to 7)

1042 <1> ;

1043 00001941 D0E7 <1> shl bh, 1 ; WORD OFFSET

1044 00001943 0FB6F7 <1> movzx esi, bh

1045 00001946 0FB796[56580100] <1> movzx edx, word [esi+CURSOR\_POSN]

1046 0000194D C3 <1> retn

1047 <1>

1048 <1> vga\_get\_cpos:

1049 <1> ; 12/09/2016

1050 <1> ; get cursor position (vga)

1051 0000194E 0FB715[56580100] <1> movzx edx, word [CURSOR\_POSN] ; cursor pos for pg 0

1052 00001955 31C9 <1> xor ecx, ecx ; Cursor Mode = 0 (invalid)

1053 00001957 EBDA <1> jmp short read\_cursor\_2

1054 <1>

1055 <1> ACT\_DISP\_PAGE:

1056 <1> ; 07/07/2016

1057 <1> ; 26/06/2016

1058 <1> ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)

1059 <1> ;

1060 <1> ; VIDEO.ASM - 06/10/85 VIDEO DISPLAY BIOS

1061 <1> ;

1062 <1> ;-----------------------------------------------------

1063 <1> ; ACT\_DISP\_PAGE

1064 <1> ; THIS ROUTINE SETS THE ACTIVE DISPLAY PAGE, ALLOWING

1065 <1> ; THE FULL USE OF THE MEMORY SET ASIDE FOR THE VIDEO ATTACHMENT

1066 <1> ; INPUT

1067 <1> ; AL HAS THE NEW ACTIVE DISPLAY PAGE

1068 <1> ; OUTPUT

1069 <1> ; THE 6845 IS RESET TO DISPLAY THAT PAGE

1070 <1> ;-----------------------------------------------------

1071 <1> ; 07/07/2016

1072 00001959 3C07 <1> cmp al, 7 ; > 7 = invalid video page number

1073 0000195B 0F87F3FBFFFF <1> ja VIDEO\_RETURN

1074 00001961 803D[C25E0000]03 <1> cmp byte [CRT\_MODE], 3

1075 00001968 7408 <1> je short adp\_1

1076 0000196A 20C0 <1> and al, al

1077 0000196C 0F85E2FBFFFF <1> jnz VIDEO\_RETURN

1078 <1> ;sub al, al ; 0 ; force to page 0

1079 <1> adp\_1:

1080 00001972 E805000000 <1> call set\_active\_page

1081 00001977 E9D8FBFFFF <1> jmp VIDEO\_RETURN

1082 <1>

1083 <1> set\_active\_page: ; tty\_sw

1084 <1> ; 09/12/2017

1085 <1> ; 26/07/2016

1086 <1> ; 26/06/2016

1087 <1> ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)

1088 <1> ; 30/06/2015

1089 <1> ; 04/03/2014 (act\_disp\_page --> tty\_sw)

1090 <1> ; 10/12/2013

1091 <1> ; 04/12/2013

1092 <1> ;

1093 0000197C A2[66580100] <1> mov [ACTIVE\_PAGE], al ; save active page value ; [ptty]

1094 <1> \_set\_active\_page:

1095 <1> ; 27/06/2015

1096 00001981 0FB6D8 <1> movzx ebx, al

1097 <1> ;

1098 <1> ;cbw ; 07/09/2014 (ah=0)

1099 00001984 28E4 <1> sub ah, ah ; 09/12/2017

1100 00001986 66F725[D4650100] <1> mul word [CRT\_LEN] ; get saved length of regen buffer

1101 <1> ; display page times regen length

1102 <1> ; 10/12/2013

1103 0000198D 66A3[54580100] <1> mov [CRT\_START], ax ; save start address for later

1104 00001993 6689C1 <1> mov cx, ax ; start address to cx

1105 <1> \_M16:

1106 <1> ;sar cx, 1

1107 00001996 66D1E9 <1> shr cx, 1 ; divide by 2 for 6845 handling

1108 00001999 B40C <1> mov ah, 12 ; 6845 register for start address

1109 0000199B E8D3030000 <1> call m16

1110 <1> ;sal bx, 1

1111 <1> ; 01/09/2014

1112 000019A0 D0E3 <1> shl bl, 1 ; \*2 for word offset

1113 000019A2 81C3[56580100] <1> add ebx, CURSOR\_POSN

1114 000019A8 668B13 <1> mov dx, [ebx] ; get cursor for this page

1115 <1> ; 16/01/2016

1116 <1> ;call m18

1117 <1> ;retn

1118 000019AB E9AF030000 <1> jmp m18

1119 <1>

1120 <1> position:

1121 <1> ; 24/06/2016

1122 <1> ; 12/05/2016 - TRDOS 386 (TRDOS v2.0)

1123 <1> ; 27/06/2015

1124 <1> ; 02/09/2014

1125 <1> ; 30/08/2014 (Retro UNIX 386 v1)

1126 <1> ; 04/12/2013 (Retro UNIX 8086 v1)

1127 <1> ;

1128 <1> ; VIDEO.ASM - 06/10/85 VIDEO DISPLAY BIOS

1129 <1> ;

1130 <1> ;-----------------------------------------

1131 <1> ; POSITION

1132 <1> ; THIS SERVICE ROUTINE CALCULATES THE REGEN BUFFER ADDRESS

1133 <1> ; OF A CHARACTER IN THE ALPHA MODE

1134 <1> ; INPUT

1135 <1> ; AX = ROW, COLUMN POSITION

1136 <1> ; OUTPUT

1137 <1> ; AX = OFFSET OF CHAR POSITION IN REGEN BUFFER

1138 <1> ;-----------------------------------------

1139 <1>

1140 <1> ; DX = ROW, COLUMN POSITION

1141 000019B0 0FB605[C45E0000] <1> movzx eax, byte [CRT\_COLS] ; 24/06/2016

1142 000019B7 F6E6 <1> mul dh ; row value

1143 000019B9 30F6 <1> xor dh, dh ; 0

1144 000019BB 6601D0 <1> add ax, dx ; add column value to the result

1145 000019BE 66D1E0 <1> shl ax, 1 ; \* 2 for attribute bytes

1146 <1> ; EAX = AX = OFFSET OF CHAR POSITION IN REGEN BUFFER

1147 000019C1 C3 <1> retn

1148 <1>

1149 <1> find\_position:

1150 <1> ; 24/06/2016

1151 <1> ; 12/05/2016 - TRDOS 386 (TRDOS v2.0)

1152 <1> ; 27/06/2015

1153 <1> ; 07/09/2014

1154 <1> ; 02/09/2014

1155 <1> ; 30/08/2014 (Retro UNIX 386 v1)

1156 <1> ; VIDEO.ASM - 06/10/85 VIDEO DISPLAY BIOS

1157 <1>

1158 000019C2 0FB6CF <1> movzx ecx, bh ; video page number

1159 000019C5 89CE <1> mov esi, ecx

1160 000019C7 66D1E6 <1> shl si, 1

1161 000019CA 668B96[56580100] <1> mov dx, [esi+CURSOR\_POSN]

1162 000019D1 740C <1> jz short p21

1163 000019D3 6631F6 <1> xor si, si

1164 <1> p20:

1165 000019D6 660335[D4650100] <1> add si, [CRT\_LEN] ; 24/06/2016

1166 <1> ;add si, 80\*25\*2 ; add length of buffer for one page

1167 000019DD E2F7 <1> loop p20

1168 <1> p21:

1169 000019DF 6621D2 <1> and dx, dx

1170 000019E2 7407 <1> jz short p22

1171 000019E4 E8C7FFFFFF <1> call position ; determine location in regen in page

1172 000019E9 01C6 <1> add esi, eax ; add location to start of regen page

1173 <1> p22:

1174 <1> ;mov dx, [addr\_6845] ; get base address of active display

1175 <1> ;mov dx, 03D4h ; I/O address of color card

1176 <1> ;add dx, 6 ; point at status port

1177 000019EB 66BADA03 <1> mov dx, 03DAh ; status port

1178 <1> ; cx = 0

1179 000019EF C3 <1> retn

1180 <1>

1181 <1> SCROLL\_UP:

1182 <1> ; 07/07/2016

1183 <1> ; 26/06/2016

1184 <1> ; 12/05/2016

1185 <1> ; 30/01/2016

1186 <1> ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)

1187 <1> ; 07/09/2014

1188 <1> ; 02/09/2014

1189 <1> ; 01/09/2014 (Retro UNIX 386 v1 - beginning)

1190 <1> ; 04/04/2014

1191 <1> ; 04/12/2013

1192 <1> ;

1193 <1> ; VIDEO.ASM - 06/10/85 VIDEO DISPLAY BIOS

1194 <1> ;

1195 <1> ;----------------------------------------------

1196 <1> ; SCROLL UP

1197 <1> ; THIS ROUTINE MOVES A BLOCK OF CHARACTERS UP

1198 <1> ; ON THE SCREEN

1199 <1> ; INPUT

1200 <1> ; (AH) = CURRENT CRT MODE

1201 <1> ; (AL) = NUMBER OF ROWS TO SCROLL

1202 <1> ; (CX) = ROW/COLUMN OF UPPER LEFT CORNER

1203 <1> ; (DX) = ROW/COLUMN OF LOWER RIGHT CORNER

1204 <1> ; (BH) = ATTRIBUTE TO BE USED ON BLANKED LINE

1205 <1> ; (DS) = DATA SEGMENT

1206 <1> ; (ES) = REGEN BUFFER SEGMENT

1207 <1> ; OUTPUT

1208 <1> ; NONE -- THE REGEN BUFFER IS MODIFIED

1209 <1> ;--------------------------------------------

1210 <1>

1211 <1> ; 07/07/2016

1212 000019F0 38F5 <1> cmp ch, dh

1213 000019F2 0F875CFBFFFF <1> ja VIDEO\_RETURN

1214 000019F8 38D1 <1> cmp cl, dl

1215 000019FA 0F8754FBFFFF <1> ja VIDEO\_RETURN

1216 <1> ;

1217 00001A00 E805000000 <1> call \_scroll\_up

1218 00001A05 E94AFBFFFF <1> jmp VIDEO\_RETURN

1219 <1>

1220 <1> \_scroll\_up: ; from 'write\_tty'

1221 <1> ;

1222 <1> ; cl = left upper column

1223 <1> ; ch = left upper row

1224 <1> ; dl = right lower column

1225 <1> ; dh = right lower row

1226 <1> ;

1227 <1> ; al = line count

1228 <1> ; bl = attribute to be used on blanked line

1229 <1> ; bh = video page number (0 to 7)

1230 <1>

1231 00001A0A E896000000 <1> call test\_line\_count ; 16/01/2016

1232 <1>

1233 00001A0F 8A25[C25E0000] <1> mov ah, [CRT\_MODE] ; current video mode

1234 <1> ;cmp ah, 4

1235 <1> ;jb short n0

1236 <1> ;cmp byte [CRT\_MODE], 4

1237 00001A15 80FC04 <1> cmp ah, 4 ; 07/07/2016

1238 00001A18 0F8320050000 <1> jnb GRAPHICS\_UP ; 26/06/2016

1239 <1>

1240 <1> ;cmp ah, 7 ; TEST FOR BW CARD

1241 <1> ;jne GRAPHICS\_UP

1242 <1> n0:

1243 <1> ; 07/07/2016

1244 00001A1E 80FF07 <1> cmp bh, 7 ; video page number

1245 00001A21 7606 <1> jna short n1

1246 00001A23 8A3D[66580100] <1> mov bh, [ACTIVE\_PAGE]

1247 <1> n1:

1248 00001A29 88DC <1> mov ah, bl ; attribute

1249 00001A2B 6650 <1> push ax ; \*

1250 <1> ;mov esi, [CRT\_BASE]

1251 00001A2D BE00800B00 <1> mov esi, 0B8000h

1252 00001A32 3A3D[66580100] <1> cmp bh, [ACTIVE\_PAGE]

1253 00001A38 750B <1> jne short n2

1254 <1> ;

1255 00001A3A 66A1[54580100] <1> mov ax, [CRT\_START]

1256 00001A40 6601C6 <1> add si, ax

1257 00001A43 EB11 <1> jmp short n4

1258 <1> n2:

1259 00001A45 20FF <1> and bh, bh

1260 00001A47 740D <1> jz short n4

1261 00001A49 88F8 <1> mov al, bh

1262 <1> n3:

1263 00001A4B 660335[D4650100] <1> add si, [CRT\_LEN]

1264 00001A52 FEC8 <1> dec al

1265 00001A54 75F5 <1> jnz short n3

1266 <1> n4:

1267 00001A56 E85D000000 <1> call scroll\_position ; 16/01/2016

1268 00001A5B 7420 <1> jz short n6

1269 <1>

1270 00001A5D 01CE <1> add esi, ecx ; from address for scroll

1271 00001A5F 88F5 <1> mov ch, dh ; #rows in block

1272 00001A61 28C5 <1> sub ch, al ; #rows to be moved

1273 <1> n5:

1274 00001A63 E894000000 <1> call n10 ; 16/01/2016

1275 <1>

1276 00001A68 51 <1> push ecx

1277 00001A69 0FB60D[C45E0000] <1> movzx ecx, byte [CRT\_COLS]

1278 00001A70 00C9 <1> add cl, cl

1279 00001A72 01CE <1> add esi, ecx ; next line

1280 00001A74 01CF <1> add edi, ecx

1281 00001A76 59 <1> pop ecx

1282 <1>

1283 00001A77 FECD <1> dec ch ; count of lines to move

1284 00001A79 75E8 <1> jnz short n5 ; row loop

1285 <1> ; ch = 0

1286 00001A7B 88C6 <1> mov dh, al ; #rows

1287 <1> n6:

1288 <1> ; attribute in ah

1289 00001A7D B020 <1> mov al, ' ' ; fill with blanks

1290 <1> n7:

1291 00001A7F E885000000 <1> call n11 ; 16/01/2016

1292 <1>

1293 00001A84 8A0D[C45E0000] <1> mov cl, [CRT\_COLS]

1294 00001A8A 00C9 <1> add cl, cl

1295 00001A8C 01CF <1> add edi, ecx

1296 <1>

1297 00001A8E FECE <1> dec dh

1298 00001A90 75ED <1> jnz short n7

1299 <1> n16:

1300 00001A92 3A3D[66580100] <1> cmp bh, [ACTIVE\_PAGE]

1301 00001A98 750A <1> jne short n8

1302 <1>

1303 <1> ;cmp byte [CRT\_MODE], 7 ; is this the black and white card

1304 <1> ;je short n8 ; if so, skip the mode reset

1305 <1>

1306 00001A9A A0[C35E0000] <1> mov al, [CRT\_MODE\_SET] ; get the value of mode set

1307 00001A9F 66BAD803 <1> mov dx, 03D8h ; always set color card port

1308 00001AA3 EE <1> out dx, al

1309 <1> n8:

1310 00001AA4 C3 <1> retn

1311 <1>

1312 <1> test\_line\_count:

1313 <1> ; 12/05/2016

1314 <1> ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)

1315 <1> ; 07/09/2014 (scroll\_up)

1316 00001AA5 08C0 <1> or al, al

1317 00001AA7 740E <1> jz short al\_set2

1318 00001AA9 6652 <1> push dx

1319 00001AAB 28EE <1> sub dh, ch ; subtract upper row from lower row number

1320 00001AAD FEC6 <1> inc dh ; adjust difference by 1

1321 00001AAF 38C6 <1> cmp dh, al ; line count = amount of rows in window?

1322 00001AB1 7502 <1> jne short al\_set1 ; if not the we're all set

1323 00001AB3 30C0 <1> xor al, al ; otherwise set al to zero

1324 <1> al\_set1:

1325 00001AB5 665A <1> pop dx

1326 <1> al\_set2:

1327 00001AB7 C3 <1> retn

1328 <1>

1329 <1> scroll\_position:

1330 <1> ; 26/06/2016

1331 <1> ; 30/01/2016

1332 <1> ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)

1333 <1> ; 07/09/2014 (scroll\_up)

1334 <1>

1335 00001AB8 6652 <1> push dx

1336 00001ABA 6689CA <1> mov dx, cx ; now, upper left position in DX

1337 00001ABD E8EEFEFFFF <1> call position

1338 00001AC2 01C6 <1> add esi, eax

1339 00001AC4 89F7 <1> mov edi, esi

1340 00001AC6 665A <1> pop dx ; lower right position in DX

1341 00001AC8 6629CA <1> sub dx, cx

1342 00001ACB FEC6 <1> inc dh ; dh = #rows

1343 00001ACD FEC2 <1> inc dl ; dl = #cols in block

1344 00001ACF 59 <1> pop ecx ; return address

1345 00001AD0 6658 <1> pop ax ; \* ; al = line count, ah = attribute

1346 00001AD2 51 <1> push ecx ; return address

1347 00001AD3 0FB7C8 <1> movzx ecx, ax

1348 00001AD6 8A25[C45E0000] <1> mov ah, [CRT\_COLS]

1349 00001ADC F6E4 <1> mul ah ; determine offset to from address

1350 00001ADE 6601C0 <1> add ax, ax ; \*2 for attribute byte

1351 <1> ;

1352 00001AE1 6650 <1> push ax ; offset

1353 00001AE3 6652 <1> push dx

1354 <1> ;

1355 <1> ; 04/04/2014

1356 00001AE5 66BADA03 <1> mov dx, 3DAh ; guaranteed to be color card here

1357 <1> n9: ; wait\_display\_enable

1358 00001AE9 EC <1> in al, dx ; get port

1359 00001AEA A808 <1> test al, RVRT ; wait for vertical retrace

1360 00001AEC 74FB <1> jz short n9 ; wait\_display\_enable

1361 00001AEE B025 <1> mov al, 25h

1362 00001AF0 B2D8 <1> mov dl, 0D8h ; address control port

1363 00001AF2 EE <1> out dx, al ; turn off video during vertical retrace

1364 00001AF3 665A <1> pop dx ; #rows, #cols

1365 00001AF5 6658 <1> pop ax ; offset

1366 00001AF7 6691 <1> xchg ax, cx ;

1367 <1> ; ecx = offset, al = line count, ah = attribute

1368 <1> ;

1369 00001AF9 08C0 <1> or al, al

1370 00001AFB C3 <1> retn

1371 <1> n10:

1372 <1> ; Move rows

1373 00001AFC 88D1 <1> mov cl, dl ; get # of cols to move

1374 00001AFE 56 <1> push esi

1375 00001AFF 57 <1> push edi ; save start address

1376 <1> n10r:

1377 00001B00 66A5 <1> movsw ; move that line on screen

1378 00001B02 FEC9 <1> dec cl

1379 00001B04 75FA <1> jnz short n10r

1380 00001B06 5F <1> pop edi

1381 00001B07 5E <1> pop esi ; recover addresses

1382 00001B08 C3 <1> retn

1383 <1> n11:

1384 <1> ; Clear rows

1385 <1> ; dh = #rows

1386 00001B09 88D1 <1> mov cl, dl ; get # of cols to clear

1387 00001B0B 57 <1> push edi ; save address

1388 <1> n11r:

1389 00001B0C 66AB <1> stosw ; store fill character

1390 00001B0E FEC9 <1> dec cl

1391 00001B10 75FA <1> jnz short n11r

1392 00001B12 5F <1> pop edi ; recover address

1393 00001B13 C3 <1> retn

1394 <1>

1395 <1> SCROLL\_DOWN:

1396 <1> ; 07/07/2016

1397 <1> ; 27/06/2016

1398 <1> ; 26/06/2016

1399 <1> ; 12/05/2016

1400 <1> ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)

1401 <1> ;

1402 <1> ; VIDEO.ASM - 06/10/85 VIDEO DISPLAY BIOS

1403 <1>

1404 <1> ;------------------------------------------

1405 <1> ; SCROLL DOWN

1406 <1> ; THIS ROUTINE MOVES THE CHARACTERS WITHIN A DEFINED

1407 <1> ; BLOCK DOWN ON THE SCREEN, FILLING THE TOP LINES

1408 <1> ; WITH A DEFINED CHARACTER

1409 <1> ; INPUT

1410 <1> ; (AH) = CURRENT CRT MODE

1411 <1> ; (AL) = NUMBER OF LINES TO SCROLL

1412 <1> ; (CX) = UPPER LEFT CORNER OF RECION

1413 <1> ; (DX) = LOWER RIGHT CORNER OF REGION

1414 <1> ; (BH) = FILL CHARACTER

1415 <1> ; (DS) = DATA SEGMENT

1416 <1> ; (ES) = REGEN SEGMENT

1417 <1> ; OUTPUT

1418 <1> ; NONE -- SCREEN IS SCROLLED

1419 <1> ;------------------------------------------

1420 <1>

1421 <1> ; 07/07/2016

1422 00001B14 38F5 <1> cmp ch, dh

1423 00001B16 0F8738FAFFFF <1> ja VIDEO\_RETURN

1424 00001B1C 38D1 <1> cmp cl, dl

1425 00001B1E 0F8730FAFFFF <1> ja VIDEO\_RETURN

1426 <1> ;

1427 00001B24 E805000000 <1> call \_scroll\_down

1428 00001B29 E926FAFFFF <1> jmp VIDEO\_RETURN

1429 <1>

1430 <1> \_scroll\_down: ; 27/06/2016

1431 <1>

1432 <1> ; cl = left upper column

1433 <1> ; ch = left upper row

1434 <1> ; dl = right lower column

1435 <1> ; dh = right lower row

1436 <1> ;

1437 <1> ; al = line count

1438 <1> ; bl = attribute to be used on blanked line

1439 <1> ; bh = video page number (0 to 7)

1440 <1>

1441 <1> ; !!!!

1442 00001B2E FD <1> std ; DIRECTION FOR SCROLL DOWN

1443 <1> ; !!!!

1444 00001B2F E871FFFFFF <1> call test\_line\_count ; 16/01/2016

1445 <1>

1446 00001B34 8A25[C25E0000] <1> mov ah, [CRT\_MODE] ; current video mode

1447 <1> ;cmp ah, 4

1448 <1> ;jb short \_n0

1449 <1> ;cmp byte [CRT\_MODE], 4

1450 00001B3A 80FC04 <1> cmp ah, 4 ; 07/07/2016

1451 00001B3D 0F83DF070000 <1> jnb GRAPHICS\_DOWN ; 26/06/2016

1452 <1>

1453 <1> ;cmp ah, 7 ; TEST FOR BW CARD

1454 <1> ;jne GRAPHICS\_DOWN

1455 <1> \_n0:

1456 <1> ; 07/07/2016

1457 00001B43 80FF07 <1> cmp bh, 7 ; video page number

1458 00001B46 7606 <1> jna short n12

1459 00001B48 8A3D[66580100] <1> mov bh, [ACTIVE\_PAGE]

1460 <1> ;

1461 <1> n12: ; CONTINUE\_DOWN

1462 00001B4E 88DC <1> mov ah, bl

1463 00001B50 6650 <1> push ax ; \* ; save attribute in ah

1464 00001B52 6689D0 <1> mov ax, dx ; LOWER RIGHT CORNER

1465 00001B55 E85EFFFFFF <1> call scroll\_position ; GET REGEN LOCATION

1466 00001B5A 741F <1> jz short n14

1467 00001B5C 29CE <1> sub esi, ecx ; SI IS FROM ADDRESS

1468 00001B5E 88F5 <1> mov ch, dh ; #rows in block

1469 00001B60 28C5 <1> sub ch, al ; #rows to be moved

1470 <1> n13:

1471 00001B62 E895FFFFFF <1> call n10 ; MOVE ONE ROW

1472 <1>

1473 00001B67 51 <1> push ecx

1474 00001B68 8A0D[C45E0000] <1> mov cl, [CRT\_COLS]

1475 00001B6E 00C9 <1> add cl, cl

1476 00001B70 29CE <1> sub esi, ecx ; next line

1477 00001B72 29CF <1> sub edi, ecx

1478 00001B74 59 <1> pop ecx

1479 <1>

1480 00001B75 FECD <1> dec ch ; count of lines to move

1481 00001B77 75E9 <1> jnz short n13 ; row loop

1482 <1> ; ch = 0

1483 00001B79 88C6 <1> mov dh, al ; #rows

1484 <1> n14:

1485 <1> ; attribute in ah

1486 00001B7B B020 <1> mov al, ' ' ; fill with blanks

1487 <1> n15:

1488 00001B7D E887FFFFFF <1> call n11 ; 16/01/2016

1489 <1>

1490 00001B82 8A0D[C45E0000] <1> mov cl, [CRT\_COLS]

1491 00001B88 00C9 <1> add cl, cl

1492 00001B8A 29CF <1> sub edi, ecx

1493 <1>

1494 00001B8C FECE <1> dec dh

1495 00001B8E 75ED <1> jnz short n15

1496 <1> ;

1497 00001B90 E9FDFEFFFF <1> jmp n16 ; 27/06/2016

1498 <1>

1499 <1> ; cmp bh, [ACTIVE\_PAGE]

1500 <1> ; jne short n16

1501 <1> ;

1502 <1> ; ;cmp byte [CRT\_MODE], 7 ; is this the black and white card

1503 <1> ; ;je short n16 ; if so, skip the mode reset

1504 <1> ;

1505 <1> ; mov al, [CRT\_MODE\_SET] ; get the value of mode set

1506 <1> ; mov dx, 03D8h ; always set color card port

1507 <1> ; out dx, al

1508 <1> ;n16:

1509 <1> ; ; !!!!

1510 <1> ; cld ; Clear direction flag !

1511 <1> ; ; !!!!

1512 <1> ; retn

1513 <1>

1514 <1> READ\_AC\_CURRENT:

1515 <1> ; 08/07/2016

1516 <1> ; 26/06/2016

1517 <1> ; 12/05/2016

1518 <1> ; 18/01/2016

1519 <1> ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)

1520 <1> ;

1521 <1> ; VIDEO.ASM - 06/10/85 VIDEO DISPLAY BIOS

1522 <1> ;

1523 <1> ; 08/07/2016

1524 00001B95 803D[C25E0000]07 <1> cmp byte [CRT\_MODE], 7 ; 6!?

1525 00001B9C 7607 <1> jna short read\_ac\_c

1526 00001B9E 31C0 <1> xor eax, eax

1527 00001BA0 E9B4F9FFFF <1> jmp \_video\_return

1528 <1> read\_ac\_c:

1529 00001BA5 E805000000 <1> call \_read\_ac\_current

1530 <1> ; 12/05/2016

1531 <1> ;jmp VIDEO\_RETURN

1532 00001BAA E9AAF9FFFF <1> jmp \_video\_return

1533 <1>

1534 <1> ;------------------------------------------------------------------------

1535 <1> ; READ\_AC\_CURRENT :

1536 <1> ; THIS ROUTINE READS THE ATTRIBUTE AND CHARACTER AT THE CURRENT :

1537 <1> ; CURSOR POSITION AND RETURNS THEM TO THE CALLER :

1538 <1> ; INPUT :

1539 <1> ; (AH) = CURRENT CRT MODE :

1540 <1> ; (BH) = DISPLAY PAGE ( ALPHA MODES ONLY ) :

1541 <1> ; (DS) = DATA SEGMENT :

1542 <1> ; (ES) = REGEN SEGMENT :

1543 <1> ; OUTPUT :

1544 <1> ; (AL) = CHARACTER READ :

1545 <1> ; (AH) = ATTRIBUTE READ :

1546 <1> ;------------------------------------------------------------------------

1547 <1>

1548 <1> \_read\_ac\_current:

1549 <1> ; 26/06/2016

1550 <1> ; 12/05/2016

1551 <1> ; 18/01/2016

1552 <1>

1553 <1> ;mov ah, [CRT\_MODE] ; current video mode

1554 <1> ;cmp ah, 4

1555 <1> ;jb short p10

1556 00001BAF 803D[C25E0000]04 <1> cmp byte [CRT\_MODE], 4

1557 00001BB6 0F83BB080000 <1> jnb GRAPHICS\_READ ; 26/06/2016

1558 <1>

1559 <1> ;cmp ah, 7 ; TEST FOR BW CARD

1560 <1> ;jne GRAPHICS\_READ

1561 <1> p10:

1562 00001BBC E801FEFFFF <1> call find\_position ; GET REGEN LOCATION AND PORT ADDRESS

1563 <1> ;

1564 <1> ; esi = regen location

1565 <1> ; dx = status port

1566 <1> ;

1567 00001BC1 8A25[C25E0000] <1> mov ah, [CRT\_MODE]

1568 00001BC7 80EC02 <1> sub ah, 2

1569 00001BCA D0EC <1> shr ah, 1

1570 00001BCC 7515 <1> jnz short p13

1571 <1>

1572 <1> ; WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80

1573 <1> p11:

1574 00001BCE FB <1> sti ; enable interrupts first

1575 00001BCF 3A3D[66580100] <1> cmp bh, [ACTIVE\_PAGE]

1576 00001BD5 750C <1> jne short p13

1577 00001BD7 FA <1> cli ; block interrupts for single loop

1578 00001BD8 EC <1> in al, dx ; get status from the adapter

1579 00001BD9 A801 <1> test al, RHRZ ; is horizontal retrace low

1580 00001BDB 75F1 <1> jnz short p11 ; wait until it is

1581 <1> p12: ; wait for either retrace high

1582 00001BDD EC <1> in al, dx ; get status again

1583 00001BDE A809 <1> test al, RVRT+RHRZ ; is horizontal or vertical retrace high

1584 00001BE0 74FB <1> jz short p12 ; wait until either retrace active

1585 00001BE2 FB <1> sti

1586 <1> p13:

1587 00001BE3 81C600800B00 <1> add esi, 0B8000h

1588 00001BE9 668B06 <1> mov ax, [esi]

1589 <1>

1590 00001BEC C3 <1> retn ; 18/01/2016

1591 <1>

1592 <1> WRITE\_AC\_CURRENT:

1593 <1> ; 08/07/2016

1594 <1> ; 26/06/2016

1595 <1> ; 24/06/2016

1596 <1> ; 12/05/2016

1597 <1> ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)

1598 <1> ;

1599 <1> ; VIDEO.ASM - 06/10/85 VIDEO DISPLAY BIOS

1600 <1> ;

1601 <1> ;----------------------------------------------------------------

1602 <1> ; WRITE\_AC\_CURRENT :

1603 <1> ; THTS ROUTINE WRITES THE ATTRIBUTE AND CHARACTER :

1604 <1> ; AT THE CURRENT CURSOR POSITION :

1605 <1> ; INPUT :

1606 <1> ; (AH) = CURRENT CRT MODE :

1607 <1> ; (BH) = DISPLAY PAGE :

1608 <1> ; (CX) = COUNT OF CHARACTERS TO WRITE :

1609 <1> ; (AL) = CHAR TO WRITE :

1610 <1> ; (BL) = ATTRIBUTE OF CHAR TO WRITE :

1611 <1> ; (DS) = DATA SEGMENT :

1612 <1> ; (ES) = REGEN SEGMENT :

1613 <1> ; OUTPUT :

1614 <1> ; DISPLAY REGEN BUFFER UPDATED :

1615 <1> ;----------------------------------------------------------------

1616 <1>

1617 <1> ; 08/07/2016

1618 00001BED 803D[C25E0000]07 <1> cmp byte [CRT\_MODE], 7 ; 6!?

1619 00001BF4 760A <1> jna short write\_ac\_c

1620 <1>

1621 00001BF6 E8F20A0000 <1> call vga\_write\_char\_attr

1622 00001BFB E954F9FFFF <1> jmp VIDEO\_RETURN

1623 <1>

1624 <1> write\_ac\_c:

1625 00001C00 E834000000 <1> call \_write\_c\_current

1626 <1>

1627 00001C05 0FB6F7 <1> movzx esi, bh ; video page number (0 to 7)

1628 00001C08 889E[CB5E0000] <1> mov [esi+chr\_attrib], bl ; color/attribute

1629 <1>

1630 00001C0E E941F9FFFF <1> jmp VIDEO\_RETURN

1631 <1>

1632 <1> WRITE\_C\_CURRENT:

1633 <1> ; 08/07/2016

1634 <1> ; 26/06/2016

1635 <1> ; 12/05/2016

1636 <1> ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)

1637 <1> ;

1638 <1> ; VIDEO.ASM - 06/10/85 VIDEO DISPLAY BIOS

1639 <1> ;

1640 <1> ;----------------------------------------------------------------

1641 <1> ; WRITE\_C\_CURRENT :

1642 <1> ; THIS ROUTINE WRITES THE CHARACTER AT :

1643 <1> ; THE CURRENT CURSOR POSITION, ATTRIBUTE UNCHANGED :

1644 <1> ; INPUT :

1645 <1> ; (AH) = CURRENT CRT MODE :

1646 <1> ; (BH) = DISPLAY PAGE :

1647 <1> ; (CX) = COUNT OF CHARACTERS TO WRITE :

1648 <1> ; (AL) = CHAR TO WRITE :

1649 <1> ; (DS) = DATA SEGMENT :

1650 <1> ; (ES) = REGEN SEGMENT :

1651 <1> ; OUTPUT :

1652 <1> ; DISPLAY REGEN BUFFER UPDATED :

1653 <1> ;----------------------------------------------------------------

1654 <1>

1655 <1> ; 08/07/2016

1656 00001C13 803D[C25E0000]07 <1> cmp byte [CRT\_MODE], 7 ; 6!?

1657 00001C1A 760A <1> jna short write\_c\_c

1658 <1>

1659 00001C1C E8CC0A0000 <1> call vga\_write\_char\_only

1660 00001C21 E92EF9FFFF <1> jmp VIDEO\_RETURN

1661 <1>

1662 <1> write\_c\_c:

1663 <1> ;and bh, 7 ; video page number (<= 7)

1664 00001C26 0FB6F7 <1> movzx esi, bh

1665 00001C29 8A9E[CB5E0000] <1> mov bl, [esi+chr\_attrib]

1666 <1>

1667 00001C2F E805000000 <1> call \_write\_c\_current

1668 00001C34 E91BF9FFFF <1> jmp VIDEO\_RETURN

1669 <1>

1670 <1> \_write\_c\_current: ; from 'write\_tty'

1671 <1> ; 26/06/2016

1672 <1> ; 24/06/2016

1673 <1> ; 12/05/2016

1674 <1> ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)

1675 <1> ; 30/08/2014 (Retro UNIX 386 v1)

1676 <1> ; 18/01/2014

1677 <1> ; 04/12/2013

1678 <1> ;

1679 <1> ; VIDEO.ASM - 06/10/85 VIDEO DISPLAY BIOS

1680 <1>

1681 <1> ;mov ah, [CRT\_MODE] ; current video mode

1682 <1> ;cmp ah, 4

1683 <1> ;jb short p40

1684 00001C39 803D[C25E0000]04 <1> cmp byte [CRT\_MODE], 4

1685 00001C40 0F8381070000 <1> jnb GRAPHICS\_WRITE ; 26/06/2016

1686 <1>

1687 <1> ;cmp ah, 7 ; TEST FOR BW CARD

1688 <1> ;jne GRAPHICS\_WRITE

1689 <1> p40:

1690 <1> ; al = character

1691 <1> ; bl = color/attribute

1692 <1> ; bh = video page

1693 <1> ; cx = count of characters to write

1694 00001C46 6652 <1> push dx

1695 00001C48 88DC <1> mov ah, bl ; color/attribute (12/05/2016)

1696 00001C4A 6650 <1> push ax ; save character & attribute/color

1697 00001C4C 6651 <1> push cx

1698 00001C4E E86FFDFFFF <1> call find\_position ; get regen location and port address

1699 00001C53 6659 <1> pop cx

1700 <1> ; esi = regen location

1701 <1> ; dx = status port

1702 <1> ;

1703 00001C55 81C600800B00 <1> add esi, 0B8000h ; 30/08/2014 (crt\_base)

1704 <1> ;

1705 00001C5B 8A25[C25E0000] <1> mov ah, [CRT\_MODE]

1706 00001C61 80EC02 <1> sub ah, 2

1707 00001C64 D0EC <1> shr ah, 1

1708 00001C66 7519 <1> jnz short p44 ; 26/06/2016

1709 <1>

1710 <1> ; WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80

1711 <1> p41:

1712 00001C68 FB <1> sti ; enable interrupts first

1713 00001C69 3A3D[66580100] <1> cmp bh, [ACTIVE\_PAGE]

1714 00001C6F 7510 <1> jne short p44

1715 00001C71 FA <1> cli ; block interrupts for single loop

1716 00001C72 EC <1> in al, dx ; get status from the adapter

1717 00001C73 A808 <1> test al, RVRT ; check for vertical retrace first

1718 00001C75 7509 <1> jnz short p43 ; Do fast write now if vertical retrace

1719 00001C77 A801 <1> test al, RHRZ ; is horizontal retrace low

1720 00001C79 75ED <1> jnz short p41 ; wait until it is

1721 <1> p42: ; wait for either retrace high

1722 00001C7B EC <1> in al, dx ; get status again

1723 00001C7C A809 <1> test al, RVRT+RHRZ ; is horizontal or vertical retrace high

1724 00001C7E 74FB <1> jz short p42 ; wait until either retrace active

1725 <1> p43:

1726 00001C80 FB <1> sti

1727 <1> p44:

1728 00001C81 668B0424 <1> mov ax, [esp] ; restore the character (al) & attribute (ah)

1729 00001C85 668906 <1> mov [esi], ax

1730 <1>

1731 00001C88 6649 <1> dec cx

1732 00001C8A 7404 <1> jz short p45

1733 <1>

1734 00001C8C 46 <1> inc esi

1735 00001C8D 46 <1> inc esi

1736 00001C8E EBD8 <1> jmp short p41

1737 <1> p45:

1738 00001C90 6658 <1> pop ax

1739 00001C92 665A <1> pop dx

1740 00001C94 C3 <1> retn

1741 <1>

1742 <1> ; 09/07/2016

1743 <1> ; 26/06/2016

1744 <1> ; 24/06/2016

1745 <1> ; 12/05/2016

1746 <1> ; 18/01/2016

1747 <1> ; 16/01/2016 - TRDOS 386 (TRDOS v2.0)

1748 <1> ; 30/06/2015

1749 <1> ; 27/06/2015

1750 <1> ; 11/03/2015

1751 <1> ; 02/09/2014

1752 <1> ; 30/08/2014

1753 <1> ; VIDEO FUNCTIONS

1754 <1> ; (write\_tty - Retro UNIX 8086 v1 - U9.ASM, 01/02/2014)

1755 <1>

1756 <1> WRITE\_TTY:

1757 <1> ; 09/12/2017

1758 <1> ; 09/07/2016

1759 <1> ; 01/07/2016

1760 <1> ; 26/06/2016

1761 <1> ; 24/06/2016

1762 <1> ; 13/05/2016

1763 <1> ; 12/05/2016

1764 <1> ; 30/01/2016

1765 <1> ; 18/01/2016

1766 <1> ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)

1767 <1> ; 13/08/2015

1768 <1> ; 02/09/2014

1769 <1> ; 30/08/2014 (Retro UNIX 386 v1 - beginning)

1770 <1> ; 01/02/2014 (Retro UNIX 8086 v1 - last update)

1771 <1> ; 03/12/2013 (Retro UNIX 8086 v1 - beginning)

1772 <1> ; (Modified registers: EAX, EBX, ECX, EDX, ESI, EDI)

1773 <1> ;

1774 <1> ; INPUT -> AL = Character to be written

1775 <1> ; BL = Color (Forecolor, Backcolor)

1776 <1> ; BH = Video Page (0 to 7)

1777 <1>

1778 <1> ; 09/07/2016

1779 00001C95 803D[C25E0000]07 <1> cmp byte [CRT\_MODE], 7

1780 00001C9C 760A <1> jna short write\_tty\_cga

1781 <1>

1782 00001C9E E8290D0000 <1> call vga\_write\_teletype

1783 00001CA3 E9ACF8FFFF <1> jmp VIDEO\_RETURN

1784 <1>

1785 <1> write\_tty\_cga:

1786 <1> ; 13/05/2016

1787 <1> ;call \_write\_tty

1788 <1> ; 01/07/2016

1789 00001CA8 E818000000 <1> call \_write\_tty\_m3

1790 00001CAD E9A2F8FFFF <1> jmp VIDEO\_RETURN

1791 <1>

1792 <1> RVRT equ 00001000b ; VIDEO VERTICAL RETRACE BIT

1793 <1> RHRZ equ 00000001b ; VIDEO HORIZONTAL RETRACE BIT

1794 <1>

1795 <1> ; Derived from "WRITE\_TTY" procedure of IBM "pc-at" rombios source code

1796 <1> ; (06/10/1985), 'video.asm', INT 10H, VIDEO\_IO

1797 <1> ;

1798 <1> ; 06/10/85 VIDEO DISPLAY BIOS

1799 <1> ;

1800 <1> ;--- WRITE\_TTY ------------------------------------------------------------------

1801 <1> ; :

1802 <1> ; THIS INTERFACE PROVIDES A TELETYPE LIKE INTERFACE TO THE :

1803 <1> ; VIDEO CARDS. THE INPUT CHARACTER IS WRITTEN TO THE CURRENT :

1804 <1> ; CURSOR POSITION, AND THE CURSOR IS MOVED TO THE NEXT POSITION. :

1805 <1> ; IF THE CURSOR LEAVES THE LAST COLUMN OF THE FIELD, THE COLUMN :

1806 <1> ; IS SET TO ZERO, AND THE ROW VALUE IS INCREMENTED. IF THE ROW :

1807 <1> ; ROW VALUE LEAVES THE FIELD, THE CURSOR IS PLACED ON THE LAST ROW, :

1808 <1> ; FIRST COLUMN, AND THE ENTIRE SCREEN IS SCROLLED UP ONE LINE. :

1809 <1> ; WHEN THE SCREEN IS SCROLLED UP, THE ATTRIBUTE FOR FILLING THE :

1810 <1> ; NEWLY BLANKED LINE IS READ FROM THE CURSOR POSITION ON THE PREVIOUS :

1811 <1> ; LINE BEFORE THE SCROLL, IN CHARACTER MODE. IN GRAPHICS MODE, :

1812 <1> ; THE 0 COLOR IS USED. :

1813 <1> ; ENTRY -- :

1814 <1> ; (AH) = CURRENT CRT MODE :

1815 <1> ; (AL) = CHARACTER TO BE WRITTEN :

1816 <1> ; NOTE THAT BACK SPACE, CARRIAGE RETURN, BELL AND LINE FEED ARE :

1817 <1> ; HANDLED AS COMMANDS RATHER THAN AS DISPLAY GRAPHICS CHARACTERS :

1818 <1> ; (BL) = FOREGROUND COLOR FOR CHAR WRITE IF CURRENTLY IN A GRAPHICS MODE :

1819 <1> ; EXIT -- :

1820 <1> ; ALL REGISTERS SAVED :

1821 <1> ;--------------------------------------------------------------------------------

1822 <1>

1823 <1> ; 09/12/2017

1824 <1> ; 08/07/2016

1825 <1> ; 26/06/2016

1826 <1> ; 24/06/2016

1827 <1> \_write\_tty: ; 13/05/2016

1828 00001CB2 FA <1> cli

1829 <1> ;

1830 <1> ; 01/09/2014

1831 00001CB3 803D[C25E0000]03 <1> cmp byte [CRT\_MODE], 3

1832 00001CBA 7409 <1> je short \_write\_tty\_m3

1833 <1> ;

1834 <1> set\_mode\_3:

1835 00001CBC 53 <1> push ebx

1836 00001CBD 50 <1> push eax

1837 00001CBE E8A2F8FFFF <1> call \_set\_mode

1838 00001CC3 58 <1> pop eax

1839 00001CC4 5B <1> pop ebx

1840 <1> ;

1841 <1> \_write\_tty\_m3: ; 24/06/2016 (m3 -> \_write\_tty\_m3)

1842 00001CC5 0FB6F7 <1> movzx esi, bh ; 12/05/2016

1843 00001CC8 66D1E6 <1> shl si, 1

1844 00001CCB 81C6[56580100] <1> add esi, CURSOR\_POSN

1845 00001CD1 668B16 <1> mov dx, [esi]

1846 <1> ;

1847 <1> ; dx now has the current cursor position

1848 <1> ;

1849 00001CD4 3C0D <1> cmp al, 0Dh ; CR ; is it carriage return or control character

1850 00001CD6 7636 <1> jbe short u8

1851 <1> ;

1852 <1> ; write the char to the screen

1853 <1> u0:

1854 <1> ; al = character

1855 <1> ; bl = attribute/color

1856 <1> ; bh = video page number (0 to 7)

1857 <1> ;

1858 00001CD8 66B90100 <1> mov cx, 1 ; 24/06/2016

1859 <1> ; cx = count of characters to write

1860 <1> ;

1861 00001CDC E858FFFFFF <1> call \_write\_c\_current ; 16/01/2015

1862 <1> ;

1863 <1> ; position the cursor for next char

1864 00001CE1 FEC2 <1> inc dl ; next column

1865 00001CE3 3A15[C45E0000] <1> cmp dl, [CRT\_COLS] ; test for column overflow

1866 00001CE9 755D <1> jne \_set\_cpos

1867 00001CEB B200 <1> mov dl, 0 ; column = 0

1868 <1> u10: ; (line feed found)

1869 00001CED 80FE18 <1> cmp dh, 25-1 ; check for last row

1870 00001CF0 7218 <1> jb short u6

1871 <1> ;

1872 <1> ; scroll required

1873 <1> u1:

1874 <1> ; SET CURSOR POSITION (04/12/2013)

1875 00001CF2 E851000000 <1> call \_set\_cpos

1876 <1> ;

1877 <1> ; determine value to fill with during scroll

1878 <1> u2:

1879 <1> ; bh = video page number

1880 <1> ;

1881 00001CF7 E8B3FEFFFF <1> call \_read\_ac\_current ; 18/01/2016

1882 <1> ;

1883 <1> ; al = character, ah = attribute

1884 <1> ; bh = video page number

1885 <1> u3:

1886 <1> ;;mov ax, 0601h ; scroll one line

1887 <1> ;;sub cx, cx ; upper left corner

1888 <1> ;;mov dh, 25-1 ; lower right row

1889 <1> ;;;mov dl, [CRT\_COLS]

1890 <1> ;mov dl, 80 ; lower right column

1891 <1> ;;dec dl

1892 <1> ;;mov dl, 79

1893 <1>

1894 <1> ;;call scroll\_up ; 04/12/2013

1895 <1> ;;; 11/03/2015

1896 <1> ; 02/09/2014

1897 <1> ;;;mov cx, [crt\_ulc] ; Upper left corner (0000h)

1898 <1> ;;;mov dx, [crt\_lrc] ; Lower right corner (184Fh)

1899 <1> ; 11/03/2015

1900 00001CFC 6629C9 <1> sub cx, cx

1901 00001CFF 66BA4F18 <1> mov dx, 184Fh ; dl = 79 (column), dh = 24 (row)

1902 <1> ;

1903 00001D03 B001 <1> mov al, 1 ; scroll 1 line up

1904 <1> ; ah = attribute

1905 <1> ;mov bl, al ; 12/05/2016

1906 00001D05 E900FDFFFF <1> jmp \_scroll\_up ; 16/01/2016

1907 <1> ;u4:

1908 <1> ;;int 10h ; video-call return

1909 <1> ; scroll up the screen

1910 <1> ; tty return

1911 <1> ;u5:

1912 <1> ;retn ; return to the caller

1913 <1>

1914 <1> u6: ; set-cursor-inc

1915 00001D0A FEC6 <1> inc dh ; next row

1916 <1> ; set cursor

1917 <1> ;u7:

1918 <1> ;;mov ah, 02h

1919 <1> ;;jmp short u4 ; establish the new cursor

1920 <1> ;call \_set\_cpos

1921 <1> ;jmp short u5

1922 00001D0C EB3A <1> jmp \_set\_cpos

1923 <1>

1924 <1> ; check for control characters

1925 <1> u8:

1926 00001D0E 7436 <1> je short u9

1927 00001D10 3C0A <1> cmp al, 0Ah ; is it a line feed (0Ah)

1928 00001D12 74D9 <1> je short u10

1929 00001D14 3C07 <1> cmp al, 07h ; is it a bell

1930 00001D16 747A <1> je short u11

1931 00001D18 3C08 <1> cmp al, 08h ; is it a backspace

1932 <1> ;jne short u0

1933 00001D1A 7422 <1> je short bs ; 12/12/2013

1934 <1> ; 12/12/2013 (tab stop)

1935 00001D1C 3C09 <1> cmp al, 09h ; is it a tab stop

1936 00001D1E 75B8 <1> jne short u0

1937 00001D20 88D0 <1> mov al, dl

1938 <1> ;cbw

1939 00001D22 30E4 <1> xor ah, ah ; 09/12/2017

1940 00001D24 B108 <1> mov cl, 8

1941 00001D26 F6F1 <1> div cl

1942 00001D28 28E1 <1> sub cl, ah

1943 <1> ts:

1944 <1> ; 02/09/2014

1945 <1> ; 01/09/2014

1946 00001D2A B020 <1> mov al, 20h

1947 <1> tsloop:

1948 00001D2C 6651 <1> push cx

1949 00001D2E 6650 <1> push ax

1950 <1> ;mov bh, [ACTIVE\_PAGE]

1951 00001D30 E890FFFFFF <1> call \_write\_tty\_m3 ; 24/06/2016 (m3 -> \_write\_tty\_m3)

1952 00001D35 6658 <1> pop ax ; ah = attribute/color

1953 00001D37 6659 <1> pop cx

1954 00001D39 FEC9 <1> dec cl

1955 00001D3B 75EF <1> jnz short tsloop

1956 00001D3D C3 <1> retn

1957 <1> bs:

1958 <1> ; back space found

1959 <1>

1960 00001D3E 08D2 <1> or dl, dl ; is it already at start of line

1961 <1> ;je short u7 ; set\_cursor

1962 00001D40 7406 <1> jz short \_set\_cpos

1963 00001D42 664A <1> dec dx ; no -- just move it back

1964 <1> ;jmp short u7

1965 00001D44 EB02 <1> jmp short \_set\_cpos

1966 <1>

1967 <1> ; carriage return found

1968 <1> u9:

1969 00001D46 B200 <1> mov dl, 0 ; move to first column

1970 <1> ;jmp short u7

1971 <1> ;jmp short \_set\_cpos ; 30/01/2016

1972 <1>

1973 <1> ; line feed found

1974 <1> ;u10:

1975 <1> ; cmp dh, 25-1 ; bottom of screen

1976 <1> ; jne short u6 ; no, just set the cursor

1977 <1> ; jmp u1 ; yes, scroll the screen

1978 <1>

1979 <1> \_set\_cpos:

1980 <1> ; 12/05/2016 - TRDOS 386 (TRDOS v2.0)

1981 <1> ; 27/06/2015

1982 <1> ; 01/09/2014

1983 <1> ; 30/08/2014 (Retro UNIX 386 v1)

1984 <1> ;

1985 <1> ; 04/12/2013 - 12/12/2013 (Retro UNIX 8086 v1)

1986 <1> ;

1987 <1> ; VIDEO.ASM - 06/10/85 VIDEO DISPLAY BIOS

1988 <1> ;

1989 <1> ;----------------------------------------------

1990 <1> ; SET\_CPOS

1991 <1> ; THIS ROUTINE SETS THE CURRENT CURSOR POSITION TO THE

1992 <1> ; NEW X-Y VALUES PASSED

1993 <1> ; INPUT

1994 <1> ; DX - ROW,COLUMN OF NEW CURSOR

1995 <1> ; BH - DISPLAY PAGE OF CURSOR

1996 <1> ; OUTPUT

1997 <1> ; CURSOR ID SET AT 6845 IF DISPLAY PAGE IS CURRENT DISPLAY

1998 <1> ;----------------------------------------------

1999 <1> ;

2000 00001D48 BE[56580100] <1> mov esi, CURSOR\_POSN

2001 00001D4D 0FB6C7 <1> movzx eax, bh ; BH = video page number

2002 <1> ; or al, al

2003 <1> ; jz short \_set\_cpos\_0

2004 00001D50 D0E0 <1> shl al, 1 ; word offset

2005 00001D52 01C6 <1> add esi, eax

2006 <1> ;\_set\_cpos\_0:

2007 00001D54 668916 <1> mov [esi], dx ; save the pointer

2008 00001D57 383D[66580100] <1> cmp [ACTIVE\_PAGE], bh

2009 00001D5D 7532 <1> jne short m17

2010 <1> ;call m18 ; CURSOR SET

2011 <1> ;m17: ; SET\_CPOS\_RETURN

2012 <1> ; 01/09/2014

2013 <1> ; retn

2014 <1> ; DX = row/column

2015 <1> m18:

2016 00001D5F E84CFCFFFF <1> call position ; determine location in regen buffer

2017 00001D64 668B0D[54580100] <1> mov cx, [CRT\_START]

2018 00001D6B 6601C1 <1> add cx, ax ; add char position in regen buffer

2019 <1> ; to the start address (offset) for this page

2020 00001D6E 66D1E9 <1> shr cx, 1 ; divide by 2 for char only count

2021 00001D71 B40E <1> mov ah, 14 ; register number for cursor

2022 <1> ;call m16 ; output value to the 6845

2023 <1> ;retn

2024 <1>

2025 <1> ;----- THIS ROUTINE OUTPUTS THE CX REGISTER

2026 <1> ; TO THE 6845 REGISTERS NAMED IN (AH)

2027 <1> m16:

2028 00001D73 FA <1> cli

2029 <1> ;mov dx, [addr\_6845] ; address register

2030 00001D74 66BAD403 <1> mov dx, 03D4h ; I/O address of color card

2031 00001D78 88E0 <1> mov al, ah ; get value

2032 00001D7A EE <1> out dx, al ; register set

2033 00001D7B 6642 <1> inc dx ; data register

2034 00001D7D EB00 <1> jmp $+2 ; i/o delay

2035 00001D7F 88E8 <1> mov al, ch ; data

2036 00001D81 EE <1> out dx, al

2037 00001D82 664A <1> dec dx

2038 00001D84 88E0 <1> mov al, ah

2039 00001D86 FEC0 <1> inc al ; point to other data register

2040 00001D88 EE <1> out dx, al ; set for second register

2041 00001D89 6642 <1> inc dx

2042 00001D8B EB00 <1> jmp $+2 ; i/o delay

2043 00001D8D 88C8 <1> mov al, cl ; second data value

2044 00001D8F EE <1> out dx, al

2045 00001D90 FB <1> sti

2046 <1> m17:

2047 00001D91 C3 <1> retn

2048 <1>

2049 <1> beeper:

2050 <1> ; 04/08/2016

2051 <1> ; 12/05/2016 - TRDOS 386 (TRDOS v2.0)

2052 <1> ; 30/08/2014 (Retro UNIX 386 v1)

2053 <1> ; 18/01/2014

2054 <1> ; 03/12/2013

2055 <1> ; bell found

2056 <1> u11:

2057 00001D92 FB <1> sti

2058 00001D93 3A3D[66580100] <1> cmp bh, [ACTIVE\_PAGE]

2059 00001D99 7551 <1> jne short u12 ; Do not sound the beep

2060 <1> ; if it is not written on the active page

2061 <1> beeper\_gfx: ; 04/08/2016

2062 00001D9B 66B93305 <1> mov cx, 1331 ; divisor for 896 hz tone

2063 00001D9F B31F <1> mov bl, 31 ; set count for 31/64 second for beep

2064 <1> ;call beep ; sound the pod bell

2065 <1> ;jmp short u5 ; tty\_return

2066 <1> ;retn

2067 <1>

2068 <1> TIMER equ 040h ; 8254 TIMER - BASE ADDRESS

2069 <1> PORT\_B equ 061h ; PORT B READ/WRITE DIAGNOSTIC REGISTER

2070 <1> GATE2 equ 00000001b ; TIMER 2 INPUT CATE CLOCK BIT

2071 <1> SPK2 equ 00000010b ; SPEAKER OUTPUT DATA ENABLE BIT

2072 <1>

2073 <1> beep:

2074 <1> ; 07/02/2015

2075 <1> ; 30/08/2014 (Retro UNIX 386 v1)

2076 <1> ; 18/01/2014

2077 <1> ; 03/12/2013

2078 <1> ;

2079 <1> ; TEST4.ASM - 06/10/85 POST AND BIOS UTILITY ROUTINES

2080 <1> ;

2081 <1> ; ROUTINE TO SOUND THE BEEPER USING TIMER 2 FOR TONE

2082 <1> ;

2083 <1> ; ENTRY:

2084 <1> ; (BL) = DURATION COUNTER ( 1 FOR 1/64 SECOND )

2085 <1> ; (CX) = FREQUENCY DIVISOR (1193180/FREQUENCY) (1331 FOR 886 HZ)

2086 <1> ; EXIT: :

2087 <1> ; (AX),(BL),(CX) MODIFIED.

2088 <1>

2089 00001DA1 9C <1> pushf ; 18/01/2014 ; save interrupt status

2090 00001DA2 FA <1> cli ; block interrupts during update

2091 00001DA3 B0B6 <1> mov al, 10110110b ; select timer 2, lsb, msb binary

2092 00001DA5 E643 <1> out TIMER+3, al ; write timer mode register

2093 00001DA7 EB00 <1> jmp $+2 ; I/O delay

2094 00001DA9 88C8 <1> mov al, cl ; divisor for hz (low)

2095 00001DAB E642 <1> out TIMER+2,AL ; write timer 2 count - lsb

2096 00001DAD EB00 <1> jmp $+2 ; I/O delay

2097 00001DAF 88E8 <1> mov al, ch ; divisor for hz (high)

2098 00001DB1 E642 <1> out TIMER+2, al ; write timer 2 count - msb

2099 00001DB3 E461 <1> in al, PORT\_B ; get current setting of port

2100 00001DB5 88C4 <1> mov ah, al ; save that setting

2101 00001DB7 0C03 <1> or al, GATE2+SPK2 ; gate timer 2 and turn speaker on

2102 00001DB9 E661 <1> out PORT\_B, al ; and restore interrupt status

2103 <1> ;popf ; 18/01/2014

2104 00001DBB FB <1> sti

2105 <1> g7: ; 1/64 second per count (bl)

2106 00001DBC B90B040000 <1> mov ecx, 1035 ; delay count for 1/64 of a second

2107 00001DC1 E827000000 <1> call waitf ; go to beep delay 1/64 count

2108 00001DC6 FECB <1> dec bl ; (bl) length count expired?

2109 00001DC8 75F2 <1> jnz short g7 ; no - continue beeping speaker

2110 <1> ;

2111 <1> ;pushf ; save interrupt status

2112 00001DCA FA <1> cli ; 18/01/2014 ; block interrupts during update

2113 00001DCB E461 <1> in al, PORT\_B ; get current port value

2114 <1> ;or al, not (GATE2+SPK2) ; isolate current speaker bits in case

2115 00001DCD 0CFC <1> or al, ~(GATE2+SPK2)

2116 00001DCF 20C4 <1> and ah, al ; someone turned them off during beep

2117 00001DD1 88E0 <1> mov al, ah ; recover value of port

2118 <1> ;or al, not (GATE2+SPK2) ; force speaker data off

2119 00001DD3 0CFC <1> or al, ~(GATE2+SPK2) ; isolate current speaker bits in case

2120 00001DD5 E661 <1> out PORT\_B, al ; and stop speaker timer

2121 <1> ;popf ; restore interrupt flag state

2122 00001DD7 FB <1> sti

2123 00001DD8 B90B040000 <1> mov ecx, 1035 ; force 1/64 second delay (short)

2124 00001DDD E80B000000 <1> call waitf ; minimum delay between all beeps

2125 <1> ;pushf ; save interrupt status

2126 00001DE2 FA <1> cli ; block interrupts during update

2127 00001DE3 E461 <1> in al, PORT\_B ; get current port value in case

2128 00001DE5 2403 <1> and al, GATE2+SPK2 ; someone turned them on

2129 00001DE7 08E0 <1> or al, ah ; recover value of port\_b

2130 00001DE9 E661 <1> out PORT\_B, al ; restore speaker status

2131 00001DEB 9D <1> popf ; restore interrupt flag state

2132 <1> u12:

2133 00001DEC C3 <1> retn

2134 <1>

2135 <1> REFRESH\_BIT equ 00010000b ; REFRESH TEST BIT

2136 <1>

2137 <1> WAITF:

2138 <1> waitf:

2139 <1> ; 30/08/2014 (Retro UNIX 386 v1)

2140 <1> ; 03/12/2013

2141 <1> ;

2142 <1> ; push ax ; save work register (ah)

2143 <1> ;waitf1:

2144 <1> ; use timer 1 output bits

2145 <1> ; in al, PORT\_B ; read current counter output status

2146 <1> ; and al, REFRESH\_BIT ; mask for refresh determine bit

2147 <1> ; cmp al, ah ; did it just change

2148 <1> ; je short waitf1 ; wait for a change in output line

2149 <1> ; ;

2150 <1> ; mov ah, al ; save new lflag state

2151 <1> ; loop waitf1 ; decrement half cycles till count end

2152 <1> ; ;

2153 <1> ; pop ax ; restore (ah)

2154 <1> ; retn ; return (cx)=0

2155 <1>

2156 <1> ; 06/02/2015 (unix386.s <-- dsectrm2.s)

2157 <1> ; 17/12/2014 (dsectrm2.s)

2158 <1> ; WAITF

2159 <1> ; /// IBM PC-XT Model 286 System BIOS Source Code - Test 4 - 06/10/85 ///

2160 <1> ;

2161 <1> ;---WAITF-----------------------------------------------------------------------

2162 <1> ; FIXED TIME WAIT ROUTINE (HARDWARE CONTROLLED - NOT PROCESSOR)

2163 <1> ; ENTRY:

2164 <1> ; (CX) = COUNT OF 15.085737 MICROSECOND INTERVALS TO WAIT

2165 <1> ; MEMORY REFRESH TIMER 1 OUTPUT USED AS REFERENCE

2166 <1> ; EXIT:

2167 <1> ; AFTER (CX) TIME COUNT (PLUS OR MINUS 16 MICROSECONDS)

2168 <1> ; (CX) = 0

2169 <1> ;-------------------------------------------------------------------------------

2170 <1>

2171 <1> ; Refresh period: 30 micro seconds (15-80 us)

2172 <1> ; (16/12/2014 - AWARDBIOS 1999 - ATORGS.ASM, WAIT\_REFRESH)

2173 <1>

2174 <1> ;WAITF: ; DELAY FOR (CX)\*15.085737 US

2175 00001DED 6650 <1> PUSH AX ; SAVE WORK REGISTER (AH)

2176 <1> ; 16/12/2014

2177 <1> ;shr cx, 1 ; convert to count of 30 micro seconds

2178 00001DEF D1E9 <1> shr ecx, 1 ; 21/02/2015

2179 <1> ;17/12/2014

2180 <1> ;WAITF1:

2181 <1> ; IN AL, PORT\_B ;061h ; READ CURRENT COUNTER OUTPUT STATUS

2182 <1> ; AND AL, REFRESH\_BIT ;00010000b ; MASK FOR REFRESH DETERMINE BIT

2183 <1> ; CMP AL, AH ; DID IT JUST CHANGE

2184 <1> ; JE short WAITF1 ; WAIT FOR A CHANGE IN OUTPUT LINE

2185 <1> ; MOV AH, AL ; SAVE NEW FLAG STATE

2186 <1> ; LOOP WAITF1 ; DECREMENT HALF CYCLES TILL COUNT END

2187 <1> ;

2188 <1> ; 17/12/2014

2189 <1> ;

2190 <1> ; Modification from 'WAIT\_REFRESH' procedure of AWARD BIOS - 1999

2191 <1> ;

2192 <1> ;WAIT\_REFRESH: Uses port 61, bit 4 to have CPU speed independent waiting.

2193 <1> ; INPUT: CX = number of refresh periods to wait

2194 <1> ; (refresh periods = 1 per 30 microseconds on most machines)

2195 <1> WR\_STATE\_0:

2196 00001DF1 E461 <1> IN AL,PORT\_B ; IN AL,SYS1

2197 00001DF3 A810 <1> TEST AL,010H

2198 00001DF5 74FA <1> JZ SHORT WR\_STATE\_0

2199 <1> WR\_STATE\_1:

2200 00001DF7 E461 <1> IN AL,PORT\_B ; IN AL,SYS1

2201 00001DF9 A810 <1> TEST AL,010H

2202 00001DFB 75FA <1> JNZ SHORT WR\_STATE\_1

2203 00001DFD E2F2 <1> LOOP WR\_STATE\_0

2204 <1> ;

2205 00001DFF 6658 <1> POP AX ; RESTORE (AH)

2206 00001E01 C3 <1> RETn ; (CX) = 0

2207 <1>

2208 <1> ; 09/07/2016

2209 <1> ; 01/07/2016

2210 <1> ; 24/06/2016

2211 <1> ; 23/06/2016 - TRDOS 386 (TRDOS v2.0)

2212 <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)

2213 <1> ;-------------------------------------------------------------------------------

2214 <1> ; WRITE\_STRING :

2215 <1> ; THIS ROUTINE WRITES A STRING OF CHARACTERS TO THE CRT. :

2216 <1> ; INPUT :

2217 <1> ; (AL) = WRITE STRING COMMAND 0 - 3 :

2218 <1> ; (BH) = DISPLAY PAGE (ACTIVE PAGE) :

2219 <1> ; (CX) = COUNT OF CHARACTERS TO WRITE, IF (CX) = 0 THEN RETURN :

2220 <1> ; (DX) = CURSOR POSITION FOR START OF STRING WRITE :

2221 <1> ; (BL) = ATTRIBUTE OF CHARACTER TO WRITE IF (AL) = 0 OR (AL) = 1 :

2222 <1> ; (eBP) = SOURCE STRING OFFSET :

2223 <1> ; OUTPUT :

2224 <1> ; NONE :

2225 <1> ;-------------------------------------------------------------------------------

2226 <1>

2227 <1> ; AL = 00h: Assign all characters the attribute in BL; do not update cursor

2228 <1> ; AL = 01h: Assign all characters the attribute in BL; update cursor

2229 <1> ; AL = 02h: Use attributes in string; do not update cursor

2230 <1> ; AL = 03h: Use attributes in string; update cursor

2231 <1>

2232 <1> WRITE\_STRING:

2233 <1> ; 12/09/2016

2234 <1> ; 09/07/2016

2235 <1> ;cmp byte [CRT\_MODE], 7 ; 6?!

2236 <1> ;ja VIDEO\_RETURN ; not a valid function for VGA modes

2237 <1> ;

2238 00001E02 A2[D0650100] <1> mov [w\_str\_cmd], al ; save (AL) command

2239 00001E07 3C04 <1> CMP AL, 4 ; TEST FOR INVALID WRITE STRING OPTION

2240 00001E09 0F8345F7FFFF <1> JNB VIDEO\_RETURN ; IF OPTION INVALID THEN RETURN

2241 <1>

2242 <1> ;JCXZ VIDEO\_RETURN ; IF ZERO LENGTH STRING THEN RETURN

2243 <1>

2244 00001E0F 67E35E <1> jcxz P55 ; 01/07/2016

2245 <1>

2246 <1>

2247 <1> ; 01/07/2016

2248 <1> ;and ecx, 0FFFFh

2249 <1> ; ECX = byte count

2250 <1> ;push ecx

2251 00001E12 89EE <1> mov esi, ebp ; user buffer

2252 00001E14 BF00000700 <1> mov edi, Cluster\_Buffer ; system buffer

2253 00001E19 E8A5C90000 <1> call transfer\_from\_user\_buffer

2254 <1> ;pop ecx

2255 00001E1E 0F8230F7FFFF <1> jc VIDEO\_RETURN

2256 <1> ; ecx = transfer (byte) count = character count

2257 00001E24 BD00000700 <1> mov ebp, Cluster\_Buffer

2258 <1> ; 12/09/2016

2259 00001E29 803D[C25E0000]07 <1> cmp byte [CRT\_MODE], 7 ; 6?!

2260 00001E30 0F879F000000 <1> ja vga\_write\_string

2261 <1> ;

2262 00001E36 0FB6F7 <1> movzx esi, bh ; GET CURRENT CURSOR PAGE

2263 00001E39 66D1E6 <1> SAL SI,1 ; CONVERT TO PAGE OFFSET (SI= PAGE)

2264 <1> ; \*\*\*\*\*

2265 00001E3C 66FFB6[56580100] <1> PUSH word [eSI+CURSOR\_POSN] ; SAVE CURRENT CURSOR POSITION IN STACK

2266 <1>

2267 <1> ;MOV AX,0200H ; SET NEW CURSOR POSITION

2268 <1> ;INT 10H

2269 <1> P50next:

2270 00001E43 53 <1> push ebx ; \*\*\*\*

2271 00001E44 51 <1> push ecx ; \*\*\*

2272 00001E45 56 <1> push esi ; \*\*

2273 00001E46 52 <1> push edx ; \*

2274 00001E47 E8FCFEFFFF <1> call \_set\_cpos

2275 <1> P50:

2276 00001E4C 8A4500 <1> MOV AL, [eBP] ; GET CHARACTER FROM INPUT STRING

2277 00001E4F 45 <1> INC eBP ; BUMP POINTER TO CHARACTER

2278 <1>

2279 <1> ;----- TEST FOR SPECIAL CHARACTER'S

2280 <1>

2281 00001E50 3C08 <1> CMP AL, 08H ; IS IT A BACKSPACE

2282 00001E52 740C <1> JE short P51 ; BACK\_SPACE

2283 00001E54 3C0D <1> CMP AL, 0Dh ; CR ; IS IT CARRIAGE RETURN

2284 00001E56 7408 <1> JE short P51 ; CAR\_RET

2285 00001E58 3C0A <1> CMP AL, 0Ah ; LF ; IS IT A LINE FEED

2286 00001E5A 7404 <1> JE short P51 ; LINE\_FEED

2287 00001E5C 3C07 <1> CMP AL, 07h ; IS IT A BELL

2288 00001E5E 7515 <1> JNE short P52 ; IF NOT THEN DO WRITE CHARACTER

2289 <1> P51:

2290 <1> ;MOV AH,0EH ; TTY\_CHARACTER\_WRITE

2291 <1> ;INT 10H ; WRITE TTY CHARACTER TO THE CRT

2292 <1>

2293 00001E60 E860FEFFFF <1> call \_write\_tty\_m3

2294 <1>

2295 00001E65 5A <1> pop edx ; \*

2296 00001E66 5E <1> pop esi ; \*\*

2297 <1>

2298 00001E67 668B96[56580100] <1> MOV DX, [eSI+CURSOR\_POSN] ; GET CURRENT CURSOR POSITION

2299 00001E6E EB46 <1> JMP SHORT P54 ; SET CURSOR POSITION AND CONTINUE

2300 <1> P55:

2301 00001E70 E9DFF6FFFF <1> JMP VIDEO\_RETURN

2302 <1> P52:

2303 00001E75 66B90100 <1> MOV CX, 1 ; SET CHARACTER WRITE AMOUNT TO ONE

2304 00001E79 803D[D0650100]02 <1> CMP byte [w\_str\_cmd], 2 ; IS THE ATTRIBUTE IN THE STRING

2305 00001E80 7204 <1> JB short P53 ; IF NOT THEN SKIP

2306 00001E82 8A5D00 <1> MOV BL, [eBP] ; ELSE GET NEW ATTRIBUTE

2307 00001E85 45 <1> INC eBP ; BUMP STRING POINTER

2308 <1> P53:

2309 <1> ;MOV AH,09H ; GOT\_CHARACTER

2310 <1> ;INT 10H ; WRITE CHARACTER TO THE CRT

2311 <1>

2312 00001E86 E8AEFDFFFF <1> call \_write\_c\_current

2313 <1>

2314 00001E8B 5A <1> pop edx ; \*

2315 <1>

2316 00001E8C 0FB6F7 <1> movzx esi, bh ; video page number (0 to 7)

2317 00001E8F 889E[CB5E0000] <1> mov [esi+chr\_attrib], bl ; color/attribute

2318 <1>

2319 00001E95 FEC2 <1> INC DL ; INCREMENT COLUMN COUNTER

2320 00001E97 3A15[C45E0000] <1> CMP DL, [CRT\_COLS] ; IF COLS ARE WITHIN RANGE FOR THIS MODE

2321 00001E9D 7217 <1> JB short P54 ; THEN GO TO COLUMNS SET

2322 00001E9F FEC6 <1> INC DH ; BUMP ROW COUNTER BY ONE

2323 00001EA1 28D2 <1> SUB DL, DL ; SET COLUMN COUNTER TO ZERO

2324 00001EA3 80FE19 <1> CMP DH, 25 ; IF ROWS ARE LESS THAN 25 THEN

2325 00001EA6 720E <1> JB short P54 ; GO TO ROWS\_COLUMNS\_SET

2326 <1>

2327 00001EA8 66B80A0E <1> MOV AX,0E0AH ; ELSE SCROLL SCREEN

2328 <1> ;INT 10H ; RESET ROW COUNTER TO 24

2329 <1>

2330 00001EAC E814FEFFFF <1> call \_write\_tty\_m3

2331 <1>

2332 00001EB1 66BA0018 <1> mov dx, 1800h ; Column = 0, Row = 24

2333 00001EB5 5E <1> pop esi ; \*\*

2334 <1> P54:

2335 <1> ; ROW\_COLUMNS\_SET

2336 <1> ;MOV AX,0200H ; SET NEW CURSOR POSITION COMMAND

2337 <1> ;INT 10H ; ESTABLISH NEW CURSOR POSITION

2338 <1>

2339 00001EB6 59 <1> pop ecx ; \*\*\*

2340 00001EB7 5B <1> pop ebx ; \*\*\*\*

2341 <1>

2342 <1> ;LOOP P50 ; DO IT ONCE MORE UNTIL (CX) = ZERO

2343 00001EB8 6649 <1> dec cx

2344 00001EBA 7587 <1> jnz short P50next

2345 <1>

2346 00001EBC 665A <1> POP DX ; \*\*\*\*\* ; RESTORE OLD CURSOR COORDINATES

2347 <1>

2348 00001EBE F605[D0650100]01 <1> test byte [w\_str\_cmd], 1 ; IF CURSOR WAS NOT TO BE MOVED

2349 00001EC5 0F8589F6FFFF <1> JNZ VIDEO\_RETURN ; THEN EXIT WITHOUT RESETTING OLD VALUE

2350 <1>

2351 <1> ;MOV AX,0200H ; ELSE RESTORE OLD CURSOR POSITION

2352 <1> ;INT 10H

2353 <1> ; DONE - EXIT WRITE STRING

2354 00001ECB E878FEFFFF <1> call \_set\_cpos

2355 00001ED0 E97FF6FFFF <1> JMP VIDEO\_RETURN ; RETURN TO CALLER

2356 <1>

2357 <1> vga\_write\_string:

2358 <1> ; 12/09/2016 - TRDOS 386 (TRDOS v2.0)

2359 <1> ;

2360 <1> ; derived from 'Plex86/Bochs VGABios' source code

2361 <1> ; vgabios-0.7a (2011)

2362 <1> ; by the LGPL VGABios developers Team (2001-2008)

2363 <1> ; 'vgabios.c', ' biosfn\_write\_string'

2364 <1>

2365 <1> ; INPUT :

2366 <1> ; (AL) = WRITE STRING COMMAND 0 - 3 :

2367 <1> ; (BH) = DISPLAY PAGE (ACTIVE PAGE) :

2368 <1> ; (CX) = COUNT OF CHARACTERS TO WRITE, IF (CX) = 0 THEN RETURN :

2369 <1> ; (DX) = CURSOR POSITION FOR START OF STRING WRITE :

2370 <1> ; (BL) = ATTRIBUTE OF CHARACTER TO WRITE IF (AL) = 0 OR (AL) = 1 :

2371 <1> ; (eBP) = SOURCE STRING OFFSET :

2372 <1> ; OUTPUT :

2373 <1> ; NONE :

2374 <1> ;-------------------------------------------------------------------------;

2375 <1>

2376 <1> ; AL = 00h: Assign all characters the attribute in BL; do not update cursor

2377 <1> ; AL = 01h: Assign all characters the attribute in BL; update cursor

2378 <1> ; AL = 02h: Use attributes in string; do not update cursor

2379 <1> ; AL = 03h: Use attributes in string; update cursor

2380 <1>

2381 <1> ; biosfn\_write\_string(GET\_AL(),GET\_BH(),GET\_BL(),CX,GET\_DH(),GET\_DL(),ES,BP);

2382 <1> ; static void biosfn\_write\_string (flag,page,attr,count,row,col,seg,offset)

2383 <1>

2384 <1> ; // Read curs info for the page

2385 <1> ; biosfn\_get\_cursor\_pos(page,&dummy,&oldcurs);

2386 <1> ; bh = video page = 0

2387 <1> ;movzx esi, word [CURSOR\_POSN] ; current cursor position for video page 0

2388 <1>

2389 <1> ; // if row=0xff special case : use current cursor position

2390 <1> ; if(row==0xff)

2391 <1> ; {col=oldcurs&0x00ff;

2392 <1> ; row=(oldcurs&0xff00)>>8;

2393 <1> ; }

2394 <1>

2395 <1> ;mov al, [w\_str\_cmd]

2396 <1>

2397 00001ED5 80FEFF <1> cmp dh, 0FFh

2398 00001ED8 7407 <1> je short vga\_wstr\_1 ; user current cursor position

2399 <1> vga\_wstr\_0:

2400 <1> ; set cursor position

2401 00001EDA 668915[56580100] <1> mov [CURSOR\_POSN], dx ; save cursor pos for pg 0

2402 <1> vga\_wstr\_1:

2403 00001EE1 66FF35[56580100] <1> push word [CURSOR\_POSN] ; \*

2404 <1>

2405 <1> ; ebp = string offset in system buffer (user buffer was copied to)

2406 <1>

2407 <1> ; while(count--!=0)

2408 <1> ; {

2409 <1> ; car=read\_byte(seg,offset++);

2410 <1> ; if((flag&0x02)!=0)

2411 <1> ; attr=read\_byte(seg,offset++);

2412 <1> ; biosfn\_write\_teletype(car,page,attr,WITH\_ATTR);

2413 <1> ; }

2414 <1>

2415 <1> ;push eax ; \*\*

2416 <1> ;test al, 2

2417 00001EE8 F605[D0650100]02 <1> test byte [w\_str\_cmd], 2

2418 00001EEF 751D <1> jnz short vga\_wstr\_3

2419 00001EF1 881D[67580100] <1> mov [ccolor], bl

2420 <1> vga\_wstr\_2:

2421 00001EF7 51 <1> push ecx

2422 00001EF8 8A4500 <1> mov al, [ebp]

2423 00001EFB E8CC0A0000 <1> call vga\_write\_teletype

2424 00001F00 59 <1> pop ecx

2425 00001F01 6649 <1> dec cx

2426 00001F03 741E <1> jz short vga\_wstr\_4

2427 00001F05 45 <1> inc ebp

2428 00001F06 8A1D[67580100] <1> mov bl, [ccolor]

2429 00001F0C EBE9 <1> jmp short vga\_wstr\_2

2430 <1> vga\_wstr\_3:

2431 00001F0E 51 <1> push ecx

2432 00001F0F 8A4500 <1> mov al, [ebp]

2433 00001F12 45 <1> inc ebp

2434 00001F13 8A5D00 <1> mov bl, [ebp]

2435 00001F16 E8B10A0000 <1> call vga\_write\_teletype

2436 00001F1B 59 <1> pop ecx

2437 00001F1C 6649 <1> dec cx

2438 00001F1E 7403 <1> jz short vga\_wstr\_4

2439 00001F20 45 <1> inc ebp

2440 00001F21 EBEB <1> jmp short vga\_wstr\_3

2441 <1> vga\_wstr\_4:

2442 <1> ; // Set back curs pos

2443 <1> ; if((flag&0x01)==0)

2444 <1> ; biosfn\_set\_cursor\_pos(page,oldcurs);

2445 <1> ; }

2446 <1> ;pop eax ; \*\*

2447 00001F23 665A <1> pop dx ; word [CURSOR\_POSN] ; \*

2448 <1> ;test al, 1

2449 00001F25 F605[D0650100]01 <1> test byte [w\_str\_cmd], 1

2450 00001F2C 0F8522F6FFFF <1> jnz VIDEO\_RETURN

2451 00001F32 668915[56580100] <1> mov [CURSOR\_POSN], dx

2452 00001F39 E916F6FFFF <1> JMP VIDEO\_RETURN

2453 <1>

2454 <1> ; 07/07/2016

2455 <1> ; 27/06/2016 - TRDOS 386 (TRDOS v2.0)

2456 <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)

2457 <1> ;------------------------------------------------------

2458 <1> ; SCROLL UP

2459 <1> ; THIS ROUTINE SCROLLS UP THE INFORMATION ON THE CRT

2460 <1> ; ENTRY ---

2461 <1> ; CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL

2462 <1> ; DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL

2463 <1> ; BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS

2464 <1> ; BH = FILL VALUE FOR BLANKED LINES

2465 <1> ; AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE FIELD)

2466 <1> ; DS = DATA SEGMENT

2467 <1> ; ES = REGEN SEGMENT

2468 <1> ; EXIT --

2469 <1> ; NOTHING, THE SCREEN IS SCROLLED

2470 <1> ;--------------------------------------------------------

2471 <1>

2472 <1> ; cl = upper left column

2473 <1> ; ch = upper left row

2474 <1> ; dl = lower rigth column

2475 <1> ; dh = lower right row

2476 <1> ;

2477 <1> ; al = line count (AL=0 means blank entire fields)

2478 <1> ; bl = fill value for blanked lines

2479 <1> ; bh = unused

2480 <1>

2481 <1> GRAPHICS\_UP:

2482 <1> ; 07/07/2016

2483 <1> ;AH = Current video mode, [CRT\_MODE]

2484 00001F3E 80FC07 <1> cmp ah, 7

2485 00001F41 7766 <1> ja short vga\_graphics\_up

2486 <1> ;je n0

2487 <1>

2488 00001F43 88C7 <1> MOV bh, al ; save line count in BH

2489 00001F45 6689C8 <1> MOV AX, CX ; GET UPPER LEFT POSITION INTO AX REG

2490 <1>

2491 <1> ;----- USE CHARACTER SUBROUTINE FOR POSITIONING

2492 <1> ;----- ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE

2493 <1>

2494 00001F48 E8D9050000 <1> CALL GRAPH\_POSN

2495 00001F4D 0FB7F8 <1> MOVzx eDI, AX ; SAVE RESULT AS DESTINATION ADDRESS

2496 <1>

2497 <1> ;----- DETERMINE SIZE OF WINDOW

2498 <1>

2499 00001F50 6629CA <1> SUB DX, CX

2500 00001F53 6681C20101 <1> ADD DX, 101h ; ADJUST VALUES

2501 00001F58 C0E602 <1> SAL DH, 2 ; MULTIPLY ROWS BY 4 AT 8 VERT DOTS/CHAR

2502 <1> ; AND EVEN/ODD ROWS

2503 <1> ;----- DETERMINE CRT MODE

2504 <1>

2505 00001F5B 803D[C25E0000]06 <1> CMP byte [CRT\_MODE], 6 ; TEST FOR MEDIUM RES

2506 00001F62 7305 <1> JNC short \_R7\_ ; FIND\_SOURCE

2507 <1>

2508 <1> ;----- MEDIUM RES UP

2509 00001F64 D0E2 <1> SAL DL, 1 ; # COLUMNS \* 2, SINCE 2 BYTES/CHAR

2510 00001F66 66D1E7 <1> SAL DI, 1 ; OFFSET \*2 SINCE 2 BYTES/CHAR

2511 <1>

2512 <1> ;----- DETERMINE THE SOURCE ADDRESS IN THE BUFFER

2513 <1> \_R7\_: ; FIND\_SOURCE

2514 00001F69 81C700800B00 <1> add edi, 0B8000h

2515 00001F6F C0E702 <1> sal bh, 2 ; multiply number of lines by 4

2516 00001F72 7431 <1> JZ short \_R11 ; IF ZERO, THEN BLANK ENTIRE FIELD

2517 00001F74 B050 <1> MOV AL, 80 ; 80 BYTES/ROW

2518 00001F76 F6E7 <1> mul bh ; determine offset to source

2519 00001F78 0FB7F0 <1> movzx esi, ax ; offset to source

2520 00001F7B 01FE <1> add eSI, eDI ; SET UP SOURCE

2521 00001F7D 88F4 <1> MOV AH, DH ; NUMBER OF ROWS IN FIELD

2522 00001F7F 28FC <1> sub ah, bh ; determine number to move

2523 <1>

2524 <1> ;----- LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS

2525 <1> \_R8: ; ROW\_LOOP

2526 00001F81 E812040000 <1> CALL \_R17 ; MOVE ONE ROW

2527 00001F86 6681EEB01F <1> SUB SI, 2000h-80 ; MOVE TO NEXT ROW

2528 00001F8B 6681EFB01F <1> SUB DI, 2000h-80

2529 00001F90 FECC <1> DEC AH ; NUMBER OF ROWS TO MOVE

2530 00001F92 75ED <1> JNZ short \_R8 ; CONTINUE TILL ALL MOVED

2531 <1>

2532 <1> ;----- FILL IN THE VACATED LINE(S)

2533 <1> \_R9: ; CLEAR ENTRY

2534 00001F94 88D8 <1> mov al, bl ; attribute to fill with

2535 <1> \_R10\_:

2536 00001F96 E819040000 <1> CALL \_R18 ; CLEAR THAT ROW

2537 00001F9B 6681EFB01F <1> SUB DI, 2000h-80 ; POINT TO NEXT LINE

2538 00001FA0 FECF <1> dec bh ; number of lines to fill

2539 00001FA2 75F2 <1> JNZ short \_R10\_ ; CLEAR LOOP

2540 00001FA4 C3 <1> retn ; EVERYYHING DONE

2541 <1>

2542 <1> \_R11: ; BLANK\_FIELD

2543 00001FA5 88F7 <1> mov bh, dh ; set blank count to everything in field

2544 00001FA7 EBEB <1> JMP short \_R9 ; CLEAR THE FIELD

2545 <1>

2546 <1> vga\_graphics\_up:

2547 <1> ; 08/08/2016

2548 <1> ; 07/08/2016

2549 <1> ; 04/08/2016

2550 <1> ; 01/08/2016

2551 <1> ; 31/07/2016

2552 <1> ; 07/07/2016 - TRDOS 386 (TRDOS v2.0)

2553 <1> ;

2554 <1> ; derived from 'Plex86/Bochs VGABios' source code

2555 <1> ; vgabios-0.7a (2011)

2556 <1> ; by the LGPL VGABios developers Team (2001-2008)

2557 <1> ; 'vgabios.c', 'biosfn\_scroll'

2558 <1> ;

2559 <1>

2560 <1> ; cl = upper left column

2561 <1> ; ch = upper left row

2562 <1> ; dl = lower rigth column

2563 <1> ; dh = lower right row

2564 <1> ;

2565 <1> ; al = line count (AL=0 means blank entire fields)

2566 <1> ; bl = fill value for blanked lines

2567 <1> ; bh = unused

2568 <1> ;

2569 <1> ; ah = [CRT\_MODE], current video mode

2570 <1>

2571 00001FA9 88C7 <1> mov bh, al ; 31/07/2016

2572 00001FAB BE[E65E0000] <1> mov esi, vga\_g\_modes

2573 00001FB0 89F7 <1> mov edi, esi

2574 00001FB2 83C708 <1> add edi, vga\_g\_mode\_count

2575 <1> vga\_g\_up\_0:

2576 00001FB5 AC <1> lodsb

2577 00001FB6 38E0 <1> cmp al, ah ; [CRT\_MODE]

2578 00001FB8 7405 <1> je short vga\_g\_up\_1

2579 00001FBA 39FE <1> cmp esi, edi

2580 00001FBC 72F7 <1> jb short vga\_g\_up\_0

2581 <1> ;xor bh, bh ; 31/07/2016)

2582 00001FBE C3 <1> retn ; nothing to do

2583 <1> vga\_g\_up\_1:

2584 00001FBF 88F8 <1> mov al, bh ; 31/07/2016

2585 00001FC1 83C64F <1> add esi, vga\_g\_memmodel - (vga\_g\_modes + 1)

2586 <1> ; [ESI] = VGA memory model number (LINEAR8, PLANAR4, PLANAR1)

2587 <1>

2588 <1> ; if(rlr>=nbrows)rlr=nbrows-1;

2589 <1> ; if(clr>=nbcols)clr=nbcols-1;

2590 <1> ; if(nblines>nbrows)nblines=0;

2591 <1> ; cols=clr-cul+1;

2592 <1>

2593 00001FC4 3A35[CA5E0000] <1> cmp dh, [VGA\_ROWS]

2594 00001FCA 7208 <1> jb short vga\_g\_up\_2

2595 00001FCC 8A35[CA5E0000] <1> mov dh, [VGA\_ROWS]

2596 00001FD2 FECE <1> dec dh

2597 <1> vga\_g\_up\_2:

2598 00001FD4 3A15[C45E0000] <1> cmp dl, [CRT\_COLS] ; = [VGA\_COLS]

2599 00001FDA 7208 <1> jb short vga\_g\_up\_3

2600 00001FDC 8A15[C45E0000] <1> mov dl, [CRT\_COLS]

2601 00001FE2 FECA <1> dec dl

2602 <1> vga\_g\_up\_3:

2603 00001FE4 3A05[CA5E0000] <1> cmp al, [VGA\_ROWS]

2604 00001FEA 7602 <1> jna short vga\_g\_up\_4

2605 00001FEC 28C0 <1> sub al, al ; 0

2606 <1> vga\_g\_up\_4:

2607 00001FEE 88D7 <1> mov bh, dl ; clr

2608 00001FF0 28CF <1> sub bh, cl ; cul

2609 00001FF2 FEC7 <1> inc bh ; cols = clr-cul+1

2610 <1>

2611 00001FF4 20C0 <1> and al, al ; nblines = 0

2612 00001FF6 755D <1> jnz short vga\_g\_up\_6

2613 00001FF8 20ED <1> and ch, ch ; rul = 0

2614 00001FFA 7559 <1> jnz short vga\_g\_up\_6

2615 00001FFC 20C9 <1> and cl, cl ; cul = 0

2616 00001FFE 7555 <1> jnz short vga\_g\_up\_6

2617 <1>

2618 00002000 6650 <1> push ax

2619 00002002 A0[CA5E0000] <1> mov al, [VGA\_ROWS]

2620 00002007 FEC8 <1> dec al

2621 00002009 38C6 <1> cmp dh, al ; rlr = nbrows-1

2622 0000200B 7546 <1> jne short vga\_g\_up\_5

2623 0000200D A0[C45E0000] <1> mov al, [CRT\_COLS] ; = VGA\_COLS

2624 00002012 FEC8 <1> dec al

2625 00002014 38C2 <1> cmp dl, al ; clr = nbcols-1

2626 00002016 753B <1> jne short vga\_g\_up\_5

2627 00002018 6658 <1> pop ax

2628 <1>

2629 0000201A 66B80502 <1> mov ax, 0205h

2630 0000201E 66BACE03 <1> mov dx, 3CEh ; VGAREG\_GRDC\_ADDRESS

2631 00002022 66EF <1> out dx, ax

2632 00002024 A0[CA5E0000] <1> mov al, [VGA\_ROWS]

2633 00002029 8A25[C45E0000] <1> mov ah, [CRT\_COLS] ; = [VGA\_COLS]

2634 0000202F F6E4 <1> mul ah

2635 00002031 0FB7D0 <1> movzx edx, ax

2636 <1> ; 08/08/2016

2637 00002034 0FB605[C65E0000] <1> movzx eax, byte [CHAR\_HEIGHT]

2638 0000203B F7E2 <1> mul edx

2639 <1> ; eax = byte count

2640 0000203D 89C1 <1> mov ecx, eax

2641 <1> ;; 07/08/2016

2642 <1> ;shl dx, 3 ; \* 8 ; \* [CHAR\_HEIGHT]

2643 <1> ;mov ecx, edx

2644 0000203F 88D8 <1> mov al, bl ; fill value for blanked lines

2645 00002041 BF00000A00 <1> mov edi, 0A0000h

2646 00002046 F3AA <1> rep stosb

2647 <1>

2648 00002048 66B80500 <1> mov ax, 5

2649 0000204C 66BACE03 <1> mov dx, 3CEh ; VGAREG\_GRDC\_ADDRESS

2650 00002050 66EF <1> out dx, ax ; 0005h

2651 <1>

2652 00002052 C3 <1> retn

2653 <1>

2654 <1> vga\_g\_up\_5:

2655 00002053 6658 <1> pop ax

2656 <1>

2657 <1> vga\_g\_up\_6:

2658 <1> ; [ESI] = VGA memory model number for current video mode

2659 <1> ;

2660 <1> ; LINEAR8 equ 5

2661 <1> ; PLANAR4 equ 4

2662 <1> ; PLANAR1 equ 3

2663 <1>

2664 00002055 803E04 <1> cmp byte [esi], PLANAR4

2665 00002058 7424 <1> je short vga\_g\_up\_planar

2666 0000205A 803E03 <1> cmp byte [esi], PLANAR1

2667 0000205D 741F <1> je short vga\_g\_up\_planar

2668 <1> vga\_g\_up\_linear8:

2669 <1> ; 07/07/2016 (TEMPORARY)

2670 <1> ;

2671 <1> ; cl = upper left column ; cul

2672 <1> ; ch = upper left row ; rul

2673 <1> ; dl = lower rigth column ; clr

2674 <1> ; dh = lower right row ; rlr

2675 <1>

2676 <1> vga\_g\_up\_l0:

2677 <1> ;{for(i=rul;i<=rlr;i++)

2678 <1> ; if((i+nblines>rlr)||(nblines==0))

2679 0000205F 08C0 <1> or al, al

2680 00002061 7414 <1> jz short vga\_g\_up\_l2

2681 00002063 88C4 <1> mov ah, al

2682 00002065 00EC <1> add ah, ch ; i+nblines

2683 <1> ;jc short vga\_g\_up\_l2

2684 00002067 38F4 <1> cmp ah, dh

2685 00002069 770C <1> ja short vga\_g\_up\_l2

2686 <1> ; else

2687 <1> ; vgamem\_copy\_pl4(cul,i+nblines,i,cols,nbcols,cheight);

2688 0000206B E8F2000000 <1> call vgamem\_copy\_l8

2689 <1> vga\_g\_up\_l1:

2690 00002070 FEC5 <1> inc ch

2691 00002072 38F5 <1> cmp ch, dh

2692 00002074 76E9 <1> jna short vga\_g\_up\_l0

2693 00002076 C3 <1> retn

2694 <1> vga\_g\_up\_l2:

2695 <1> ; vgamem\_fill\_pl4(cul,i,cols,nbcols,cheight,attr);

2696 00002077 E850010000 <1> call vgamem\_fill\_l8

2697 0000207C EBF2 <1> jmp short vga\_g\_up\_l1

2698 <1>

2699 <1> vga\_g\_up\_planar:

2700 <1> ; cl = upper left column ; cul

2701 <1> ; ch = upper left row ; rul

2702 <1> ; dl = lower rigth column ; clr

2703 <1> ; dh = lower right row ; rlr

2704 <1> vga\_g\_up\_pl0:

2705 <1> ;{for(i=rul;i<=rlr;i++)

2706 <1> ; if((i+nblines>rlr)||(nblines==0))

2707 0000207E 20C0 <1> and al, al

2708 00002080 7414 <1> jz short vga\_g\_up\_pl2

2709 00002082 88C4 <1> mov ah, al

2710 00002084 00EC <1> add ah, ch ; i+nblines

2711 <1> ;jc short vga\_g\_up\_pl2

2712 00002086 38F4 <1> cmp ah, dh

2713 00002088 770C <1> ja short vga\_g\_up\_pl2

2714 <1> ; else

2715 <1> ; vgamem\_copy\_pl4(cul,i+nblines,i,cols,nbcols,cheight);

2716 0000208A E80E000000 <1> call vgamem\_copy\_pl4

2717 <1> vga\_g\_up\_pl1:

2718 0000208F FEC5 <1> inc ch

2719 00002091 38F5 <1> cmp ch, dh

2720 00002093 76E9 <1> jna short vga\_g\_up\_pl0

2721 00002095 C3 <1> retn

2722 <1> vga\_g\_up\_pl2:

2723 <1> ; vgamem\_fill\_pl4(cul,i,cols,nbcols,cheight,attr);

2724 00002096 E870000000 <1> call vgamem\_fill\_pl4

2725 0000209B EBF2 <1> jmp short vga\_g\_up\_pl1

2726 <1>

2727 <1> vgamem\_copy\_pl4:

2728 <1> ; 08/08/2016

2729 <1> ; 07/08/2016

2730 <1> ; 07/07/2016 - TRDOS 386 (TRDOS v2.0)

2731 <1> ;

2732 <1> ; derived from 'Plex86/Bochs VGABios' source code

2733 <1> ; vgabios-0.7a (2011)

2734 <1> ; by the LGPL VGABios developers Team (2001-2008)

2735 <1> ; 'vgabios.c', 'vgamem\_copy\_pl4'

2736 <1> ;

2737 <1> ; vgamem\_copy\_pl4(xstart,ysrc,ydest,cols,nbcols,cheight)

2738 <1> ; cl = xstart, ah = ysrc (i+nblines), ch = ydest (i),

2739 <1> ; bh = cols, [CRT\_COLS] = nbcols, [CHAR\_HEIGHT] = cheight

2740 <1>

2741 <1> ; src=ysrc\*cheight\*nbcols+xstart;

2742 <1> ; dest=ydest\*cheight\*nbcols+xstart;

2743 <1>

2744 0000209D 52 <1> push edx

2745 0000209E 50 <1> push eax

2746 <1>

2747 <1> ; outw(VGAREG\_GRDC\_ADDRESS, 0x0105)

2748 0000209F 66B80501 <1> mov ax, 0105h

2749 000020A3 66BACE03 <1> mov dx, 3CEh ; VGAREG\_GRDC\_ADDRESS

2750 000020A7 66EF <1> out dx, ax

2751 <1>

2752 <1> ; 07/08/2016

2753 <1> ;mov ah, [esp+1]

2754 <1> ;movzx edx, ah ; ysrc

2755 000020A9 0FB6542401 <1> movzx edx, byte [esp+1]

2756 <1> ; 08/08/2016

2757 000020AE 0FB605[C65E0000] <1> movzx eax, byte [CHAR\_HEIGHT]

2758 000020B5 8A25[C45E0000] <1> mov ah, [CRT\_COLS] ; nbcols

2759 000020BB F6E4 <1> mul ah

2760 <1> ;; 07/08/2016

2761 <1> ;movzx eax, byte [CRT\_COLS]

2762 <1> ;shl ax, 3 ; \* 8 ; \* [CHAR\_HEIGHT]

2763 000020BD 50 <1> push eax ; cheight \* nbcols

2764 000020BE F7E2 <1> mul edx ; \* ysrc

2765 <1> ; eax = ysrc \* cheight \* nbcols

2766 <1> ; edx = 0

2767 000020C0 88CA <1> mov dl, cl ; edx = xstart

2768 000020C2 01D0 <1> add eax, edx

2769 000020C4 89C6 <1> mov esi, eax ; src

2770 000020C6 88EA <1> mov dl, ch ; ydest

2771 000020C8 58 <1> pop eax ; cheight \* nbcols

2772 000020C9 F7E2 <1> mul edx

2773 <1> ; eax = ydest \* cheight \* nbcols

2774 000020CB 88CA <1> mov dl, cl ; edx = xstart

2775 000020CD 01D0 <1> add eax, edx

2776 000020CF 89C7 <1> mov edi, eax ; dest

2777 <1> ; esi = src

2778 <1> ; edi = dest

2779 <1> ; for(i=0;i<cheight;i++)

2780 <1> ; {

2781 <1> ; memcpyb(0xa000,dest+i\*nbcols,0xa000,src+i\*nbcols,cols);

2782 <1> ; }

2783 000020D1 51 <1> push ecx

2784 000020D2 B900000A00 <1> mov ecx, 0A0000h

2785 000020D7 01CE <1> add esi, ecx

2786 000020D9 01CF <1> add edi, ecx

2787 <1> ; 08/08/2016

2788 000020DB 8A35[C65E0000] <1> mov dh, [CHAR\_HEIGHT]

2789 <1> ;; 07/08/2016

2790 <1> ;mov dh, 8 ; 07/08/2016

2791 000020E1 28D2 <1> sub dl, dl ; i

2792 <1> vgamem\_copy\_pl4\_0:

2793 000020E3 56 <1> push esi

2794 000020E4 57 <1> push edi

2795 000020E5 0FB605[C45E0000] <1> movzx eax, byte [CRT\_COLS]

2796 000020EC F6E2 <1> mul dl

2797 <1> ; eax = i \* nbcols

2798 000020EE 01C7 <1> add edi, eax ; dest+i\*nbcols

2799 000020F0 01C6 <1> add esi, eax

2800 000020F2 0FB6CF <1> movzx ecx, bh ; cols

2801 000020F5 F3A4 <1> rep movsb

2802 000020F7 5F <1> pop edi

2803 000020F8 5E <1> pop esi

2804 000020F9 FECE <1> dec dh

2805 000020FB 75E6 <1> jnz short vgamem\_copy\_pl4\_0

2806 <1> vgamem\_copy\_pl4\_1:

2807 000020FD 59 <1> pop ecx

2808 <1>

2809 <1> ; outw(VGAREG\_GRDC\_ADDRESS, 0x0005);

2810 000020FE 66B80500 <1> mov ax, 0005h

2811 00002102 66BACE03 <1> mov dx, 3CEh ; VGAREG\_GRDC\_ADDRESS

2812 00002106 66EF <1> out dx, ax

2813 <1>

2814 00002108 58 <1> pop eax

2815 00002109 5A <1> pop edx

2816 <1>

2817 0000210A C3 <1> retn

2818 <1>

2819 <1> vgamem\_fill\_pl4:

2820 <1> ; 08/08/2016

2821 <1> ; 07/08/2016

2822 <1> ; 04/08/2016

2823 <1> ; 07/07/2016 - TRDOS 386 (TRDOS v2.0)

2824 <1> ;

2825 <1> ; derived from 'Plex86/Bochs VGABios' source code

2826 <1> ; vgabios-0.7a (2011)

2827 <1> ; by the LGPL VGABios developers Team (2001-2008)

2828 <1> ; 'vgabios.c', 'vgamem\_fill\_pl4'

2829 <1> ;

2830 <1> ; vgamem\_fill\_pl4(xstart,ystart,cols,nbcols,cheight,attr)

2831 <1> ; cl = xstart, edi = ch = ystart, bh = cols,

2832 <1> ; [CRT\_COLS] = nbcols, [CHAR\_HEIGHT] = cheight, attr = 0

2833 <1>

2834 <1> ; dest=ystart\*cheight\*nbcols+xstart;

2835 0000210B 52 <1> push edx

2836 0000210C 50 <1> push eax

2837 <1>

2838 <1> ; outw(VGAREG\_GRDC\_ADDRESS, 0x0205)

2839 0000210D 66B80502 <1> mov ax, 0205h

2840 00002111 66BACE03 <1> mov dx, 3CEh ; VGAREG\_GRDC\_ADDRESS

2841 00002115 66EF <1> out dx, ax

2842 <1>

2843 <1> ; 08/08/2016

2844 00002117 0FB605[C65E0000] <1> movzx eax, byte [CHAR\_HEIGHT]

2845 0000211E F6E5 <1> mul ch

2846 <1> ;; 07/08/2016

2847 <1> ;movzx eax, ch

2848 <1> ;shl ax, 3 ; \* 8 ; \* [CHAR\_HEIGHT]

2849 00002120 0FB615[C45E0000] <1> movzx edx, byte [CRT\_COLS] ; = [VGA\_COLS]

2850 00002127 F7E2 <1> mul edx

2851 <1> ; edx = 0

2852 00002129 88CA <1> mov dl, cl

2853 0000212B 01D0 <1> add eax, edx

2854 0000212D 89C7 <1> mov edi, eax

2855 <1> ; edi = dest

2856 <1> ; for(i=0;i<cheight;i++)

2857 <1> ; {

2858 <1> ; memsetb(0xa000,dest+i\*nbcols,attr,cols);

2859 <1> ; }

2860 0000212F 81C700000A00 <1> add edi, 0A0000h

2861 00002135 51 <1> push ecx

2862 <1> ; 08/08/2016

2863 00002136 8A35[C65E0000] <1> mov dh, [CHAR\_HEIGHT]

2864 <1> ;; 07/08/2016

2865 <1> ;mov dh, 8 ; 07/08/2016

2866 0000213C 28D2 <1> sub dl, dl ; i

2867 <1> vgamem\_fill\_pl4\_0:

2868 0000213E 57 <1> push edi

2869 0000213F 0FB605[C45E0000] <1> movzx eax, byte [CRT\_COLS]

2870 00002146 F6E2 <1> mul dl

2871 <1> ; eax = i \* nbcols

2872 00002148 01C7 <1> add edi, eax ; dest+i\*nbcols

2873 0000214A 88D8 <1> mov al, bl ; attr ; 04/08/2016

2874 0000214C 0FB6CF <1> movzx ecx, bh ; cols

2875 0000214F F3AA <1> rep stosb

2876 00002151 5F <1> pop edi

2877 00002152 75EA <1> jnz short vgamem\_fill\_pl4\_0

2878 <1> vgamem\_fill\_pl4\_1:

2879 00002154 59 <1> pop ecx

2880 <1>

2881 <1> ; outw(VGAREG\_GRDC\_ADDRESS, 0x0005);

2882 00002155 66B80500 <1> mov ax, 0005h

2883 00002159 66BACE03 <1> mov dx, 3CEh ; VGAREG\_GRDC\_ADDRESS

2884 0000215D 66EF <1> out dx, ax

2885 <1>

2886 0000215F 58 <1> pop eax

2887 00002160 5A <1> pop edx

2888 <1>

2889 00002161 C3 <1> retn

2890 <1>

2891 <1> vgamem\_copy\_l8:

2892 <1> ; 08/08/2016

2893 <1> ; 07/08/2016

2894 <1> ; 06/08/2016

2895 <1> ; 07/07/2016 - TRDOS 386 (TRDOS v2.0)

2896 <1> ;

2897 <1> ; TEMPORARY

2898 <1> ;

2899 <1> ; derived from 'Plex86/Bochs VGABios' source code

2900 <1> ; vgabios-0.7a (2011)

2901 <1> ; by the LGPL VGABios developers Team (2001-2008)

2902 <1> ; 'vgabios.c', 'vgamem\_copy\_pl4'

2903 <1> ;

2904 <1> ; vgamem\_copy\_pl4(xstart,ysrc,ydest,cols,nbcols,cheight)

2905 <1> ; cl = xstart, ah = ysrc (i+nblines), ch = ydest (i),

2906 <1> ; bh = cols, [CRT\_COLS] = nbcols, [CHAR\_HEIGHT] = cheight

2907 <1>

2908 <1> ; src=ysrc\*cheight\*nbcols+xstart;

2909 <1> ; dest=ydest\*cheight\*nbcols+xstart;

2910 <1>

2911 00002162 52 <1> push edx

2912 00002163 50 <1> push eax

2913 <1>

2914 <1> ; outw(VGAREG\_GRDC\_ADDRESS, 0x0105)

2915 <1> ;mov ax, 0105h

2916 <1> ;mov dx, 3CEh ; VGAREG\_GRDC\_ADDRESS

2917 <1> ;out dx, ax

2918 <1>

2919 <1> ;mov ah, [esp+1]

2920 <1>

2921 00002164 0FB6D4 <1> movzx edx, ah ; ysrc

2922 <1> ; 08/08/2016

2923 00002167 0FB605[C65E0000] <1> movzx eax, byte [CHAR\_HEIGHT]

2924 0000216E 8A25[C45E0000] <1> mov ah, [CRT\_COLS] ; nbcols

2925 00002174 F6E4 <1> mul ah

2926 <1> ;; 07/08/2016

2927 <1> ;movzx eax, byte [CRT\_COLS]

2928 <1> ;shl ax, 3 ; \* 8 ; \* [CHAR\_HEIGHT]

2929 00002176 50 <1> push eax ; cheight \* nbcols

2930 00002177 F7E2 <1> mul edx ; \* ysrc

2931 <1> ; eax = ysrc \* cheight \* nbcols

2932 <1> ; edx = 0

2933 00002179 88CA <1> mov dl, cl ; edx = xstart

2934 0000217B 01D0 <1> add eax, edx

2935 0000217D 89C6 <1> mov esi, eax ; src

2936 0000217F 66C1E603 <1> shl si, 3 ; \* 8 ; 06/08/2016

2937 00002183 88EA <1> mov dl, ch ; ydest

2938 00002185 58 <1> pop eax ; cheight \* nbcols

2939 00002186 F7E2 <1> mul edx

2940 <1> ; eax = ydest \* cheight \* nbcols

2941 00002188 88CA <1> mov dl, cl ; edx = xstart

2942 0000218A 01D0 <1> add eax, edx

2943 0000218C 89C7 <1> mov edi, eax ; dest

2944 0000218E 66C1E703 <1> shl di, 3 ; \* 8 ; 06/08/2016

2945 <1> ; esi = src

2946 <1> ; edi = dest

2947 <1> ; for(i=0;i<cheight;i++)

2948 <1> ; {

2949 <1> ; memcpyb(0xa000,dest+i\*nbcols,0xa000,src+i\*nbcols,cols);

2950 <1> ; }

2951 00002192 51 <1> push ecx

2952 00002193 B900000A00 <1> mov ecx, 0A0000h

2953 00002198 01CE <1> add esi, ecx

2954 0000219A 01CF <1> add edi, ecx

2955 <1> ; 08/08/2016

2956 0000219C 8A35[C65E0000] <1> mov dh, [CHAR\_HEIGHT]

2957 <1> ;; 07/08/2016

2958 <1> ;mov dh, 8 ; 07/08/2016

2959 000021A2 28D2 <1> sub dl, dl ; i

2960 <1> vgamem\_copy\_l8\_0:

2961 000021A4 56 <1> push esi

2962 000021A5 57 <1> push edi

2963 000021A6 0FB605[C45E0000] <1> movzx eax, byte [CRT\_COLS]

2964 000021AD F6E2 <1> mul dl

2965 <1> ; eax = i \* nbcols

2966 000021AF 66C1E003 <1> shl ax, 3 ; \* 8 ; 06/08/2016

2967 000021B3 01C7 <1> add edi, eax ; dest+i\*nbcols

2968 000021B5 01C6 <1> add esi, eax

2969 000021B7 0FB6CF <1> movzx ecx, bh ; cols

2970 000021BA 66C1E103 <1> shl cx, 3 ; \* 8 ; 06/08/2016

2971 000021BE F3A4 <1> rep movsb

2972 000021C0 5F <1> pop edi

2973 000021C1 5E <1> pop esi

2974 000021C2 FEC2 <1> inc dl ; 06/08/2016

2975 000021C4 FECE <1> dec dh

2976 000021C6 75DC <1> jnz short vgamem\_copy\_l8\_0

2977 <1> vgamem\_copy\_l8\_1:

2978 000021C8 59 <1> pop ecx

2979 <1>

2980 <1> ;; outw(VGAREG\_GRDC\_ADDRESS, 0x0005);

2981 <1> ;mov ax, 0005h

2982 <1> ;mov dx, 3CEh ; VGAREG\_GRDC\_ADDRESS

2983 <1> ;out dx, ax

2984 <1>

2985 000021C9 58 <1> pop eax

2986 000021CA 5A <1> pop edx

2987 <1>

2988 000021CB C3 <1> retn

2989 <1>

2990 <1> vgamem\_fill\_l8:

2991 <1> ; 08/08/2016

2992 <1> ; 07/08/2016

2993 <1> ; 06/08/2016

2994 <1> ; 04/08/2016

2995 <1> ; 07/07/2016 - TRDOS 386 (TRDOS v2.0)

2996 <1> ;

2997 <1> ; TEMPORARY

2998 <1> ;

2999 <1> ; derived from 'Plex86/Bochs VGABios' source code

3000 <1> ; vgabios-0.7a (2011)

3001 <1> ; by the LGPL VGABios developers Team (2001-2008)

3002 <1> ; 'vgabios.c', 'vgamem\_fill\_pl4'

3003 <1> ;

3004 <1> ; vgamem\_fill\_pl4(xstart,ystart,cols,nbcols,cheight,attr)

3005 <1> ; cl = xstart, edi = ch = ystart, bh = cols,

3006 <1> ; [CRT\_COLS] = nbcols, [CHAR\_HEIGHT] = cheight, attr = 0

3007 <1>

3008 <1> ; dest=ystart\*cheight\*nbcols+xstart;

3009 000021CC 52 <1> push edx

3010 000021CD 50 <1> push eax

3011 <1>

3012 <1> ;; outw(VGAREG\_GRDC\_ADDRESS, 0x0205)

3013 <1> ;mov ax, 0205h

3014 <1> ;mov dx, 3CEh ; VGAREG\_GRDC\_ADDRESS

3015 <1> ;out dx, ax

3016 <1>

3017 <1> ; 08/08/2016

3018 000021CE 0FB605[C65E0000] <1> movzx eax, byte [CHAR\_HEIGHT]

3019 000021D5 F6E5 <1> mul ch

3020 <1> ;; 07/08/2016

3021 <1> ;movzx eax, ch

3022 <1> ;shl ax, 3 ; \* 8 ; \* [CHAR\_HEIGHT]

3023 000021D7 0FB615[C45E0000] <1> movzx edx, byte [CRT\_COLS] ; = [VGA\_COLS]

3024 000021DE F7E2 <1> mul edx

3025 <1> ; edx = 0

3026 000021E0 88CA <1> mov dl, cl

3027 000021E2 01D0 <1> add eax, edx

3028 000021E4 89C7 <1> mov edi, eax

3029 000021E6 66C1E703 <1> shl di, 3 ; \* 8 ; 06/08/2016

3030 <1> ; edi = dest

3031 <1> ; for(i=0;i<cheight;i++)

3032 <1> ; {

3033 <1> ; memsetb(0xa000,dest+i\*nbcols,attr,cols);

3034 <1> ; }

3035 000021EA 81C700000A00 <1> add edi, 0A0000h

3036 000021F0 51 <1> push ecx

3037 <1> ; 08/08/2016

3038 000021F1 8A35[C65E0000] <1> mov dh, [CHAR\_HEIGHT]

3039 <1> ;; 07/08/2016

3040 <1> ;mov dh, 8 ; 07/08/2016

3041 000021F7 28D2 <1> sub dl, dl ; i

3042 <1> vgamem\_fill\_l8\_0:

3043 000021F9 57 <1> push edi

3044 000021FA 0FB605[C45E0000] <1> movzx eax, byte [CRT\_COLS]

3045 00002201 F6E2 <1> mul dl

3046 <1> ; eax = i \* nbcols

3047 00002203 66C1E003 <1> shl ax, 3 ; \* 8 ; 06/08/2016

3048 00002207 01C7 <1> add edi, eax ; dest+i\*nbcols

3049 00002209 88D8 <1> mov al, bl ; attr ; 04/08/2016

3050 0000220B 0FB6CF <1> movzx ecx, bh ; cols

3051 0000220E 66C1E103 <1> shl cx, 3 ; \* 8 ; 06/08/2016

3052 00002212 F3AA <1> rep stosb

3053 00002214 5F <1> pop edi

3054 00002215 FEC2 <1> inc dl ; 06/08/2016

3055 00002217 FECE <1> dec dh

3056 00002219 75DE <1> jnz short vgamem\_fill\_l8\_0

3057 <1> vgamem\_fill\_l8\_1:

3058 0000221B 59 <1> pop ecx

3059 <1>

3060 <1> ;; outw(VGAREG\_GRDC\_ADDRESS, 0x0005);

3061 <1> ;mov ax, 0005h

3062 <1> ;mov dx, 3CEh ; VGAREG\_GRDC\_ADDRESS

3063 <1> ;out dx, ax

3064 <1>

3065 0000221C 58 <1> pop eax

3066 0000221D 5A <1> pop edx

3067 <1>

3068 0000221E C3 <1> retn

3069 <1>

3070 <1> vga\_graphics\_down:

3071 <1> ; 08/08/2016

3072 <1> ; 07/08/2016

3073 <1> ; 31/07/2016

3074 <1> ; 07/07/2016 - TRDOS 386 (TRDOS v2.0)

3075 <1> ;

3076 <1> ; derived from 'Plex86/Bochs VGABios' source code

3077 <1> ; vgabios-0.7a (2011)

3078 <1> ; by the LGPL VGABios developers Team (2001-2008)

3079 <1> ; 'vgabios.c', 'biosfn\_scroll'

3080 <1> ;

3081 <1>

3082 <1> ; cl = upper left column

3083 <1> ; ch = upper left row

3084 <1> ; dl = lower rigth column

3085 <1> ; dh = lower right row

3086 <1> ;

3087 <1> ; al = line count (AL=0 means blank entire fields)

3088 <1> ; bl = fill value for blanked lines

3089 <1> ; bh = unused

3090 <1> ;

3091 <1> ; ah = [CRT\_MODE], current video mode

3092 <1>

3093 0000221F FC <1> cld ; !!! Clear direction flag !!!

3094 <1>

3095 00002220 88C7 <1> mov bh, al ; 31/07/2016

3096 <1>

3097 00002222 BE[DE5E0000] <1> mov esi, vga\_modes

3098 00002227 89F7 <1> mov edi, esi

3099 00002229 83C710 <1> add edi, vga\_mode\_count

3100 <1> vga\_g\_down\_0:

3101 0000222C AC <1> lodsb

3102 0000222D 38E0 <1> cmp al, ah ; [CRT\_MODE]

3103 0000222F 7405 <1> je short vga\_g\_down\_1

3104 00002231 39FE <1> cmp esi, edi

3105 00002233 72F7 <1> jb short vga\_g\_down\_0

3106 <1> ; xor bh, bh ; 31/07/2016

3107 00002235 C3 <1> retn ; nothing to do

3108 <1> vga\_g\_down\_1:

3109 00002236 88F8 <1> mov al, bh ; 31/07/2016

3110 00002238 83C64F <1> add esi, vga\_memmodel - (vga\_modes + 1)

3111 <1> ; [ESI] = VGA memory model number (LINEAR8, PLANAR4, PLANAR1)

3112 <1>

3113 <1> ; if(rlr>=nbrows)rlr=nbrows-1;

3114 <1> ; if(clr>=nbcols)clr=nbcols-1;

3115 <1> ; if(nblines>nbrows)nblines=0;

3116 <1> ; cols=clr-cul+1;

3117 <1>

3118 0000223B 3A35[CA5E0000] <1> cmp dh, [VGA\_ROWS]

3119 00002241 7208 <1> jb short vga\_g\_down\_2

3120 00002243 8A35[CA5E0000] <1> mov dh, [VGA\_ROWS]

3121 00002249 FECE <1> dec dh

3122 <1> vga\_g\_down\_2:

3123 0000224B 3A15[C45E0000] <1> cmp dl, [CRT\_COLS] ; = [VGA\_COLS]

3124 00002251 7208 <1> jb short vga\_g\_down\_3

3125 00002253 8A15[C45E0000] <1> mov dl, [CRT\_COLS]

3126 00002259 FECA <1> dec dl

3127 <1> vga\_g\_down\_3:

3128 0000225B 3A05[CA5E0000] <1> cmp al, [VGA\_ROWS]

3129 00002261 7602 <1> jna short vga\_g\_down\_4

3130 00002263 28C0 <1> sub al, al ; 0

3131 <1> vga\_g\_down\_4:

3132 00002265 88F7 <1> mov bh, dh ; clr

3133 00002267 28CF <1> sub bh, cl ; cul

3134 00002269 FEC7 <1> inc bh ; cols = clr-cul+1

3135 <1>

3136 0000226B 20C0 <1> and al, al ; nblines = 0

3137 0000226D 755B <1> jnz short vga\_g\_down\_6

3138 0000226F 20ED <1> and ch, ch ; rul = 0

3139 00002271 7557 <1> jnz short vga\_g\_down\_6

3140 00002273 20C9 <1> and cl, cl ; cul = 0

3141 00002275 7553 <1> jnz short vga\_g\_down\_6

3142 <1>

3143 00002277 6650 <1> push ax

3144 00002279 A0[CA5E0000] <1> mov al, [VGA\_ROWS]

3145 0000227E FEC8 <1> dec al

3146 00002280 38C6 <1> cmp dh, al ; rlr = nbrows-1

3147 00002282 7544 <1> jne short vga\_g\_down\_5

3148 00002284 A0[C45E0000] <1> mov al, [CRT\_COLS] ; = VGA\_COLS

3149 00002289 FEC8 <1> dec al

3150 0000228B 38C2 <1> cmp dl, al ; clr = nbcols-1

3151 0000228D 7539 <1> jne short vga\_g\_down\_5

3152 0000228F 6658 <1> pop ax

3153 <1>

3154 00002291 66B80502 <1> mov ax, 0205h

3155 00002295 66BACE03 <1> mov dx, 3CEh ; VGAREG\_GRDC\_ADDRESS

3156 00002299 66EF <1> out dx, ax

3157 0000229B A0[CA5E0000] <1> mov al, [VGA\_ROWS]

3158 000022A0 8A25[C45E0000] <1> mov ah, [CRT\_COLS] ; = [VGA\_COLS]

3159 000022A6 F6E4 <1> mul ah

3160 000022A8 0FB7D0 <1> movzx edx, ax

3161 <1> ; 08/08/2016

3162 000022AB 0FB605[C65E0000] <1> movzx eax, byte [CHAR\_HEIGHT]

3163 000022B2 F7E2 <1> mul edx

3164 <1> ; eax = byte count

3165 000022B4 89C1 <1> mov ecx, eax

3166 <1> ;; 07/08/2016

3167 <1> ;shl dx, 3 ; \* 8 ; \* [CHAR\_HEIGHT]

3168 <1> ;mov ecx, edx

3169 000022B6 88D8 <1> mov al, bl ; fill value for blanked lines

3170 000022B8 BF00000A00 <1> mov edi, 0A0000h

3171 000022BD F3AA <1> rep stosb

3172 <1>

3173 000022BF B005 <1> mov al, 5

3174 000022C1 66BACE03 <1> mov dx, 3CEh ; VGAREG\_GRDC\_ADDRESS

3175 000022C5 66EF <1> out dx, ax ; 0005h

3176 <1>

3177 000022C7 C3 <1> retn

3178 <1>

3179 <1> vga\_g\_down\_5:

3180 000022C8 6658 <1> pop ax

3181 <1>

3182 <1> vga\_g\_down\_6:

3183 <1> ; [ESI] = VGA memory model number for current video mode

3184 <1> ;

3185 <1> ; LINEAR8 equ 5

3186 <1> ; PLANAR4 equ 4

3187 <1> ; PLANAR1 equ 3

3188 <1>

3189 000022CA 803E04 <1> cmp byte [esi], PLANAR4

3190 000022CD 742C <1> je short vga\_g\_down\_planar

3191 000022CF 803E03 <1> cmp byte [esi], PLANAR1

3192 000022D2 7427 <1> je short vga\_g\_down\_planar

3193 <1> vga\_g\_down\_linear8:

3194 <1> ; 07/07/2016 (TEMPORARY)

3195 <1> ;

3196 <1> ; cl = upper left column ; cul

3197 <1> ; ch = upper left row ; rul

3198 <1> ; dl = lower rigth column ; clr

3199 <1> ; dh = lower right row ; rlr

3200 <1>

3201 <1> vga\_g\_down\_l0:

3202 <1> ;{for(i=rlr;i>=rul;i--)

3203 <1> ; if((i<rul+nblines)||(nblines==0))

3204 000022D4 08C0 <1> or al, al

3205 000022D6 741C <1> jz short vga\_g\_down\_l2

3206 000022D8 88C4 <1> mov ah, al

3207 000022DA 00EC <1> add ah, ch

3208 <1> ;jc short vga\_g\_down\_l2

3209 000022DC 86EE <1> xchg ch, dh

3210 000022DE 38E5 <1> cmp ch, ah

3211 000022E0 7212 <1> jb short vga\_g\_down\_l2

3212 000022E2 88EC <1> mov ah, ch

3213 000022E4 28C4 <1> sub ah, al ; ah = i - nblines

3214 <1> ; else

3215 <1> ; vgamem\_copy\_pl4(cul,i,i-nblines,cols,nbcols,cheight);

3216 000022E6 E877FEFFFF <1> call vgamem\_copy\_l8

3217 <1> vga\_g\_down\_l1:

3218 000022EB 86F5 <1> xchg dh, ch

3219 000022ED FECE <1> dec dh

3220 000022EF 38EE <1> cmp dh, ch

3221 000022F1 73E1 <1> jnb short vga\_g\_down\_l0

3222 000022F3 C3 <1> retn

3223 <1>

3224 <1> vga\_g\_down\_l2:

3225 <1> ; vgamem\_fill\_pl4(cul,i,cols,nbcols,cheight,attr);

3226 000022F4 E8D3FEFFFF <1> call vgamem\_fill\_l8

3227 000022F9 EBF0 <1> jmp short vga\_g\_down\_l1

3228 <1>

3229 <1> vga\_g\_down\_planar:

3230 <1> ; cl = upper left column ; cul

3231 <1> ; ch = upper left row ; rul

3232 <1> ; dl = lower rigth column ; clr

3233 <1> ; dh = lower right row ; rlr

3234 <1> vga\_g\_down\_pl0:

3235 <1> ;{for(i=rlr;i>=rul;i--)

3236 <1> ; if((i<rul+nblines)||(nblines==0))

3237 000022FB 08C0 <1> or al, al

3238 000022FD 741C <1> jz short vga\_g\_down\_pl2

3239 000022FF 88C4 <1> mov ah, al

3240 00002301 00EC <1> add ah, ch

3241 <1> ;jc short vga\_g\_down\_pl2

3242 00002303 86EE <1> xchg ch, dh

3243 00002305 38E5 <1> cmp ch, ah

3244 00002307 7212 <1> jb short vga\_g\_down\_pl2

3245 00002309 88EC <1> mov ah, ch

3246 0000230B 28C4 <1> sub ah, al ; ah = i - nblines

3247 <1> ; else

3248 <1> ; vgamem\_copy\_pl4(cul,i,i-nblines,cols,nbcols,cheight);

3249 0000230D E88BFDFFFF <1> call vgamem\_copy\_pl4

3250 <1> vga\_g\_down\_pl1:

3251 00002312 86F5 <1> xchg dh, ch

3252 00002314 FECE <1> dec dh

3253 00002316 38EE <1> cmp dh, ch

3254 00002318 73E1 <1> jnb short vga\_g\_down\_pl0

3255 0000231A C3 <1> retn

3256 <1>

3257 <1> vga\_g\_down\_pl2:

3258 <1> ; vgamem\_fill\_pl4(cul,i,cols,nbcols,cheight,attr);

3259 0000231B E8EBFDFFFF <1> call vgamem\_fill\_pl4

3260 00002320 EBF0 <1> jmp short vga\_g\_down\_pl1

3261 <1>

3262 <1> ; 07/07/2016

3263 <1> ; 27/06/2016 - TRDOS 386 (TRDOS v2.0)

3264 <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)

3265 <1> ;------------------------------------------------------

3266 <1> ; SCROLL DOWN

3267 <1> ; THIS ROUTINE SCROLLS DOWN THE INFORMATION ON THE CRT

3268 <1> ; ENTRY --

3269 <1> ; CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL

3270 <1> ; DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL

3271 <1> ; BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS

3272 <1> ; BH = FILL VALUE FOR BLANKED LINES

3273 <1> ; AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE FIELD)

3274 <1> ; DS = DATA SEGMENT

3275 <1> ; ES = REGEN SEGMENT

3276 <1> ; EXIT --

3277 <1> ; NOTHING, THE SCREEN IS SCROLLED

3278 <1> ;--------------------------------------------------------

3279 <1>

3280 <1> ; cl = upper left column

3281 <1> ; ch = upper left row

3282 <1> ; dl = lower rigth column

3283 <1> ; dh = lower right row

3284 <1> ;

3285 <1> ; al = line count (AL=0 means blank entire fields)

3286 <1> ; bl = fill value for blanked lines

3287 <1> ; bh = unused

3288 <1>

3289 <1> GRAPHICS\_DOWN:

3290 <1> ; 07/07/2016

3291 <1> ;AH = Current video mode, [CRT\_MODE]

3292 <1> ;STD ; SET DIRECTION

3293 00002322 80FC07 <1> cmp ah, 7

3294 00002325 0F87F4FEFFFF <1> ja vga\_graphics\_down

3295 <1> ;je \_n0

3296 <1>

3297 0000232B 88C7 <1> MOV bh, al ; save line count in BH

3298 0000232D 6689D0 <1> MOV AX, DX ; GET LOWER RIGHT POSITION INTO AX REG

3299 <1>

3300 <1> ;----- USE CHARACTER SUBROUTINE FOR POSITIONING

3301 <1> ;----- ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE

3302 <1>

3303 00002330 E8F1010000 <1> CALL GRAPH\_POSN

3304 00002335 0FB7F8 <1> MOVzx eDI, AX ; SAVE RESULT AS DESTINATION ADDRESS

3305 <1>

3306 <1> ;----- DETERMINE SIZE OF WINDOW

3307 <1>

3308 00002338 6629CA <1> SUB DX, CX

3309 0000233B 6681C20101 <1> ADD DX, 101h ; ADJUST VALUES

3310 00002340 C0E602 <1> SAL DH, 2 ; MULTIPLY ROWS BY 4 AT 8 VERT DOTS/CHAR

3311 <1> ; AND EVEN/ODD ROWS

3312 <1>

3313 <1> ;----- DETERMINE CRT MODE

3314 <1>

3315 00002343 803D[C25E0000]06 <1> CMP byte [CRT\_MODE], 6 ; TEST FOR MEDIUM RES

3316 0000234A 7307 <1> JNC short \_R12 ; FIND\_SOURCE\_DOWN

3317 <1>

3318 <1> ;----- MEDIUM RES DOWN

3319 0000234C D0E2 <1> SAL DL, 1 ; # COLUMNS \* 2, SINCE 2 BYTES/CHAR

3320 0000234E 66D1E7 <1> SAL DI, 1 ; OFFSET \*2 SINCE 2 BYTES/CHAR

3321 00002351 6647 <1> INC DI ; POINT TO LAST BYTE

3322 <1>

3323 <1> ;----- DETERMINE THE SOURCE ADDRESS IN THE BUFFER

3324 <1>

3325 <1> \_R12: ; FIND\_SOURCE\_DOWN

3326 00002353 81C700800B00 <1> add edi, 0B8000h

3327 00002359 6681C7F000 <1> ADD DI, 240 ; POINT TO LAST ROW OF PIXELS

3328 0000235E C0E702 <1> sal bh, 2 ; multiply number of lines by 4

3329 00002361 74(06) <1> JZ short 6 ; IF ZERO, THEN BLANK ENTIRE FIELD

3330 00002363 B050 <1> MOV AL, 80 ; 80 BYTES/ROW

3331 00002365 F6E7 <1> mul bh ; determine offset to source

3332 00002367 89FE <1> MOV eSI, eDI ; SET UP SOURCE

3333 00002369 6629C6 <1> SUB SI, AX ; SUBTRACT THE OFFSET

3334 0000236C 88F4 <1> MOV AH, DH ; NUMBER OF ROWS IN FIELD

3335 0000236E 28FC <1> sub ah, bh ; determine number to move

3336 <1>

3337 <1> ;----- LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS

3338 <1>

3339 <1> \_R13: ; ROW\_LOOP\_DOWN

3340 00002370 E823000000 <1> CALL \_R17 ; MOVE ONE ROW

3341 00002375 6681EE5020 <1> SUB SI, 2000h+80 ; MOVE TO NEXT ROW

3342 0000237A 6681EF5020 <1> SUB DI, 2000h+80

3343 0000237F FECC <1> DEC AH ; NUMBER OF ROWS TO MOVE

3344 00002381 75ED <1> JNZ short \_R13 ; CONTINUE TILL ALL MOVED

3345 <1>

3346 <1> ;----- FILL IN THE VACATED LINE(S)

3347 <1> \_R14: ; CLEAR\_ENTRY\_DOWN

3348 00002383 88D8 <1> mov al, bl ; attribute to fill with

3349 <1> \_R15\_: ; CLEAR\_LOOP\_DOWN

3350 00002385 E82A000000 <1> CALL \_R18 ; CLEAR A ROW

3351 0000238A 6681EF5020 <1> SUB DI, 2000h+80 ; POINT TO NEXT LINE

3352 0000238F FECF <1> dec bh ; number of lines to fill

3353 00002391 75F2 <1> JNZ short \_R15\_ ; CLEAR\_LOOP\_DOWN

3354 <1>

3355 00002393 C3 <1> retn ; EVERYYHING DONE

3356 <1>

3357 <1> \_R16: ; BLANK\_FIELD\_DOWN

3358 00002394 88F7 <1> mov bh, dh ; set blank count to everything in field

3359 00002396 EBEB <1> JMP short \_R14 ; CLEAR THE FIELD

3360 <1>

3361 <1> ; 27/06/2016 - TRDOS 386 (TRDOS v2.0)

3362 <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)

3363 <1>

3364 <1> ;----- ROUTINE TO MOVE ONE ROW OF INFORMATION

3365 <1>

3366 <1> \_R17:

3367 00002398 0FB6CA <1> MOVzx ecx, DL ; NUMBER OF BYTES IN THE ROW

3368 0000239B 56 <1> PUSH eSI

3369 0000239C 57 <1> PUSH eDI ; SAVE POINTERS

3370 0000239D F3A4 <1> REP MOVSB ; MOVE THE EVEN FIELD

3371 0000239F 5F <1> POP eDI

3372 000023A0 5E <1> POP eSI

3373 000023A1 6681C60020 <1> ADD SI, 2000h

3374 000023A6 6681C70020 <1> ADD DI, 2000h ; POINT TO THE ODD FIELD

3375 000023AB 56 <1> PUSH eSI

3376 000023AC 57 <1> PUSH eDI ; SAVE THE POINTERS

3377 000023AD 88D1 <1> MOV CL, DL ; COUNT BACK

3378 000023AF F3A4 <1> REP MOVSB ; MOVE THE ODD FIELD

3379 000023B1 5F <1> POP eDI

3380 000023B2 5E <1> POP eSI ; POINTERS BACK

3381 000023B3 C3 <1> RETn ; RETURN TO CALLER

3382 <1>

3383 <1> ;----- CLEAR A SINGLE ROW

3384 <1>

3385 <1> \_R18:

3386 000023B4 0FB6CA <1> MOVzx ecx, DL ; NUMBER OF BYTES IN FIELD

3387 000023B7 57 <1> PUSH eDI ; SAVE POINTER

3388 000023B8 F3AA <1> REP STOSB ; STORE THE NEW VALUE

3389 000023BA 5F <1> POP eDI ; POINTER BACK

3390 000023BB 6681C70020 <1> ADD DI, 2000h ; POINT TO ODD FIELD

3391 000023C0 57 <1> PUSH eDI

3392 000023C1 88D1 <1> MOV CL, DL

3393 000023C3 F3AA <1> REP STOSB ; FILL THE ODD FIELD

3394 000023C5 5F <1> POP eDI

3395 000023C6 C3 <1> RETn ; RETURN TO CALLER

3396 <1>

3397 <1> ; 04/07/2016

3398 <1> ; 01/07/2016

3399 <1> ; 30/06/2016 - TRDOS 386 (TRDOS v2.0)

3400 <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)

3401 <1> ;--------------------------------------------------

3402 <1> ; GRAPHICS WRITE

3403 <1> ; THIS ROUTINE WRITES THE ASCII CHARACTER TO THE CURRENT

3404 <1> ; POSITION ON THE SCREEN.

3405 <1> ; ENTRY --

3406 <1> ; AL = CHARACTER TO WRITE

3407 <1> ; BL = COLOR ATTRIBUTE TO BE USED FOR FOREGROUND COLOR

3408 <1> ; IF BIT 7 IS SET, THE CHAR IS XOR'D INTO THE REGEN BUFFER

3409 <1> ; (0 IS USED FOR THE BACKGROUND COLOR)

3410 <1> ; CX = NUMBER OF CHARS TO WRITE

3411 <1> ; DS = DATA SEGMENT

3412 <1> ; ES = REGEN SEGMENT

3413 <1> ; EXIT --

3414 <1> ; NOTHING IS RETURNED

3415 <1> ;

3416 <1> ; GRAPHICS READ

3417 <1> ; THIS ROUTINE READS THE ASCII CHARACTER AT THE CURRENT CURSOR

3418 <1> ; POSITION ON THE SCREEN BY MATCHING THE DOTS ON THE SCREEN TO THE

3419 <1> ; CHARACTER GENERATOR CODE POINTS

3420 <1> ; ENTRY --

3421 <1> ; NONE (0 IS ASSUMED AS THE BACKGROUND COLOR)

3422 <1> ; EXIT --

3423 <1> ; AL = CHARACTER READ AT THAT POSITION (0 RETURNED IF NONE FOUND)

3424 <1> ;

3425 <1> ; FOR BOTH ROUTINES, THE IMAGES USED TO FORM CHARS ARE CONTAINED IN ROM

3426 <1> ; FOR THE 1ST 128 CHARS. TO ACCESS CHARS IN THE SECOND HALF, THE USER

3427 <1> ; MUST INITIALIZE THE VECTOR AT INTERRUPT 1FH (LOCATION 0007CH) TO

3428 <1> ; POINT TO THE USER SUPPLIED TABLE OF GRAPHIC IMAGES (8X8 BOXES).

3429 <1> ; FAILURE TO DO SO WILL CAUSE IN STRANGE RESULTS

3430 <1> ;-----------------------------------------------------

3431 <1>

3432 <1> GRAPHICS\_WRITE:

3433 000023C7 25FF000000 <1> and eax, 0FFh ; ZERO TO HIGH OF CODE POINT

3434 000023CC 50 <1> PUSH eAX ; SAVE CODE POINT VALUE

3435 <1>

3436 <1> ;----- DETERMINE POSITION IN REGEN BUFFER TO PUT CODE POINTS

3437 <1>

3438 000023CD E84D010000 <1> CALL S26 ; FIND LOCATION IN REGEN BUFFER

3439 000023D2 89C7 <1> MOV eDI, eAX ; REGEN POINTER IN DI

3440 <1>

3441 <1> ;----- DETERMINE REGION TO GET CODE POINTS FROM

3442 <1>

3443 000023D4 58 <1> POP eAX ; RECOVER CODE POINT

3444 <1>

3445 000023D5 BE[B82C0100] <1> MOV eSI, CRT\_CHAR\_GEN ; OFFSET OF IMAGES

3446 <1>

3447 <1> ;----- DETERMINE GRAPHICS MODE IN OPERATION

3448 <1> ; DETERMINE\_MODE

3449 000023DA 66C1E003 <1> SAL AX, 3 ; MULTIPLY CODE POINT VALUE BY 8

3450 000023DE 01C6 <1> ADD eSI, eAX ; SI HAS OFFSET OF DESIRED CODES

3451 <1>

3452 000023E0 803D[C25E0000]06 <1> CMP byte [CRT\_MODE], 6

3453 000023E7 7231 <1> JC short S6 ; TEST FOR MEDIUM RESOLUTION MODE

3454 <1>

3455 <1> ;----- HIGH RESOLUTION MODE

3456 <1>

3457 000023E9 81C700800B00 <1> add edi, 0B8000h

3458 <1> S1: ; HIGH\_CHAR

3459 000023EF 57 <1> PUSH eDI ; SAVE REGEN POINTER

3460 000023F0 56 <1> PUSH eSI ; SAVE CODE POINTER

3461 000023F1 B604 <1> MOV DH, 4 ; NUMBER OF TIMES THROUGH LOOP

3462 <1> S2:

3463 000023F3 AC <1> LODSB ; GET BYTE FROM CODE POINTS

3464 000023F4 F6C380 <1> TEST BL, 80H ; SHOULD WE USE THE FUNCTION

3465 000023F7 7515 <1> JNZ short S5 ; TO PUT CHAR IN

3466 000023F9 AA <1> STOSB ; STORE IN REGEN BUFFER

3467 000023FA AC <1> LODSB

3468 <1> S4:

3469 000023FB 8887FF1F0000 <1> MOV [eDI+2000H-1], AL ; STORE IN SECOND HALF

3470 00002401 83C74F <1> ADD eDI, 79 ; MOVE TO NEXT ROW IN REGEN

3471 00002404 FECE <1> DEC DH ; DONE WITH LOOP

3472 00002406 75EB <1> JNZ short S2

3473 00002408 5E <1> POP eSI

3474 00002409 5F <1> POP eDI ; RECOVER REGEN POINTER

3475 0000240A 47 <1> INC eDI ; POINT TO NEXT CHAR POSITION

3476 0000240B E2E2 <1> LOOP S1 ; MORE CHARS TO WRITE

3477 0000240D C3 <1> retn

3478 <1>

3479 <1> S5:

3480 0000240E 3207 <1> XOR AL, [eDI] ; EXCLUSIVE OR WITH CURRENT

3481 00002410 AA <1> STOSB ; STORE THE CODE POINT

3482 00002411 AC <1> LODSB ; AGAIN FOR ODD FIELD

3483 00002412 3287FF1F0000 <1> XOR AL, [eDI+2000H-1]

3484 00002418 EBE1 <1> JMP short S4 ; BACK TO MAINSTREAM

3485 <1>

3486 <1> ;----- MEDIUM RESOLUTION WRITE

3487 <1> S6: ; MED\_RES\_WRITE

3488 0000241A 88DA <1> MOV DL, BL ; SAVE HIGH COLOR BIT

3489 0000241C 66D1E7 <1> SAL DI, 1 ; OFFSET\*2 SINCE 2 BYTES/CHAR

3490 <1> ; EXPAND BL TO FULL WORD OF COLOR

3491 0000241F 80E303 <1> AND BL, 3 ; ISOLATE THE COLOR BITS ( LOW 2 BITS )

3492 00002422 B055 <1> MOV AL, 055H ; GET BIT CONVERSION MULTIPLIER

3493 00002424 F6E3 <1> MUL BL ; EXPAND 2 COLOR BITS TO 4 REPLICATIONS

3494 00002426 88C3 <1> MOV BL, AL ; PLACE BACK IN WORK REGISTER

3495 00002428 88C7 <1> MOV BH, AL ; EXPAND TO 8 REPLICATIONS OF COLOR BITS

3496 0000242A 81C700800B00 <1> add edi, 0B8000h

3497 <1> S7: ; MED\_CHAR

3498 00002430 57 <1> PUSH eDI ; SAVE REGEN POINTER

3499 00002431 56 <1> PUSH eSI ; SAVE THE CODE POINTER

3500 00002432 B604 <1> MOV DH, 4 ; NUMBER OF LOOPS

3501 <1> S8:

3502 00002434 AC <1> LODSB ; GET CODE POINT

3503 00002435 E8B3000000 <1> CALL S21 ; DOUBLE UP ALL THE BITS

3504 0000243A 6621D8 <1> AND AX, BX ; CONVERT TO FOREGROUND COLOR ( 0 BACK )

3505 0000243D 86E0 <1> XCHG AH, AL ; SWAP HIGH/LOW BYTES FOR WORD MOVE

3506 0000243F F6C280 <1> TEST DL, 80H ; IS THIS XOR FUNCTION

3507 00002442 7403 <1> JZ short S9 ; NO, STORE IT IN AS IS

3508 00002444 663307 <1> XOR AX, [eDI] ; DO FUNCTION WITH LOW/HIGH

3509 <1> S9:

3510 00002447 668907 <1> MOV [eDI], AX ; STORE FIRST BYTE HIGH, SECOND LOW

3511 0000244A AC <1> LODSB ; GET CODE POINT

3512 0000244B E89D000000 <1> CALL S21

3513 00002450 6621D8 <1> AND AX, BX ; CONVERT TO COLOR

3514 00002453 86E0 <1> XCHG AH, AL ; SWAP HIGH/LOW BYTES FOR WORD MOVE

3515 00002455 F6C280 <1> TEST DL, 80H ; AGAIN, IS THIS XOR FUNCTION

3516 00002458 7407 <1> JZ short \_S10 ; NO, JUST STORE THE VALUES

3517 0000245A 66338700200000 <1> XOR AX, [eDI+2000H] ; FUNCTION WITH FIRST HALF LOW

3518 <1> \_S10:

3519 00002461 66898700200000 <1> MOV [eDI+2000H], AX ; STORE SECOND PORTION HIGH

3520 00002468 6683C750 <1> ADD DI, 80 ; POINT TO NEXT LOCATION

3521 0000246C FECE <1> DEC DH

3522 0000246E 75C4 <1> JNZ short S8 ; KEEP GOING

3523 00002470 5E <1> POP eSI ; RECOVER CODE POINTER

3524 00002471 5F <1> POP eDI ; RECOVER REGEN POINTER

3525 00002472 47 <1> INC eDI ; POINT TO NEXT CHAR POSITION

3526 00002473 47 <1> INC eDI

3527 00002474 E2BA <1> LOOP S7 ; MORE TO WRITE

3528 00002476 C3 <1> retn

3529 <1>

3530 <1> ; 04/07/2016

3531 <1> ; 01/07/2016

3532 <1> ; 30/06/2016 - TRDOS 386 (TRDOS v2.0)

3533 <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)

3534 <1> ;----------------------------------------

3535 <1> ; GRAPHICS READ

3536 <1> ;----------------------------------------

3537 <1> GRAPHICS\_READ:

3538 00002477 E8A3000000 <1> CALL S26 ; CONVERTED TO OFFSET IN REGEN

3539 0000247C 89C6 <1> MOV eSI, eAX ; SAVE IN SI

3540 0000247E 81C600800B00 <1> add esi, 0B8000h ; 01/07/2016

3541 00002484 83EC08 <1> SUB eSP, 8 ; ALLOCATE SPACE FOR THE READ CODE POINT

3542 00002487 89E5 <1> MOV eBP, eSP ; POINTER TO SAVE AREA

3543 <1>

3544 <1> ;----- DETERMINE GRAPHICS MODES

3545 00002489 B604 <1> mov dh, 4 ; number of passes ; 01/07/2016

3546 0000248B 803D[C25E0000]06 <1> CMP byte [CRT\_MODE], 6

3547 00002492 7219 <1> JC short S12 ; MEDIUM RESOLUTION

3548 <1>

3549 <1> ;----- HIGH RESOLUTION READ

3550 <1> ;----- GET VALUES FROM REGEN BUFFER AND CONVERT TO CODE POINT

3551 <1> ;MOV DH,4 ; NUMBER OF PASSES

3552 <1> S11:

3553 00002494 8A06 <1> MOV AL, [eSI] ; GET FIRST BYTE

3554 00002496 884500 <1> MOV [eBP], AL ; SAVE IN STORAGE AREA

3555 00002499 45 <1> INC eBP ; NEXT LOCATION

3556 0000249A 8A8600200000 <1> MOV AL, [eSI+2000H] ; GET LOWER REGION BYTE

3557 000024A0 884500 <1> MOV [eBP], AL ; ADJUST AND STORE

3558 000024A3 45 <1> INC eBP

3559 000024A4 83C650 <1> ADD eSI, 80 ; POINTER INTO REGEN

3560 000024A7 FECE <1> DEC DH ; LOOP CONTROL

3561 000024A9 75E9 <1> JNZ short S11 ; DO IT SOME MORE

3562 000024AB EB1D <1> JMP SHORT S14 ; GO MATCH THE SAVED CODE POINTS

3563 <1>

3564 <1> ;----- MEDIUM RESOLUTION READ

3565 <1> S12:

3566 000024AD 66D1E6 <1> SAL SI, 1 ; OFFSET\*2 SINCE 2 BYTES/CHAR

3567 <1> ;MOV DH, 4 ; NUMBER OF PASSES

3568 <1> S13:

3569 000024B0 E84D000000 <1> CALL S23 ; GET BYTES FROM REGEN INTO SINGLE SAVE

3570 000024B5 81C6FE1F0000 <1> ADD eSI, 2000H-2 ; GO TO LOWER REGION

3571 000024BB E842000000 <1> CALL S23 ; GET THIS PAIR INTO SAVE

3572 000024C0 81EEB21F0000 <1> SUB eSI, 2000H-80+2 ; ADJUST POINTER BACK INTO UPPER

3573 000024C6 FECE <1> DEC DH

3574 000024C8 75E6 <1> JNZ short S13 ; KEEP GOING UNTIL ALL 8 DONE

3575 <1>

3576 <1> ;----- SAVE AREA HAS CHARACTER IN IT, MATCH IT

3577 <1> S14: ; FIND\_CHAR

3578 000024CA BF[B82C0100] <1> MOV eDI, CRT\_CHAR\_GEN ; ESTABLISH ADDRESSING

3579 000024CF 83ED08 <1> SUB eBP, 8 ; ADJUST POINTER TO START OF SAVE AREA

3580 000024D2 89EE <1> MOV eSI, eBP

3581 <1> S15:

3582 000024D4 66B80001 <1> mov ax, 256 ; NUMBER TO TEST AGAINST

3583 <1> S16:

3584 000024D8 56 <1> PUSH eSI ; SAVE SAVE AREA POINTER

3585 000024D9 57 <1> PUSH eDI ; SAVE CODE POINTER

3586 <1> ;MOV eCX, 4 ; NUMBER OF WORDS TO MATCH

3587 <1> ;REPE CMPSW ; COMPARE THE 8 BYTES AS WORDS

3588 000024DA A7 <1> cmpsd ; compare first 4 bytes

3589 000024DB 7501 <1> jne short S17 ;

3590 000024DD A7 <1> cmpsd ; compare last 4 bytes

3591 <1> S17:

3592 000024DE 5F <1> POP eDI ; RECOVER THE POINTERS

3593 000024DF 5E <1> POP eSI

3594 <1> ;JZ short S18 ; IF ZERO FLAG SET, THEN MATCH OCCURRED

3595 000024E0 7407 <1> je short S18

3596 <1> ; ; NO MATCH, MOVE ON TO NEXT

3597 000024E2 83C708 <1> ADD eDI, 8 ; NEXT CODE POINT

3598 000024E5 6648 <1> dec ax ; LOOP CONTROL

3599 000024E7 75EF <1> JNZ short S16 ; DO ALL OF THEM

3600 <1>

3601 <1> ;----- CHARACTER IS FOUND ( AL=0 IF NOT FOUND )

3602 <1> S18:

3603 000024E9 83C408 <1> ADD eSP, 8 ; READJUST THE STACK, THROW AWAY SAVE

3604 000024EC C3 <1> retn ; ALL DONE

3605 <1>

3606 <1> ; 30/06/2016 - TRDOS 386 (TRDOS v2.0)

3607 <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)

3608 <1> ;--------------------------------------------

3609 <1> ; EXPAND BYTE

3610 <1> ; THIS ROUTINE TAKES THE BYTE IN AL AND DOUBLES ALL

3611 <1> ; OF THE BITS, TURNING THE 8 BITS INTO 16 BITS.

3612 <1> ; THE RESULT IS LEFT IN AX

3613 <1> ;--------------------------------------------

3614 <1> S21:

3615 000024ED 6651 <1> PUSH CX ; SAVE REGISTER

3616 <1> ;MOV CX, 8 ; SHIFT COUNT REGISTER FOR ONE BYTE

3617 000024EF B108 <1> mov cl, 8

3618 <1> S22:

3619 000024F1 D0C8 <1> ROR AL,1 ; SHIFT BITS, LOW BIT INTO CARRY FLAG

3620 000024F3 66D1DD <1> RCR BP,1 ; MOVE CARRY FLAG (LOW BIT INTO RESULTS

3621 000024F6 66D1FD <1> SAR BP,1 ; SIGN EXTEND HIGH BIT (DOUBLE IT)

3622 <1> ;LOOP S22 ; REPEAT FOR ALL 8 BITS

3623 000024F9 FEC9 <1> dec cl

3624 000024FB 75F4 <1> jnz short S22

3625 000024FD 6695 <1> XCHG AX, BP ; MOVE RESULTS TO PARAMETER REGISTER

3626 000024FF 6659 <1> POP CX ; RECOVER REGISTER

3627 00002501 C3 <1> RETn ; ALL DONE

3628 <1>

3629 <1> ; 01/07/2016 - TRDOS 386 (TRDOS v2.0)

3630 <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)

3631 <1> ;--------------------------------------------------

3632 <1> ; MED\_READ\_BYTE

3633 <1> ; THIS ROUTINE WILL TAKE 2 BYTES FROM THE REGEN BUFFER,

3634 <1> ; COMPARE AGAINST THE CURRENT FOREGROUND COLOR, AND PLACE

3635 <1> ; THE CORRESPONDING ON/OFF BIT PATTERN INTO THE CURRENT

3636 <1> ; POSITION IN THE SAVE AREA

3637 <1> ; ENTRY --

3638 <1> ; SI,DS = POINTER TO REGEN AREA OF INTEREST

3639 <1> ; BX = EXPANDED FOREGROUND COLOR

3640 <1> ; BP = POINTER TO SAVE AREA

3641 <1> ; EXIT --

3642 <1> ; SI AND BP ARE INCREMENTED

3643 <1> ;----------------------------------------------------

3644 <1> S23:

3645 00002502 66AD <1> LODSW ; GET FIRST BYTE AND SECOND BYTES

3646 00002504 86C4 <1> XCHG AL, AH ; SWAP FOR COMPARE

3647 00002506 66B900C0 <1> MOV CX, 0C000H ; 2 BIT MASK TO TEST THE ENTRIES

3648 0000250A B200 <1> MOV DL, 0 ; RESULT REGISTER

3649 <1> S24:

3650 0000250C 6685C8 <1> TEST AX, CX ; IS THIS SECTION BACKCROUND?

3651 0000250F 7401 <1> JZ short S25 ; IF ZERO, IT IS BACKGROUND (CARRY=0)

3652 00002511 F9 <1> STC ; WASN'T, SO SET CARRY

3653 <1> S25:

3654 00002512 D0D2 <1> RCL DL, 1 ; MOVE THAT BIT INTO THE RESULT

3655 00002514 66C1E902 <1> SHR CX, 2 ; MOVE THE MASK TO THE RIGHT BY 2 BITS

3656 00002518 73F2 <1> JNC short S24 ; DO IT AGAIN IF MASK DIDN'T FALL OUT

3657 0000251A 885500 <1> MOV [eBP], DL ; STORE RESULT IN SAVE AREA

3658 0000251D 45 <1> INC eBP ; ADJUST POINTER

3659 0000251E C3 <1> RETn ; ALL DONE

3660 <1>

3661 <1> ; 30/06/2016 - TRDOS 386 (TRDOS v2.0)

3662 <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)

3663 <1> ;-----------------------------------------

3664 <1> ; V4\_POSITION

3665 <1> ; THIS ROUTINE TAKES THE CURSOR POSITION CONTAINED IN

3666 <1> ; THE MEMORY LOCATION, AND CONVERTS IT INTO AN OFFSET

3667 <1> ; INTO THE REGEN BUFFER, ASSUMING ONE BYTE/CHAR.

3668 <1> ; FOR MEDIUM RESOLUTION GRAPHICS, THE NUMBER MUST

3669 <1> ; BE DOUBLED.

3670 <1> ; ENTRY -- NO REGISTERS,MEMORY LOCATION @CURSOR\_POSN IS USED

3671 <1> ; EXIT--

3672 <1> ; AX CONTAINS OFFSET INTO REGEN BUFFER

3673 <1> ;-----------------------------------------

3674 <1> S26:

3675 0000251F 0FB705[56580100] <1> movzx eax, word [CURSOR\_POSN] ; GET CURRENT CURSOR

3676 <1> GRAPH\_POSN:

3677 00002526 53 <1> PUSH eBX ; SAVE REGISTER

3678 00002527 0FB6D8 <1> movzx ebx, al ; SAVE A COPY OF CURRENT CURSOR

3679 0000252A A0[C45E0000] <1> MOV AL, [CRT\_COLS] ; GET BYTES PER COLUMN

3680 0000252F F6E4 <1> MUL AH ; MULTIPLY BY ROWS

3681 00002531 66C1E002 <1> SHL AX, 2 ; MULTIPLY \* 4 SINCE 4 ROWS/BYTE

3682 00002535 01D8 <1> ADD eAX, eBX ; DETERMINE OFFSET

3683 00002537 5B <1> POP eBX ; RECOVER POINTER

3684 00002538 C3 <1> RETn ; ALL DONE

3685 <1>

3686 <1> ; 09/07/2016

3687 <1> ; 01/07/2016 - TRDOS 386 (TRDOS v2.0)

3688 <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)

3689 <1> ;---------------------------------------------

3690 <1> ; SET\_COLOR

3691 <1> ; THIS ROUTINE WILL ESTABLISH THE BACKGROUND COLOR, THE OVERSCAN COLOR,

3692 <1> ; AND THE FOREGROUND COLOR SET FOR MEDIUM RESOLUTION GRAPHICS

3693 <1> ; INPUT

3694 <1> ; (BH) HAS COLOR ID

3695 <1> ; IF BH=0, THE BACKGROUND COLOR VALUE IS SET

3696 <1> ; FROM THE LOW BITS OF BL (0-31)

3697 <1> ; IF BH=1, THE PALETTE SELECTION IS MADE

3698 <1> ; BASED ON THE LOW BIT OF BL:

3699 <1> ; 0 = GREEN, RED, YELLOW FOR COLORS 1,2,3

3700 <1> ; 1 = BLUE, CYAN, MAGENTA FOR COLORS 1,2,3

3701 <1> ; (BL) HAS THE COLOR VALUE TO BE USED

3702 <1> ; OUTPUT

3703 <1> ; THE COLOR SELECTION IS UPDATED

3704 <1> ;----------------------------------------------

3705 <1> SET\_COLOR:

3706 00002539 803D[C25E0000]07 <1> cmp byte [CRT\_MODE], 7 ; 09/07/2016

3707 00002540 0F870EF0FFFF <1> ja VIDEO\_RETURN ; nothing to do for VGA modes

3708 <1>

3709 <1> ;MOV DX, [ADDR\_6845] ; I/O PORT FOR PALETTE

3710 <1> ;mov dx, 3D4h

3711 <1> ;ADD DX,5 ; OVERSCAN PORT

3712 00002546 66BAD903 <1> mov dx, 3D9h

3713 0000254A A0[C55E0000] <1> MOV AL, [CRT\_PALETTE] ; GET THE CURRENT PALETTE VALUE

3714 0000254F 08FF <1> OR BH, BH ; IS THIS COLOR 0?

3715 00002551 7512 <1> JNZ short M20 ; OUTPUT COLOR 1

3716 <1>

3717 <1> ;----- HANDLE COLOR 0 BY SETTING THE BACKGROUND COLOR

3718 <1>

3719 00002553 24E0 <1> AND AL, 0E0H ; TURN OFF LOW 5 BITS OF CURRENT

3720 00002555 80E31F <1> AND BL, 01FH ; TURN OFF HIGH 3 BITS OF INPUT VALUE

3721 00002558 08D8 <1> OR AL, BL ; PUT VALUE INTO REGISTER

3722 <1> M19: ; OUTPUT THE PALETTE

3723 0000255A EE <1> OUT DX, AL ; OUTPUT COLOR SELECTION TO 3D9 PORT

3724 0000255B A2[C55E0000] <1> MOV [CRT\_PALETTE], AL ; SAVE THE COLOR VALUE

3725 00002560 E9EFEFFFFF <1> JMP VIDEO\_RETURN

3726 <1>

3727 <1> ;----- HANDLE COLOR 1 BY SELECTING THE PALETTE TO BE USED

3728 <1>

3729 <1> M20:

3730 00002565 24DF <1> AND AL, 0DFH ; TURN OFF PALETTE SELECT BIT

3731 00002567 D0EB <1> SHR BL, 1 ; TEST THE LOW ORDER BIT OF BL

3732 00002569 73EF <1> JNC short M19 ; ALREADY DONE

3733 0000256B 0C20 <1> OR AL, 20H ; TURN ON PALETTE SELECT BIT

3734 0000256D EBEB <1> JMP short M19 ; GO DO IT

3735 <1>

3736 <1> ; 09/07/2016

3737 <1> ; 01/07/2016 - TRDOS 386 (TRDOS v2.0)

3738 <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)

3739 <1> ;--------------------------------------------

3740 <1> ; READ DOT -- WRITE DOT

3741 <1> ; THESE ROUTINES WILL WRITE A DOT, OR READ THE

3742 <1> ; DOT AT THE INDICATED LOCATION

3743 <1> ; ENTRY --

3744 <1> ; DX = ROW (0-199) (THE ACTUAL VALUE DEPENDS ON THE MODE)

3745 <1> ; CX = COLUMN ( 0-639) ( THE VALUES ARE NOT RANGE CHECKED )

3746 <1> ; AL = DOT VALUE TO WRITE (1,2 OR 4 BITS DEPENDING ON MODE,

3747 <1> ; REQUIRED FOR WRITE DOT ONLY, RIGHT JUSTIFIED)

3748 <1> ; BIT 7 OF AL = 1 INDICATES XOR THE VALUE INTO THE LOCATION

3749 <1> ; DS = DATA SEGMENT

3750 <1> ; ES = REGEN SEGMENT

3751 <1> ;

3752 <1> ; EXIT

3753 <1> ; AL = DOT VALUE READ, RIGHT JUSTIFIED, READ ONLY

3754 <1> ;----------------------------------------------

3755 <1>

3756 <1> READ\_DOT:

3757 <1> ; 09/07/2016

3758 0000256F 8A25[C25E0000] <1> mov ah, [CRT\_MODE]

3759 00002575 80FC07 <1> cmp ah, 7 ; 6!?

3760 00002578 760A <1> jna short read\_dot\_cga

3761 <1>

3762 0000257A E8CB030000 <1> call vga\_read\_pixel

3763 <1> ; al = pixel value

3764 0000257F E9D5EFFFFF <1> jmp \_video\_return

3765 <1>

3766 <1> read\_dot\_cga:

3767 <1> ;je VIDEO\_RETURN ; 7

3768 00002584 80FC04 <1> cmp ah, 4 ; graphics ?

3769 00002587 0F82C7EFFFFF <1> jb VIDEO\_RETURN ; no, text mode, nothing to do

3770 <1>

3771 0000258D E855000000 <1> CALL R3 ; DETERMINE BYTE POSITION OF DOT

3772 00002592 8A06 <1> MOV AL, [eSI] ; GET THE BYTE

3773 00002594 20E0 <1> AND AL, AH ; MASK OFF THE OTHER BITS IN THE BYTE

3774 00002596 D2E0 <1> SHL AL, CL ; LEFT JUSTIFY THE VALUE

3775 00002598 88F1 <1> MOV CL, DH ; GET NUMBER OF BITS IN RESULT

3776 0000259A D2C0 <1> ROL AL, CL ; RIGHT JUSTIFY THE RESULT

3777 <1> ;JMP VIDEO\_RETURN ; RETURN FROM VIDEO I/O

3778 0000259C 0FB6C0 <1> movzx eax, al

3779 0000259F E9B5EFFFFF <1> jmp \_video\_return

3780 <1>

3781 <1> ; 09/07/2016

3782 <1> ; 01/07/2016 - TRDOS 386 (TRDOS v2.0)

3783 <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)

3784 <1>

3785 <1> WRITE\_DOT:

3786 <1> ; 09/07/2016

3787 000025A4 8A25[C25E0000] <1> mov ah, [CRT\_MODE]

3788 000025AA 80FC07 <1> cmp ah, 7 ; 6!?

3789 000025AD 760A <1> jna short write\_dot\_cga

3790 <1>

3791 000025AF E805030000 <1> call vga\_write\_pixel

3792 000025B4 E99BEFFFFF <1> jmp VIDEO\_RETURN

3793 <1>

3794 <1> write\_dot\_cga:

3795 <1> ;je VIDEO\_RETURN ; 7

3796 000025B9 80FC04 <1> cmp ah, 4 ; graphics ?

3797 000025BC 0F8292EFFFFF <1> jb VIDEO\_RETURN ; no, text mode, nothing to do

3798 <1>

3799 <1> ;PUSH AX ; SAVE DOT VALUE

3800 000025C2 6650 <1> PUSH AX ; TWICE

3801 000025C4 E81E000000 <1> CALL R3 ; DETERMINE BYTE POSITION OF THE DOT

3802 000025C9 D2E8 <1> SHR AL, CL ; SHIFT TO SET UP THE BITS FOR OUTPUT

3803 000025CB 20E0 <1> AND AL, AH ; STRIP OFF THE OTHER BITS

3804 000025CD 8A0E <1> MOV CL, [eSI] ; GET THE CURRENT BYTE

3805 000025CF 665B <1> POP BX ; RECOVER XOR FLAG

3806 000025D1 F6C380 <1> TEST BL, 80H ; IS IT ON

3807 000025D4 750D <1> JNZ short R2 ; YES, XOR THE DOT

3808 000025D6 F6D4 <1> NOT AH ; SET MASK TO REMOVE THE INDICATED BITS

3809 000025D8 20E1 <1> AND CL, AH

3810 000025DA 08C8 <1> OR AL, CL ; OR IN THE NEW VALUE OF THOSE BITS

3811 <1> R1: ; FINISH\_DOT

3812 000025DC 8806 <1> MOV [eSI], AL ; RESTORE THE BYTE IN MEMORY

3813 <1> ;POP AX

3814 000025DE E971EFFFFF <1> JMP VIDEO\_RETURN ; RETURN FROM VIDEO I/O

3815 <1> R2: ; XOR\_DOT

3816 000025E3 30C8 <1> XOR AL, CL ; EXCLUSIVE OR THE DOTS

3817 000025E5 EBF5 <1> JMP short R1 ; FINISH UP THE WRITING

3818 <1>

3819 <1> ; 01/07/2016 - TRDOS 386 (TRDOS v2.0)

3820 <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)

3821 <1>

3822 <1> ;----------------------------------------------

3823 <1> ; THIS SUBROUTINE DETERMINES THE REGEN BYTE LOCATION OF THE

3824 <1> ; INDICATED ROW COLUMN VALUE IN GRAPHICS MODE.

3825 <1> ; ENTRY --

3826 <1> ; DX = ROW VALUE (0-199)

3827 <1> ; CX = COLUMN VALUE (0-639)

3828 <1> ; EXIT --

3829 <1> ; SI = OFFSET INTO REGEN BUFFER FOR BYTE OF INTEREST

3830 <1> ; AH = MASK TO STRIP OFF THE BITS OF INTEREST

3831 <1> ; CL = BITS TO SHIFT TO RIGHT JUSTIFY THE MASK IN AH

3832 <1> ; DH = # BITS IN RESULT

3833 <1> ; BX = MODIFIED

3834 <1> ;-----------------------------------------------

3835 <1> R3:

3836 <1>

3837 <1> ;----- DETERMINE 1ST BYTE IN INDICATED ROW BY MULTIPLYING ROW VALUE BY 40

3838 <1> ;----- ( LOW BIT OF ROW DETERMINES EVEN/ODD, 80 BYTES/ROW )

3839 <1>

3840 000025E7 0FB7F0 <1> movzx esi, ax ; WILL SAVE AL AND AH DURING OPERATION

3841 000025EA B028 <1> MOV AL, 40

3842 000025EC F6E2 <1> MUL DL ; AX= ADDRESS OF START OF INDICATED ROW

3843 000025EE A808 <1> TEST AL, 08H ; TEST FOR EVEN/ODD ROW CALCULATED

3844 000025F0 7404 <1> JZ short R4 ; JUMP IF EVEN ROW

3845 000025F2 6605D81F <1> ADD AX, 2000H-40 ; OFFSET TO LOCATION OF ODD ROWS ADJUST

3846 <1> R4: ; EVEN\_ROW

3847 000025F6 6696 <1> XCHG SI, AX ; MOVE POINTER TO (SI) AND RECOVER (AX)

3848 000025F8 81C600800B00 <1> add esi, 0B8000h

3849 000025FE 6689CA <1> MOV DX, CX ; COLUMN VALUE TO DX

3850 <1>

3851 <1> ;----- DETERMINE GRAPHICS MODE CURRENTLY IN EFFECT

3852 <1>

3853 <1> ; SET UP THE REGISTERS ACCORDING TO THE MODE

3854 <1> ; CH = MASK FOR LOW OF COLUMN ADDRESS ( 7/3 FOR HIGH/MED RES )

3855 <1> ; CL = # OF ADDRESS BITS IN COLUMN VALUE ( 3/2 FOR H/M )

3856 <1> ; BL = MASK TO SELECT BITS FROM POINTED BYTE ( 80H/C0H FOR H/M )

3857 <1> ; BH = NUMBER OF VALID BITS IN POINTED BYTE ( 1/2 FOR H/M )

3858 <1>

3859 00002601 66BBC002 <1> MOV BX, 2C0H

3860 00002605 66B90203 <1> MOV CX, 302H ; SET PARMS FOR MED RES

3861 00002609 803D[C25E0000]06 <1> CMP byte [CRT\_MODE], 6

3862 00002610 7208 <1> JC short R5 ; HANDLE IF MED RES

3863 00002612 66BB8001 <1> MOV BX, 180H

3864 00002616 66B90307 <1> MOV CX, 703H ; SET PARMS FOR HIGH RES

3865 <1>

3866 <1> ;----- DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK

3867 <1> R5:

3868 0000261A 20D5 <1> AND CH, DL ; ADDRESS OF PEL WITHIN BYTE TO CH

3869 <1>

3870 <1> ;----- DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN

3871 <1>

3872 0000261C 66D3EA <1> SHR DX, CL ; SHIFT BY CORRECT AMOUNT

3873 0000261F 6601D6 <1> ADD SI, DX ; INCREMENT THE POINTER

3874 00002622 88FE <1> MOV DH, BH ; GET THE # OF BITS IN RESULT TO DH

3875 <1>

3876 <1> ;----- MULTIPLY BH (VALID BITS IN BYTE) BY CH (BIT OFFSET)

3877 <1>

3878 00002624 28C9 <1> SUB CL, CL ; ZERO INTO STORAGE LOCATION

3879 <1> R6:

3880 00002626 D0C8 <1> ROR AL, 1 ; LEFT JUSTIFY VALUE IN AL (FOR WRITE)

3881 00002628 00E9 <1> ADD CL, CH ; ADD IN THE BIT OFFSET VALUE

3882 0000262A FECF <1> DEC BH ; LOOP CONTROL

3883 0000262C 75F8 <1> JNZ short R6 ; ON EXIT, CL HAS COUNT TO RESTORE BITS

3884 0000262E 88DC <1> MOV AH, BL ; GET MASK TO AH

3885 00002630 D2EC <1> SHR AH, CL ; MOVE THE MASK TO CORRECT LOCATION

3886 00002632 C3 <1> RETn ; RETURN WITH EVERYTHING SET UP

3887 <1>

3888 <1> load\_dac\_palette:

3889 <1> ; 29/07/2016

3890 <1> ; 23/07/2016

3891 <1> ; 03/07/2016 (TRDOS 386 = TRDOS v2.0)

3892 <1> ; (set\_mode\_vga)

3893 <1> ; derived from 'Plex86/Bochs VGABios' source code

3894 <1> ; vgabios-0.7a (2011)

3895 <1> ; by the LGPL VGABios developers Team (2001-2008)

3896 <1> ; 'vgabios.c', 'load\_dac\_palette'

3897 <1> ;

3898 <1> ; Oracle VirtualBox 5.0.24 VGABios Source Code

3899 <1> ; ('vgabios.c', 'vgatables.h', 'vgafonts.h', 'vgarom.asm')

3900 <1> ;

3901 <1> ; INPUT -> AH = DAC selection number (3, 2 or 1)

3902 <1> ; OUTPUT -> ECX = 0, AX = 0

3903 <1> ; (Modifed registers: EAX, ECX, EDX, ESI)

3904 <1> ;

3905 00002633 66BAC803 <1> mov dx, 3C8h ; VGAREG\_DAC\_WRITE\_ADDRESS

3906 00002637 28C0 <1> sub al, al ; 0

3907 00002639 EE <1> out dx, al ; 0 ; color index, always 0 at the beginning

3908 0000263A 6642 <1> inc dx ; 3C9h ; VGAREG\_DAC\_DATA

3909 0000263C B900010000 <1> mov ecx, 256 ; always 256\*3 values

3910 <1> ;push esi

3911 00002641 88E0 <1> mov al, ah

3912 00002643 B43F <1> mov ah, 3Fh ; 3Fh except DAC selection number 3

3913 00002645 3C02 <1> cmp al, 2

3914 00002647 7414 <1> je short l\_dac\_p\_2

3915 00002649 7719 <1> ja short l\_dac\_p\_3

3916 0000264B 20C0 <1> and al, al

3917 0000264D 7507 <1> jnz short l\_dac\_p\_1

3918 <1> l\_dac\_p\_0:

3919 0000264F BE[78270100] <1> mov esi, palette0

3920 00002654 EB15 <1> jmp short l\_dac\_p\_4

3921 <1> l\_dac\_p\_1:

3922 00002656 BE[38280100] <1> mov esi, palette1

3923 0000265B EB0E <1> jmp short l\_dac\_p\_4

3924 <1> l\_dac\_p\_2:

3925 0000265D BE[F8280100] <1> mov esi, palette2

3926 00002662 EB07 <1> jmp short l\_dac\_p\_4

3927 <1> l\_dac\_p\_3:

3928 00002664 B4FF <1> mov ah, 0FFh ; dac registers

3929 00002666 BE[B8290100] <1> mov esi, palette3

3930 <1> l\_dac\_p\_4:

3931 0000266B AC <1> lodsb

3932 0000266C EE <1> out dx, al ; Red

3933 0000266D AC <1> lodsb

3934 0000266E EE <1> out dx, al ; Green

3935 0000266F AC <1> lodsb

3936 00002670 EE <1> out dx, al ; Blue

3937 00002671 20E4 <1> and ah, ah

3938 00002673 7405 <1> jz short l\_dac\_p\_5

3939 00002675 FECC <1> dec ah

3940 00002677 E2F2 <1> loop l\_dac\_p\_4

3941 <1> ;pop esi

3942 00002679 C3 <1> retn

3943 <1> l\_dac\_p\_5:

3944 <1> ; 29/07/2016

3945 0000267A FEC9 <1> dec cl

3946 0000267C 7407 <1> jz short l\_dac\_p\_7

3947 <1> ;

3948 0000267E 28C0 <1> sub al, al ; 0

3949 <1> l\_dac\_p\_6:

3950 00002680 EE <1> out dx, al ; outb(VGAREG\_DAC\_DATA,0);

3951 00002681 EE <1> out dx, al

3952 00002682 EE <1> out dx, al

3953 00002683 E2FB <1> loop l\_dac\_p\_6

3954 <1> l\_dac\_p\_7:

3955 <1> ;pop esi

3956 00002685 C3 <1> retn

3957 <1>

3958 <1> gray\_scale\_summing:

3959 <1> ; 03/07/2016 (TRDOS 386 = TRDOS v2.0)

3960 <1> ; (set\_mode\_vga)

3961 <1> ; derived from 'Plex86/Bochs VGABios' source code

3962 <1> ; vgabios-0.7a (2011)

3963 <1> ; by the LGPL VGABios developers Team (2001-2008)

3964 <1> ; 'vgabios.c', 'biosfn\_perform\_gray\_scale\_summing'

3965 <1> ;

3966 <1> ; Oracle VirtualBox 5.0.24 VGABios Source Code

3967 <1> ; ('vgabios.c', 'vgatables.h', 'vgafonts.h', 'vgarom.asm')

3968 <1> ;

3969 <1>

3970 <1> ; INPUT -> EBX = Start address (color index <= 255)

3971 <1> ; ECX = Count (<= 256)

3972 <1> ; OUTPUT -> (E)CX = 0

3973 <1> ; (Modifed registers: EAX, ECX, EDX, EBX)

3974 <1>

3975 00002686 66BADA03 <1> mov dx, 3DAh ; VGAREG\_ACTL\_RESET

3976 0000268A EC <1> in al, dx

3977 0000268B 30C0 <1> xor al, al ; 0

3978 0000268D 66BAC003 <1> mov dx, 3C0h ; VGAREG\_ACTL\_ADDRESS

3979 00002691 EE <1> out dx, al ; clear bit 5

3980 <1> ; (while loading palette registers)

3981 <1> ; set read address and switch to read mode

3982 <1> g\_s\_s\_1:

3983 00002692 66BAC703 <1> mov dx, 3C7h ; VGAREG\_DAC\_READ\_ADDRESS

3984 00002696 88D8 <1> mov al, bl

3985 00002698 EE <1> out dx, al

3986 <1> ; get 6-bit wide RGB data values

3987 <1> ; intensity = (0.3\*Red)+(0.59\*Green)+(0.11\*Blue)

3988 <1> ; i = ( ( 77\*r + 151\*g + 28\*b ) + 0x80 ) >> 8;

3989 00002699 66BAC903 <1> mov dx, 3C9h ; VGAREG\_DAC\_DATA

3990 0000269D EC <1> in al, dx ; red

3991 0000269E B44D <1> mov ah, 77 ; 0.3\* Red

3992 000026A0 F6E4 <1> mul ah

3993 000026A2 6650 <1> push ax

3994 000026A4 EC <1> in al, dx ; green

3995 000026A5 B497 <1> mov ah, 151 ; 0.59 \* Green

3996 000026A7 F6E4 <1> mul ah

3997 000026A9 6650 <1> push ax

3998 000026AB EC <1> in al, dx ; blue

3999 000026AC B41C <1> mov ah, 28 ; 0.11 \* Blue

4000 000026AE F6E4 <1> mul ah

4001 000026B0 665A <1> pop dx

4002 000026B2 6601D0 <1> add ax, dx

4003 000026B5 665A <1> pop dx

4004 000026B7 6601D0 <1> add ax, dx

4005 000026BA 66058000 <1> add ax, 80h

4006 000026BE B03F <1> mov al, 3Fh

4007 000026C0 38C4 <1> cmp ah, al

4008 000026C2 7602 <1> jna short g\_s\_s\_2

4009 000026C4 88C4 <1> mov ah, al

4010 <1> g\_s\_s\_2:

4011 000026C6 66BAC803 <1> mov dx, 3C8h ; VGAREG\_DAC\_WRITE\_ADDRESS

4012 000026CA 88D8 <1> mov al, bl ; color index

4013 000026CC EE <1> out dx, al

4014 000026CD 88E0 <1> mov al, ah ; intensity

4015 000026CF 6642 <1> inc dx ; 3C9h ; VGAREG\_DAC\_DATA

4016 000026D1 EE <1> out dx, al ; R (R=G=B)

4017 000026D2 88E0 <1> mov al, ah ; intensity

4018 000026D4 EE <1> out dx, al ; G (R=G=B)

4019 000026D5 88E0 <1> mov al, ah ; intensity

4020 000026D7 EE <1> out dx, al ; B (R=G=B)

4021 000026D8 6649 <1> dec cx

4022 000026DA 7404 <1> jz short g\_s\_s\_3

4023 000026DC FEC3 <1> inc bl ; next color index value

4024 000026DE EBB2 <1> jmp short g\_s\_s\_1

4025 <1> g\_s\_s\_3:

4026 000026E0 66BADA03 <1> mov dx, 3DAh ; VGAREG\_ACTL\_RESET

4027 000026E4 EC <1> in al, dx

4028 000026E5 B020 <1> mov al, 20h

4029 000026E7 66BAC003 <1> mov dx, 3C0h ; VGAREG\_ACTL\_ADDRESS

4030 000026EB EE <1> out dx, al ; 20h -> set bit 5

4031 <1> ; (after loading palette regs)

4032 000026EC C3 <1> retn

4033 <1>

4034 <1> vga\_write\_char\_attr:

4035 <1> vga\_write\_char\_only:

4036 <1> ; 08/07/2016 (TRDOS 386 = TRDOS v2.0)

4037 <1> ;

4038 <1> ; derived from 'Plex86/Bochs VGABios' source code

4039 <1> ; vgabios-0.7a (2011)

4040 <1> ; by the LGPL VGABios developers Team (2001-2008)

4041 <1> ; 'vgabios.c', 'biosfn\_write\_char\_attr'

4042 <1> ; 'biosfn\_write\_char\_only'

4043 <1>

4044 <1> ; INPUT ->

4045 <1> ; [CRT\_MODE] = current video mode (>7)

4046 <1> ; CX = Count of characters to write

4047 <1> ; AL = Character to write

4048 <1> ; BL = Color of character

4049 <1> ; OUTPUT ->

4050 <1> ; Regen buffer updated

4051 <1>

4052 000026ED 8A25[C25E0000] <1> mov ah, [CRT\_MODE]

4053 000026F3 668B15[56580100] <1> mov dx, [CURSOR\_POSN] ; cursor pos for page 0

4054 <1>

4055 000026FA BE[DE5E0000] <1> mov esi, vga\_modes

4056 000026FF 89F7 <1> mov edi, esi

4057 00002701 83C710 <1> add edi, vga\_mode\_count

4058 <1> vga\_wca\_0:

4059 00002704 AC <1> lodsb

4060 00002705 38E0 <1> cmp al, ah ; [CRT\_MODE]

4061 00002707 7405 <1> je short vga\_wca\_2

4062 00002709 39FE <1> cmp esi, edi

4063 0000270B 72F7 <1> jb short vga\_wca\_0

4064 <1> vga\_wca\_1:

4065 0000270D C3 <1> retn ; nothing to do

4066 <1> vga\_wca\_2:

4067 0000270E 83C64F <1> add esi, vga\_memmodel - (vga\_modes + 1)

4068 <1> ; [ESI] = VGA memory model number (LINEAR8, PLANAR4, PLANAR1)

4069 <1>

4070 <1> ; biosfn\_write\_char\_attr (car,page,attr,count)

4071 <1> ; AL = car, page = 0, BL = attr, CX = count

4072 00002711 803E04 <1> cmp byte [esi], PLANAR4

4073 00002714 741D <1> je short vga\_wca\_planar

4074 00002716 803E03 <1> cmp byte [esi], PLANAR1

4075 00002719 7418 <1> je short vga\_wca\_planar

4076 <1> vga\_wca\_linear8:

4077 <1> ; while((count-->0) && (xcurs<nbcols))

4078 <1> ; CX = count

4079 0000271B 6621C9 <1> and cx, cx

4080 0000271E 74ED <1> jz short vga\_wca\_1

4081 00002720 3A15[C45E0000] <1> cmp dl, [CRT\_COLS]

4082 00002726 73E5 <1> jnb short vga\_wca\_1

4083 <1> ; write\_gfx\_char\_lin(car,attr,xcurs,ycurs,nbcols);

4084 <1> ; AL = car, BL = attr, DL = xcurs, DH = ycurs,

4085 <1> ; [CRT\_COLS] = nbcols

4086 00002728 E81E000000 <1> call write\_gfx\_char\_lin

4087 0000272D 6649 <1> dec cx ; count

4088 0000272F FEC2 <1> inc dl ; xcurs

4089 00002731 EBE8 <1> jmp short vga\_wca\_linear8

4090 <1> vga\_wca\_planar:

4091 <1> ; while((count-->0) && (xcurs<nbcols))

4092 <1> ; CX = count

4093 00002733 6621C9 <1> and cx, cx

4094 00002736 74D5 <1> jz short vga\_wca\_1

4095 00002738 3A15[C45E0000] <1> cmp dl, [CRT\_COLS]

4096 0000273E 73CD <1> jnb short vga\_wca\_1

4097 <1> ; write\_gfx\_char\_pl4(car,attr,xcurs,ycurs,nbcols,cheight);

4098 <1> ; AL = car, BL = attr, DL = xcurs, DH = ycurs,

4099 <1> ; [CRT\_COLS] = nbcols, [CHAR\_HEIGHT] = cheight

4100 00002740 E89D000000 <1> call write\_gfx\_char\_pl4

4101 00002745 6649 <1> dec cx ; count

4102 00002747 FEC2 <1> inc dl ; xcurs

4103 00002749 EBE8 <1> jmp short vga\_wca\_planar

4104 <1>

4105 <1> write\_gfx\_char\_lin:

4106 <1> ; 08/08/2016

4107 <1> ; 31/07/2016

4108 <1> ; 08/07/2016 (TRDOS 386 = TRDOS v2.0)

4109 <1> ;

4110 <1> ; derived from 'Plex86/Bochs VGABios' source code

4111 <1> ; vgabios-0.7a (2011)

4112 <1> ; by the LGPL VGABios developers Team (2001-2008)

4113 <1> ; 'vgabios.c', 'write\_gfx\_char\_lin'

4114 <1>

4115 <1> ; write\_gfx\_char\_lin(car,attr,xcurs,ycurs,nbcols)

4116 <1> ; INPUT ->

4117 <1> ; AL = car, BL = attr, DL = xcurs, DH = ycurs,

4118 <1> ; [CRT\_COLS] = nbcols

4119 <1> ; OUTPUT ->

4120 <1> ; Regen buffer updated

4121 <1>

4122 0000274B 51 <1> push ecx

4123 0000274C 53 <1> push ebx

4124 0000274D 52 <1> push edx

4125 0000274E 50 <1> push eax

4126 <1> ; addr=xcurs\*8+ycurs\*nbcols\*64;

4127 <1> ; 08/08/2016

4128 0000274F 0FB6F0 <1> movzx esi, al ; car

4129 00002752 0FB6C6 <1> movzx eax, dh ; ycurs

4130 00002755 8A25[C45E0000] <1> mov ah, [CRT\_COLS] ; nbcols

4131 0000275B F6E4 <1> mul ah

4132 <1> ;shl ax, 6 ; \* 64

4133 0000275D 66C1E003 <1> shl ax, 3 ; \* 8

4134 <1> ;sub dh, dh

4135 <1> ;shl dx, 3 ; xcurs \* 8

4136 <1> ;movzx edi, dx

4137 00002761 0FB6FA <1> movzx edi, dl

4138 00002764 66C1E703 <1> shl di, 3 ; xcurs \* 8

4139 00002768 30F6 <1> xor dh, dh

4140 0000276A 8A15[C65E0000] <1> mov dl, [CHAR\_HEIGHT]

4141 00002770 66F7E2 <1> mul dx

4142 <1> ; eax = ycurs\*nbcols\*8\*[CHAR\_HEIGHT]

4143 00002773 01C7 <1> add edi, eax ; addr

4144 00002775 81C700000A00 <1> add edi, 0A0000h

4145 <1> ;shl si, 3 ; car \* 8

4146 0000277B 30E4 <1> xor ah, ah

4147 0000277D A0[C65E0000] <1> mov al, [CHAR\_HEIGHT]

4148 00002782 66F7E6 <1> mul si

4149 00002785 6689C6 <1> mov si, ax

4150 <1> ;; esi = src = car \* 8

4151 <1> ; esi = src = car \* [CHAR\_HEIGHT]

4152 <1> ; i = 0

4153 <1> ;add esi, vgafont8 ; fdata [src+i]

4154 <1> ; 08/08/2016

4155 00002788 A1[E6650100] <1> mov eax, [VGA\_INT43H]

4156 0000278D 3D[B8420100] <1> cmp eax, vgafont16

4157 00002792 740F <1> je short wgfxl\_0

4158 00002794 3D[B8340100] <1> cmp eax, vgafont14

4159 00002799 7408 <1> je short wgfxl\_0

4160 0000279B 81C6[B82C0100] <1> add esi, vgafont8

4161 000027A1 EB02 <1> jmp short wgfxl\_1

4162 <1> wgfxl\_0:

4163 000027A3 01C6 <1> add esi, eax

4164 <1> wgfxl\_1:

4165 000027A5 28FF <1> sub bh, bh ; i = 0

4166 <1> wgfxl\_2:

4167 <1> ; for(i=0;i<8;i++)

4168 000027A7 57 <1> push edi ; addr

4169 000027A8 0FB605[C45E0000] <1> movzx eax, byte [CRT\_COLS] ; nbcols

4170 000027AF F6E7 <1> mul bh ; nbcols\*i

4171 000027B1 66C1E003 <1> shl ax, 3 ; i\*nbcols\*8

4172 <1> ; dest=addr+i\*nbcols\*8;

4173 000027B5 01C7 <1> add edi, eax ; dest + j ; j = 0

4174 000027B7 B180 <1> mov cl, 80h ; mask = 0x80;

4175 <1> ; esi = fdata + src + i

4176 <1> ; for(j=0;j<8;j++)

4177 000027B9 29D2 <1> sub edx, edx ; j = 0

4178 <1> wgfxl\_3:

4179 000027BB 8A06 <1> mov al, [esi] ; al = fdata[src+i]

4180 000027BD 20C8 <1> and al, cl ; if (fdata[src+i] & mask)

4181 000027BF 7402 <1> jz short wgfxl\_4 ; data = 0, zf = 1

4182 000027C1 88D8 <1> mov al, bl ; data = attr;

4183 <1> wgfxl\_4:

4184 <1> ; write\_byte(0xa000,dest+j,data);

4185 000027C3 AA <1> stosb ; dest + j (+ 0A0000h)

4186 <1> ;inc dl ; j++

4187 <1> ;cmp dl, 8

4188 000027C4 80FA07 <1> cmp dl, 7

4189 000027C7 720E <1> jb short wgfxl\_5

4190 000027C9 5F <1> pop edi

4191 <1> ; 08/08/2016

4192 <1> ;cmp bh, 7

4193 <1> ;jnb short wgfxl\_6

4194 000027CA FEC7 <1> inc bh ; i++

4195 000027CC 3A3D[C65E0000] <1> cmp bh, [CHAR\_HEIGHT]

4196 000027D2 7309 <1> jnb short wgfxl\_6

4197 000027D4 46 <1> inc esi

4198 000027D5 EBD0 <1> jmp short wgfxl\_2

4199 <1> wgfxl\_5:

4200 000027D7 D0E9 <1> shr cl, 1 ; mask >>= 1;

4201 000027D9 FEC2 <1> inc dl ; j++

4202 000027DB EBDE <1> jmp short wgfxl\_3

4203 <1> wgfxl\_6:

4204 000027DD 58 <1> pop eax

4205 000027DE 5A <1> pop edx

4206 000027DF 5B <1> pop ebx

4207 000027E0 59 <1> pop ecx

4208 000027E1 C3 <1> retn

4209 <1>

4210 <1> write\_gfx\_char\_pl4:

4211 <1> ; 08/08/2016

4212 <1> ; 08/07/2016 (TRDOS 386 = TRDOS v2.0)

4213 <1> ;

4214 <1> ; derived from 'Plex86/Bochs VGABios' source code

4215 <1> ; vgabios-0.7a (2011)

4216 <1> ; by the LGPL VGABios developers Team (2001-2008)

4217 <1> ; 'vgabios.c', 'write\_gfx\_char\_pl4'

4218 <1>

4219 <1> ; write\_gfx\_char\_pl4(car,attr,xcurs,ycurs,nbcols,cheight)

4220 <1> ; INPUT ->

4221 <1> ; AL = car, BL = attr, DL = xcurs, DH = ycurs,

4222 <1> ; [CRT\_COLS] = nbcols, [CHAR\_HEIGHT] = cheight

4223 <1> ; OUTPUT ->

4224 <1> ; Regen buffer updated

4225 <1>

4226 000027E2 51 <1> push ecx

4227 000027E3 53 <1> push ebx

4228 000027E4 52 <1> push edx

4229 000027E5 50 <1> push eax

4230 <1> wgfxpl\_f0:

4231 <1> ; switch(cheight)

4232 000027E6 8A25[C65E0000] <1> mov ah, [CHAR\_HEIGHT]

4233 000027EC 80FC10 <1> cmp ah, 16 ; case 16:

4234 000027EF 7507 <1> jne short wgfxpl\_f1

4235 <1> ; fdata = &vgafont16;

4236 000027F1 BE[B8420100] <1> mov esi, vgafont16

4237 000027F6 EB13 <1> jmp short wgfxpl\_f3

4238 <1> wgfxpl\_f1:

4239 000027F8 80FC0E <1> cmp ah, 14 ; case 14:

4240 000027FB 7507 <1> jne short wgfxpl\_f2

4241 000027FD BE[B8340100] <1> mov esi, vgafont14

4242 00002802 EB07 <1> jmp short wgfxpl\_f3

4243 <1> wgfxpl\_f2:

4244 <1> ; default:

4245 <1> ; fdata = &vgafont8;

4246 00002804 BE[B82C0100] <1> mov esi, vgafont8

4247 00002809 B408 <1> mov ah, 8

4248 <1> wgfxpl\_f3:

4249 <1> ; al = car

4250 0000280B F6E4 <1> mul ah ; ah = cheight

4251 0000280D 25FFFF0000 <1> and eax, 0FFFFh ; car \* cheight

4252 <1> ; src = car \* cheight;

4253 00002812 01C6 <1> add esi, eax ; esi = fdata[src+i]

4254 <1> ; addr=xcurs\*8+ycurs\*nbcols\*64;

4255 00002814 88F0 <1> mov al, dh ; ycurs

4256 00002816 8A25[C45E0000] <1> mov ah, [CRT\_COLS] ; nbcols

4257 0000281C F6E4 <1> mul ah

4258 <1> ; 08/08/2016

4259 <1> ;shl ax, 6 ; \* 64

4260 0000281E 66C1E003 <1> shl ax, 3 ; \* 8

4261 <1> ;sub dh, dh ; 0

4262 <1> ;shl dx, 3 ; xcurs \* 8

4263 <1> ;movzx edi, dx

4264 00002822 0FB6FA <1> movzx edi, dl

4265 00002825 66C1E703 <1> shl di, 3 ; xcurs \* 8

4266 00002829 30F6 <1> xor dh, dh

4267 0000282B 8A15[C65E0000] <1> mov dl, [CHAR\_HEIGHT]

4268 00002831 66F7E2 <1> mul dx

4269 <1> ; eax = ycurs\*nbcols\*8\*[CHAR\_HEIGHT]

4270 00002834 01C7 <1> add edi, eax ; addr

4271 00002836 81C700000A00 <1> add edi, 0A0000h

4272 <1> ;

4273 <1> ; outw(VGAREG\_SEQU\_ADDRESS, 0x0f02);

4274 <1> ; outw(VGAREG\_GRDC\_ADDRESS, 0x0205);

4275 0000283C 66BAC403 <1> mov dx, 3C4h ; VGAREG\_SEQU\_ADDRESS

4276 00002840 66B8020F <1> mov ax, 0F02h

4277 00002844 66EF <1> out dx, ax

4278 00002846 66BACE03 <1> mov dx, 3CEh ; VGAREG\_GRDC\_ADDRESS

4279 0000284A 66B80502 <1> mov ax, 0205h

4280 0000284E 66EF <1> out dx, ax

4281 <1> ;

4282 00002850 66BACE03 <1> mov dx, 3CEh ; VGAREG\_GRDC\_ADDRESS

4283 00002854 F6C380 <1> test bl, 80h ; if(attr&0x80)

4284 00002857 7406 <1> jz short wgfxpl\_f4 ; else

4285 <1> ; outw(VGAREG\_GRDC\_ADDRESS, 0x1803);

4286 00002859 66B80318 <1> mov ax, 1803h

4287 0000285D EB04 <1> jmp short wgfxpl\_f5

4288 <1> wgfxpl\_f4:

4289 <1> ; outw(VGAREG\_GRDC\_ADDRESS, 0x0003);

4290 0000285F 66B80300 <1> mov ax, 0003h

4291 <1> wgfxpl\_f5:

4292 00002863 66EF <1> out dx, ax

4293 <1> ;

4294 00002865 28FF <1> sub bh, bh ; i = 0

4295 <1> wgfxpl\_0:

4296 <1> ; for(i=0;i<cheight;i++)

4297 00002867 57 <1> push edi ; addr

4298 00002868 0FB605[C45E0000] <1> movzx eax, byte [CRT\_COLS] ; nbcols

4299 0000286F F6E7 <1> mul bh ; nbcols\*i

4300 <1> ; dest=addr+i\*nbcols

4301 00002871 01C7 <1> add edi, eax ; dest

4302 00002873 B580 <1> mov ch, 80h ; mask = 0x80;

4303 <1> ; for(j=0;j<8;j++)

4304 00002875 28C9 <1> sub cl, cl ; j = 0

4305 <1> wgfxpl\_1:

4306 00002877 D2ED <1> shr ch, cl ; mask=0x80>>j;

4307 <1> ;

4308 <1> ; outw(VGAREG\_GRDC\_ADDRESS, (mask << 8) | 0x08);

4309 <1> ; read\_byte(0xa000,dest);

4310 <1> ;mov dx, 3CEh ; VGAREG\_GRDC\_ADDRESS

4311 00002879 88EC <1> mov ah, ch

4312 0000287B B008 <1> mov al, 8

4313 0000287D 66EF <1> out dx, ax

4314 0000287F 8A07 <1> mov al, [edi] ; ? (io delay?)

4315 <1> ;

4316 00002881 28C0 <1> sub al, al ; attr = 0

4317 <1> ; if (fdata[src+i] & mask)

4318 00002883 842E <1> test byte [esi], ch

4319 00002885 7404 <1> jz short wgfxpl\_2 ; zf = 1

4320 <1> ; write\_byte(0xa000,dest,attr&0x0f);

4321 00002887 88D8 <1> mov al, bl ; attr;

4322 00002889 240F <1> and al, 0Fh ; attr&0x0f

4323 <1> wgfxpl\_2:

4324 <1> ; write\_byte(0xa000,dest,0x00);

4325 0000288B 8807 <1> mov [edi], al ; dest (+ 0A0000h)

4326 0000288D FEC1 <1> inc cl ; j++

4327 0000288F 80F908 <1> cmp cl, 8

4328 00002892 72E3 <1> jb short wgfxpl\_1

4329 00002894 5F <1> pop edi

4330 <1> ; 08/08/2016

4331 <1> ;cmp bh, 7

4332 <1> ;jnb short wgfxpl\_3

4333 00002895 FEC7 <1> inc bh ; i++

4334 00002897 3A3D[C65E0000] <1> cmp bh, [CHAR\_HEIGHT]

4335 0000289D 7303 <1> jnb short wgfxpl\_3

4336 0000289F 46 <1> inc esi

4337 000028A0 EBC5 <1> jmp short wgfxpl\_0

4338 <1> wgfxpl\_3:

4339 <1> ;mov dx, 3CEh ; VGAREG\_GRDC\_ADDRESS

4340 000028A2 66B808FF <1> mov ax, 0FF08h

4341 000028A6 66EF <1> out dx, ax

4342 000028A8 66B80500 <1> mov ax, 0005h

4343 000028AC 66EF <1> out dx, ax

4344 000028AE 66B80300 <1> mov ax, 0003h

4345 000028B2 66EF <1> out dx, ax

4346 <1> ;

4347 000028B4 58 <1> pop eax

4348 000028B5 5A <1> pop edx

4349 000028B6 5B <1> pop ebx

4350 000028B7 59 <1> pop ecx

4351 000028B8 C3 <1> retn

4352 <1>

4353 <1> vga\_write\_pixel:

4354 <1> ; 09/07/2016 (TRDOS 386 = TRDOS v2.0)

4355 <1> ;

4356 <1> ; derived from 'Plex86/Bochs VGABios' source code

4357 <1> ; vgabios-0.7a (2011)

4358 <1> ; by the LGPL VGABios developers Team (2001-2008)

4359 <1> ; 'vgabios.c', 'biosfn\_write\_pixel'

4360 <1>

4361 <1> ; INPUT ->

4362 <1> ; DX = row (0-239)

4363 <1> ; CX = column (0-799)

4364 <1> ; AL = pixel value

4365 <1> ; (AH = [CRT\_MODE])

4366 <1> ; OUTPUT ->

4367 <1> ; none

4368 <1>

4369 000028B9 88C3 <1> mov bl, al ; pixel value

4370 <1> ;mov ah, [CRT\_MODE]

4371 000028BB BE[DE5E0000] <1> mov esi, vga\_modes

4372 000028C0 89F7 <1> mov edi, esi

4373 000028C2 83C710 <1> add edi, vga\_mode\_count

4374 <1> vga\_wp\_0:

4375 000028C5 AC <1> lodsb

4376 000028C6 38E0 <1> cmp al, ah ; [CRT\_MODE]

4377 000028C8 7405 <1> je short vga\_wp\_1

4378 000028CA 39FE <1> cmp esi, edi

4379 000028CC 72F7 <1> jb short vga\_wp\_0

4380 000028CE C3 <1> retn ; nothing to do

4381 <1> vga\_wp\_1:

4382 000028CF 83C64F <1> add esi, vga\_memmodel - (vga\_modes + 1)

4383 <1> ; [ESI] = VGA memory model number (LINEAR8, PLANAR4, PLANAR1)

4384 000028D2 BF00000A00 <1> mov edi, 0A0000h

4385 <1> ;

4386 000028D7 803E04 <1> cmp byte [esi], PLANAR4

4387 000028DA 741D <1> je short vga\_wp\_planar

4388 000028DC 803E03 <1> cmp byte [esi], PLANAR1

4389 000028DF 7418 <1> je short vga\_wp\_planar

4390 <1> vga\_wp\_linear8:

4391 <1> ; addr=CX+DX\*(read\_word(BIOSMEM\_SEG,BIOSMEM\_NB\_COLS)\*8);

4392 000028E1 0FB605[C45E0000] <1> movzx eax, byte [CRT\_COLS] ; = [VGA\_COLS] ; nbcols

4393 000028E8 66C1E003 <1> shl ax, 3 ; \* 8

4394 000028EC 66F7E2 <1> mul dx

4395 000028EF 50 <1> push eax

4396 <1> ;mov edi, 0A0000h

4397 000028F0 6601CF <1> add di, cx

4398 000028F3 58 <1> pop eax

4399 000028F4 01C7 <1> add edi, eax ; addr

4400 <1> ; write\_byte(0xa000,addr,AL);

4401 000028F6 881F <1> mov [edi], bl

4402 000028F8 C3 <1> retn

4403 <1> vga\_wp\_planar:

4404 <1> ; addr = CX/8+DX\*read\_word(BIOSMEM\_SEG,BIOSMEM\_NB\_COLS);

4405 000028F9 0FB7C1 <1> movzx eax, cx

4406 000028FC 66C1E803 <1> shr ax, 3 ; CX/8

4407 00002900 50 <1> push eax

4408 00002901 28E4 <1> sub ah, ah ; 0

4409 00002903 A0[C45E0000] <1> mov al, [CRT\_COLS] ; = [VGA\_COLS] ; nbcols

4410 00002908 66F7E2 <1> mul dx

4411 <1> ;mov edi, 0A0000h

4412 0000290B 6601C7 <1> add di, ax

4413 0000290E 58 <1> pop eax

4414 0000290F 01C7 <1> add edi, eax ; addr

4415 00002911 80E107 <1> and cl, 7

4416 00002914 B580 <1> mov ch, 80h ; mask

4417 00002916 D2ED <1> shr ch, cl ; mask = 0x80 >> (CX & 0x07);

4418 <1>

4419 <1> ; outw(VGAREG\_GRDC\_ADDRESS, (mask << 8) | 0x08);

4420 00002918 66BACE03 <1> mov dx, 3CEh ; VGAREG\_GRDC\_ADDRESS

4421 0000291C 88EC <1> mov ah, ch

4422 0000291E B008 <1> mov al, 8

4423 00002920 66EF <1> out dx, ax

4424 <1> ; outw(VGAREG\_GRDC\_ADDRESS, 0x0205);

4425 00002922 66B80502 <1> mov ax, 0205h

4426 00002926 66EF <1> out dx, ax

4427 <1> ; data = read\_byte(0xa000,addr);

4428 00002928 8A07 <1> mov al, [edi] ; (delay?)

4429 <1> ; if (AL & 0x80)

4430 <1> ; {

4431 <1> ; outw(VGAREG\_GRDC\_ADDRESS, 0x1803);

4432 <1> ; }

4433 0000292A F6C380 <1> test bl, 80h

4434 0000292D 7406 <1> jz short vga\_wp\_2

4435 0000292F 66B80318 <1> mov ax, 1803h

4436 00002933 66EF <1> out dx, ax

4437 <1> vga\_wp\_2:

4438 <1> ; write\_byte(0xa000,addr,AL);

4439 00002935 881F <1> mov [edi], bl

4440 <1> ;

4441 <1> ;mov dx, 3CEh ; VGAREG\_GRDC\_ADDRESS

4442 00002937 66B808FF <1> mov ax, 0FF08h

4443 0000293B 66EF <1> out dx, ax

4444 0000293D 66B80500 <1> mov ax, 0005h

4445 00002941 66EF <1> out dx, ax

4446 00002943 66B80300 <1> mov ax, 0003h

4447 00002947 66EF <1> out dx, ax

4448 <1> ;

4449 00002949 C3 <1> retn

4450 <1>

4451 <1> vga\_read\_pixel:

4452 <1> ; 09/07/2016 (TRDOS 386 = TRDOS v2.0)

4453 <1> ;

4454 <1> ; derived from 'Plex86/Bochs VGABios' source code

4455 <1> ; vgabios-0.7a (2011)

4456 <1> ; by the LGPL VGABios developers Team (2001-2008)

4457 <1> ; 'vgabios.c', 'biosfn\_read\_pixel'

4458 <1>

4459 <1> ; INPUT ->

4460 <1> ; DX = row (0-239)

4461 <1> ; CX = column (0-799)

4462 <1> ; (AH = [CRT\_MODE])

4463 <1> ; OUTPUT ->

4464 <1> ; AL = pixel value

4465 <1>

4466 <1> ;mov ah, [CRT\_MODE]

4467 0000294A BE[DE5E0000] <1> mov esi, vga\_modes

4468 0000294F 89F7 <1> mov edi, esi

4469 00002951 83C710 <1> add edi, vga\_mode\_count

4470 <1> vga\_rp\_0:

4471 00002954 AC <1> lodsb

4472 00002955 38E0 <1> cmp al, ah ; [CRT\_MODE]

4473 00002957 7405 <1> je short vga\_rp\_1

4474 00002959 39FE <1> cmp esi, edi

4475 0000295B 72F7 <1> jb short vga\_rp\_0

4476 0000295D C3 <1> retn ; nothing to do

4477 <1> vga\_rp\_1:

4478 0000295E 83C64F <1> add esi, vga\_memmodel - (vga\_modes + 1)

4479 <1> ; [ESI] = VGA memory model number (LINEAR8, PLANAR4, PLANAR1)

4480 00002961 BF00000A00 <1> mov edi, 0A0000h

4481 <1> ;

4482 00002966 803E04 <1> cmp byte [esi], PLANAR4

4483 00002969 741D <1> je short vga\_rp\_planar

4484 0000296B 803E03 <1> cmp byte [esi], PLANAR1

4485 0000296E 7418 <1> je short vga\_rp\_planar

4486 <1> vga\_rp\_linear8:

4487 <1> ; addr=CX+DX\*(read\_word(BIOSMEM\_SEG,BIOSMEM\_NB\_COLS)\*8);

4488 00002970 0FB605[C45E0000] <1> movzx eax, byte [CRT\_COLS] ; = [VGA\_COLS] ; nbcols

4489 00002977 66C1E003 <1> shl ax, 3 ; \* 8

4490 0000297B 66F7E2 <1> mul dx

4491 0000297E 50 <1> push eax

4492 <1> ;mov edi, 0A0000h

4493 0000297F 6601CF <1> add di, cx

4494 00002982 58 <1> pop eax

4495 00002983 01C7 <1> add edi, eax ; addr

4496 <1> ; attr=read\_byte(0xa000,addr);

4497 00002985 8A07 <1> mov al, [edi] ; pixel value

4498 00002987 C3 <1> retn

4499 <1> vga\_rp\_planar:

4500 <1> ; addr = CX/8+DX\*read\_word(BIOSMEM\_SEG,BIOSMEM\_NB\_COLS);

4501 00002988 0FB7C1 <1> movzx eax, cx

4502 0000298B 66C1E803 <1> shr ax, 3 ; CX/8

4503 0000298F 50 <1> push eax

4504 00002990 28E4 <1> sub ah, ah ; 0

4505 00002992 A0[C45E0000] <1> mov al, [CRT\_COLS] ; = [VGA\_COLS] ; nbcols

4506 00002997 66F7E2 <1> mul dx

4507 <1> ;mov edi, 0A0000h

4508 0000299A 6601C7 <1> add di, ax

4509 0000299D 58 <1> pop eax

4510 0000299E 01C7 <1> add edi, eax ; addr

4511 000029A0 80E107 <1> and cl, 7

4512 000029A3 B580 <1> mov ch, 80h ; mask

4513 000029A5 D2ED <1> shr ch, cl ; mask = 0x80 >> (CX & 0x07);

4514 <1> ; attr = 0x00;

4515 000029A7 30DB <1> xor bl, bl ; attr = bl = 0,

4516 000029A9 30C9 <1> xor cl, cl ; i = cl = 0

4517 <1> ; for(i=0;i<4;i++)

4518 <1> ; {

4519 <1> ; outw(VGAREG\_GRDC\_ADDRESS, (i << 8) | 0x04);

4520 <1> ; data = read\_byte(0xa000,addr) & mask;

4521 <1> ; if (data > 0) attr |= (0x01 << i);

4522 <1> ; }

4523 <1> vga\_rp\_2:

4524 000029AB 88CC <1> mov ah, cl ; i << 8

4525 000029AD B004 <1> mov al, 4 ; | 0x04

4526 000029AF 66BACE03 <1> mov dx, 3CEh ; VGAREG\_GRDC\_ADDRESS

4527 000029B3 66EF <1> out dx, ax

4528 <1> ; data = read\_byte(0xa000,addr) & mask;

4529 000029B5 8A07 <1> mov al, [edi]

4530 000029B7 20E8 <1> and al, ch ; & mask

4531 <1> ; if (data > 0) attr |= (0x01 << i);

4532 000029B9 08C0 <1> or al, al

4533 000029BB 7408 <1> jz short vga\_rp\_3 ; al = 0

4534 000029BD B701 <1> mov bh, 1

4535 000029BF D2E7 <1> shl bh, cl ; (0x01 << i)

4536 000029C1 08FB <1> or bl, bh ; attr |= (0x01 << i)

4537 000029C3 88D8 <1> mov al, bl ; pixel value

4538 <1> vga\_rp\_3:

4539 000029C5 C3 <1> retn

4540 <1>

4541 <1> vga\_beeper:

4542 <1> ; 04/08/2016 (TRDOS 386 = TRDOS v2.0)

4543 000029C6 FB <1> sti

4544 <1> ;mov bh, [ACTIVE\_PAGE]

4545 000029C7 E9CFF3FFFF <1> jmp beeper\_gfx

4546 <1>

4547 <1> vga\_write\_teletype:

4548 <1> ; 09/12/2017

4549 <1> ; 06/08/2016

4550 <1> ; 04/08/2016

4551 <1> ; 01/08/2016

4552 <1> ; 31/07/2016

4553 <1> ; 09/07/2016 (TRDOS 386 = TRDOS v2.0)

4554 <1> ;

4555 <1> ; derived from 'Plex86/Bochs VGABios' source code

4556 <1> ; vgabios-0.7a (2011)

4557 <1> ; by the LGPL VGABios developers Team (2001-2008)

4558 <1> ; 'vgabios.c', 'biosfn\_write\_teletype'

4559 <1> ; 'biosfn\_write\_char\_only'

4560 <1>

4561 <1> ; INPUT ->

4562 <1> ; [CRT\_MODE] = current video mode (>7)

4563 <1> ; AL = Character to write

4564 <1> ; BL = Color of character

4565 <1> ; OUTPUT ->

4566 <1> ; Regen buffer updated

4567 <1>

4568 <1> ; biosfn\_write\_teletype (car, page, attr, flag)

4569 <1> ; car = character (AL)

4570 <1> ; page = 0

4571 <1> ; attr = color (BL)

4572 <1> ; 'flag' not used

4573 <1>

4574 000029CC 8A25[C25E0000] <1> mov ah, [CRT\_MODE]

4575 000029D2 88C7 <1> mov bh, al ; character

4576 000029D4 668B15[56580100] <1> mov dx, [CURSOR\_POSN] ; cursor pos for page 0

4577 <1>

4578 000029DB BE[E65E0000] <1> mov esi, vga\_g\_modes

4579 000029E0 89F7 <1> mov edi, esi

4580 000029E2 83C708 <1> add edi, vga\_g\_mode\_count

4581 <1> vga\_wtty\_0:

4582 000029E5 AC <1> lodsb

4583 000029E6 38E0 <1> cmp al, ah ; [CRT\_MODE]

4584 000029E8 7405 <1> je short vga\_wtty\_2

4585 000029EA 39FE <1> cmp esi, edi

4586 000029EC 72F7 <1> jb short vga\_wtty\_0

4587 <1> vga\_wtty\_1:

4588 000029EE C3 <1> retn ; nothing to do

4589 <1> vga\_wtty\_2:

4590 000029EF 80FF07 <1> cmp bh, 07h ; bell (beep)

4591 000029F2 74D2 <1> je short vga\_beeper ; u11

4592 000029F4 80FF08 <1> cmp bh, 08h ; backspace

4593 000029F7 7508 <1> jne short vga\_wtty\_3

4594 <1> ; if(xcurs>0)xcurs--;

4595 000029F9 08D2 <1> or dl, dl ; xcurs (column)

4596 000029FB 74F1 <1> jz short vga\_wtty\_1

4597 000029FD FECA <1> dec dl ; xcurs--;

4598 000029FF EB59 <1> jmp short vga\_wtty\_12

4599 <1> vga\_wtty\_3:

4600 00002A01 80FF0D <1> cmp bh, 0Dh ; carriage return (\r)

4601 00002A04 7504 <1> jne short vga\_wtty\_4

4602 <1> ; xcurs=0;

4603 00002A06 28D2 <1> sub dl, dl ; 0

4604 00002A08 EB50 <1> jmp short vga\_wtty\_12

4605 <1> vga\_wtty\_4:

4606 00002A0A 80FF0A <1> cmp bh, 0Ah ; new line (\n)

4607 00002A0D 7504 <1> jne short vga\_wtty\_5

4608 <1> ; ycurs++;

4609 00002A0F FEC6 <1> inc dh ; next row

4610 00002A11 EB62 <1> jmp short vga\_wtty\_11

4611 <1> vga\_wtty\_5:

4612 00002A13 80FF09 <1> cmp bh, 09h ; tab stop

4613 00002A16 7527 <1> jne short vga\_wtty\_8

4614 00002A18 88D0 <1> mov al, dl

4615 <1> ;cbw

4616 00002A1A 30E4 <1> xor ah, ah ; 09/12/2017

4617 00002A1C B108 <1> mov cl, 8

4618 00002A1E F6F1 <1> div cl

4619 00002A20 28E1 <1> sub cl, ah

4620 <1> ;

4621 00002A22 B720 <1> mov bh, 20h ; space

4622 <1> vga\_wtty\_6: ; tab stop loop

4623 00002A24 6651 <1> push cx

4624 00002A26 6653 <1> push bx

4625 00002A28 E812000000 <1> call vga\_wtty\_8

4626 00002A2D 665B <1> pop bx ; bh = character, bl = color

4627 00002A2F 6659 <1> pop cx

4628 00002A31 FEC9 <1> dec cl

4629 00002A33 7409 <1> jz short vga\_wtty\_7

4630 00002A35 668B15[56580100] <1> mov dx, [CURSOR\_POSN] ; new cursor position (pg 0)

4631 00002A3C EBE6 <1> jmp short vga\_wtty\_6

4632 <1> vga\_wtty\_7:

4633 00002A3E C3 <1> retn

4634 <1> ;

4635 <1> vga\_wtty\_8:

4636 00002A3F 83C64F <1> add esi, vga\_g\_memmodel - (vga\_g\_modes + 1)

4637 <1> ; [ESI] = VGA memory model number (LINEAR8, PLANAR4, PLANAR1)

4638 00002A42 BF00000A00 <1> mov edi, 0A0000h

4639 <1> ;

4640 00002A47 88F8 <1> mov al, bh ; character

4641 <1> ;

4642 00002A49 803E04 <1> cmp byte [esi], PLANAR4

4643 00002A4C 7414 <1> je short vga\_wtty\_planar

4644 00002A4E 803E03 <1> cmp byte [esi], PLANAR1

4645 00002A51 740F <1> je short vga\_wtty\_planar

4646 <1> vga\_wtty\_linear8:

4647 <1> ; write\_gfx\_char\_lin(car,attr,xcurs,ycurs,nbcols);

4648 <1> ; AL = car, BL = attr (color), DL = xcurs, DH = ycurs,

4649 <1> ; [CRT\_COLS] = nbcols

4650 00002A53 E8F3FCFFFF <1> call write\_gfx\_char\_lin

4651 00002A58 EB0D <1> jmp short vga\_wtty\_9

4652 <1>

4653 <1> vga\_wtty\_12:

4654 <1> ; 09/07/2016

4655 <1> ; set cursor position

4656 <1> ; NOTE: Hardware cursor position will not be set

4657 <1> ; in any VGA modes (>7)

4658 <1> ; But, cursor position will be saved into

4659 <1> ; [CURSOR\_POSN].

4660 <1> ; TRDOS 386 (TRDOS v2.0) uses only one page

4661 <1> ; (page 0) for all graphics modes.

4662 <1>

4663 00002A5A 668915[56580100] <1> mov [CURSOR\_POSN], dx ; save cursor pos for pg 0

4664 <1> ; 04/08/2016

4665 <1> ;mov bh, [ACTIVE\_PAGE] ; = 0

4666 <1> ;call \_set\_cpos

4667 00002A61 C3 <1> retn

4668 <1>

4669 <1> vga\_wtty\_planar:

4670 <1> ; write\_gfx\_char\_pl4(car,attr,xcurs,ycurs,nbcols,cheight);

4671 <1> ; AL = car, BL = attr (color), DL = xcurs, DH = ycurs,

4672 <1> ; [CRT\_COLS]= nbcols, [CHAR\_HEIGHT] = cheight

4673 00002A62 E87BFDFFFF <1> call write\_gfx\_char\_pl4

4674 <1> vga\_wtty\_9:

4675 00002A67 FEC2 <1> inc dl ; xcurs++;

4676 <1> vga\_wtty\_10:

4677 <1> ; Do we need to wrap ?

4678 <1> ; if(xcurs==nbcols)

4679 00002A69 3A15[C45E0000] <1> cmp dl, [CRT\_COLS] ; [VGA\_COLS]

4680 00002A6F 7204 <1> jb short vga\_wtty\_11 ; no

4681 00002A71 28D2 <1> sub dl, dl ; xcurs=0;

4682 00002A73 FEC6 <1> inc dh ; ycurs++;

4683 <1> vga\_wtty\_11:

4684 <1> ; Do we need to scroll ?

4685 <1> ; if(ycurs==nbrows)

4686 00002A75 3A35[CA5E0000] <1> cmp dh, [VGA\_ROWS]

4687 00002A7B 72DD <1> jb short vga\_wtty\_12 ; no

4688 <1> ;

4689 <1> ; biosfn\_scroll (nblines,attr,rul,cul,rlr,clr,page,dir)

4690 <1> ; al = nblines = 1, bl = attr (color) = 0

4691 <1> ; ch = rul, cl = cul, dh = rlr, dl = clr, page = 0

4692 <1> ; dir = SCROLL\_UP

4693 <1>

4694 00002A7D B001 <1> mov al, 1

4695 00002A7F 28DB <1> sub bl, bl ; 0 ; blank/black line (attr=0) will be used

4696 00002A81 6629C9 <1> sub cx, cx ; 0,0

4697 <1>

4698 <1> ; 06/08/2016

4699 00002A84 8A35[CA5E0000] <1> mov dh, [VGA\_ROWS]

4700 00002A8A FECE <1> dec dh ; nbrows -1

4701 <1>

4702 00002A8C 6652 <1> push dx ; 04/08/2016

4703 00002A8E 8A15[C45E0000] <1> mov dl, [CRT\_COLS]

4704 00002A94 FECA <1> dec dl ; nbcols -1

4705 <1>

4706 00002A96 8A25[C25E0000] <1> mov ah, [CRT\_MODE]

4707 <1>

4708 <1> ; biosfn\_scroll(0x01,0x00,0,0,nbrows-1,nbcols-1,page,SCROLL\_UP);

4709 00002A9C E808F5FFFF <1> call vga\_graphics\_up

4710 <1> ; 04/08/2016

4711 00002AA1 665A <1> pop dx

4712 <1> ;dec dh ; ycurs-=1

4713 00002AA3 EBB5 <1> jmp short vga\_wtty\_12

4714 <1>

4715 <1> font\_setup:

4716 <1> ; 09/07/2016

4717 <1> ; character generator (font loading) functions

4718 <1> ;

4719 <1> ; derived from 'Plex86/Bochs VGABios' source code

4720 <1> ; vgabios-0.7a (2011)

4721 <1> ; by the LGPL VGABios developers Team (2001-2008)

4722 <1> ; 'vgabios.c', 'int10\_func'

4723 <1>

4724 <1> ; AX = 1100H ; Load User-Defined Font (EGA/VGA)

4725 <1> ;

4726 <1> ; BH height of each character (bytes per character definition)

4727 <1> ; (BL font block to load (EGA: 0-3; VGA: 0-7))

4728 <1> ; CX number of characters to redefine (<=256)

4729 <1> ; DX ASCII code of the first character defined at ES:BP

4730 <1> ; EBP address of font-definition information

4731 <1> ; (in user's memory space)

4732 <1>

4733 <1> ; case 0x11:

4734 <1> ; switch(GET\_AL())

4735 <1> ; {

4736 <1> ; case 0x00:

4737 <1> ; case 0x10:

4738 <1> ; biosfn\_load\_text\_user\_pat(GET\_AL(),ES,BP,CX,DX,GET\_BL(),GET\_BH());

4739 <1> ; break;

4740 <1>

4741 <1> ; AX = 1110H ; Load and Activate User-Defined Font (EGA/VGA)

4742 00002AA5 08C0 <1> or al, al ; 0

4743 00002AA7 7404 <1> jz short font\_setup\_0

4744 00002AA9 3C10 <1> cmp al, 10h

4745 00002AAB 7511 <1> jne short font\_setup\_1

4746 <1> font\_setup\_0:

4747 00002AAD E8B7000000 <1> call transfer\_user\_fonts

4748 00002AB2 721C <1> jc short font\_setup\_error

4749 00002AB4 E8C2000000 <1> call load\_text\_user\_pat

4750 00002AB9 E996EAFFFF <1> jmp VIDEO\_RETURN

4751 <1> font\_setup\_1:

4752 <1> ; AX = 1101H ; Load ROM 8x14 Character Set (EGA/VGA)

4753 <1> ; case 0x01:

4754 <1> ; case 0x11:

4755 <1> ; biosfn\_load\_text\_8\_14\_pat(GET\_AL(),GET\_BL());

4756 <1> ; break;

4757 00002ABE 3C01 <1> cmp al, 1

4758 00002AC0 7404 <1> je short font\_setup\_2

4759 00002AC2 3C11 <1> cmp al, 11h

4760 00002AC4 7511 <1> jne short font\_setup\_3

4761 <1> font\_setup\_2:

4762 <1> ; AX = 1111H ; Load and Activate ROM 8x14 Character Set (EGA/VGA)

4763 <1> ; (BL = font block to load (EGA: 0-3; VGA: 0-7))

4764 00002AC6 E8EE010000 <1> call load\_text\_8\_14\_pat

4765 00002ACB E984EAFFFF <1> jmp VIDEO\_RETURN

4766 <1> font\_setup\_error:

4767 00002AD0 29C0 <1> sub eax, eax ; 0 -> fonts could not be loaded

4768 00002AD2 E982EAFFFF <1> jmp \_video\_return

4769 <1> font\_setup\_3:

4770 <1> ; AX = 1102H ; Load ROM 8x8 Character Set (EGA/VGA)

4771 <1> ; case 0x02:

4772 <1> ; case 0x12:

4773 <1> ; biosfn\_load\_text\_8\_8\_pat(GET\_AL(),GET\_BL());

4774 <1> ; break;

4775 00002AD7 3C02 <1> cmp al, 2

4776 00002AD9 7404 <1> je short font\_setup\_4

4777 00002ADB 3C12 <1> cmp al, 12h

4778 00002ADD 750A <1> jne short font\_setup\_5

4779 <1> font\_setup\_4:

4780 <1> ; AX = 1112H ; Load and Activate ROM 8x8 Character Set (EGA/VGA)

4781 <1> ; (BL = font block to load (EGA: 0-3; VGA: 0-7))

4782 00002ADF E805020000 <1> call load\_text\_8\_8\_pat

4783 00002AE4 E96BEAFFFF <1> jmp VIDEO\_RETURN

4784 <1> font\_setup\_5:

4785 <1> ; AX = 1104H ; Load ROM 8x16 Character Set (EGA/VGA)

4786 <1> ; case 0x04:

4787 <1> ; case 0x14:

4788 <1> ; biosfn\_load\_text\_8\_16\_pat(GET\_AL(),GET\_BL());

4789 <1> ; break;

4790 00002AE9 3C04 <1> cmp al, 4

4791 00002AEB 7404 <1> je short font\_setup\_6

4792 00002AED 3C14 <1> cmp al, 14h

4793 00002AEF 750A <1> jne short font\_setup\_7

4794 <1> font\_setup\_6:

4795 <1> ; AX = 1114H ; Load and Activate ROM 8x16 Character Set (EGA/VGA)

4796 <1> ; (BL = font block to load (EGA: 0-3; VGA: 0-7))

4797 00002AF1 E823020000 <1> call load\_text\_8\_16\_pat

4798 00002AF6 E959EAFFFF <1> jmp VIDEO\_RETURN

4799 <1> font\_setup\_7:

4800 <1> ; Note: AX=1120h (Setup INT 1Fh, EXT\_PTR) is not needed

4801 <1> ; for TRDOS 386 (TRDIOS v2.0) video functionality;

4802 <1> ; because, originally EXT\_PTR (font address) was used for

4803 <1> ; chars 80h to 0FFh (after the first 128 ASCII char fonts), for

4804 <1> ; CGA graphics mode; currenty, 'vgafont8' address has 256 chars!

4805 <1> ;

4806 <1> ; case 0x20:

4807 <1> ; biosfn\_load\_gfx\_8\_8\_chars(ES,BP);

4808 <1> ; break;

4809 <1> ; case 0x21:

4810 <1> ; biosfn\_load\_gfx\_user\_chars(ES,BP,CX,GET\_BL(),GET\_DL());

4811 <1> ; break;

4812 <1> ; AX = 1121H ; Setup User-Defined Font for Graphics Mode (VGA)

4813 <1> ; BL screen rows code: 00H = user-specified (in DL)

4814 <1> ; 01H = 14 rows

4815 <1> ; 02H = 25 rows

4816 <1> ; 03H = 43 rows

4817 <1> ; CX bytes per character definition

4818 <1> ; DL (when BL=0) custom number of character rows on screen

4819 <1> ; EBP address of font-definition information (user's mem space)

4820 <1>

4821 00002AFB 3C21 <1> cmp al, 21h

4822 00002AFD 751A <1> jne short font\_setup\_9

4823 <1>

4824 <1> ; TRDOS 386 modification !

4825 <1> ; dh = 0 -> 256 characters

4826 <1> ; dh = 80h -> 128 characters

4827 <1> ; (If DH <> 0 and DH <> 80h -> invalid)

4828 00002AFF 20F6 <1> and dh, dh

4829 00002B01 7405 <1> jz short font\_setup\_8 ; 256 characters

4830 00002B03 80FE80 <1> cmp dh, 80h ; 128 characters

4831 00002B06 75C8 <1> jne short font\_setup\_error ; invalid !

4832 <1> font\_setup\_8:

4833 00002B08 E85C000000 <1> call transfer\_user\_fonts

4834 00002B0D 72C1 <1> jc short font\_setup\_error

4835 <1> ; ebp = user's font data address in system's memory space

4836 00002B0F E836020000 <1> call load\_gfx\_user\_chars

4837 00002B14 E93BEAFFFF <1> jmp VIDEO\_RETURN

4838 <1> font\_setup\_9:

4839 <1> ; case 0x22:

4840 <1> ; biosfn\_load\_gfx\_8\_14\_chars(GET\_BL());

4841 <1> ; break;

4842 00002B19 3C22 <1> cmp al, 22h

4843 00002B1B 750A <1> jne short font\_setup\_10

4844 00002B1D E866020000 <1> call load\_gfx\_8\_14\_chars

4845 00002B22 E92DEAFFFF <1> jmp VIDEO\_RETURN

4846 <1> font\_setup\_10:

4847 <1> ; case 0x23:

4848 <1> ; biosfn\_load\_gfx\_8\_8\_dd\_chars(GET\_BL());

4849 <1> ; break;

4850 00002B27 3C23 <1> cmp al, 23h

4851 00002B29 750A <1> jne short font\_setup\_11

4852 00002B2B E899020000 <1> call load\_gfx\_8\_8\_chars

4853 00002B30 E91FEAFFFF <1> jmp VIDEO\_RETURN

4854 <1> font\_setup\_11:

4855 <1> ; case 0x24:

4856 <1> ; biosfn\_load\_gfx\_8\_16\_chars(GET\_BL());

4857 <1> ; break;

4858 00002B35 3C24 <1> cmp al, 24h

4859 00002B37 750A <1> jne short font\_setup\_12

4860 00002B39 E8CC020000 <1> call load\_gfx\_8\_16\_chars

4861 00002B3E E911EAFFFF <1> jmp VIDEO\_RETURN

4862 <1> font\_setup\_12:

4863 <1> ; case 0x30:

4864 <1> ; biosfn\_get\_font\_info(GET\_BH(),&ES,&BP,&CX,&DX);

4865 <1> ; break;

4866 00002B43 3C30 <1> cmp al, 30h

4867 00002B45 750A <1> jne short font\_setup\_13

4868 00002B47 E8FF020000 <1> call get\_font\_info

4869 <1> ; eax = return value (info: 4 bytes for 4 parms)

4870 <1> ; eax = 0 -> invalid function (input)

4871 00002B4C E908EAFFFF <1> jmp \_video\_return

4872 <1> font\_setup\_13:

4873 00002B51 3C03 <1> cmp al, 03h ; AX = 1103h

4874 00002B53 750D <1> jne short font\_setup\_14

4875 <1> ; biosfn\_set\_text\_block\_specifier:

4876 <1> ; BL = font block selector code

4877 <1> ; NOTE: TRDOS 386 only uses and sets font block 0

4878 <1> ; (It is as BL = 0 for TRDOS 386)

4879 00002B55 66BAC403 <1> mov dx, 3C4h ; VGAREG\_SEQU\_ADDRESS

4880 <1> ;mov ah, bl

4881 00002B59 28E4 <1> sub ah, ah ; 0

4882 <1> ;mov al, 03h

4883 00002B5B 66EF <1> out dx, ax

4884 00002B5D E9F2E9FFFF <1> jmp VIDEO\_RETURN

4885 <1>

4886 <1> font\_setup\_14:

4887 00002B62 29C0 <1> sub eax, eax ; 0 = invalid function

4888 00002B64 E9F0E9FFFF <1> jmp \_video\_return

4889 <1>

4890 <1> transfer\_user\_fonts:

4891 <1> ; 09/07/2016

4892 <1> ;and ecx, 0FFFFh

4893 <1> ; ECX = byte count

4894 <1> ;push ecx

4895 00002B69 89EE <1> mov esi, ebp ; user buffer

4896 00002B6B BF00000700 <1> mov edi, Cluster\_Buffer ; system buffer

4897 00002B70 E84EBC0000 <1> call transfer\_from\_user\_buffer

4898 <1> ;pop ecx

4899 <1> ; ecx = transfer (byte) count = character count

4900 00002B75 BD00000700 <1> mov ebp, Cluster\_Buffer

4901 <1> ; jc VIDEO\_RETURN -> failed

4902 00002B7A C3 <1> retn

4903 <1>

4904 <1> load\_text\_user\_pat:

4905 <1> ; 26/07/2016

4906 <1> ; 09/07/2016

4907 <1> ; load user defined (EGA/VGA) text fonts

4908 <1> ;

4909 <1> ; derived from 'Plex86/Bochs VGABios' source code

4910 <1> ; vgabios-0.7a (2011)

4911 <1> ; by the LGPL VGABios developers Team (2001-2008)

4912 <1> ; 'vgabios.c', 'biosfn\_load\_text\_user\_pat'

4913 <1>

4914 <1> ; biosfn\_load\_text\_user\_pat (AL,ES,BP,CX,DX,BL,BH)

4915 <1>

4916 <1> ; get\_font\_access();

4917 <1> ; blockaddr = ((BL & 0x03) << 14) + ((BL & 0x04) << 11);

4918 <1> ; for(i=0;i<CX;i++)

4919 <1> ; {

4920 <1> ; src = BP + i \* BH;

4921 <1> ; dest = blockaddr + (DX + i) \* 32;

4922 <1> ; memcpyb(0xA000, dest, ES, src, BH);

4923 <1> ; }

4924 <1> ; release\_font\_access();

4925 <1> ; if(AL>=0x10)

4926 <1> ; {

4927 <1> ; set\_scan\_lines(BH);

4928 <1> ; }

4929 <1>

4930 00002B7B 50 <1> push eax

4931 00002B7C E83C000000 <1> call get\_font\_access

4932 00002B81 28DB <1> sub bl, bl ; i = 0

4933 <1> ltup\_1:

4934 00002B83 88D8 <1> mov al, bl

4935 00002B85 F6E7 <1> mul bh

4936 00002B87 0FB7F0 <1> movzx esi, ax

4937 00002B8A 01EE <1> add esi, ebp

4938 00002B8C 88D8 <1> mov al, bl

4939 00002B8E 28E4 <1> sub ah, ah

4940 00002B90 6601D0 <1> add ax, dx ; (DX + i)

4941 00002B93 66C1E005 <1> shl ax, 5 ; \* 32

4942 00002B97 0FB7F8 <1> movzx edi, ax

4943 00002B9A 81C700000A00 <1> add edi, 0A0000h

4944 00002BA0 51 <1> push ecx

4945 00002BA1 0FB6CF <1> movzx ecx, bh

4946 00002BA4 F3A4 <1> rep movsb

4947 00002BA6 59 <1> pop ecx

4948 00002BA7 FEC3 <1> inc bl

4949 00002BA9 38CB <1> cmp bl, cl

4950 00002BAB 75D6 <1> jne short ltup\_1

4951 <1> ;

4952 00002BAD E840000000 <1> call release\_font\_access

4953 <1> ;

4954 00002BB2 58 <1> pop eax

4955 <1> ; if(AL>=0x10)

4956 00002BB3 3C10 <1> cmp al, 10h

4957 00002BB5 7205 <1> jb short ltup\_2

4958 <1> ; set\_scan\_lines(BH);

4959 00002BB7 E875000000 <1> call set\_scan\_lines

4960 <1> ltup\_2:

4961 00002BBC C3 <1> retn

4962 <1>

4963 <1> get\_font\_access:

4964 <1> ; 09/07/2016

4965 <1> ;

4966 <1> ; derived from 'Plex86/Bochs VGABios' source code

4967 <1> ; vgabios-0.7a (2011)

4968 <1> ; by the LGPL VGABios developers Team (2001-2008)

4969 <1> ; 'vgabios.c', 'get\_font\_access'

4970 <1>

4971 <1> ; get\_font\_access()

4972 00002BBD 52 <1> push edx

4973 00002BBE 66BAC403 <1> mov dx, 3C4h ; VGAREG\_SEQU\_ADDRESS

4974 00002BC2 66B80001 <1> mov ax, 0100h

4975 00002BC6 66EF <1> out dx, ax

4976 00002BC8 66B80204 <1> mov ax, 0402h

4977 00002BCC 66EF <1> out dx, ax

4978 00002BCE 66B80407 <1> mov ax, 0704h

4979 00002BD2 66EF <1> out dx, ax

4980 00002BD4 66B80003 <1> mov ax, 0300h

4981 00002BD8 66EF <1> out dx, ax

4982 00002BDA 66BACE03 <1> mov dx, 3CEh ; VGAREG\_GRDC\_ADDRESS

4983 00002BDE 66B80402 <1> mov ax, 0204h

4984 00002BE2 66EF <1> out dx, ax

4985 00002BE4 66B80500 <1> mov ax, 0005h

4986 00002BE8 66EF <1> out dx, ax

4987 00002BEA 66B80604 <1> mov ax, 0406h

4988 00002BEE 66EF <1> out dx, ax

4989 00002BF0 5A <1> pop edx

4990 00002BF1 C3 <1> retn

4991 <1>

4992 <1> release\_font\_access:

4993 <1> ; 29/07/2016

4994 <1> ; 09/07/2016

4995 <1> ;

4996 <1> ; derived from 'Plex86/Bochs VGABios' source code

4997 <1> ; vgabios-0.7a (2011)

4998 <1> ; by the LGPL VGABios developers Team (2001-2008)

4999 <1> ; 'vgabios.c', 'release\_font\_access'

5000 <1>

5001 00002BF2 66BAC403 <1> mov dx, 3C4h ; VGAREG\_SEQU\_ADDRESS

5002 00002BF6 66B80001 <1> mov ax, 0100h

5003 00002BFA 66EF <1> out dx, ax

5004 00002BFC 66B80203 <1> mov ax, 0302h

5005 00002C00 66EF <1> out dx, ax

5006 00002C02 66B80403 <1> mov ax, 0304h

5007 00002C06 66EF <1> out dx, ax

5008 00002C08 66B80003 <1> mov ax, 0300h

5009 00002C0C 66EF <1> out dx, ax

5010 00002C0E 66BACC03 <1> mov dx, 3CCh ; VGAREG\_READ\_MISC\_OUTPUT

5011 00002C12 EC <1> in al, dx

5012 00002C13 2401 <1> and al, 01h

5013 00002C15 C0E002 <1> shl al, 2

5014 00002C18 0C0A <1> or al, 0Ah

5015 00002C1A 88C4 <1> mov ah, al

5016 00002C1C B006 <1> mov al, 06h

5017 00002C1E 66BACE03 <1> mov dx, 3CEh ; VGAREG\_GRDC\_ADDRESS

5018 00002C22 66EF <1> out dx, ax

5019 00002C24 66B80400 <1> mov ax, 0004h

5020 00002C28 66EF <1> out dx, ax

5021 00002C2A 66B80510 <1> mov ax, 1005h

5022 00002C2E 66EF <1> out dx, ax

5023 00002C30 C3 <1> retn

5024 <1>

5025 <1> set\_scan\_lines:

5026 <1> ; 09/07/2016

5027 <1> ;

5028 <1> ; derived from 'Plex86/Bochs VGABios' source code

5029 <1> ; vgabios-0.7a (2011)

5030 <1> ; by the LGPL VGABios developers Team (2001-2008)

5031 <1> ; 'vgabios.c', 'set\_scan\_lines'

5032 <1>

5033 <1> ; set\_scan\_lines(lines)

5034 <1> ; BH = lines

5035 <1>

5036 <1> ; outb(crtc\_addr, 0x09);

5037 00002C31 66BAD403 <1> mov dx, 3D4h ; CRTC\_ADDRESS = 3D4h (always)

5038 00002C35 B009 <1> mov al, 09h

5039 00002C37 EE <1> out dx, al

5040 <1> ; crtc\_r9 = inb(crtc\_addr+1);

5041 00002C38 6642 <1> inc dx ; 3D5h

5042 00002C3A EC <1> in al, dx

5043 <1> ; crtc\_r9 = (crtc\_r9 & 0xe0) | (lines - 1);

5044 00002C3B 24E0 <1> and al, 0E0h

5045 00002C3D FECF <1> dec bh ; lines - 1

5046 00002C3F 08F8 <1> or al, bh

5047 <1> ; outb(crtc\_addr+1, crtc\_r9);

5048 00002C41 EE <1> out dx, al

5049 <1> ;inc bh

5050 <1> ; if(lines==8)

5051 <1> ;cmp bh, 8

5052 00002C42 80FF07 <1> cmp bh, 7

5053 00002C45 7506 <1> jne short ssl\_1

5054 <1> ; biosfn\_set\_cursor\_shape(0x06,0x07);

5055 00002C47 66B90706 <1> mov cx, 0607h

5056 00002C4B EB06 <1> jmp short ssl\_2

5057 <1> ssl\_1:

5058 <1> ; biosfn\_set\_cursor\_shape(lines-4,lines-3);

5059 00002C4D 88F9 <1> mov cl, bh ; lines - 1

5060 00002C4F 88CD <1> mov ch, cl ; lines - 1 (16 -> 15)

5061 00002C51 FECD <1> dec ch ; lines - 2 (16 -> 14)

5062 <1> ssl\_2:

5063 <1> ; CH = start line, CL = stop line

5064 00002C53 B40A <1> mov ah, 10 ; 6845 register for cursor set

5065 00002C55 66890D[DB5E0000] <1> mov [CURSOR\_MODE], cx ; save in data area

5066 00002C5C E812F1FFFF <1> call m16 ; output cx register

5067 <1> ; write\_word(BIOSMEM\_SEG,BIOSMEM\_CHAR\_HEIGHT, lines);

5068 00002C61 FEC7 <1> inc bh ; lines

5069 00002C63 883D[C65E0000] <1> mov [CHAR\_HEIGHT], bh

5070 <1> ; outb(crtc\_addr, 0x12);

5071 00002C69 66BAD403 <1> mov dx, 3D4h ; CRTC\_ADDRESS

5072 00002C6D B012 <1> mov al, 12h

5073 00002C6F EE <1> out dx, al

5074 <1> ; vde = inb(crtc\_addr+1);

5075 00002C70 6642 <1> inc dx

5076 00002C72 EC <1> in al, dx

5077 00002C73 88C4 <1> mov ah, al

5078 <1> ; outb(crtc\_addr, 0x07);

5079 00002C75 664A <1> dec dx

5080 00002C77 B007 <1> mov al, 07h

5081 00002C79 EE <1> out dx, al

5082 <1> ; ovl = inb(crtc\_addr+1);

5083 00002C7A 6642 <1> inc dx

5084 00002C7C EC <1> in al, dx

5085 <1> ; vde += (((ovl & 0x02) << 7) + ((ovl & 0x40) << 3) + 1);

5086 00002C7D 88E2 <1> mov dl, ah ; vde

5087 00002C7F 88C6 <1> mov dh, al ; ovl

5088 00002C81 6683E002 <1> and ax, 02h

5089 00002C85 66C1E007 <1> shl ax, 7

5090 00002C89 6689C1 <1> mov cx, ax ; (ovl & 0x02) << 7)

5091 00002C8C 88F0 <1> mov al, dh ; ovl

5092 00002C8E 6683E040 <1> and ax, 40h

5093 00002C92 66C1E003 <1> shl ax, 3 ; (ovl & 0x40) << 3)

5094 00002C96 6640 <1> inc ax ; + 1

5095 00002C98 6601C8 <1> add ax, cx

5096 00002C9B 30F6 <1> xor dh, dh

5097 00002C9D 6601D0 <1> add ax, dx ; + vde

5098 <1> ; rows = vde / lines;

5099 00002CA0 F6F7 <1> div bh

5100 <1> ;dec al ; rows -1

5101 <1> ; write\_byte(BIOSMEM\_SEG,BIOSMEM\_NB\_ROWS, rows-1);

5102 00002CA2 A2[CA5E0000] <1> mov [VGA\_ROWS], al ; rows (not 'rows-1' !)

5103 <1> ; write\_word(BIOSMEM\_SEG,BIOSMEM\_PAGE\_SIZE, rows \* cols \* 2);

5104 00002CA7 8A25[C45E0000] <1> mov ah, [CRT\_COLS]

5105 00002CAD F6E4 <1> mul ah

5106 00002CAF 66D1E0 <1> shl ax, 1

5107 00002CB2 66A3[D4650100] <1> mov [CRT\_LEN], ax

5108 00002CB8 C3 <1> retn

5109 <1>

5110 <1> load\_text\_8\_14\_pat:

5111 <1> ; 26/07/2016

5112 <1> ; 25/07/2016

5113 <1> ; 23/07/2016

5114 <1> ; 09/07/2016

5115 <1> ; load user defined (EGA/VGA) text fonts

5116 <1> ;

5117 <1> ; derived from 'Plex86/Bochs VGABios' source code

5118 <1> ; vgabios-0.7a (2011)

5119 <1> ; by the LGPL VGABios developers Team (2001-2008)

5120 <1> ; 'vgabios.c', 'biosfn\_load\_text\_8\_14\_pat'

5121 <1>

5122 <1> ; biosfn\_load\_text\_8\_14\_pat (AL,BL)

5123 <1>

5124 <1> ; get\_font\_access();

5125 <1> ; blockaddr = ((BL & 0x03) << 14) + ((BL & 0x04) << 11);

5126 <1> ; for(i=0;i<0x100;i++)

5127 <1> ; {

5128 <1> ; src = i \* 14;

5129 <1> ; dest = blockaddr + i \* 32;

5130 <1> ; memcpyb(0xA000, dest, 0xC000, vgafont14+src, 14);

5131 <1> ; }

5132 <1> ; release\_font\_access();

5133 <1> ; if(AL>=0x10)

5134 <1> ; {

5135 <1> ; set\_scan\_lines(14);

5136 <1> ; }

5137 <1>

5138 00002CB9 50 <1> push eax

5139 00002CBA E8FEFEFFFF <1> call get\_font\_access

5140 <1>

5141 <1> ; blockaddr = ((BL & 0x03) << 14) + ((BL & 0x04) << 11);

5142 <1> ;mov dl, bl

5143 <1> ;and dl, 3

5144 <1> ;shl dx, 14

5145 <1> ;xchg dx, bx

5146 <1> ;and dl, 4

5147 <1> ;shl dx, 11

5148 <1> ;add dx, bx

5149 <1>

5150 <1> ;xor dx, dx ; blockaddr = 0

5151 <1> ; Always block 0 for TRDOS 386 ! (blockaddr=0(

5152 <1>

5153 00002CBF 28DB <1> sub bl, bl ; i = 0

5154 00002CC1 B70E <1> mov bh, 14

5155 00002CC3 BE[B8340100] <1> mov esi, vgafont14

5156 00002CC8 BF00000A00 <1> mov edi, 0A0000h

5157 <1> lt8\_14\_1:

5158 <1> ;mov al, bl

5159 <1> ;mul bh

5160 <1> ;movzx esi, ax

5161 <1> ;add esi, vgafont14

5162 <1> ;mov al, bl

5163 <1> ;sub ah, ah

5164 <1> ;shl ax, 5 ; \* 32

5165 <1> ;;add ax, dx ; blockaddr + i \* 32;

5166 <1> ;movzx edi, ax ; dest

5167 <1> ;add edi, 0A0000h

5168 00002CCD 0FB6CF <1> movzx ecx, bh

5169 00002CD0 F3A4 <1> rep movsb

5170 00002CD2 83C712 <1> add edi, 18 ; 32 - 14

5171 00002CD5 FEC3 <1> inc bl

5172 00002CD7 75F4 <1> jnz short lt8\_14\_1

5173 <1> ;

5174 00002CD9 E814FFFFFF <1> call release\_font\_access

5175 <1> ;

5176 00002CDE 58 <1> pop eax

5177 <1> ; if(AL>=0x10)

5178 00002CDF 3C10 <1> cmp al, 10h

5179 00002CE1 7205 <1> jb short lt8\_14\_4

5180 <1> ; BH = 14

5181 <1> ; set\_scan\_lines(14);

5182 00002CE3 E849FFFFFF <1> call set\_scan\_lines

5183 <1> lt8\_14\_4:

5184 00002CE8 C3 <1> retn

5185 <1>

5186 <1> load\_text\_8\_8\_pat:

5187 <1> ; 26/07/2016

5188 <1> ; 25/07/2016

5189 <1> ; 23/07/2016

5190 <1> ; 09/07/2016

5191 <1> ; load user defined (EGA/VGA) text fonts

5192 <1> ;

5193 <1> ; derived from 'Plex86/Bochs VGABios' source code

5194 <1> ; vgabios-0.7a (2011)

5195 <1> ; by the LGPL VGABios developers Team (2001-2008)

5196 <1> ; 'vgabios.c', 'biosfn\_load\_text\_8\_8\_pat'

5197 <1>

5198 <1> ; biosfn\_load\_text\_8\_8\_pat (AL,BL)

5199 <1>

5200 <1> ; get\_font\_access();

5201 <1> ; blockaddr = ((BL & 0x03) << 14) + ((BL & 0x04) << 11);

5202 <1> ; for(i=0;i<0x100;i++)

5203 <1> ; {

5204 <1> ; src = i \* 8;

5205 <1> ; dest = blockaddr + i \* 32;

5206 <1> ; memcpyb(0xA000, dest, 0xC000, vgafont8+src, 8);

5207 <1> ; }

5208 <1> ; release\_font\_access();

5209 <1> ; if(AL>=0x10)

5210 <1> ; {

5211 <1> ; set\_scan\_lines(8);

5212 <1> ; }

5213 <1>

5214 00002CE9 50 <1> push eax

5215 00002CEA E8CEFEFFFF <1> call get\_font\_access

5216 <1>

5217 <1> ; blockaddr = ((BL & 0x03) << 14) + ((BL & 0x04) << 11);

5218 <1> ;mov dl, bl

5219 <1> ;and dl, 3

5220 <1> ;shl dx, 14

5221 <1> ;xchg dx, bx

5222 <1> ;and dl, 4

5223 <1> ;shl dx, 11

5224 <1> ;add dx, bx

5225 <1>

5226 <1> ;xor dx, dx ; blockaddr = 0

5227 <1> ; Always block 0 for TRDOS 386 ! (blockaddr=0(

5228 <1>

5229 00002CEF 28DB <1> sub bl, bl ; i = 0

5230 00002CF1 B708 <1> mov bh, 8

5231 00002CF3 BE[B82C0100] <1> mov esi, vgafont8

5232 00002CF8 BF00000A00 <1> mov edi, 0A0000h

5233 <1> lt8\_8\_1:

5234 <1> ;mov al, bl

5235 <1> ;mul bh

5236 <1> ;movzx esi, ax

5237 <1> ;add esi, vgafont8

5238 <1> ;mov al, bl

5239 <1> ;sub ah, ah

5240 <1> ;shl ax, 5 ; \* 32

5241 <1> ;;add ax, dx ; blockaddr + i \* 32;

5242 <1> ;movzx edi, ax ; dest

5243 <1> ;add edi, 0A0000h

5244 00002CFD 0FB6CF <1> movzx ecx, bh

5245 00002D00 F3A4 <1> rep movsb

5246 00002D02 83C718 <1> add edi, 24 ; 32 - 8

5247 00002D05 FEC3 <1> inc bl

5248 00002D07 75F4 <1> jnz short lt8\_8\_1

5249 <1> ;

5250 00002D09 E8E4FEFFFF <1> call release\_font\_access

5251 <1> ;

5252 00002D0E 58 <1> pop eax

5253 <1> ; if(AL>=0x10)

5254 00002D0F 3C10 <1> cmp al, 10h

5255 00002D11 7205 <1> jb short lt8\_8\_2

5256 <1> ; BH = 8

5257 <1> ; set\_scan\_lines(8);

5258 00002D13 E819FFFFFF <1> call set\_scan\_lines

5259 <1> lt8\_8\_2:

5260 00002D18 C3 <1> retn

5261 <1>

5262 <1> load\_text\_8\_16\_pat:

5263 <1> ; 26/07/2016

5264 <1> ; 25/07/2016

5265 <1> ; 23/07/2016

5266 <1> ; 09/07/2016

5267 <1> ; load user defined (EGA/VGA) text fonts

5268 <1> ;

5269 <1> ; derived from 'Plex86/Bochs VGABios' source code

5270 <1> ; vgabios-0.7a (2011)

5271 <1> ; by the LGPL VGABios developers Team (2001-2008)

5272 <1> ; 'vgabios.c', 'biosfn\_load\_text\_8\_16\_pat'

5273 <1>

5274 <1> ; biosfn\_load\_text\_8\_16\_pat (AL,BL)

5275 <1>

5276 <1> ; get\_font\_access();

5277 <1> ; blockaddr = ((BL & 0x03) << 14) + ((BL & 0x04) << 11);

5278 <1> ; for(i=0;i<0x100;i++)

5279 <1> ; {

5280 <1> ; src = i \* 16;

5281 <1> ; dest = blockaddr + i \* 32;

5282 <1> ; memcpyb(0xA000, dest, 0xC000, vgafont16+src, 16);

5283 <1> ; }

5284 <1> ; release\_font\_access();

5285 <1> ; if(AL>=0x10)

5286 <1> ; {

5287 <1> ; set\_scan\_lines(16);

5288 <1> ; }

5289 <1>

5290 00002D19 50 <1> push eax

5291 00002D1A E89EFEFFFF <1> call get\_font\_access

5292 <1>

5293 <1> ; blockaddr = ((BL & 0x03) << 14) + ((BL & 0x04) << 11);

5294 <1> ;mov dl, bl

5295 <1> ;and dl, 3

5296 <1> ;shl dx, 14

5297 <1> ;xchg dx, bx

5298 <1> ;and dl, 4

5299 <1> ;shl dx, 11

5300 <1> ;add dx, bx

5301 <1>

5302 <1> ;xor dx, dx ; blockaddr = 0

5303 <1> ; Always block 0 for TRDOS 386 ! (blockaddr=0(

5304 <1>

5305 00002D1F 28DB <1> sub bl, bl ; i = 0

5306 00002D21 B710 <1> mov bh, 16

5307 00002D23 BE[B8420100] <1> mov esi, vgafont16

5308 00002D28 BF00000A00 <1> mov edi, 0A0000h

5309 00002D2D 0FB6C7 <1> movzx eax, bh

5310 <1> lt8\_16\_1:

5311 <1> ;mov al, bl

5312 <1> ;mul bh

5313 <1> ;movzx esi, ax

5314 <1> ;add esi, vgafont16

5315 <1> ;mov al, bl ; i

5316 <1> ;sub ah, ah

5317 <1> ;shl ax, 5 ; \* 32

5318 <1> ;;add ax, dx ; blockaddr + i \* 32;

5319 <1> ;movzx edi, ax ; dest

5320 <1> ;add edi, 0A0000h

5321 <1> ;movzx ecx, bh

5322 00002D30 89C1 <1> mov ecx, eax ; 16

5323 00002D32 F3A4 <1> rep movsb

5324 00002D34 01C7 <1> add edi, eax ; add edi, 16

5325 00002D36 FEC3 <1> inc bl

5326 00002D38 75F6 <1> jnz short lt8\_16\_1

5327 <1> ;

5328 00002D3A E8B3FEFFFF <1> call release\_font\_access

5329 <1> ;

5330 00002D3F 58 <1> pop eax

5331 <1> ; if(AL>=0x10)

5332 00002D40 3C10 <1> cmp al, 10h

5333 00002D42 7205 <1> jb short lt8\_16\_2

5334 <1> ; BH = 16

5335 <1> ; set\_scan\_lines(16);

5336 00002D44 E8E8FEFFFF <1> call set\_scan\_lines

5337 <1> lt8\_16\_2:

5338 00002D49 C3 <1> retn

5339 <1>

5340 <1> load\_gfx\_user\_chars:

5341 <1> ; 08/08/2016

5342 <1> ; 10/07/2016

5343 <1> ; Setup User-Defined Font for Graphics Mode (VGA)

5344 <1> ;

5345 <1> ; derived from 'Plex86/Bochs VGABios' source code

5346 <1> ; vgabios-0.7a (2011)

5347 <1> ; by the LGPL VGABios developers Team (2001-2008)

5348 <1> ; 'vgabios.c', 'biosfn\_load\_gfx\_user\_chars'

5349 <1>

5350 <1> ; biosfn\_load\_gfx\_user\_chars (ES,BP,CX,BL,DL)

5351 <1> ; /\* set 0x43 INT pointer \*/

5352 <1> ; write\_word(0x0, 0x43\*4, BP);

5353 <1> ; write\_word(0x0, 0x43\*4+2, ES);

5354 00002D4A 31C0 <1> xor eax, eax

5355 00002D4C 48 <1> dec eax ; 0FFFFFFFFh (user defined fonts)

5356 00002D4D A3[E6650100] <1> mov [VGA\_INT43H], eax

5357 <1>

5358 <1> ; BL screen rows code: 00H = user-specified (in DL)

5359 <1> ; 01H = 14 rows

5360 <1> ; 02H = 25 rows

5361 <1> ; 03H = 43 rows

5362 <1> ; CX bytes per character definition

5363 <1> ; DL (when BL=0) custom number of character rows on screen

5364 <1> ; dh = 0 -> 256 characters

5365 <1> ; dh = 80h -> 128 characters

5366 <1> ; (If DH <> 0 and DH <> 80h -> invalid)

5367 <1> ; EBP address of font-definition information (user's mem space)

5368 <1>

5369 <1> ; switch (BL) {

5370 <1> ; case 0:

5371 <1> ; write\_byte(BIOSMEM\_SEG,BIOSMEM\_NB\_ROWS, DL-1);

5372 <1> ; break;

5373 00002D52 20DB <1> and bl, bl

5374 00002D54 7508 <1> jnz short l\_gfx\_uc\_1

5375 00002D56 8815[CA5E0000] <1> mov [VGA\_ROWS], dl ; not DL-1 !

5376 00002D5C EB23 <1> jmp short l\_gfx\_uc\_4

5377 <1> l\_gfx\_uc\_1:

5378 <1> ; case 1:

5379 <1> ; write\_byte(BIOSMEM\_SEG,BIOSMEM\_NB\_ROWS, 13);

5380 <1> ; break;

5381 00002D5E FECB <1> dec bl

5382 00002D60 7509 <1> jnz short l\_gfx\_uc\_2

5383 <1> ; bl = 1

5384 00002D62 C605[CA5E0000]0E <1> mov byte [VGA\_ROWS], 14 ; not 13 !

5385 00002D69 EB16 <1> jmp short l\_gfx\_uc\_4

5386 <1> l\_gfx\_uc\_2:

5387 00002D6B FECB <1> dec bl

5388 00002D6D 740B <1> jz short l\_gfx\_uc\_3 ; bl = 2

5389 00002D6F FECB <1> dec bl

5390 00002D71 750E <1> jnz short l\_gfx\_uc\_4 ; bl > 3

5391 <1> ; bl = 3

5392 <1> ; case 3:

5393 <1> ; write\_byte(BIOSMEM\_SEG,BIOSMEM\_NB\_ROWS, 42);

5394 <1> ; break;

5395 00002D73 C605[CA5E0000]2B <1> mov byte [VGA\_ROWS], 43 ; not 42 !

5396 <1> l\_gfx\_uc\_3:

5397 <1> ; case 2:

5398 <1> ; default:

5399 <1> ; write\_byte(BIOSMEM\_SEG,BIOSMEM\_NB\_ROWS, 24);

5400 <1> ; break;

5401 <1> ; bl = 2 or bl > 3

5402 00002D7A C605[CA5E0000]19 <1> mov byte [VGA\_ROWS], 25 ; not 24 !

5403 <1> ; }

5404 <1> l\_gfx\_uc\_4:

5405 <1> ; write\_byte(BIOSMEM\_SEG, BIOSMEM\_CHAR\_HEIGHT, CX);

5406 00002D81 880D[C65E0000] <1> mov [CHAR\_HEIGHT], cl

5407 <1> ; }

5408 00002D87 C3 <1> retn

5409 <1>

5410 <1> load\_gfx\_8\_14\_chars:

5411 <1> ; 08/08/2016

5412 <1> ; 10/07/2016

5413 <1> ; Setup ROM 8x14 Font for Graphics Mode (VGA)

5414 <1> ;

5415 <1> ; derived from 'Plex86/Bochs VGABios' source code

5416 <1> ; vgabios-0.7a (2011)

5417 <1> ; by the LGPL VGABios developers Team (2001-2008)

5418 <1> ; 'vgabios.c', 'biosfn\_load\_gfx\_8\_14\_chars'

5419 <1>

5420 <1> ; biosfn\_load\_gfx\_8\_14\_chars (BL)

5421 <1> ; /\* set 0x43 INT pointer \*/

5422 <1> ; write\_word(0x0, 0x43\*4, &vgafont14);

5423 <1> ; write\_word(0x0, 0x43\*4+2, 0xC000);

5424 00002D88 C705[E6650100]- <1> mov dword [VGA\_INT43H], vgafont14

5424 00002D8E [B8340100] <1>

5425 <1>

5426 <1> ; BL screen rows code: 00H = user-specified (in DL)

5427 <1> ; 01H = 14 rows

5428 <1> ; 02H = 25 rows

5429 <1> ; 03H = 43 rows

5430 <1> ; DL (when BL=0) custom number of char rows on screen

5431 <1>

5432 <1> ; switch (BL) {

5433 <1> ; case 0:

5434 <1> ; write\_byte(BIOSMEM\_SEG,BIOSMEM\_NB\_ROWS, DL-1);

5435 <1> ; break;

5436 00002D92 20DB <1> and bl, bl

5437 00002D94 7508 <1> jnz short l\_gfx\_8\_14c\_1

5438 00002D96 8815[CA5E0000] <1> mov [VGA\_ROWS], dl ; not DL-1 !

5439 00002D9C EB23 <1> jmp short l\_gfx\_8\_14c\_4

5440 <1> l\_gfx\_8\_14c\_1:

5441 <1> ; case 1:

5442 <1> ; write\_byte(BIOSMEM\_SEG,BIOSMEM\_NB\_ROWS, 13);

5443 <1> ; break;

5444 00002D9E FECB <1> dec bl

5445 00002DA0 7509 <1> jnz short l\_gfx\_8\_14c\_2

5446 <1> ; bl = 1

5447 00002DA2 C605[CA5E0000]0E <1> mov byte [VGA\_ROWS], 14 ; not 13 !

5448 00002DA9 EB16 <1> jmp short l\_gfx\_8\_14c\_4

5449 <1> l\_gfx\_8\_14c\_2:

5450 00002DAB FECB <1> dec bl

5451 00002DAD 740B <1> jz short l\_gfx\_8\_14c\_3 ; bl = 2

5452 00002DAF FECB <1> dec bl

5453 00002DB1 750E <1> jnz short l\_gfx\_8\_14c\_4 ; bl > 3

5454 <1> ; bl = 3

5455 <1> ; case 3:

5456 <1> ; write\_byte(BIOSMEM\_SEG,BIOSMEM\_NB\_ROWS, 42);

5457 <1> ; break;

5458 00002DB3 C605[CA5E0000]2B <1> mov byte [VGA\_ROWS], 43 ; not 42 !

5459 <1> l\_gfx\_8\_14c\_3:

5460 <1> ; case 2:

5461 <1> ; default:

5462 <1> ; write\_byte(BIOSMEM\_SEG,BIOSMEM\_NB\_ROWS, 24);

5463 <1> ; break;

5464 <1> ; bl = 2 or bl > 3

5465 00002DBA C605[CA5E0000]19 <1> mov byte [VGA\_ROWS], 25 ; not 24 !

5466 <1> ; }

5467 <1> l\_gfx\_8\_14c\_4:

5468 <1> ; write\_byte(BIOSMEM\_SEG, BIOSMEM\_CHAR\_HEIGHT, 14);

5469 00002DC1 C605[C65E0000]0E <1> mov byte [CHAR\_HEIGHT], 14

5470 <1> ; }

5471 00002DC8 C3 <1> retn

5472 <1>

5473 <1> load\_gfx\_8\_8\_chars:

5474 <1> ; 08/08/2016

5475 <1> ; 10/07/2016

5476 <1> ; Setup ROM 8x14 Font for Graphics Mode (VGA)

5477 <1> ;

5478 <1> ; derived from 'Plex86/Bochs VGABios' source code

5479 <1> ; vgabios-0.7a (2011)

5480 <1> ; by the LGPL VGABios developers Team (2001-2008)

5481 <1> ; 'vgabios.c', 'biosfn\_load\_gfx\_8\_8\_dd\_chars'

5482 <1>

5483 <1> ; biosfn\_load\_gfx\_8\_8\_dd\_chars (BL)

5484 <1> ; /\* set 0x43 INT pointer \*/

5485 <1> ; write\_word(0x0, 0x43\*4, &vgafont8);

5486 <1> ; write\_word(0x0, 0x43\*4+2, 0xC000);

5487 00002DC9 C705[E6650100]- <1> mov dword [VGA\_INT43H], vgafont8

5487 00002DCF [B82C0100] <1>

5488 <1>

5489 <1> ; BL screen rows code: 00H = user-specified (in DL)

5490 <1> ; 01H = 14 rows

5491 <1> ; 02H = 25 rows

5492 <1> ; 03H = 43 rows

5493 <1> ; DL (when BL=0) custom number of char rows on screen

5494 <1>

5495 <1> ; switch (BL) {

5496 <1> ; case 0:

5497 <1> ; write\_byte(BIOSMEM\_SEG,BIOSMEM\_NB\_ROWS, DL-1);

5498 <1> ; break;

5499 00002DD3 20DB <1> and bl, bl

5500 00002DD5 7508 <1> jnz short l\_gfx\_8\_8c\_1

5501 00002DD7 8815[CA5E0000] <1> mov [VGA\_ROWS], dl ; not DL-1 !

5502 00002DDD EB23 <1> jmp short l\_gfx\_8\_8c\_4

5503 <1> l\_gfx\_8\_8c\_1:

5504 <1> ; case 1:

5505 <1> ; write\_byte(BIOSMEM\_SEG,BIOSMEM\_NB\_ROWS, 13);

5506 <1> ; break;

5507 00002DDF FECB <1> dec bl

5508 00002DE1 7509 <1> jnz short l\_gfx\_8\_8c\_2

5509 <1> ; bl = 1

5510 00002DE3 C605[CA5E0000]0E <1> mov byte [VGA\_ROWS], 14 ; not 13 !

5511 00002DEA EB16 <1> jmp short l\_gfx\_8\_8c\_4

5512 <1> l\_gfx\_8\_8c\_2:

5513 00002DEC FECB <1> dec bl

5514 00002DEE 740B <1> jz short l\_gfx\_8\_8c\_3 ; bl = 2

5515 00002DF0 FECB <1> dec bl

5516 00002DF2 750E <1> jnz short l\_gfx\_8\_8c\_4 ; bl > 3

5517 <1> ; bl = 3

5518 <1> ; case 3:

5519 <1> ; write\_byte(BIOSMEM\_SEG,BIOSMEM\_NB\_ROWS, 42);

5520 <1> ; break;

5521 00002DF4 C605[CA5E0000]2B <1> mov byte [VGA\_ROWS], 43 ; not 42 !

5522 <1> l\_gfx\_8\_8c\_3:

5523 <1> ; case 2:

5524 <1> ; default:

5525 <1> ; write\_byte(BIOSMEM\_SEG,BIOSMEM\_NB\_ROWS, 24);

5526 <1> ; break;

5527 <1> ; bl = 2 or bl > 3

5528 00002DFB C605[CA5E0000]19 <1> mov byte [VGA\_ROWS], 25 ; not 24 !

5529 <1> ; }

5530 <1> l\_gfx\_8\_8c\_4:

5531 <1> ; write\_byte(BIOSMEM\_SEG, BIOSMEM\_CHAR\_HEIGHT, 8);

5532 00002E02 C605[C65E0000]08 <1> mov byte [CHAR\_HEIGHT], 8

5533 <1> ; }

5534 00002E09 C3 <1> retn

5535 <1>

5536 <1> load\_gfx\_8\_16\_chars:

5537 <1> ; 08/08/2016

5538 <1> ; 10/07/2016

5539 <1> ; Setup ROM 8x14 Font for Graphics Mode (VGA)

5540 <1> ;

5541 <1> ; derived from 'Plex86/Bochs VGABios' source code

5542 <1> ; vgabios-0.7a (2011)

5543 <1> ; by the LGPL VGABios developers Team (2001-2008)

5544 <1> ; 'vgabios.c', 'biosfn\_load\_gfx\_8\_16\_chars'

5545 <1>

5546 <1> ; biosfn\_load\_gfx\_8\_16\_chars (BL)

5547 <1> ; /\* set 0x43 INT pointer \*/

5548 <1> ; write\_word(0x0, 0x43\*4, &vgafont16);

5549 <1> ; write\_word(0x0, 0x43\*4+2, 0xC000);

5550 00002E0A C705[E6650100]- <1> mov dword [VGA\_INT43H], vgafont16

5550 00002E10 [B8420100] <1>

5551 <1>

5552 <1> ; BL screen rows code: 00H = user-specified (in DL)

5553 <1> ; 01H = 14 rows

5554 <1> ; 02H = 25 rows

5555 <1> ; 03H = 43 rows

5556 <1> ; DL (when BL=0) custom number of char rows on screen

5557 <1>

5558 <1> ; switch (BL) {

5559 <1> ; case 0:

5560 <1> ; write\_byte(BIOSMEM\_SEG,BIOSMEM\_NB\_ROWS, DL-1);

5561 <1> ; break;

5562 00002E14 20DB <1> and bl, bl

5563 00002E16 7508 <1> jnz short l\_gfx\_8\_16c\_1

5564 00002E18 8815[CA5E0000] <1> mov [VGA\_ROWS], dl ; not DL-1 !

5565 00002E1E EB23 <1> jmp short l\_gfx\_8\_16c\_4

5566 <1> l\_gfx\_8\_16c\_1:

5567 <1> ; case 1:

5568 <1> ; write\_byte(BIOSMEM\_SEG,BIOSMEM\_NB\_ROWS, 13);

5569 <1> ; break;

5570 00002E20 FECB <1> dec bl

5571 00002E22 7509 <1> jnz short l\_gfx\_8\_16c\_2

5572 <1> ; bl = 1

5573 00002E24 C605[CA5E0000]0E <1> mov byte [VGA\_ROWS], 14 ; not 13 !

5574 00002E2B EB16 <1> jmp short l\_gfx\_8\_16c\_4

5575 <1> l\_gfx\_8\_16c\_2:

5576 00002E2D FECB <1> dec bl

5577 00002E2F 740B <1> jz short l\_gfx\_8\_16c\_3 ; bl = 2

5578 00002E31 FECB <1> dec bl

5579 00002E33 750E <1> jnz short l\_gfx\_8\_16c\_4 ; bl > 3

5580 <1> ; bl = 3

5581 <1> ; case 3:

5582 <1> ; write\_byte(BIOSMEM\_SEG,BIOSMEM\_NB\_ROWS, 42);

5583 <1> ; break;

5584 00002E35 C605[CA5E0000]2B <1> mov byte [VGA\_ROWS], 43 ; not 42 !

5585 <1> l\_gfx\_8\_16c\_3:

5586 <1> ; case 2:

5587 <1> ; default:

5588 <1> ; write\_byte(BIOSMEM\_SEG,BIOSMEM\_NB\_ROWS, 24);

5589 <1> ; break;

5590 <1> ; bl = 2 or bl > 3

5591 00002E3C C605[CA5E0000]19 <1> mov byte [VGA\_ROWS], 25 ; not 24 !

5592 <1> ; }

5593 <1> l\_gfx\_8\_16c\_4:

5594 <1> ; write\_byte(BIOSMEM\_SEG, BIOSMEM\_CHAR\_HEIGHT, 16);

5595 00002E43 C605[C65E0000]10 <1> mov byte [CHAR\_HEIGHT], 16

5596 <1> ; }

5597 00002E4A C3 <1> retn

5598 <1>

5599 <1> get\_font\_info:

5600 <1> ; 19/09/2016

5601 <1> ; 08/08/2016

5602 <1> ; 10/07/2016

5603 <1> ; Get Current Character Generator Info (VGA)

5604 <1> ;

5605 <1> ; derived from 'Plex86/Bochs VGABios' source code

5606 <1> ; vgabios-0.7a (2011)

5607 <1> ; by the LGPL VGABios developers Team (2001-2008)

5608 <1> ; 'vgabios.c', 'biosfn\_get\_font\_info'

5609 <1>

5610 <1> ; Modified for TRDOS 386 !

5611 <1> ;

5612 <1> ; INPUT ->

5613 <1> ; AX = 1130h

5614 <1> ; BL = 0 -> Get info for current VGA font

5615 <1> ; (BH = unused)

5616 <1> ; 19/09/2016

5617 <1> ; BL > 0 -> Get requested character font data

5618 <1> ; BL = 1 -> vgafont8

5619 <1> ; BL = 2 -> vgafont14

5620 <1> ; BL = 3 -> vgafont16

5621 <1> ; BL > 3 -> Invalid function (for now!)

5622 <1> ; BH = ASCII code of the first character

5623 <1> ; ECX = Number of characters from the 1st char

5624 <1> ; ECX >= 256 -> All (256-BH) characters

5625 <1> ; ECX = 0 -> All characters (BH = unused)

5626 <1> ; EDX = User's Buffer Address

5627 <1> ; OUTPUT ->

5628 <1> ; AL = height (scanlines), bytes per character

5629 <1> ; AH = screen rows

5630 <1> ; Byte 16-23 of EAX = number of columns

5631 <1> ; Byte 24-31 of EAX =

5632 <1> ; 0 -> default font (not configured yet)

5633 <1> ; 0FFh -> user defined font

5634 <1> ; 14 = vgafont14

5635 <1> ; 8 = vgafont8

5636 <1> ; 16 = vgafont16

5637 <1> ; If BL input > 0 ->

5638 <1> ; EAX = Actual transfer count

5639 <1> ;

5640 00002E4B 20DB <1> and bl, bl

5641 00002E4D 7408 <1> jz short gfi\_0

5642 <1> ; invalid function (input)

5643 00002E4F 80FB03 <1> cmp bl, 3

5644 00002E52 7642 <1> jna short gfi\_4

5645 00002E54 31C0 <1> xor eax, eax ; 0

5646 00002E56 C3 <1> retn

5647 <1> gfi\_0:

5648 00002E57 A0[C65E0000] <1> mov al, [CHAR\_HEIGHT]

5649 00002E5C 8A25[CA5E0000] <1> mov ah, [VGA\_ROWS]

5650 00002E62 C1E010 <1> shl eax, 16

5651 00002E65 A0[C45E0000] <1> mov al, [CRT\_COLS]

5652 00002E6A 8B0D[E6650100] <1> mov ecx, [VGA\_INT43H]

5653 00002E70 21C9 <1> and ecx, ecx

5654 00002E72 741E <1> jz short gfi\_2 ; 0 = default font

5655 00002E74 41 <1> inc ecx ; 0FFFFFFFFh -> 0 (user defined font)

5656 00002E75 7504 <1> jnz short gfi\_1

5657 00002E77 FECC <1> dec ah ; 0FFh

5658 00002E79 EB17 <1> jmp short gfi\_2

5659 <1> gfi\_1:

5660 00002E7B 49 <1> dec ecx ; 08/08/2016

5661 00002E7C B40E <1> mov ah, 14

5662 00002E7E 81F9[B8340100] <1> cmp ecx, vgafont14

5663 00002E84 740C <1> je short gfi\_2

5664 00002E86 B408 <1> mov ah, 8

5665 00002E88 81F9[B82C0100] <1> cmp ecx, vgafont8

5666 00002E8E 7402 <1> je short gfi\_2

5667 <1> ; vgafont16

5668 00002E90 D0E4 <1> shl ah, 1 ; ah = 16

5669 <1> gfi\_2:

5670 00002E92 C1C010 <1> rol eax, 16

5671 <1> gfi\_3:

5672 00002E95 C3 <1> retn

5673 <1> gfi\_4:

5674 00002E96 89D7 <1> mov edi, edx ; \*\*

5675 00002E98 80FB02 <1> cmp bl, 2

5676 00002E9B 720B <1> jb short gfi\_5

5677 00002E9D 772F <1> ja short gfi\_7

5678 <1> ;BL = 2 -> vgafont14

5679 00002E9F BE[B8340100] <1> mov esi, vgafont14 ; \*

5680 00002EA4 B30E <1> mov bl, 14

5681 00002EA6 EB07 <1> jmp short gfi\_6

5682 <1> gfi\_5:

5683 <1> ;BL = 1 -> vgafont8

5684 00002EA8 BE[B82C0100] <1> mov esi, vgafont8 ; \*

5685 00002EAD B308 <1> mov bl, 8

5686 <1> gfi\_6:

5687 00002EAF 09C9 <1> or ecx, ecx

5688 00002EB1 7424 <1> jz short gfi\_8 ; all chars from the 00h

5689 00002EB3 88F8 <1> mov al, bh ; character index

5690 00002EB5 F6E3 <1> mul bl ; char index \* char height/size

5691 00002EB7 0FB7D0 <1> movzx edx, ax

5692 00002EBA 01D6 <1> add esi, edx ; \*

5693 00002EBC 66BAFF00 <1> mov dx, 255

5694 00002EC0 28FA <1> sub dl, bh

5695 00002EC2 6642 <1> inc dx

5696 00002EC4 39D1 <1> cmp ecx, edx

5697 00002EC6 770F <1> ja short gfi\_8

5698 00002EC8 7412 <1> je short gfi\_9

5699 00002ECA 89D1 <1> mov ecx, edx

5700 00002ECC EB0E <1> jmp short gfi\_9

5701 <1> gfi\_7:

5702 <1> ;BL = 3 -> vgafont16

5703 00002ECE BE[B8420100] <1> mov esi, vgafont16 ; \*

5704 00002ED3 B310 <1> mov bl, 16

5705 00002ED5 EBD8 <1> jmp short gfi\_6

5706 <1> gfi\_8:

5707 00002ED7 B900010000 <1> mov ecx, 256

5708 <1> gfi\_9:

5709 00002EDC 6689C8 <1> mov ax, cx ; character count

5710 00002EDF 30FF <1> xor bh, bh

5711 00002EE1 66F7E3 <1> mul bx ; char count \* char height/size

5712 00002EE4 6689C1 <1> mov cx, ax

5713 <1>

5714 <1> ; ESI = source address in system space

5715 <1> ; EDI = user's buffer address

5716 <1> ; ECX = transfer (byte) count

5717 00002EE7 E88DB80000 <1> call transfer\_to\_user\_buffer

5718 00002EEC 89C8 <1> mov eax, ecx ; actual transfer count

5719 00002EEE C3 <1> retn

5720 <1>

5721 <1> vga\_pal\_funcs:

5722 <1> ; 10/08/2016

5723 <1> ; VGA Palette functions

5724 <1> ;

5725 <1> ; derived from 'Plex86/Bochs VGABios' source code

5726 <1> ; vgabios-0.7a (2011)

5727 <1> ; by the LGPL VGABios developers Team (2001-2008)

5728 <1> ; 'vgabios.c', 'vgarom.asm'

5729 <1>

5730 00002EEF 3C00 <1> cmp al, 0

5731 00002EF1 0F848F000000 <1> je set\_single\_palette\_reg

5732 <1> vga\_palf\_1001:

5733 00002EF7 3C01 <1> cmp al, 1

5734 00002EF9 0F84B4000000 <1> je set\_overscan\_border\_color

5735 <1> vga\_palf\_1002:

5736 00002EFF 3C02 <1> cmp al, 2

5737 00002F01 0F84B0000000 <1> je set\_all\_palette\_reg

5738 <1> vga\_palf\_1003:

5739 00002F07 3C03 <1> cmp al, 3

5740 00002F09 0F84E8000000 <1> je toggle\_intensity

5741 <1> vga\_palf\_1007:

5742 00002F0F 3C07 <1> cmp al, 7

5743 00002F11 0F840D010000 <1> je get\_single\_palette\_reg

5744 00002F17 7266 <1> jb short vga\_palf\_unknown

5745 <1> vga\_palf\_1008:

5746 00002F19 3C08 <1> cmp al, 8

5747 00002F1B 0F8437010000 <1> je read\_overscan\_border\_color

5748 <1> vga\_palf\_1009:

5749 00002F21 3C09 <1> cmp al, 9

5750 00002F23 0F8433010000 <1> je get\_all\_palette\_reg

5751 <1> vga\_palf\_1010:

5752 00002F29 3C10 <1> cmp al, 10h

5753 00002F2B 0F8487010000 <1> je set\_single\_dac\_reg

5754 00002F31 724C <1> jb short vga\_palf\_unknown

5755 <1> vga\_palf\_1012:

5756 00002F33 3C12 <1> cmp al, 12h

5757 00002F35 0F8498010000 <1> je set\_all\_dac\_reg

5758 00002F3B 7242 <1> jb short vga\_palf\_unknown

5759 <1> vga\_palf\_1013:

5760 00002F3D 3C13 <1> cmp al, 13h

5761 00002F3F 0F84CC010000 <1> je select\_video\_dac\_color\_page

5762 <1> vga\_palf\_1015:

5763 00002F45 3C15 <1> cmp al, 15h

5764 00002F47 0F8412020000 <1> je read\_single\_dac\_reg

5765 00002F4D 7230 <1> jb short vga\_palf\_unknown

5766 <1> vga\_palf\_1017:

5767 00002F4F 3C17 <1> cmp al, 17h

5768 00002F51 0F8428020000 <1> je read\_all\_dac\_reg

5769 00002F57 7226 <1> jb short vga\_palf\_unknown

5770 <1> vga\_palf\_1018:

5771 00002F59 3C18 <1> cmp al, 18h

5772 00002F5B 0F845E020000 <1> je set\_pel\_mask

5773 <1> vga\_palf\_1019:

5774 00002F61 3C19 <1> cmp al, 19h

5775 00002F63 0F8462020000 <1> je read\_pel\_mask

5776 <1> vga\_palf\_101A:

5777 00002F69 3C1A <1> cmp al, 1Ah

5778 00002F6B 0F8468020000 <1> je read\_video\_dac\_state

5779 <1> vga\_palf\_101B:

5780 00002F71 3C1B <1> cmp al, 1Bh

5781 <1> ;jne short vga\_palf\_unknown

5782 00002F73 770A <1> ja short vga\_palf\_unknown

5783 <1>

5784 00002F75 E80CF7FFFF <1> call gray\_scale\_summing

5785 00002F7A E9D5E5FFFF <1> jmp VIDEO\_RETURN

5786 <1>

5787 <1> vga\_palf\_unknown:

5788 00002F7F 29C0 <1> sub eax, eax ; 0 = invalid function

5789 00002F81 E9D3E5FFFF <1> jmp \_video\_return

5790 <1>

5791 <1> set\_single\_palette\_reg:

5792 <1> ; 10/08/2016

5793 <1> ; Set One Palette Register

5794 <1> ; BL = register number to set

5795 <1> ; (a 4-bit attribute nibble: 00h-0Fh)

5796 <1> ; BH = 6-bit RGB color to display

5797 <1> ; for that attribute

5798 <1>

5799 00002F86 80FB14 <1> cmp bl, 14h

5800 <1> ;ja short no\_actl\_reg1

5801 00002F89 0F87C5E5FFFF <1> ja VIDEO\_RETURN

5802 00002F8F 6650 <1> push ax

5803 00002F91 6652 <1> push dx

5804 00002F93 66BADA03 <1> mov dx, 3DAh ; VGAREG\_ACTL\_RESET

5805 00002F97 EC <1> in al, dx

5806 00002F98 66BAC003 <1> mov dx, 3C0h ; VGAREG\_ACTL\_ADDRESS

5807 00002F9C 88D8 <1> mov al, bl

5808 00002F9E EE <1> out dx, al

5809 00002F9F 88F8 <1> mov al, bh

5810 00002FA1 EE <1> out dx, al

5811 00002FA2 B020 <1> mov al, 20h

5812 00002FA4 EE <1> out dx, al

5813 <1> ; ifdef VBOX

5814 00002FA5 66BADA03 <1> mov dx, 3DAh ; VGAREG\_ACTL\_RESET

5815 00002FA9 EC <1> in al, dx

5816 <1> ; endif ; VBOX

5817 00002FAA 665A <1> pop dx

5818 00002FAC 6658 <1> pop ax

5819 <1> ;no\_actl\_reg1:

5820 00002FAE E9A1E5FFFF <1> jmp VIDEO\_RETURN

5821 <1>

5822 <1> set\_overscan\_border\_color:

5823 <1> ; 10/08/2016

5824 <1> ; Set Overscan/Border Color Register

5825 <1> ; BH = 6-bit RGB color to display

5826 <1> ; for that attribute

5827 <1>

5828 00002FB3 B311 <1> mov bl, 11h

5829 00002FB5 EBCF <1> jmp short set\_single\_palette\_reg

5830 <1>

5831 <1> set\_all\_palette\_reg:

5832 <1> ; 10/08/2016

5833 <1> ; Set All Palette Registers and Overscan

5834 <1> ; EDX = Address of 17 bytes;

5835 <1> ; an rgbRGB value for each of 16 palette

5836 <1> ; registers plus one for the border.

5837 <1>

5838 00002FB7 89D6 <1> mov esi, edx ; user buffer

5839 00002FB9 B911000000 <1> mov ecx, 17

5840 00002FBE 89E7 <1> mov edi, esp

5841 00002FC0 83EC14 <1> sub esp, 20

5842 00002FC3 E8FBB70000 <1> call transfer\_from\_user\_buffer

5843 <1> ;jc VIDEO\_RETURN

5844 <1>

5845 00002FC8 66BADA03 <1> mov dx, 3DAh ; VGAREG\_ACTL\_RESET

5846 00002FCC EC <1> in al, dx

5847 00002FCD B100 <1> mov cl, 0

5848 00002FCF 66BAC003 <1> mov dx, 3C0h ; VGAREG\_ACTL\_ADDRESS

5849 <1> set\_palette\_loop:

5850 00002FD3 88C8 <1> mov al, cl

5851 00002FD5 EE <1> out dx, al

5852 00002FD6 8A07 <1> mov al, [edi]

5853 00002FD8 EE <1> out dx, al

5854 00002FD9 47 <1> inc edi

5855 00002FDA FEC1 <1> inc cl

5856 00002FDC 80F910 <1> cmp cl, 10h

5857 00002FDF 75F2 <1> jne short set\_palette\_loop

5858 00002FE1 B011 <1> mov al, 11h

5859 00002FE3 EE <1> out dx, al

5860 00002FE4 8A07 <1> mov al, [edi]

5861 00002FE6 EE <1> out dx, al

5862 00002FE7 B020 <1> mov al, 20h

5863 00002FE9 EE <1> out dx, al

5864 <1> ; ifdef VBOX

5865 00002FEA 66BADA03 <1> mov dx, 3DAh ; VGAREG\_ACTL\_RESET

5866 00002FEE EC <1> in al, dx

5867 <1> ; endif ; VBOX

5868 00002FEF 83C414 <1> add esp, 20

5869 00002FF2 E95DE5FFFF <1> jmp VIDEO\_RETURN

5870 <1>

5871 <1> toggle\_intensity:

5872 <1> ; 10/08/2016

5873 <1> ; Select Foreground Blink or Bold Background

5874 <1> ; BL = 00h = enable bold backgrounds

5875 <1> ; (16 background colors)

5876 <1> ; 01h = enable blinking foreground

5877 <1> ; (8 background colors)

5878 <1>

5879 00002FF7 66BADA03 <1> mov dx, 3DAh ; VGAREG\_ACTL\_RESET

5880 00002FFB EC <1> in al, dx

5881 00002FFC 66BAC003 <1> mov dx, 3C0h ; VGAREG\_ACTL\_ADDRESS

5882 00003000 B010 <1> mov al, 10h

5883 00003002 EE <1> out dx, al

5884 00003003 66BAC103 <1> mov dx, 3C1h ; VGAREG\_ACTL\_READ\_DATA

5885 00003007 EC <1> in al, dx

5886 00003008 24F7 <1> and al, 0F7h

5887 0000300A 80E301 <1> and bl, 01h

5888 0000300D C0E303 <1> shl bl, 3

5889 00003010 08D8 <1> or al, bl

5890 00003012 66BAC003 <1> mov dx, 3C0h ; VGAREG\_ACTL\_ADDRESS

5891 00003016 EE <1> out dx, al

5892 00003017 B020 <1> mov al, 20h

5893 00003019 EE <1> out dx, al

5894 <1> ; ifdef VBOX

5895 0000301A 66BADA03 <1> mov dx, 3DAh ; VGAREG\_ACTL\_RESET

5896 0000301E EC <1> in al, dx

5897 <1> ; endif ; VBOX

5898 0000301F E930E5FFFF <1> jmp VIDEO\_RETURN

5899 <1>

5900 <1> get\_single\_palette\_reg:

5901 <1> ; 10/08/2016

5902 <1> ; Read One Palette Register

5903 <1> ; INPUT:

5904 <1> ; BL = Palette register to read (00h-0Fh)

5905 <1> ; OUTPUT:

5906 <1> ; BH = Current rgbRGB value of specified register

5907 <1> ; for that attribute

5908 <1>

5909 00003024 80FB14 <1> cmp bl, 14h

5910 <1> ;ja short no\_actl\_reg2

5911 00003027 0F8727E5FFFF <1> ja VIDEO\_RETURN

5912 <1>

5913 0000302D 66BADA03 <1> mov dx, 3DAh ; VGAREG\_ACTL\_RESET

5914 00003031 EC <1> in al, dx

5915 00003032 66BAC003 <1> mov dx, 3C0h ; VGAREG\_ACTL\_ADDRESS

5916 00003036 88D8 <1> mov al, bl

5917 00003038 EE <1> out dx, al

5918 00003039 66BAC103 <1> mov dx, 3C1h ; VGAREG\_ACTL\_READ\_DATA

5919 0000303D EC <1> in al, dx

5920 0000303E 8844240D <1> mov [esp+13], al ; bh

5921 00003042 66BADA03 <1> mov dx, 3DAh ; VGAREG\_ACTL\_RESET

5922 00003046 EC <1> in al, dx

5923 00003047 66BAC003 <1> mov dx, 3C0h ; VGAREG\_ACTL\_ADDRESS

5924 0000304B B020 <1> mov al, 20h

5925 0000304D EE <1> out dx, al

5926 <1> ; ifdef VBOX

5927 0000304E 66BADA03 <1> mov dx, 3DAh ; VGAREG\_ACTL\_RESET

5928 00003052 EC <1> in al, dx

5929 <1> ; endif ; VBOX

5930 00003053 E9FCE4FFFF <1> jmp VIDEO\_RETURN

5931 <1>

5932 <1> read\_overscan\_border\_color:

5933 <1> ; 10/08/2016

5934 <1> ; Read Overscan Register

5935 <1> ; OUTPUT:

5936 <1> ; BH = current rgbRGB value

5937 <1> ; of the overscan/border register

5938 <1>

5939 00003058 B311 <1> mov bl, 11h

5940 0000305A EBC8 <1> jmp short get\_single\_palette\_reg

5941 <1>

5942 <1> get\_all\_palette\_reg:

5943 <1> ; 10/08/2016

5944 <1> ; Read All Palette Registers

5945 <1> ; EDX = Address of 17-byte buffer

5946 <1> ; to receive data

5947 <1>

5948 0000305C 89D7 <1> mov edi, edx

5949 0000305E 89E3 <1> mov ebx, esp

5950 00003060 89DE <1> mov esi, ebx

5951 00003062 83EC14 <1> sub esp, 20

5952 <1>

5953 00003065 B100 <1> mov cl, 0

5954 <1> get\_palette\_loop:

5955 00003067 66BADA03 <1> mov dx, 3DAh ; VGAREG\_ACTL\_RESET

5956 0000306B EC <1> in al, dx

5957 0000306C 66BAC003 <1> mov dx, 3C0h ; VGAREG\_ACTL\_ADDRESS

5958 00003070 88C8 <1> mov al, cl

5959 00003072 EE <1> out dx, al

5960 00003073 66BAC103 <1> mov dx, 3C1h ; VGAREG\_ACTL\_READ\_DATA

5961 00003077 EC <1> in al, dx

5962 00003078 8803 <1> mov [ebx], al

5963 0000307A 43 <1> inc ebx

5964 0000307B FEC1 <1> inc cl

5965 0000307D 80F910 <1> cmp cl, 10h

5966 00003080 75E5 <1> jne short get\_palette\_loop

5967 00003082 66BADA03 <1> mov dx, 3DAh ; VGAREG\_ACTL\_RESET

5968 00003086 EC <1> in al, dx

5969 00003087 66BAC003 <1> mov dx, 3C0h ; VGAREG\_ACTL\_ADDRESS

5970 0000308B B011 <1> mov al, 11h

5971 0000308D EE <1> out dx, al

5972 0000308E 66BAC103 <1> mov dx, 3C1h ; VGAREG\_ACTL\_READ\_DATA

5973 00003092 EC <1> in al, dx

5974 00003093 8803 <1> mov [ebx], al

5975 00003095 66BADA03 <1> mov dx, 3DAh ; VGAREG\_ACTL\_RESET

5976 00003099 EC <1> in al, dx

5977 0000309A 66BAC003 <1> mov dx, 3C0h ; VGAREG\_ACTL\_ADDRESS

5978 0000309E B020 <1> mov al, 20h

5979 000030A0 EE <1> out dx, al

5980 <1> ; ifdef VBOX

5981 000030A1 66BADA03 <1> mov dx, 3DAh ; VGAREG\_ACTL\_RESET

5982 000030A5 EC <1> in al, dx

5983 <1> ; endif ; VBOX

5984 <1>

5985 000030A6 B911000000 <1> mov ecx, 17 ; transfer (byte) count

5986 <1> ; ESI = source address in system space

5987 <1> ; EDI = user's buffer address

5988 000030AB E8C9B60000 <1> call transfer\_to\_user\_buffer

5989 <1>

5990 000030B0 83C414 <1> add esp, 20

5991 000030B3 E99CE4FFFF <1> jmp VIDEO\_RETURN

5992 <1>

5993 <1> set\_single\_dac\_reg:

5994 <1> ; 10/08/2016

5995 <1> ; Set One DAC Color Register

5996 <1> ; BX = color register to set (0-255)

5997 <1> ; CH = green value (00h-3Fh)

5998 <1> ; CL = blue value (00h-3Fh)

5999 <1> ; DH = red value (00h-3Fh)

6000 <1>

6001 000030B8 6652 <1> push dx

6002 000030BA 66BAC803 <1> mov dx, 3C8h ; VGAREG\_DAC\_WRITE\_ADDRESS

6003 000030BE 88D8 <1> mov al, bl

6004 000030C0 EE <1> out dx, al

6005 <1> ;mov dx, 3C9h ; VGAREG\_DAC\_DATA

6006 000030C1 6642 <1> inc dx

6007 000030C3 6658 <1> pop ax

6008 000030C5 88E0 <1> mov al, ah

6009 000030C7 EE <1> out dx, al

6010 000030C8 88E8 <1> mov al, ch

6011 000030CA EE <1> out dx, al

6012 000030CB 88C8 <1> mov al, cl

6013 000030CD EE <1> out dx, al

6014 000030CE E981E4FFFF <1> jmp VIDEO\_RETURN

6015 <1>

6016 <1> set\_all\_dac\_reg:

6017 <1> ; 12/08/2016

6018 <1> ; 11/08/2016

6019 <1> ; 10/08/2016

6020 <1> ; Set a Block of DAC Color Register

6021 <1> ; BX = first DAC register to set (0-00FFh)

6022 <1> ; ECX = number of registers to set (0-00FFh)

6023 <1> ; EDX = addr of a table of R,G,B values

6024 <1> ; (it will be CX\*3 bytes long)

6025 <1>

6026 000030D3 89D6 <1> mov esi, edx ; user buffer

6027 000030D5 89CA <1> mov edx, ecx

6028 000030D7 66D1E1 <1> shl cx, 1 ; \*2

6029 000030DA 01D1 <1> add ecx, edx ; ecx = 3\*ecx

6030 000030DC 89E5 <1> mov ebp, esp

6031 000030DE 89EF <1> mov edi, ebp

6032 000030E0 29CF <1> sub edi, ecx

6033 000030E2 6683E7FC <1> and di, 0FFFCh ; (dword alignment)

6034 000030E6 89FC <1> mov esp, edi

6035 000030E8 E8D6B60000 <1> call transfer\_from\_user\_buffer

6036 <1> ;jc VIDEO\_RETURN

6037 <1>

6038 000030ED 89D1 <1> mov ecx, edx

6039 000030EF 66BAC803 <1> mov dx, 3C8h ; VGAREG\_DAC\_WRITE\_ADDRESS

6040 000030F3 88D8 <1> mov al, bl

6041 000030F5 EE <1> out dx, al

6042 000030F6 66BAC903 <1> mov dx, 3C9h ; VGAREG\_DAC\_DATA

6043 <1> set\_dac\_loop:

6044 000030FA 8A07 <1> mov al, [edi]

6045 000030FC EE <1> out dx, al

6046 000030FD 47 <1> inc edi

6047 000030FE 8A07 <1> mov al, [edi]

6048 00003100 EE <1> out dx, al

6049 00003101 47 <1> inc edi

6050 00003102 8A07 <1> mov al, [edi]

6051 00003104 EE <1> out dx, al

6052 00003105 47 <1> inc edi

6053 00003106 6649 <1> dec cx

6054 00003108 75F0 <1> jnz short set\_dac\_loop

6055 0000310A 89EC <1> mov esp, ebp

6056 0000310C E943E4FFFF <1> jmp VIDEO\_RETURN

6057 <1>

6058 <1> select\_video\_dac\_color\_page:

6059 <1> ; 10/08/2016

6060 <1> ; DAC Color Paging Functions

6061 <1> ; BL = 00H = select color paging mode

6062 <1> ; BH = paging mode

6063 <1> ; 00h = 4 blocks of 64 registers

6064 <1> ; 01h = 16 blocks of 16 registers

6065 <1> ; BL = 01H = activate color page

6066 <1> ; BH = DAC color page number

6067 <1> ; 00h-03h (4-page/64-reg mode)

6068 <1> ; 00h-0Fh (16-page/16-reg mode)

6069 <1>

6070 00003111 66BADA03 <1> mov dx, 3DAh ; VGAREG\_ACTL\_RESET

6071 00003115 EC <1> in al, dx

6072 00003116 66BAC003 <1> mov dx, 3C0h ; VGAREG\_ACTL\_ADDRESS

6073 0000311A B010 <1> mov al, 10h

6074 0000311C EE <1> out dx, al

6075 0000311D 66BAC103 <1> mov dx, 3C1h ; VGAREG\_ACTL\_READ\_DATA

6076 00003121 EC <1> in al, dx

6077 00003122 80E301 <1> and bl, 01h

6078 00003125 750E <1> jnz short set\_dac\_page

6079 00003127 247F <1> and al, 07Fh

6080 00003129 C0E707 <1> shl bh, 7

6081 0000312C 08F8 <1> or al, bh

6082 0000312E 66BAC003 <1> mov dx, 3C0h ; VGAREG\_ACTL\_ADDRESS

6083 00003132 EE <1> out dx, al

6084 00003133 EB1D <1> jmp short set\_actl\_normal

6085 <1> set\_dac\_page:

6086 00003135 6650 <1> push ax

6087 00003137 66BADA03 <1> mov dx, 3DAh ; VGAREG\_ACTL\_RESET

6088 0000313B EC <1> in al, dx

6089 0000313C 66BAC003 <1> mov dx, 3C0h ; VGAREG\_ACTL\_ADDRESS

6090 00003140 B014 <1> mov al, 14h

6091 00003142 EE <1> out dx, al

6092 00003143 6658 <1> pop ax

6093 00003145 2480 <1> and al, 80h

6094 00003147 7503 <1> jnz short set\_dac\_16\_page

6095 00003149 C0E702 <1> shl bh, 2

6096 <1> set\_dac\_16\_page:

6097 0000314C 80E70F <1> and bh, 0Fh

6098 0000314F 88F8 <1> mov al, bh

6099 00003151 EE <1> out dx, al

6100 <1> set\_actl\_normal:

6101 00003152 B020 <1> mov al, 20h

6102 00003154 EE <1> out dx, al

6103 <1> ; ifdef VBOX

6104 00003155 66BADA03 <1> mov dx, 3DAh ; VGAREG\_ACTL\_RESET

6105 00003159 EC <1> in al, dx

6106 <1> ; endif ; VBOX

6107 0000315A E9F5E3FFFF <1> jmp VIDEO\_RETURN

6108 <1>

6109 <1> read\_single\_dac\_reg:

6110 <1> ; 10/08/2016

6111 <1> ; Read One DAC Color Register

6112 <1> ; INPUT:

6113 <1> ; BX = color register to read (0-255)

6114 <1> ; OUTPUT:

6115 <1> ; CH = green value (00h-3Fh)

6116 <1> ; CL = blue value (00h-3Fh)

6117 <1> ; DH = red value (00h-3Fh)

6118 <1>

6119 0000315F 66BAC703 <1> mov dx, 3C7h ; VGAREG\_DAC\_READ\_ADDRESS

6120 00003163 88D8 <1> mov al, bl

6121 00003165 EE <1> out dx, al

6122 00003166 66BAC903 <1> mov dx, 3C9h ; VGAREG\_DAC\_DATA

6123 0000316A EC <1> in al, dx

6124 0000316B 88442415 <1> mov [esp+21], al ; dh

6125 0000316F EC <1> in al, dx

6126 00003170 88C5 <1> mov ch, al

6127 00003172 EC <1> in al, dx

6128 00003173 88C1 <1> mov cl, al

6129 00003175 66894C2410 <1> mov [esp+16], cx ; cx

6130 0000317A E9D5E3FFFF <1> jmp VIDEO\_RETURN

6131 <1>

6132 <1> read\_all\_dac\_reg:

6133 <1> ; 12/08/2016

6134 <1> ; 11/08/2016

6135 <1> ; 10/08/2016

6136 <1> ; Read a Block of DAC Color Registers

6137 <1> ; BX = first DAC register to read (0-00FFh)

6138 <1> ; ECX = number of registers to read (0-00FFh)

6139 <1> ; EDX = addr of a buffer to hold R,G,B values

6140 <1> ; (CX\*3 bytes long)

6141 <1>

6142 0000317F 89D7 <1> mov edi, edx ; user buffer

6143 00003181 89CA <1> mov edx, ecx

6144 00003183 66D1E2 <1> shl dx, 1 ; \*2

6145 00003186 01CA <1> add edx, ecx ; edx = 3\*ecx

6146 00003188 89E5 <1> mov ebp, esp

6147 0000318A 89EE <1> mov esi, ebp

6148 0000318C 29D6 <1> sub esi, edx

6149 0000318E 6683E6FC <1> and si, 0FFFCh ; (dword alignment)

6150 00003192 89F4 <1> mov esp, esi

6151 00003194 52 <1> push edx ; 3\*ecx

6152 00003195 66BAC703 <1> mov dx, 3C7h ; VGAREG\_DAC\_READ\_ADDRESS

6153 00003199 88D8 <1> mov al, bl

6154 0000319B EE <1> out dx, al

6155 0000319C 66BAC903 <1> mov dx, 3C9h ; VGAREG\_DAC\_DATA

6156 000031A0 89F3 <1> mov ebx, esi

6157 <1> read\_dac\_loop:

6158 000031A2 EC <1> in al, dx

6159 000031A3 8803 <1> mov [ebx], al

6160 000031A5 43 <1> inc ebx

6161 000031A6 EC <1> in al, dx

6162 000031A7 8803 <1> mov [ebx], al

6163 000031A9 43 <1> inc ebx

6164 000031AA EC <1> in al, dx

6165 000031AB 8803 <1> mov [ebx], al

6166 000031AD 43 <1> inc ebx

6167 000031AE 6649 <1> dec cx

6168 000031B0 75F0 <1> jnz short read\_dac\_loop

6169 000031B2 59 <1> pop ecx ; 3\*ecx

6170 <1> ; ECX = transfer (byte) count

6171 <1> ; ESI = source address in system space

6172 <1> ; EDI = user's buffer address

6173 000031B3 E8C1B50000 <1> call transfer\_to\_user\_buffer

6174 000031B8 89EC <1> mov esp, ebp

6175 000031BA E995E3FFFF <1> jmp VIDEO\_RETURN

6176 <1>

6177 <1> set\_pel\_mask:

6178 <1> ; 10/08/2016

6179 <1> ; BL = mask value

6180 000031BF 66BAC603 <1> mov dx, 3C6h ; VGAREG\_PEL\_MASK

6181 000031C3 88D8 <1> mov al, bl

6182 000031C5 EE <1> out dx, al

6183 000031C6 E989E3FFFF <1> jmp VIDEO\_RETURN

6184 <1>

6185 <1> read\_pel\_mask:

6186 <1> ; 10/08/2016

6187 <1> ; Output: BL = mask value

6188 000031CB 66BAC603 <1> mov dx, 3C6h ; VGAREG\_PEL\_MASK

6189 000031CF EC <1> in al, dx

6190 000031D0 8844240C <1> mov [esp+12], al ; bl

6191 000031D4 E97BE3FFFF <1> jmp VIDEO\_RETURN

6192 <1>

6193 <1> read\_video\_dac\_state:

6194 <1> ; 10/08/2016

6195 <1> ; Query DAC Color Paging State

6196 <1> ; Output:

6197 <1> ; BH = current active DAC color page

6198 <1> ; BL = current active DAC paging mode

6199 <1>

6200 000031D9 66BADA03 <1> mov dx, 3DAh ; VGAREG\_ACTL\_RESET

6201 000031DD EC <1> in al, dx

6202 000031DE 66BAC003 <1> mov dx, 3C0h ; VGAREG\_ACTL\_ADDRESS

6203 000031E2 B010 <1> mov al, 10h

6204 000031E4 EE <1> out dx, al

6205 000031E5 66BAC103 <1> mov dx, 3C1h ; VGAREG\_ACTL\_READ\_DATA

6206 000031E9 EC <1> in al, dx

6207 000031EA 88C3 <1> mov bl, al

6208 000031EC C0EB07 <1> shr bl, 7

6209 000031EF 66BADA03 <1> mov dx, 3DAh ; VGAREG\_ACTL\_RESET

6210 000031F3 EC <1> in al, dx

6211 000031F4 66BAC003 <1> mov dx, 3C0h ; VGAREG\_ACTL\_ADDRESS

6212 000031F8 B014 <1> mov al, 14h

6213 000031FA EE <1> out dx, al

6214 000031FB 66BAC103 <1> mov dx, 3C1h ; VGAREG\_ACTL\_READ\_DATA

6215 000031FF EC <1> in al, dx

6216 00003200 88C7 <1> mov bh, al

6217 00003202 80E70F <1> and bh, 0Fh

6218 00003205 F6C301 <1> test bl, 01

6219 00003208 7503 <1> jnz short get\_dac\_16\_page

6220 0000320A C0EF02 <1> shr bh, 2

6221 <1> get\_dac\_16\_page:

6222 0000320D 66BADA03 <1> mov dx, 3DAh ; VGAREG\_ACTL\_RESET

6223 00003211 EC <1> in al, dx

6224 00003212 66BAC003 <1> mov dx, 3C0h ; VGAREG\_ACTL\_ADDRESS

6225 00003216 B020 <1> mov al, 20h

6226 00003218 EE <1> out dx, al

6227 <1> ; ifdef VBOX

6228 00003219 66BADA03 <1> mov dx, 3DAh ; VGAREG\_ACTL\_RESET

6229 0000321D EC <1> in al, dx

6230 <1> ; endif ; VBOX

6231 0000321E 66895C240C <1> mov [esp+12], bx ; bx

6232 00003223 E92CE3FFFF <1> jmp VIDEO\_RETURN

6233 <1>

6234 <1> ; % include 'vidata.s' ; VIDEO DATA

6235 <1>

6236 <1> ; /// End Of VIDEO FUNCTIONS ///

1942

1943 setup\_rtc\_int:

1944 ; source: http://wiki.osdev.org/RTC

1945 00003228 FA cli ; disable interrupts

1946 ; default int frequency is 1024 Hz (Lower 4 bits of register A is 0110b or 6)

1947 ; in order to change this ...

1948 ; frequency = 32768 >> (rate-1) --> 32768 >> 5 = 1024

1949 ; (rate must be above 2 and not over 15)

1950 ; new rate = 15 --> 32768 >> (15-1) = 2 Hz

1951 00003229 B08A mov al, 8Ah

1952 0000322B E670 out 70h, al ; set index to register A, disable NMI

1953 0000322D 90 nop

1954 0000322E E471 in al, 71h ; get initial value of register A

1955 00003230 88C4 mov ah, al

1956 00003232 80E4F0 and ah, 0F0h

1957 00003235 B08A mov al, 8Ah

1958 00003237 E670 out 70h, al ; reset index to register A

1959 00003239 88E0 mov al, ah

1960 0000323B 0C0F or al, 0Fh ; new rate (0Fh -> 15)

1961 0000323D E671 out 71h, al ; write only our rate to A. Note, rate is the bottom 4 bits.

1962 ; enable RTC interrupt

1963 0000323F B08B mov al, 8Bh ;

1964 00003241 E670 out 70h, al ; select register B and disable NMI

1965 00003243 90 nop

1966 00003244 E471 in al, 71h ; read the current value of register B

1967 00003246 88C4 mov ah, al ;

1968 00003248 B08B mov al, 8Bh ;

1969 0000324A E670 out 70h, al ; set the index again (a read will reset the index to register B)

1970 0000324C 88E0 mov al, ah ;

1971 0000324E 0C40 or al, 40h ;

1972 00003250 E671 out 71h, al ; write the previous value ORed with 0x40. This turns on bit 6 of register B

1973 00003252 FB sti

1974 00003253 C3 retn

1975

1976 ; Write memory information

1977 ; 29/01/2016

1978 ; 06/11/2014

1979 ; 14/08/2015

1980 memory\_info:

1981 00003254 A1[3C580100] mov eax, [memory\_size] ; in pages

1982 00003259 50 push eax

1983 0000325A C1E00C shl eax, 12 ; in bytes

1984 0000325D BB0A000000 mov ebx, 10

1985 00003262 89D9 mov ecx, ebx ; 10

1986 00003264 BE[C9180100] mov esi, mem\_total\_b\_str

1987 00003269 E8BD000000 call bintdstr

1988 0000326E 58 pop eax

1989 0000326F B107 mov cl, 7

1990 00003271 BE[ED180100] mov esi, mem\_total\_p\_str

1991 00003276 E8B0000000 call bintdstr

1992 ; 14/08/2015

1993 0000327B E8C8000000 call calc\_free\_mem

1994 ; edx = calculated free pages

1995 ; ecx = 0

1996 00003280 A1[40580100] mov eax, [free\_pages]

1997 00003285 39D0 cmp eax, edx ; calculated free mem value

1998 ; and initial free mem value are same or not?

1999 00003287 751D jne short pmim ; print mem info with '?' if not

2000 00003289 52 push edx ; free memory in pages

2001 ;mov eax, edx

2002 0000328A C1E00C shl eax, 12 ; convert page count

2003 ; to byte count

2004 0000328D B10A mov cl, 10

2005 0000328F BE[0D190100] mov esi, free\_mem\_b\_str

2006 00003294 E892000000 call bintdstr

2007 00003299 58 pop eax

2008 0000329A B107 mov cl, 7

2009 0000329C BE[31190100] mov esi, free\_mem\_p\_str

2010 000032A1 E885000000 call bintdstr

2011 pmim:

2012 000032A6 BE[B7180100] mov esi, msg\_memory\_info

2013 ;

2014 000032AB B407 mov ah, 07h ; Black background,

2015 ; light gray forecolor

2016 print\_kmsg: ; 29/01/2016

2017 000032AD 8825[67580100] mov [ccolor], ah

2018 pkmsg\_loop:

2019 000032B3 AC lodsb

2020 000032B4 08C0 or al, al

2021 000032B6 7410 jz short pkmsg\_ok

2022 000032B8 56 push esi

2023 ; 13/05/2016

2024 000032B9 0FB61D[67580100] movzx ebx, byte [ccolor]

2025 ; Video page 0 (bh=0)

2026 000032C0 E8EDE9FFFF call \_write\_tty

2027 000032C5 5E pop esi

2028 000032C6 EBEB jmp short pkmsg\_loop

2029 pkmsg\_ok:

2030 000032C8 C3 retn

2031

2032 ; Convert binary number to hexadecimal string

2033 ; 10/05/2015

2034 ; dsectpm.s (28/02/2015)

2035 ; Retro UNIX 386 v1 - Kernel v0.2.0.6

2036 ; 01/12/2014

2037 ; 25/11/2014

2038 ;

2039 bytetohex:

2040 ; INPUT ->

2041 ; AL = byte (binary number)

2042 ; OUTPUT ->

2043 ; AX = hexadecimal string

2044 ;

2045 000032C9 53 push ebx

2046 000032CA 31DB xor ebx, ebx

2047 000032CC 88C3 mov bl, al

2048 000032CE C0EB04 shr bl, 4

2049 000032D1 8A9B[1B330000] mov bl, [ebx+hexchrs]

2050 000032D7 86D8 xchg bl, al

2051 000032D9 80E30F and bl, 0Fh

2052 000032DC 8AA3[1B330000] mov ah, [ebx+hexchrs]

2053 000032E2 5B pop ebx

2054 000032E3 C3 retn

2055

2056 wordtohex:

2057 ; INPUT ->

2058 ; AX = word (binary number)

2059 ; OUTPUT ->

2060 ; EAX = hexadecimal string

2061 ;

2062 000032E4 53 push ebx

2063 000032E5 31DB xor ebx, ebx

2064 000032E7 86E0 xchg ah, al

2065 000032E9 6650 push ax

2066 000032EB 88E3 mov bl, ah

2067 000032ED C0EB04 shr bl, 4

2068 000032F0 8A83[1B330000] mov al, [ebx+hexchrs]

2069 000032F6 88E3 mov bl, ah

2070 000032F8 80E30F and bl, 0Fh

2071 000032FB 8AA3[1B330000] mov ah, [ebx+hexchrs]

2072 00003301 C1E010 shl eax, 16

2073 00003304 6658 pop ax

2074 00003306 5B pop ebx

2075 00003307 EBC0 jmp short bytetohex

2076 ;mov bl, al

2077 ;shr bl, 4

2078 ;mov bl, [ebx+hexchrs]

2079 ;xchg bl, al

2080 ;and bl, 0Fh

2081 ;mov ah, [ebx+hexchrs]

2082 ;pop ebx

2083 ;retn

2084

2085 dwordtohex:

2086 ; INPUT ->

2087 ; EAX = dword (binary number)

2088 ; OUTPUT ->

2089 ; EDX:EAX = hexadecimal string

2090 ;

2091 00003309 50 push eax

2092 0000330A C1E810 shr eax, 16

2093 0000330D E8D2FFFFFF call wordtohex

2094 00003312 89C2 mov edx, eax

2095 00003314 58 pop eax

2096 00003315 E8CAFFFFFF call wordtohex

2097 0000331A C3 retn

2098

2099 ; 10/05/2015

2100 hex\_digits:

2101 hexchrs:

2102 0000331B 303132333435363738- db '0123456789ABCDEF'

2102 00003324 39414243444546

2103

2104 ; Convert binary number to decimal/numeric string

2105 ; 06/11/2014

2106 ; Temporary Code

2107 ;

2108

2109 bintdstr:

2110 ; EAX = binary number

2111 ; ESI = decimal/numeric string address

2112 ; EBX = divisor (10)

2113 ; ECX = string length (<=10)

2114 0000332B 01CE add esi, ecx

2115 btdstr0:

2116 0000332D 4E dec esi

2117 0000332E 31D2 xor edx, edx

2118 00003330 F7F3 div ebx

2119 00003332 80C230 add dl, 30h

2120 00003335 8816 mov [esi], dl

2121 00003337 FEC9 dec cl

2122 00003339 740C jz short btdstr2 ; 08/09/2016

2123 0000333B 09C0 or eax, eax

2124 0000333D 75EE jnz short btdstr0

2125 btdstr1:

2126 0000333F 4E dec esi

2127 00003340 C60620 mov byte [esi], 20h ; blank space

2128 00003343 FEC9 dec cl

2129 00003345 75F8 jnz short btdstr1

2130 btdstr2:

2131 00003347 C3 retn

2132

2133 ; Calculate free memory pages on M.A.T.

2134 ; 06/11/2014

2135 ; Temporary Code

2136 ;

2137

2138 calc\_free\_mem:

2139 00003348 31D2 xor edx, edx

2140 ;xor ecx, ecx

2141 0000334A 668B0D[50580100] mov cx, [mat\_size] ; in pages

2142 00003351 C1E10A shl ecx, 10 ; 1024 dwords per page

2143 00003354 BE00001000 mov esi, MEM\_ALLOC\_TBL

2144 cfm0:

2145 00003359 AD lodsd

2146 0000335A 51 push ecx

2147 0000335B B920000000 mov ecx, 32

2148 cfm1:

2149 00003360 D1E8 shr eax, 1

2150 00003362 7301 jnc short cfm2

2151 00003364 42 inc edx

2152 cfm2:

2153 00003365 E2F9 loop cfm1

2154 00003367 59 pop ecx

2155 00003368 E2EF loop cfm0

2156 0000336A C3 retn

2157

2158 %include 'diskio.s' ; 07/03/2015

1 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2 <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0 - diskio.s

3 <1> ; ----------------------------------------------------------------------------

4 <1> ; Last Update: 09/12/2017

5 <1> ; ----------------------------------------------------------------------------

6 <1> ; Beginning: 24/01/2016

7 <1> ; ----------------------------------------------------------------------------

8 <1> ; Assembler: NASM version 2.11 (trdos386.s)

9 <1> ; ----------------------------------------------------------------------------

10 <1> ; Turkish Rational DOS

11 <1> ; Operating System Project v2.0 by ERDOGAN TAN (Beginning: 04/01/2016)

12 <1> ;

13 <1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan

14 <1> ; diskio.inc (22/08/2015)

15 <1> ;

16 <1> ; Derived from 'IBM PC-XT-286' BIOS source code (1986)

17 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

18 <1>

19 <1> ; Retro UNIX 386 v1 Kernel - DISKIO.INC

20 <1> ; Last Modification: 22/08/2015

21 <1> ; (Initialized Disk Parameters Data is in 'DISKDATA.INC')

22 <1> ; (Uninitialized Disk Parameters Data is in 'DISKBSS.INC')

23 <1>

24 <1> ; DISK I/O SYSTEM - Erdogan Tan (Retro UNIX 386 v1 project)

25 <1>

26 <1> ; ///////// DISK I/O SYSTEM ///////////////

27 <1>

28 <1> ; 06/02/2015

29 <1> diskette\_io:

30 0000336B 9C <1> pushfd

31 0000336C 0E <1> push cs

32 0000336D E809000000 <1> call DISKETTE\_IO\_1

33 00003372 C3 <1> retn

34 <1>

35 <1> ;;;;;; DISKETTE I/O ;;;;;;;;;;;;;;;;;;;; 06/02/2015 ;;;

36 <1> ;//////////////////////////////////////////////////////

37 <1>

38 <1> ; DISKETTE I/O - Erdogan Tan (Retro UNIX 386 v1 project)

39 <1> ; 20/02/2015

40 <1> ; 06/02/2015 (unix386.s)

41 <1> ; 16/12/2014 - 02/01/2015 (dsectrm2.s)

42 <1> ;

43 <1> ; Code (DELAY) modifications - AWARD BIOS 1999 (ADISK.EQU, COMMON.MAC)

44 <1> ;

45 <1> ; ADISK.EQU

46 <1>

47 <1> ;----- Wait control constants

48 <1>

49 <1> ;amount of time to wait while RESET is active.

50 <1>

51 <1> WAITCPU\_RESET\_ON EQU 21 ;Reset on must last at least 14us

52 <1> ;at 250 KBS xfer rate.

53 <1> ;see INTEL MCS, 1985, pg. 5-456

54 <1>

55 <1> WAITCPU\_FOR\_STATUS EQU 100 ;allow 30 microseconds for

56 <1> ;status register to become valid

57 <1> ;before re-reading.

58 <1>

59 <1> ;After sending a byte to NEC, status register may remain

60 <1> ;incorrectly set for 24 us.

61 <1>

62 <1> WAITCPU\_RQM\_LOW EQU 24 ;number of loops to check for

63 <1> ;RQM low.

64 <1>

65 <1> ; COMMON.MAC

66 <1> ;

67 <1> ; Timing macros

68 <1> ;

69 <1>

70 <1> %macro SIODELAY 0 ; SHORT IODELAY

71 <1> jmp short $+2

72 <1> %endmacro

73 <1>

74 <1> %macro IODELAY 0 ; NORMAL IODELAY

75 <1> jmp short $+2

76 <1> jmp short $+2

77 <1> %endmacro

78 <1>

79 <1> %macro NEWIODELAY 0

80 <1> out 0ebh,al

81 <1> %endmacro

82 <1>

83 <1> ; (According to) AWARD BIOS 1999 - ATORGS.ASM (dw -> equ, db -> equ)

84 <1> ;;; WAIT\_FOR\_MEM

85 <1> ;WAIT\_FDU\_INT\_LO equ 017798 ; 2.5 secs in 30 micro units.

86 <1> ;WAIT\_FDU\_INT\_HI equ 1

87 <1> WAIT\_FDU\_INT\_LH equ 83334 ; 27/02/2015 (2.5 seconds waiting)

88 <1> ;;; WAIT\_FOR\_PORT

89 <1> ;WAIT\_FDU\_SEND\_LO equ 16667 ; .5 secons in 30 us units.

90 <1> ;WAIT\_FDU\_SEND\_HI equ 0

91 <1> WAIT\_FDU\_SEND\_LH equ 16667 ; 27/02/2015

92 <1> ;Time to wait while waiting for each byte of NEC results = .5

93 <1> ;seconds. .5 seconds = 500,000 micros. 500,000/30 = 16,667.

94 <1> ;WAIT\_FDU\_RESULTS\_LO equ 16667 ; .5 seconds in 30 micro units.

95 <1> ;WAIT\_FDU\_RESULTS\_HI equ 0

96 <1> WAIT\_FDU\_RESULTS\_LH equ 16667 ; 27/02/2015

97 <1> ;;; WAIT\_REFRESH

98 <1> ;amount of time to wait for head settle, per unit in parameter

99 <1> ;table = 1 ms.

100 <1> WAIT\_FDU\_HEAD\_SETTLE equ 33 ; 1 ms in 30 micro units.

101 <1>

102 <1>

103 <1> ; //////////////// DISKETTE I/O ////////////////

104 <1>

105 <1> ; 11/12/2014 (copy from IBM PC-XT Model 286 BIOS - POSTEQU.INC)

106 <1>

107 <1> ;----------------------------------------

108 <1> ; EQUATES USED BY POST AND BIOS :

109 <1> ;----------------------------------------

110 <1>

111 <1> ;--------- 8042 KEYBOARD INTERFACE AND DIAGNOSTIC CONTROL REGISTERS ------------

112 <1> ;PORT\_A EQU 060H ; 8042 KEYBOARD SCAN CODE/CONTROL PORT

113 <1> ;PORT\_B EQU 061H ; PORT B READ/WRITE DIAGNOSTIC REGISTER

114 <1> ;REFRESH\_BIT EQU 00010000B ; REFRESH TEST BIT

115 <1>

116 <1> ;----------------------------------------

117 <1> ; CMOS EQUATES FOR THIS SYSTEM :

118 <1> ;-------------------------------------------------------------------------------

119 <1> ;CMOS\_PORT EQU 070H ; I/O ADDRESS OF CMOS ADDRESS PORT

120 <1> ;CMOS\_DATA EQU 071H ; I/O ADDRESS OF CMOS DATA PORT

121 <1> ;NMI EQU 10000000B ; DISABLE NMI INTERRUPTS MASK -

122 <1> ; HIGH BIT OF CMOS LOCATION ADDRESS

123 <1>

124 <1> ;---------- CMOS TABLE LOCATION ADDRESS'S ## -----------------------------------

125 <1> CMOS\_DISKETTE EQU 010H ; DISKETTE DRIVE TYPE BYTE ;

126 <1> ; EQU 011H ; - RESERVED ;C

127 <1> CMOS\_DISK EQU 012H ; FIXED DISK TYPE BYTE ;H

128 <1> ; EQU 013H ; - RESERVED ;E

129 <1> CMOS\_EQUIP EQU 014H ; EQUIPMENT WORD LOW BYTE ;C

130 <1>

131 <1> ;---------- DISKETTE EQUATES ---------------------------------------------------

132 <1> INT\_FLAG EQU 10000000B ; INTERRUPT OCCURRENCE FLAG

133 <1> DSK\_CHG EQU 10000000B ; DISKETTE CHANGE FLAG MASK BIT

134 <1> DETERMINED EQU 00010000B ; SET STATE DETERMINED IN STATE BITS

135 <1> HOME EQU 00010000B ; TRACK 0 MASK

136 <1> SENSE\_DRV\_ST EQU 00000100B ; SENSE DRIVE STATUS COMMAND

137 <1> TRK\_SLAP EQU 030H ; CRASH STOP (48 TPI DRIVES)

138 <1> QUIET\_SEEK EQU 00AH ; SEEK TO TRACK 10

139 <1> ;MAX\_DRV EQU 2 ; MAX NUMBER OF DRIVES

140 <1> HD12\_SETTLE EQU 15 ; 1.2 M HEAD SETTLE TIME

141 <1> HD320\_SETTLE EQU 20 ; 320 K HEAD SETTLE TIME

142 <1> MOTOR\_WAIT EQU 37 ; 2 SECONDS OF COUNTS FOR MOTOR TURN OFF

143 <1>

144 <1> ;---------- DISKETTE ERRORS ----------------------------------------------------

145 <1> ;TIME\_OUT EQU 080H ; ATTACHMENT FAILED TO RESPOND

146 <1> ;BAD\_SEEK EQU 040H ; SEEK OPERATION FAILED

147 <1> BAD\_NEC EQU 020H ; DISKETTE CONTROLLER HAS FAILED

148 <1> BAD\_CRC EQU 010H ; BAD CRC ON DISKETTE READ

149 <1> MED\_NOT\_FND EQU 00CH ; MEDIA TYPE NOT FOUND

150 <1> DMA\_BOUNDARY EQU 009H ; ATTEMPT TO DMA ACROSS 64K BOUNDARY

151 <1> BAD\_DMA EQU 008H ; DMA OVERRUN ON OPERATION

152 <1> MEDIA\_CHANGE EQU 006H ; MEDIA REMOVED ON DUAL ATTACH CARD

153 <1> RECORD\_NOT\_FND EQU 004H ; REQUESTED SECTOR NOT FOUND

154 <1> WRITE\_PROTECT EQU 003H ; WRITE ATTEMPTED ON WRITE PROTECT DISK

155 <1> BAD\_ADDR\_MARK EQU 002H ; ADDRESS MARK NOT FOUND

156 <1> BAD\_CMD EQU 001H ; BAD COMMAND PASSED TO DISKETTE I/O

157 <1>

158 <1> ;---------- DISK CHANGE LINE EQUATES -------------------------------------------

159 <1> NOCHGLN EQU 001H ; NO DISK CHANGE LINE AVAILABLE

160 <1> CHGLN EQU 002H ; DISK CHANGE LINE AVAILABLE

161 <1>

162 <1> ;---------- MEDIA/DRIVE STATE INDICATORS ---------------------------------------

163 <1> TRK\_CAPA EQU 00000001B ; 80 TRACK CAPABILITY

164 <1> FMT\_CAPA EQU 00000010B ; MULTIPLE FORMAT CAPABILITY (1.2M)

165 <1> DRV\_DET EQU 00000100B ; DRIVE DETERMINED

166 <1> MED\_DET EQU 00010000B ; MEDIA DETERMINED BIT

167 <1> DBL\_STEP EQU 00100000B ; DOUBLE STEP BIT

168 <1> RATE\_MSK EQU 11000000B ; MASK FOR CLEARING ALL BUT RATE

169 <1> RATE\_500 EQU 00000000B ; 500 KBS DATA RATE

170 <1> RATE\_300 EQU 01000000B ; 300 KBS DATA RATE

171 <1> RATE\_250 EQU 10000000B ; 250 KBS DATA RATE

172 <1> STRT\_MSK EQU 00001100B ; OPERATION START RATE MASK

173 <1> SEND\_MSK EQU 11000000B ; MASK FOR SEND RATE BITS

174 <1>

175 <1> ;---------- MEDIA/DRIVE STATE INDICATORS COMPATIBILITY -------------------------

176 <1> M3D3U EQU 00000000B ; 360 MEDIA/DRIVE NOT ESTABLISHED

177 <1> M3D1U EQU 00000001B ; 360 MEDIA,1.2DRIVE NOT ESTABLISHED

178 <1> M1D1U EQU 00000010B ; 1.2 MEDIA/DRIVE NOT ESTABLISHED

179 <1> MED\_UNK EQU 00000111B ; NONE OF THE ABOVE

180 <1>

181 <1> ;---------- INTERRUPT EQUATES --------------------------------------------------

182 <1> ;EOI EQU 020H ; END OF INTERRUPT COMMAND TO 8259

183 <1> ;INTA00 EQU 020H ; 8259 PORT

184 <1> INTA01 EQU 021H ; 8259 PORT

185 <1> INTB00 EQU 0A0H ; 2ND 8259

186 <1> INTB01 EQU 0A1H ;

187 <1>

188 <1> ;-------------------------------------------------------------------------------

189 <1> DMA08 EQU 008H ; DMA STATUS REGISTER PORT ADDRESS

190 <1> DMA EQU 000H ; DMA CH.0 ADDRESS REGISTER PORT ADDRESS

191 <1> DMA18 EQU 0D0H ; 2ND DMA STATUS PORT ADDRESS

192 <1> DMA1 EQU 0C0H ; 2ND DMA CH.0 ADDRESS REGISTER ADDRESS

193 <1> ;-------------------------------------------------------------------------------

194 <1> ;TIMER EQU 040H ; 8254 TIMER - BASE ADDRESS

195 <1>

196 <1> ;-------------------------------------------------------------------------------

197 <1> DMA\_PAGE EQU 081H ; START OF DMA PAGE REGISTERS

198 <1>

199 <1> ; 06/02/2015 (unix386.s, protected mode modifications)

200 <1> ; (unix386.s <-- dsectrm2.s)

201 <1> ; 11/12/2014 (copy from IBM PC-XT Model 286 BIOS - DSEG.INC)

202 <1>

203 <1> ; 27/05/2016 - TRDOS 386 (TRDOS v2.0)

204 <1> ; 10/12/2014

205 <1> ;

206 <1> ;int40h:

207 <1> ; pushf

208 <1> ; push cs

209 <1> ; ;cli

210 <1> ; call DISKETTE\_IO\_1

211 <1> ; retn

212 <1>

213 <1> ; DSKETTE ----- 04/21/86 DISKETTE BIOS

214 <1> ; (IBM PC XT Model 286 System BIOS Source Code, 04-21-86)

215 <1> ;

216 <1>

217 <1> ;-- INT13H ---------------------------------------------------------------------

218 <1> ; DISKETTE I/O

219 <1> ; THIS INTERFACE PROVIDES ACCESS TO THE 5 1/4 INCH 360 KB,

220 <1> ; 1.2 MB, 720 KB AND 1.44 MB DISKETTE DRIVES.

221 <1> ; INPUT

222 <1> ; (AH) = 00H RESET DISKETTE SYSTEM

223 <1> ; HARD RESET TO NEC, PREPARE COMMAND, RECALIBRATE REQUIRED

224 <1> ; ON ALL DRIVES

225 <1> ;-------------------------------------------------------------------------------

226 <1> ; (AH)= 01H READ THE STATUS OF THE SYSTEM INTO (AH)

227 <1> ; @DISKETTE\_STATUS FROM LAST OPERATION IS USED

228 <1> ;-------------------------------------------------------------------------------

229 <1> ; REGISTERS FOR READ/WRITE/VERIFY/FORMAT

230 <1> ; (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)

231 <1> ; (DH) - HEAD NUMBER (0-1 ALLOWED, NOT VALUE CHECKED)

232 <1> ; (CH) - TRACK NUMBER (NOT VALUE CHECKED)

233 <1> ; MEDIA DRIVE TRACK NUMBER

234 <1> ; 320/360 320/360 0-39

235 <1> ; 320/360 1.2M 0-39

236 <1> ; 1.2M 1.2M 0-79

237 <1> ; 720K 720K 0-79

238 <1> ; 1.44M 1.44M 0-79

239 <1> ; (CL) - SECTOR NUMBER (NOT VALUE CHECKED, NOT USED FOR FORMAT)

240 <1> ; MEDIA DRIVE SECTOR NUMBER

241 <1> ; 320/360 320/360 1-8/9

242 <1> ; 320/360 1.2M 1-8/9

243 <1> ; 1.2M 1.2M 1-15

244 <1> ; 720K 720K 1-9

245 <1> ; 1.44M 1.44M 1-18

246 <1> ; (AL) NUMBER OF SECTORS (NOT VALUE CHECKED)

247 <1> ; MEDIA DRIVE MAX NUMBER OF SECTORS

248 <1> ; 320/360 320/360 8/9

249 <1> ; 320/360 1.2M 8/9

250 <1> ; 1.2M 1.2M 15

251 <1> ; 720K 720K 9

252 <1> ; 1.44M 1.44M 18

253 <1> ;

254 <1> ; (ES:BX) - ADDRESS OF BUFFER (NOT REQUIRED FOR VERIFY)

255 <1> ;

256 <1> ;-------------------------------------------------------------------------------

257 <1> ; (AH)= 02H READ THE DESIRED SECTORS INTO MEMORY

258 <1> ;-------------------------------------------------------------------------------

259 <1> ; (AH)= 03H WRITE THE DESIRED SECTORS FROM MEMORY

260 <1> ;-------------------------------------------------------------------------------

261 <1> ; (AH)= 04H VERIFY THE DESIRED SECTORS

262 <1> ;-------------------------------------------------------------------------------

263 <1> ; (AH)= 05H FORMAT THE DESIRED TRACK

264 <1> ; (ES,BX) MUST POINT TO THE COLLECTION OF DESIRED ADDRESS FIELDS

265 <1> ; FOR THE TRACK. EACH FIELD IS COMPOSED OF 4 BYTES, (C,H,R,N),

266 <1> ; WHERE C = TRACK NUMBER, H=HEAD NUMBER, R = SECTOR NUMBER,

267 <1> ; N= NUMBER OF BYTES PER SECTOR (00=128,01=256,02=512,03=1024),

268 <1> ; THERE MUST BE ONE ENTRY FOR EVERY SECTOR ON THE TRACK.

269 <1> ; THIS INFORMATION IS USED TO FIND THE REQUESTED SECTOR DURING

270 <1> ; READ/WRITE ACCESS.

271 <1> ; PRIOR TO FORMATTING A DISKETTE, IF THERE EXISTS MORE THAN

272 <1> ; ONE SUPPORTED MEDIA FORMAT TYPE WITHIN THE DRIVE IN QUESTION,

273 <1> ; THEN "SET DASD TYPE" (INT 13H, AH = 17H) OR 'SET MEDIA TYPE'

274 <1> ; (INT 13H, AH = 18H) MUST BE CALLED TO SET THE DISKETTE TYPE

275 <1> ; THAT IS TO BE FORMATTED. IF "SET DASD TYPE" OR "SET MEDIA TYPE"

276 <1> ; IS NOT CALLED, THE FORMAT ROUTINE WILL ASSUME THE

277 <1> ; MEDIA FORMAT TO BE THE MAXIMUM CAPACITY OF THE DRIVE.

278 <1> ;

279 <1> ; THESE PARAMETERS OF DISK BASE MUST BE CHANGED IN ORDER TO

280 <1> ; FORMAT THE FOLLOWING MEDIAS:

281 <1> ; ---------------------------------------------

282 <1> ; : MEDIA : DRIVE : PARM 1 : PARM 2 :

283 <1> ; ---------------------------------------------

284 <1> ; : 320K : 320K/360K/1.2M : 50H : 8 :

285 <1> ; : 360K : 320K/360K/1.2M : 50H : 9 :

286 <1> ; : 1.2M : 1.2M : 54H : 15 :

287 <1> ; : 720K : 720K/1.44M : 50H : 9 :

288 <1> ; : 1.44M : 1.44M : 6CH : 18 :

289 <1> ; ---------------------------------------------

290 <1> ; NOTES: - PARM 1 = GAP LENGTH FOR FORMAT

291 <1> ; - PARM 2 = EOT (LAST SECTOR ON TRACK)

292 <1> ; - DISK BASE IS POINTED BY DISK POINTER LOCATED

293 <1> ; AT ABSOLUTE ADDRESS 0:78.

294 <1> ; - WHEN FORMAT OPERATIONS ARE COMPLETE, THE PARAMETERS

295 <1> ; SHOULD BE RESTORED TO THEIR RESPECTIVE INITIAL VALUES.

296 <1> ;-------------------------------------------------------------------------------

297 <1> ; (AH) = 08H READ DRIVE PARAMETERS

298 <1> ; REGISTERS

299 <1> ; INPUT

300 <1> ; (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)

301 <1> ; \*\* 27/05/2016 - TRDOS 386 (TRDOS v2.0) \*\*

302 <1> ; \*\* EBX = Buffer address for floppy disk parameters table \*\*

303 <1> ; OUTPUT

304 <1> ; (ES:DI) POINTS TO DRIVE PARAMETER TABLE

305 <1> ; \*\*\* TRDOS 386 note: floppy disk parameter table (16 bytes)

306 <1> ; will be returned to user in EBX, buffer address \*\*\* 27/05/2016 \*\*\*

307 <1> ;

308 <1> ; (CH) - LOW ORDER 8 OF 10 BITS MAXIMUM NUMBER OF TRACKS

309 <1> ; (CL) - BITS 7 & 6 - HIGH ORDER TWO BITS OF MAXIMUM TRACKS

310 <1> ; BITS 5 THRU 0 - MAXIMUM SECTORS PER TRACK

311 <1> ; (DH) - MAXIMUM HEAD NUMBER

312 <1> ; (DL) - NUMBER OF DISKETTE DRIVES INSTALLED

313 <1> ; (BH) - 0

314 <1> ; (BL) - BITS 7 THRU 4 - 0

315 <1> ; BITS 3 THRU 0 - VALID DRIVE TYPE VALUE IN CMOS

316 <1> ; (AX) - 0

317 <1> ; UNDER THE FOLLOWING CIRCUMSTANCES:

318 <1> ; (1) THE DRIVE NUMBER IS INVALID,

319 <1> ; (2) THE DRIVE TYPE IS UNKNOWN AND CMOS IS NOT PRESENT,

320 <1> ; (3) THE DRIVE TYPE IS UNKNOWN AND CMOS IS BAD,

321 <1> ; (4) OR THE DRIVE TYPE IS UNKNOWN AND THE CMOS DRIVE TYPE IS INVALID

322 <1> ; THEN ES,AX,BX,CX,DH,DI=0 ; DL=NUMBER OF DRIVES.

323 <1> ; IF NO DRIVES ARE PRESENT THEN: ES,AX,BX,CX,DX,DI=0.

324 <1> ; @DISKETTE\_STATUS = 0 AND CY IS RESET.

325 <1> ;-------------------------------------------------------------------------------

326 <1> ; (AH)= 15H READ DASD TYPE

327 <1> ; OUTPUT REGISTERS

328 <1> ; (AH) - ON RETURN IF CARRY FLAG NOT SET, OTHERWISE ERROR

329 <1> ; 00 - DRIVE NOT PRESENT

330 <1> ; 01 - DISKETTE, NO CHANGE LINE AVAILABLE

331 <1> ; 02 - DISKETTE, CHANGE LINE AVAILABLE

332 <1> ; 03 - RESERVED (FIXED DISK)

333 <1> ; (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)

334 <1> ;-------------------------------------------------------------------------------

335 <1> ; (AH)= 16H DISK CHANGE LINE STATUS

336 <1> ; OUTPUT REGISTERS

337 <1> ; (AH) - 00 - DISK CHANGE LINE NOT ACTIVE

338 <1> ; 06 - DISK CHANGE LINE ACTIVE & CARRY BIT ON

339 <1> ; (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)

340 <1> ;-------------------------------------------------------------------------------

341 <1> ; (AH)= 17H SET DASD TYPE FOR FORMAT

342 <1> ; INPUT REGISTERS

343 <1> ; (AL) - 00 - NOT USED

344 <1> ; 01 - DISKETTE 320/360K IN 360K DRIVE

345 <1> ; 02 - DISKETTE 360K IN 1.2M DRIVE

346 <1> ; 03 - DISKETTE 1.2M IN 1.2M DRIVE

347 <1> ; 04 - DISKETTE 720K IN 720K DRIVE

348 <1> ; (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED:

349 <1> ; (DO NOT USE WHEN DISKETTE ATTACH CARD USED)

350 <1> ;-------------------------------------------------------------------------------

351 <1> ; (AH)= 18H SET MEDIA TYPE FOR FORMAT

352 <1> ; INPUT REGISTERS

353 <1> ; (CH) - LOW ORDER 8 OF 10 BITS MAXIMUM TRACKS

354 <1> ; (CL) - BITS 7 & 6 - HIGH ORDER TWO BITS OF MAXIMUM TRACKS

355 <1> ; BITS 5 THRU 0 - MAXIMUM SECTORS PER TRACK

356 <1> ; (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHACKED)

357 <1> ; OUTPUT REGISTERS:

358 <1> ; (ES:DI) - POINTER TO DRIVE PARAMETERS TABLE FOR THIS MEDIA TYPE,

359 <1> ; UNCHANGED IF (AH) IS NON-ZERO

360 <1> ; (AH) - 00H, CY = 0, TRACK AND SECTORS/TRACK COMBINATION IS SUPPORTED

361 <1> ; - 01H, CY = 1, FUNCTION IS NOT AVAILABLE

362 <1> ; - 0CH, CY = 1, TRACK AND SECTORS/TRACK COMBINATION IS NOT SUPPORTED

363 <1> ; - 80H, CY = 1, TIME OUT (DISKETTE NOT PRESENT)

364 <1> ;-------------------------------------------------------------------------------

365 <1> ; DISK CHANGE STATUS IS ONLY CHECKED WHEN A MEDIA SPECIFIED IS OTHER

366 <1> ; THAN 360 KB DRIVE. IF THE DISK CHANGE LINE IS FOUND TO BE

367 <1> ; ACTIVE THE FOLLOWING ACTIONS TAKE PLACE:

368 <1> ; ATTEMPT TO RESET DISK CHANGE LINE TO INACTIVE STATE.

369 <1> ; IF ATTEMPT SUCCEEDS SET DASD TYPE FOR FORMAT AND RETURN DISK

370 <1> ; CHANGE ERROR CODE

371 <1> ; IF ATTEMPT FAILS RETURN TIMEOUT ERROR CODE AND SET DASD TYPE

372 <1> ; TO A PREDETERMINED STATE INDICATING MEDIA TYPE UNKNOWN.

373 <1> ; IF THE DISK CHANGE LINE IN INACTIVE PERFORM SET DASD TYPE FOR FORMAT.

374 <1> ;

375 <1> ; DATA VARIABLE -- @DISK\_POINTER

376 <1> ; DOUBLE WORD POINTER TO THE CURRENT SET OF DISKETTE PARAMETERS

377 <1> ;-------------------------------------------------------------------------------

378 <1> ; OUTPUT FOR ALL FUNCTIONS

379 <1> ; AH = STATUS OF OPERATION

380 <1> ; STATUS BITS ARE DEFINED IN THE EQUATES FOR @DISKETTE\_STATUS

381 <1> ; VARIABLE IN THE DATA SEGMENT OF THIS MODULE

382 <1> ; CY = 0 SUCCESSFUL OPERATION (AH=0 ON RETURN, EXCEPT FOR READ DASD

383 <1> ; TYPE AH=(15)).

384 <1> ; CY = 1 FAILED OPERATION (AH HAS ERROR REASON)

385 <1> ; FOR READ/WRITE/VERIFY

386 <1> ; DS,BX,DX,CX PRESERVED

387 <1> ; NOTE: IF AN ERROR IS REPORTED BY THE DISKETTE CODE, THE APPROPRIATE

388 <1> ; ACTION IS TO RESET THE DISKETTE, THEN RETRY THE OPERATION.

389 <1> ; ON READ ACCESSES, NO MOTOR START DELAY IS TAKEN, SO THAT

390 <1> ; THREE RETRIES ARE REQUIRED ON READS TO ENSURE THAT THE

391 <1> ; PROBLEM IS NOT DUE TO MOTOR START-UP.

392 <1> ;-------------------------------------------------------------------------------

393 <1> ;

394 <1> ; DISKETTE STATE MACHINE - ABSOLUTE ADDRESS 40:90 (DRIVE A) & 91 (DRIVE B)

395 <1> ;

396 <1> ; -----------------------------------------------------------------

397 <1> ; | | | | | | | | |

398 <1> ; | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

399 <1> ; | | | | | | | | |

400 <1> ; -----------------------------------------------------------------

401 <1> ; | | | | | | | |

402 <1> ; | | | | | -----------------

403 <1> ; | | | | | |

404 <1> ; | | | | RESERVED |

405 <1> ; | | | | PRESENT STATE

406 <1> ; | | | | 000: 360K IN 360K DRIVE UNESTABLISHED

407 <1> ; | | | | 001: 360K IN 1.2M DRIVE UNESTABLISHED

408 <1> ; | | | | 010: 1.2M IN 1.2M DRIVE UNESTABLISHED

409 <1> ; | | | | 011: 360K IN 360K DRIVE ESTABLISHED

410 <1> ; | | | | 100: 360K IN 1.2M DRIVE ESTABLISHED

411 <1> ; | | | | 101: 1.2M IN 1.2M DRIVE ESTABLISHED

412 <1> ; | | | | 110: RESERVED

413 <1> ; | | | | 111: NONE OF THE ABOVE

414 <1> ; | | | |

415 <1> ; | | | ------> MEDIA/DRIVE ESTABLISHED

416 <1> ; | | |

417 <1> ; | | --------------> DOUBLE STEPPING REQUIRED (360K IN 1.2M

418 <1> ; | | DRIVE)

419 <1> ; | |

420 <1> ; ------------------------------> DATA TRANSFER RATE FOR THIS DRIVE:

421 <1> ;

422 <1> ; 00: 500 KBS

423 <1> ; 01: 300 KBS

424 <1> ; 10: 250 KBS

425 <1> ; 11: RESERVED

426 <1> ;

427 <1> ;

428 <1> ;-------------------------------------------------------------------------------

429 <1> ; STATE OPERATION STARTED - ABSOLUTE ADDRESS 40:92 (DRIVE A) & 93 (DRIVE B)

430 <1> ;-------------------------------------------------------------------------------

431 <1> ; PRESENT CYLINDER NUMBER - ABSOLUTE ADDRESS 40:94 (DRIVE A) & 95 (DRIVE B)

432 <1> ;-------------------------------------------------------------------------------

433 <1>

434 <1> struc MD

435 00000000 <res 00000001> <1> .SPEC1 resb 1 ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE

436 00000001 <res 00000001> <1> .SPEC2 resb 1 ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE

437 00000002 <res 00000001> <1> .OFF\_TIM resb 1 ; WAIT TIME AFTER OPERATION TILL MOTOR OFF

438 00000003 <res 00000001> <1> .BYT\_SEC resb 1 ; 512 BYTES/SECTOR

439 00000004 <res 00000001> <1> .SEC\_TRK resb 1 ; EOT (LAST SECTOR ON TRACK)

440 00000005 <res 00000001> <1> .GAP resb 1 ; GAP LENGTH

441 00000006 <res 00000001> <1> .DTL resb 1 ; DTL

442 00000007 <res 00000001> <1> .GAP3 resb 1 ; GAP LENGTH FOR FORMAT

443 00000008 <res 00000001> <1> .FIL\_BYT resb 1 ; FILL BYTE FOR FORMAT

444 00000009 <res 00000001> <1> .HD\_TIM resb 1 ; HEAD SETTLE TIME (MILLISECONDS)

445 0000000A <res 00000001> <1> .STR\_TIM resb 1 ; MOTOR START TIME (1/8 SECONDS)

446 0000000B <res 00000001> <1> .MAX\_TRK resb 1 ; MAX. TRACK NUMBER

447 0000000C <res 00000001> <1> .RATE resb 1 ; DATA TRANSFER RATE

448 <1> endstruc

449 <1>

450 <1> BIT7OFF EQU 7FH

451 <1> BIT7ON EQU 80H

452 <1>

453 <1> ;;int13h: ; 16/02/2015

454 <1> ;; 16/02/2015 - 21/02/2015

455 <1> int40h:

456 00003373 9C <1> pushfd

457 00003374 0E <1> push cs

458 00003375 E801000000 <1> call DISKETTE\_IO\_1

459 0000337A C3 <1> retn

460 <1>

461 <1> DISKETTE\_IO\_1:

462 <1>

463 0000337B FB <1> STI ; INTERRUPTS BACK ON

464 0000337C 55 <1> PUSH eBP ; USER REGISTER

465 0000337D 57 <1> PUSH eDI ; USER REGISTER

466 0000337E 52 <1> PUSH eDX ; HEAD #, DRIVE # OR USER REGISTER

467 0000337F 53 <1> PUSH eBX ; BUFFER OFFSET PARAMETER OR REGISTER

468 00003380 51 <1> PUSH eCX ; TRACK #-SECTOR # OR USER REGISTER

469 00003381 89E5 <1> MOV eBP,eSP ; BP => PARAMETER LIST DEP. ON AH

470 <1> ; [BP] = SECTOR #

471 <1> ; [BP+1] = TRACK #

472 <1> ; [BP+2] = BUFFER OFFSET

473 <1> ; FOR RETURN OF DRIVE PARAMETERS:

474 <1> ; CL/[BP] = BITS 7&6 HI BITS OF MAX CYL

475 <1> ; BITS 0-5 MAX SECTORS/TRACK

476 <1> ; CH/[BP+1] = LOW 8 BITS OF MAX CYL.

477 <1> ; BL/[BP+2] = BITS 7-4 = 0

478 <1> ; BITS 3-0 = VALID CMOS TYPE

479 <1> ; BH/[BP+3] = 0

480 <1> ; DL/[BP+4] = # DRIVES INSTALLED

481 <1> ; DH/[BP+5] = MAX HEAD #

482 <1> ; DI/[BP+6] = OFFSET TO DISK BASE

483 00003383 06 <1> push es ; 06/02/2015

484 00003384 1E <1> PUSH DS ; BUFFER SEGMENT PARM OR USER REGISTER

485 00003385 56 <1> PUSH eSI ; USER REGISTERS

486 <1> ;CALL DDS ; SEGMENT OF BIOS DATA AREA TO DS

487 <1> ;mov cx, cs

488 <1> ;mov ds, cx

489 00003386 66B91000 <1> mov cx, KDATA

490 0000338A 8ED9 <1> mov ds, cx

491 0000338C 8EC1 <1> mov es, cx

492 <1>

493 <1> ;CMP AH,(FNC\_TAE-FNC\_TAB)/2 ; CHECK FOR > LARGEST FUNCTION

494 0000338E 80FC19 <1> cmp ah,(FNC\_TAE-FNC\_TAB)/4 ; 18/02/2015

495 00003391 7202 <1> JB short OK\_FUNC ; FUNCTION OK

496 00003393 B414 <1> MOV AH,14H ; REPLACE WITH KNOWN INVALID FUNCTION

497 <1> OK\_FUNC:

498 00003395 80FC01 <1> CMP AH,1 ; RESET OR STATUS ?

499 00003398 760C <1> JBE short OK\_DRV ; IF RESET OR STATUS DRIVE ALWAYS OK

500 0000339A 80FC08 <1> CMP AH,8 ; READ DRIVE PARMS ?

501 0000339D 7407 <1> JZ short OK\_DRV ; IF SO DRIVE CHECKED LATER

502 0000339F 80FA01 <1> CMP DL,1 ; DRIVES 0 AND 1 OK

503 000033A2 7602 <1> JBE short OK\_DRV ; IF 0 OR 1 THEN JUMP

504 000033A4 B414 <1> MOV AH,14H ; REPLACE WITH KNOWN INVALID FUNCTION

505 <1> OK\_DRV:

506 000033A6 31C9 <1> xor ecx, ecx

507 <1> ;mov esi, ecx ; 08/02/2015

508 000033A8 89CF <1> mov edi, ecx ; 08/02/2015

509 000033AA 88E1 <1> MOV CL,AH ; CL = FUNCTION

510 <1> ;XOR CH,CH ; CX = FUNCTION

511 <1> ;SHL CL, 1 ; FUNCTION TIMES 2

512 000033AC C0E102 <1> SHL CL, 2 ; 20/02/2015 ; FUNCTION TIMES 4 (for 32 bit offset)

513 000033AF BB[E7330000] <1> MOV eBX,FNC\_TAB ; LOAD START OF FUNCTION TABLE

514 000033B4 01CB <1> ADD eBX,eCX ; ADD OFFSET INTO TABLE => ROUTINE

515 000033B6 88F4 <1> MOV AH,DH ; AX = HEAD #,# OF SECTORS OR DASD TYPE

516 000033B8 30F6 <1> XOR DH,DH ; DX = DRIVE #

517 000033BA 6689C6 <1> MOV SI,AX ; SI = HEAD #,# OF SECTORS OR DASD TYPE

518 000033BD 6689D7 <1> MOV DI,DX ; DI = DRIVE #

519 <1> ;

520 <1> ; 11/12/2014

521 000033C0 8815[E55C0000] <1> mov [cfd], dl ; current floppy drive (for 'GET\_PARM')

522 <1> ;

523 000033C6 8A25[C0580100] <1> MOV AH, [DSKETTE\_STATUS] ; LOAD STATUS TO AH FOR STATUS FUNCTION

524 000033CC C605[C0580100]00 <1> MOV byte [DSKETTE\_STATUS],0 ; INITIALIZE FOR ALL OTHERS

525 <1>

526 <1> ; THROUGHOUT THE DISKETTE BIOS, THE FOLLOWING INFORMATION IS CONTAINED IN

527 <1> ; THE FOLLOWING MEMORY LOCATIONS AND REGISTERS. NOT ALL DISKETTE BIOS

528 <1> ; FUNCTIONS REQUIRE ALL OF THESE PARAMETERS.

529 <1> ;

530 <1> ; DI : DRIVE #

531 <1> ; SI-HI : HEAD #

532 <1> ; SI-LOW : # OF SECTORS OR DASD TYPE FOR FORMAT

533 <1> ; ES : BUFFER SEGMENT

534 <1> ; [BP] : SECTOR #

535 <1> ; [BP+1] : TRACK #

536 <1> ; [BP+2] : BUFFER OFFSET

537 <1> ;

538 <1> ; ACROSS CALLS TO SUBROUTINES THE CARRY FLAG (CY=1), WHERE INDICATED IN

539 <1> ; SUBROUTINE PROLOGUES, REPRESENTS AN EXCEPTION RETURN (NORMALLY AN ERROR

540 <1> ; CONDITION). IN MOST CASES, WHEN CY = 1, @DSKETTE\_STATUS CONTAINS THE

541 <1> ; SPECIFIC ERROR CODE.

542 <1> ;

543 <1> ; (AH) = @DSKETTE\_STATUS

544 000033D3 FF13 <1> CALL dWORD [eBX] ; CALL THE REQUESTED FUNCTION

545 000033D5 5E <1> POP eSI ; RESTORE ALL REGISTERS

546 000033D6 1F <1> POP DS

547 000033D7 07 <1> pop es ; 06/02/2015

548 000033D8 59 <1> POP eCX

549 000033D9 5B <1> POP eBX

550 000033DA 5A <1> POP eDX

551 000033DB 5F <1> POP eDI

552 000033DC 89E5 <1> MOV eBP, eSP

553 000033DE 50 <1> PUSH eAX

554 000033DF 9C <1> PUSHFd

555 000033E0 58 <1> POP eAX

556 <1> ;MOV [BP+6], AX

557 000033E1 89450C <1> mov [ebp+12], eax ; 18/02/2015, flags

558 000033E4 58 <1> POP eAX

559 000033E5 5D <1> POP eBP

560 000033E6 CF <1> IRETd

561 <1>

562 <1> ;-------------------------------------------------------------------------------

563 <1> ; DW --> dd (06/02/2015)

564 000033E7 [4B340000] <1> FNC\_TAB dd DSK\_RESET ; AH = 00H; RESET

565 000033EB [C4340000] <1> dd DSK\_STATUS ; AH = 01H; STATUS

566 000033EF [D5340000] <1> dd DSK\_READ ; AH = 02H; READ

567 000033F3 [E6340000] <1> dd DSK\_WRITE ; AH = 03H; WRITE

568 000033F7 [F7340000] <1> dd DSK\_VERF ; AH = 04H; VERIFY

569 000033FB [08350000] <1> dd DSK\_FORMAT ; AH = 05H; FORMAT

570 000033FF [8D350000] <1> dd FNC\_ERR ; AH = 06H; INVALID

571 00003403 [8D350000] <1> dd FNC\_ERR ; AH = 07H; INVALID

572 00003407 [9A350000] <1> dd DSK\_PARMS ; AH = 08H; READ DRIVE PARAMETERS

573 0000340B [8D350000] <1> dd FNC\_ERR ; AH = 09H; INVALID

574 0000340F [8D350000] <1> dd FNC\_ERR ; AH = 0AH; INVALID

575 00003413 [8D350000] <1> dd FNC\_ERR ; AH = 0BH; INVALID

576 00003417 [8D350000] <1> dd FNC\_ERR ; AH = 0CH; INVALID

577 0000341B [8D350000] <1> dd FNC\_ERR ; AH = 0DH; INVALID

578 0000341F [8D350000] <1> dd FNC\_ERR ; AH = 0EH; INVALID

579 00003423 [8D350000] <1> dd FNC\_ERR ; AH = 0FH; INVALID

580 00003427 [8D350000] <1> dd FNC\_ERR ; AH = 10H; INVALID

581 0000342B [8D350000] <1> dd FNC\_ERR ; AH = 11H; INVALID

582 0000342F [8D350000] <1> dd FNC\_ERR ; AH = 12H; INVALID

583 00003433 [8D350000] <1> dd FNC\_ERR ; AH = 13H; INVALID

584 00003437 [8D350000] <1> dd FNC\_ERR ; AH = 14H; INVALID

585 0000343B [72360000] <1> dd DSK\_TYPE ; AH = 15H; READ DASD TYPE

586 0000343F [9D360000] <1> dd DSK\_CHANGE ; AH = 16H; CHANGE STATUS

587 00003443 [D7360000] <1> dd FORMAT\_SET ; AH = 17H; SET DASD TYPE

588 00003447 [5A370000] <1> dd SET\_MEDIA ; AH = 18H; SET MEDIA TYPE

589 <1> FNC\_TAE EQU $ ; END

590 <1>

591 <1> ;-------------------------------------------------------------------------------

592 <1> ; DISK\_RESET (AH = 00H)

593 <1> ; RESET THE DISKETTE SYSTEM.

594 <1> ;

595 <1> ; ON EXIT: @DSKETTE\_STATUS, CY REFLECT STATUS OF OPERATION

596 <1> ;-------------------------------------------------------------------------------

597 <1> DSK\_RESET:

598 0000344B 66BAF203 <1> MOV DX,03F2H ; ADAPTER CONTROL PORT

599 0000344F FA <1> CLI ; NO INTERRUPTS

600 00003450 A0[BE580100] <1> MOV AL,[MOTOR\_STATUS] ; GET DIGITAL OUTPUT REGISTER REFLECTION

601 00003455 243F <1> AND AL,00111111B ; KEEP SELECTED AND MOTOR ON BITS

602 00003457 C0C004 <1> ROL AL,4 ; MOTOR VALUE TO HIGH NIBBLE

603 <1> ; DRIVE SELECT TO LOW NIBBLE

604 0000345A 0C08 <1> OR AL,00001000B ; TURN ON INTERRUPT ENABLE

605 0000345C EE <1> OUT DX,AL ; RESET THE ADAPTER

606 0000345D C605[BD580100]00 <1> MOV byte [SEEK\_STATUS],0 ; SET RECALIBRATE REQUIRED ON ALL DRIVES

607 <1> ;JMP $+2 ; WAIT FOR I/O

608 <1> ;JMP $+2 ; WAIT FOR I/O (TO INSURE MINIMUM

609 <1> ; PULSE WIDTH)

610 <1> ; 19/12/2014

611 <1> NEWIODELAY

611 00003464 E6EB <2> out 0ebh,al

612 <1>

613 <1> ; 17/12/2014

614 <1> ; AWARD BIOS 1999 - RESETDRIVES (ADISK.ASM)

615 00003466 B915000000 <1> mov ecx, WAITCPU\_RESET\_ON ; cx = 21 -- Min. 14 micro seconds !?

616 <1> wdw1:

617 <1> NEWIODELAY ; 27/02/2015

617 0000346B E6EB <2> out 0ebh,al

618 0000346D E2FC <1> loop wdw1

619 <1> ;

620 0000346F 0C04 <1> OR AL,00000100B ; TURN OFF RESET BIT

621 00003471 EE <1> OUT DX,AL ; RESET THE ADAPTER

622 <1> ; 16/12/2014

623 <1> IODELAY

623 00003472 EB00 <2> jmp short $+2

623 00003474 EB00 <2> jmp short $+2

624 <1> ;

625 <1> ;STI ; ENABLE THE INTERRUPTS

626 00003476 E83C0C0000 <1> CALL WAIT\_INT ; WAIT FOR THE INTERRUPT

627 0000347B 723E <1> JC short DR\_ERR ; IF ERROR, RETURN IT

628 0000347D 66B9C000 <1> MOV CX,11000000B ; CL = EXPECTED @NEC\_STATUS

629 <1> NXT\_DRV:

630 00003481 6651 <1> PUSH CX ; SAVE FOR CALL

631 00003483 B8[B9340000] <1> MOV eAX, DR\_POP\_ERR ; LOAD NEC\_OUTPUT ERROR ADDRESS

632 00003488 50 <1> PUSH eAX ; "

633 00003489 B408 <1> MOV AH,08H ; SENSE INTERRUPT STATUS COMMAND

634 0000348B E81A0B0000 <1> CALL NEC\_OUTPUT

635 00003490 58 <1> POP eAX ; THROW AWAY ERROR RETURN

636 00003491 E8510C0000 <1> CALL RESULTS ; READ IN THE RESULTS

637 00003496 6659 <1> POP CX ; RESTORE AFTER CALL

638 00003498 7221 <1> JC short DR\_ERR ; ERROR RETURN

639 0000349A 3A0D[C1580100] <1> CMP CL, [NEC\_STATUS] ; TEST FOR DRIVE READY TRANSITION

640 000034A0 7519 <1> JNZ short DR\_ERR ; EVERYTHING OK

641 000034A2 FEC1 <1> INC CL ; NEXT EXPECTED @NEC\_STATUS

642 000034A4 80F9C3 <1> CMP CL,11000011B ; ALL POSSIBLE DRIVES CLEARED

643 000034A7 76D8 <1> JBE short NXT\_DRV ; FALL THRU IF 11000100B OR >

644 <1> ;

645 000034A9 E869030000 <1> CALL SEND\_SPEC ; SEND SPECIFY COMMAND TO NEC

646 <1> RESBAC:

647 000034AE E81D090000 <1> CALL SETUP\_END ; VARIOUS CLEANUPS

648 000034B3 6689F3 <1> MOV BX,SI ; GET SAVED AL TO BL

649 000034B6 88D8 <1> MOV AL,BL ; PUT BACK FOR RETURN

650 000034B8 C3 <1> RETn

651 <1> DR\_POP\_ERR:

652 000034B9 6659 <1> POP CX ; CLEAR STACK

653 <1> DR\_ERR:

654 000034BB 800D[C0580100]20 <1> OR byte [DSKETTE\_STATUS],BAD\_NEC ; SET ERROR CODE

655 000034C2 EBEA <1> JMP SHORT RESBAC ; RETURN FROM RESET

656 <1>

657 <1> ;-------------------------------------------------------------------------------

658 <1> ; DISK\_STATUS (AH = 01H)

659 <1> ; DISKETTE STATUS.

660 <1> ;

661 <1> ; ON ENTRY: AH : STATUS OF PREVIOUS OPERATION

662 <1> ;

663 <1> ; ON EXIT: AH, @DSKETTE\_STATUS, CY REFLECT STATUS OF PREVIOUS OPERATION.

664 <1> ;-------------------------------------------------------------------------------

665 <1> DSK\_STATUS:

666 000034C4 8825[C0580100] <1> MOV [DSKETTE\_STATUS],AH ; PUT BACK FOR SETUP END

667 000034CA E801090000 <1> CALL SETUP\_END ; VARIOUS CLEANUPS

668 000034CF 6689F3 <1> MOV BX,SI ; GET SAVED AL TO BL

669 000034D2 88D8 <1> MOV AL,BL ; PUT BACK FOR RETURN

670 000034D4 C3 <1> RETn

671 <1>

672 <1> ;-------------------------------------------------------------------------------

673 <1> ; DISK\_READ (AH = 02H)

674 <1> ; DISKETTE READ.

675 <1> ;

676 <1> ; ON ENTRY: DI : DRIVE #

677 <1> ; SI-HI : HEAD #

678 <1> ; SI-LOW : # OF SECTORS

679 <1> ; ES : BUFFER SEGMENT

680 <1> ; [BP] : SECTOR #

681 <1> ; [BP+1] : TRACK #

682 <1> ; [BP+2] : BUFFER OFFSET

683 <1> ;

684 <1> ; ON EXIT: @DSKETTE\_STATUS, CY REFLECT STATUS OF OPERATION

685 <1> ;-------------------------------------------------------------------------------

686 <1>

687 <1> ; 06/02/2015, ES:BX -> EBX (unix386.s)

688 <1>

689 <1> DSK\_READ:

690 000034D5 8025[BE580100]7F <1> AND byte [MOTOR\_STATUS],01111111B ; INDICATE A READ OPERATION

691 000034DC 66B846E6 <1> MOV AX,0E646H ; AX = NEC COMMAND, DMA COMMAND

692 000034E0 E83C040000 <1> CALL RD\_WR\_VF ; COMMON READ/WRITE/VERIFY

693 000034E5 C3 <1> RETn

694 <1>

695 <1> ;-------------------------------------------------------------------------------

696 <1> ; DISK\_WRITE (AH = 03H)

697 <1> ; DISKETTE WRITE.

698 <1> ;

699 <1> ; ON ENTRY: DI : DRIVE #

700 <1> ; SI-HI : HEAD #

701 <1> ; SI-LOW : # OF SECTORS

702 <1> ; ES : BUFFER SEGMENT

703 <1> ; [BP] : SECTOR #

704 <1> ; [BP+1] : TRACK #

705 <1> ; [BP+2] : BUFFER OFFSET

706 <1> ;

707 <1> ; ON EXIT: @DSKETTE\_STATUS, CY REFLECT STATUS OF OPERATION

708 <1> ;-------------------------------------------------------------------------------

709 <1>

710 <1> ; 06/02/2015, ES:BX -> EBX (unix386.s)

711 <1>

712 <1> DSK\_WRITE:

713 000034E6 66B84AC5 <1> MOV AX,0C54AH ; AX = NEC COMMAND, DMA COMMAND

714 000034EA 800D[BE580100]80 <1> OR byte [MOTOR\_STATUS],10000000B ; INDICATE WRITE OPERATION

715 000034F1 E82B040000 <1> CALL RD\_WR\_VF ; COMMON READ/WRITE/VERIFY

716 000034F6 C3 <1> RETn

717 <1>

718 <1> ;-------------------------------------------------------------------------------

719 <1> ; DISK\_VERF (AH = 04H)

720 <1> ; DISKETTE VERIFY.

721 <1> ;

722 <1> ; ON ENTRY: DI : DRIVE #

723 <1> ; SI-HI : HEAD #

724 <1> ; SI-LOW : # OF SECTORS

725 <1> ; ES : BUFFER SEGMENT

726 <1> ; [BP] : SECTOR #

727 <1> ; [BP+1] : TRACK #

728 <1> ; [BP+2] : BUFFER OFFSET

729 <1> ;

730 <1> ; ON EXIT: @DSKETTE\_STATUS, CY REFLECT STATUS OF OPERATION

731 <1> ;-------------------------------------------------------------------------------

732 <1> DSK\_VERF:

733 000034F7 8025[BE580100]7F <1> AND byte [MOTOR\_STATUS],01111111B ; INDICATE A READ OPERATION

734 000034FE 66B842E6 <1> MOV AX,0E642H ; AX = NEC COMMAND, DMA COMMAND

735 00003502 E81A040000 <1> CALL RD\_WR\_VF ; COMMON READ/WRITE/VERIFY

736 00003507 C3 <1> RETn

737 <1>

738 <1> ;-------------------------------------------------------------------------------

739 <1> ; DISK\_FORMAT (AH = 05H)

740 <1> ; DISKETTE FORMAT.

741 <1> ;

742 <1> ; ON ENTRY: DI : DRIVE #

743 <1> ; SI-HI : HEAD #

744 <1> ; SI-LOW : # OF SECTORS

745 <1> ; ES : BUFFER SEGMENT

746 <1> ; [BP] : SECTOR #

747 <1> ; [BP+1] : TRACK #

748 <1> ; [BP+2] : BUFFER OFFSET

749 <1> ; @DISK\_POINTER POINTS TO THE PARAMETER TABLE OF THIS DRIVE

750 <1> ;

751 <1> ; ON EXIT: @DSKETTE\_STATUS, CY REFLECT STATUS OF OPERATION

752 <1> ;-------------------------------------------------------------------------------

753 <1> DSK\_FORMAT:

754 00003508 E853030000 <1> CALL XLAT\_NEW ; TRANSLATE STATE TO PRESENT ARCH.

755 0000350D E84F050000 <1> CALL FMT\_INIT ; ESTABLISH STATE IF UNESTABLISHED

756 00003512 800D[BE580100]80 <1> OR byte [MOTOR\_STATUS], 10000000B ; INDICATE WRITE OPERATION

757 00003519 E897050000 <1> CALL MED\_CHANGE ; CHECK MEDIA CHANGE AND RESET IF SO

758 0000351E 725D <1> JC short FM\_DON ; MEDIA CHANGED, SKIP

759 00003520 E8F2020000 <1> CALL SEND\_SPEC ; SEND SPECIFY COMMAND TO NEC

760 00003525 E8FD050000 <1> CALL CHK\_LASTRATE ; ZF=1 ATTEMPT RATE IS SAME AS LAST RATE

761 0000352A 7405 <1> JZ short FM\_WR ; YES, SKIP SPECIFY COMMAND

762 0000352C E8D4050000 <1> CALL SEND\_RATE ; SEND DATA RATE TO CONTROLLER

763 <1> FM\_WR:

764 00003531 E88A060000 <1> CALL FMTDMA\_SET ; SET UP THE DMA FOR FORMAT

765 00003536 7245 <1> JC short FM\_DON ; RETURN WITH ERROR

766 00003538 B44D <1> MOV AH,04DH ; ESTABLISH THE FORMAT COMMAND

767 0000353A E8E7060000 <1> CALL NEC\_INIT ; INITIALIZE THE NEC

768 0000353F 723C <1> JC short FM\_DON ; ERROR - EXIT

769 00003541 B8[7D350000] <1> MOV eAX, FM\_DON ; LOAD ERROR ADDRESS

770 00003546 50 <1> PUSH eAX ; PUSH NEC\_OUT ERROR RETURN

771 00003547 B203 <1> MOV DL,3 ; BYTES/SECTOR VALUE TO NEC

772 00003549 E856090000 <1> CALL GET\_PARM

773 0000354E E8570A0000 <1> CALL NEC\_OUTPUT

774 00003553 B204 <1> MOV DL,4 ; SECTORS/TRACK VALUE TO NEC

775 00003555 E84A090000 <1> CALL GET\_PARM

776 0000355A E84B0A0000 <1> CALL NEC\_OUTPUT

777 0000355F B207 <1> MOV DL,7 ; GAP LENGTH VALUE TO NEC

778 00003561 E83E090000 <1> CALL GET\_PARM

779 00003566 E83F0A0000 <1> CALL NEC\_OUTPUT

780 0000356B B208 <1> MOV DL,8 ; FILLER BYTE TO NEC

781 0000356D E832090000 <1> CALL GET\_PARM

782 00003572 E8330A0000 <1> CALL NEC\_OUTPUT

783 00003577 58 <1> POP eAX ; THROW AWAY ERROR

784 00003578 E827070000 <1> CALL NEC\_TERM ; TERMINATE, RECEIVE STATUS, ETC,

785 <1> FM\_DON:

786 0000357D E80F030000 <1> CALL XLAT\_OLD ; TRANSLATE STATE TO COMPATIBLE MODE

787 00003582 E849080000 <1> CALL SETUP\_END ; VARIOUS CLEANUPS

788 00003587 6689F3 <1> MOV BX,SI ; GET SAVED AL TO BL

789 0000358A 88D8 <1> MOV AL,BL ; PUT BACK FOR RETURN

790 0000358C C3 <1> RETn

791 <1>

792 <1> ;-------------------------------------------------------------------------------

793 <1> ; FNC\_ERR

794 <1> ; INVALID FUNCTION REQUESTED OR INVALID DRIVE:

795 <1> ; SET BAD COMMAND IN STATUS.

796 <1> ;

797 <1> ; ON EXIT: @DSKETTE\_STATUS, CY REFLECT STATUS OF OPERATION

798 <1> ;-------------------------------------------------------------------------------

799 <1> FNC\_ERR: ; INVALID FUNCTION REQUEST

800 0000358D 6689F0 <1> MOV AX,SI ; RESTORE AL

801 00003590 B401 <1> MOV AH,BAD\_CMD ; SET BAD COMMAND ERROR

802 00003592 8825[C0580100] <1> MOV [DSKETTE\_STATUS],AH ; STORE IN DATA AREA

803 00003598 F9 <1> STC ; SET CARRY INDICATING ERROR

804 00003599 C3 <1> RETn

805 <1>

806 <1> ; 01/06/2016

807 <1> ; 28/05/2016

808 <1> ; 27/05/2016 - TRDOS 386 (TRDOS v.2.0)

809 <1> ;-------------------------------------------------------------------------------

810 <1> ; DISK\_PARMS (AH = 08H)

811 <1> ; READ DRIVE PARAMETERS.

812 <1> ;

813 <1> ; ON ENTRY: DI : DRIVE #

814 <1> ; ; 27/05/2016

815 <1> ; EBX = Buffer Address for floppy disk parameters table (16 bytes)

816 <1> ;

817 <1> ; ON EXIT: CL/[BP] = BITS 7 & 6 HI 2 BITS OF MAX CYLINDER

818 <1> ; BITS 0-5 MAX SECTORS/TRACK

819 <1> ; CH/[BP+1] = LOW 8 BITS OF MAX CYLINDER

820 <1> ; BL/[BP+2] = BITS 7-4 = 0

821 <1> ; BITS 3-0 = VALID CMOS DRIVE TYPE

822 <1> ; BH/[BP+3] = 0

823 <1> ; DL/[BP+4] = # DRIVES INSTALLED (VALUE CHECKED)

824 <1> ; DH/[BP+5] = MAX HEAD #

825 <1> ; \*\* 27/05/2016 - TRDOS 386 (TRDOS v2.0) \*\*

826 <1> ; \*\* EBX = Buffer address for floppy disk parameters table \*\*

827 <1> ; ;DI/[BP+6] = OFFSET TO DISK\_BASE

828 <1> ; ;ES = SEGMENT OF DISK\_BASE

829 <1> ;

830 <1> ; AX = 0

831 <1> ;

832 <1> ; NOTE : THE ABOVE INFORMATION IS STORED IN THE USERS STACK AT

833 <1> ; THE LOCATIONS WHERE THE MAIN ROUTINE WILL POP THEM

834 <1> ; INTO THE APPROPRIATE REGISTERS BEFORE RETURNING TO THE

835 <1> ; CALLER.

836 <1> ;-------------------------------------------------------------------------------

837 <1> DSK\_PARMS:

838 0000359A E8C1020000 <1> CALL XLAT\_NEW ; TRANSLATE STATE TO PRESENT ARCH,

839 <1> ; MOV WORD [BP+2],0 ; DRIVE TYPE = 0

840 <1> ; MOV AX, [EQUIP\_FLAG] ; LOAD EQUIPMENT FLAG FOR # DISKETTES

841 <1> ; AND AL,11000001B ; KEEP DISKETTE DRIVE BITS

842 <1> ; MOV DL,2 ; DISKETTE DRIVES = 2

843 <1> ; CMP AL,01000001B ; 2 DRIVES INSTALLED ?

844 <1> ; JZ short STO\_DL ; IF YES JUMP

845 <1> ; DEC DL ; DISKETTE DRIVES = 1

846 <1> ; CMP AL,00000001B ; 1 DRIVE INSTALLED ?

847 <1> ; JNZ short NON\_DRV ; IF NO JUMP

848 0000359F 29D2 <1> sub edx, edx

849 000035A1 66A1[F65C0000] <1> mov ax, [fd0\_type]

850 000035A7 6621C0 <1> and ax, ax

851 000035AA 0F848A000000 <1> jz NON\_DRV

852 000035B0 FEC2 <1> inc dl

853 000035B2 20E4 <1> and ah, ah

854 000035B4 7402 <1> jz short STO\_DL

855 000035B6 FEC2 <1> inc dl

856 <1> STO\_DL:

857 <1> ;MOV [BP+4],DL ; STORE NUMBER OF DRIVES

858 000035B8 895508 <1> mov [ebp+8], edx ; 20/02/2015

859 000035BB 6683FF01 <1> CMP DI,1 ; CHECK FOR VALID DRIVE

860 000035BF 777C <1> JA short NON\_DRV1 ; DRIVE INVALID

861 <1> ;MOV BYTE [BP+5],1 ; MAXIMUM HEAD NUMBER = 1

862 000035C1 C6450901 <1> mov byte [ebp+9], 1 ; 20/02/2015

863 000035C5 E8D1080000 <1> CALL CMOS\_TYPE ; RETURN DRIVE TYPE IN AL

864 <1> ;;20/02/2015

865 <1> ;;JC short CHK\_EST ; IF CMOS BAD CHECKSUM ESTABLISHED

866 <1> ;;OR AL,AL ; TEST FOR NO DRIVE TYPE

867 000035CA 740F <1> JZ short CHK\_EST ; JUMP IF SO

868 000035CC E81B020000 <1> CALL DR\_TYPE\_CHECK ; RTN CS:BX = MEDIA/DRIVE PARAM TBL

869 000035D1 7208 <1> JC short CHK\_EST ; TYPE NOT IN TABLE (POSSIBLE BAD CMOS)

870 <1> ;MOV [BP+2],AL ; STORE VALID CMOS DRIVE TYPE

871 <1> ;mov [ebp+4], al ; 06/02/2015

872 000035D3 8A4B04 <1> MOV CL, [eBX+MD.SEC\_TRK] ; GET SECTOR/TRACK

873 000035D6 8A6B0B <1> MOV CH, [eBX+MD.MAX\_TRK] ; GET MAX. TRACK NUMBER

874 000035D9 EB36 <1> JMP SHORT STO\_CX ; CMOS GOOD, USE CMOS

875 <1> CHK\_EST:

876 000035DB 8AA7[CD580100] <1> MOV AH, [DSK\_STATE+eDI] ; LOAD STATE FOR THIS DRIVE

877 000035E1 F6C410 <1> TEST AH,MED\_DET ; CHECK FOR ESTABLISHED STATE

878 000035E4 7457 <1> JZ short NON\_DRV1 ; CMOS BAD/INVALID OR UNESTABLISHED

879 <1> USE\_EST:

880 000035E6 80E4C0 <1> AND AH,RATE\_MSK ; ISOLATE STATE

881 000035E9 80FC80 <1> CMP AH,RATE\_250 ; RATE 250 ?

882 000035EC 7570 <1> JNE short USE\_EST2 ; NO, GO CHECK OTHER RATE

883 <1>

884 <1> ;----- DATA RATE IS 250 KBS, TRY 360 KB TABLE FIRST

885 <1>

886 000035EE B001 <1> MOV AL,01 ; DRIVE TYPE 1 (360KB)

887 000035F0 E8F7010000 <1> CALL DR\_TYPE\_CHECK ; RTN CS:BX = MEDIA/DRIVE PARAM TBL

888 000035F5 8A4B04 <1> MOV CL, [eBX+MD.SEC\_TRK] ; GET SECTOR/TRACK

889 000035F8 8A6B0B <1> MOV CH, [eBX+MD.MAX\_TRK] ; GET MAX. TRACK NUMBER

890 000035FB F687[CD580100]01 <1> TEST byte [DSK\_STATE+eDI],TRK\_CAPA ; 80 TRACK ?

891 00003602 740D <1> JZ short STO\_CX ; MUST BE 360KB DRIVE

892 <1>

893 <1> ;----- IT IS 1.44 MB DRIVE

894 <1>

895 <1> PARM144:

896 00003604 B004 <1> MOV AL,04 ; DRIVE TYPE 4 (1.44MB)

897 00003606 E8E1010000 <1> CALL DR\_TYPE\_CHECK ; RTN CS:BX = MEDIA/DRIVE PARAM TBL

898 0000360B 8A4B04 <1> MOV CL, [eBX+MD.SEC\_TRK] ; GET SECTOR/TRACK

899 0000360E 8A6B0B <1> MOV CH, [eBX+MD.MAX\_TRK] ; GET MAX. TRACK NUMBER

900 <1> STO\_CX:

901 00003611 894D00 <1> MOV [eBP],eCX ; SAVE POINTER IN STACK FOR RETURN

902 <1> ES\_DI:

903 <1> ;MOV [BP+6],BX ; ADDRESS OF MEDIA/DRIVE PARM TABLE

904 <1> ;mov [ebp+12], ebx ; 06/02/2015

905 <1> ;MOV AX,CS ; SEGMENT MEDIA/DRIVE PARAMETER TABLE

906 <1> ;MOV ES,AX ; ES IS SEGMENT OF TABLE

907 <1> ;

908 <1> ; 28/05/2016

909 <1> ; 27/05/2016

910 <1> ; return floppy disk parameters table to user

911 <1> ; in user's buffer, which is pointed by EBX

912 <1> ;

913 00003614 57 <1> push edi

914 00003615 8B7D04 <1> mov edi, [ebp+4] ; ebx (input), user's buffer address

915 00003618 0FB6C0 <1> movzx eax, al

916 0000361B 894504 <1> mov [ebp+4], eax ; ebx ; drive type (for floppy drives)

917 <1> ; 01/06/2016 (INT 33h, disk type return for floppy disks, in BL)

918 0000361E A3[C8650100] <1> mov [user\_buffer], eax ; 01/06/2016 (overwrite ebx return value)

919 <1> ;(INT 33h, Function 08h will replace user's buffer addr with disk type!)

920 <1> ;

921 00003623 89DE <1> mov esi, ebx ; floppy disk parameter table (16 bytes)

922 00003625 B910000000 <1> mov ecx, 16 ; 16 bytes

923 0000362A E84AB10000 <1> call transfer\_to\_user\_buffer ; trdosk6.s (16/05/2016)

924 0000362F 5F <1> pop edi

925 <1> DP\_OUT:

926 00003630 E85C020000 <1> CALL XLAT\_OLD ; TRANSLATE STATE TO COMPATIBLE MODE

927 00003635 6631C0 <1> XOR AX,AX ; CLEAR

928 00003638 F8 <1> CLC

929 00003639 C3 <1> RETn

930 <1>

931 <1> ;----- NO DRIYE PRESENT HANDLER

932 <1>

933 <1> NON\_DRV:

934 <1> ;MOV BYTE [BP+4],0 ; CLEAR NUMBER OF DRIVES

935 0000363A 895508 <1> mov [ebp+8], edx ; 0 ; 20/02/2015

936 <1> NON\_DRV1:

937 0000363D 6681FF8000 <1> CMP DI,80H ; CHECK FOR FIXED MEDIA TYPE REQUEST

938 00003642 720C <1> JB short NON\_DRV2 ; CONTINUE IF NOT REQUEST FALL THROUGH

939 <1>

940 <1> ;----- FIXED DISK REQUEST FALL THROUGH ERROR

941 <1>

942 00003644 E848020000 <1> CALL XLAT\_OLD ; ELSE TRANSLATE TO COMPATIBLE MODE

943 00003649 6689F0 <1> MOV AX,SI ; RESTORE AL

944 0000364C B401 <1> MOV AH,BAD\_CMD ; SET BAD COMMAND ERROR

945 0000364E F9 <1> STC

946 0000364F C3 <1> RETn

947 <1>

948 <1> NON\_DRV2:

949 <1> ;XOR AX,AX ; CLEAR PARMS IF NO DRIVES OR CMOS BAD

950 00003650 31C0 <1> xor eax, eax

951 00003652 66894500 <1> MOV [eBP],AX ; TRACKS, SECTORS/TRACK = 0

952 <1> ;MOV [BP+5],AH ; HEAD = 0

953 00003656 886509 <1> mov [ebp+9], ah ; 06/02/2015

954 <1> ;MOV [BP+6],AX ; OFFSET TO DISK\_BASE = 0

955 00003659 89450C <1> mov [ebp+12], eax

956 <1> ;MOV ES,AX ; ES IS SEGMENT OF TABLE

957 0000365C EBD2 <1> JMP SHORT DP\_OUT

958 <1>

959 <1> ;----- DATA RATE IS EITHER 300 KBS OR 500 KBS, TRY 1.2 MB TABLE FIRST

960 <1>

961 <1> USE\_EST2:

962 0000365E B002 <1> MOV AL,02 ; DRIVE TYPE 2 (1.2MB)

963 00003660 E887010000 <1> CALL DR\_TYPE\_CHECK ; RTN CS:BX = MEDIA/DRIVE PARAM TBL

964 00003665 8A4B04 <1> MOV CL, [eBX+MD.SEC\_TRK] ; GET SECTOR/TRACK

965 00003668 8A6B0B <1> MOV CH, [eBX+MD.MAX\_TRK] ; GET MAX. TRACK NUMBER

966 0000366B 80FC40 <1> CMP AH,RATE\_300 ; RATE 300 ?

967 0000366E 74A1 <1> JZ short STO\_CX ; MUST BE 1.2MB DRIVE

968 00003670 EB92 <1> JMP SHORT PARM144 ; ELSE, IT IS 1.44MB DRIVE

969 <1>

970 <1> ;-------------------------------------------------------------------------------

971 <1> ; DISK\_TYPE (AH = 15H)

972 <1> ; THIS ROUTINE RETURNS THE TYPE OF MEDIA INSTALLED.

973 <1> ;

974 <1> ; ON ENTRY: DI = DRIVE #

975 <1> ;

976 <1> ; ON EXIT: AH = DRIVE TYPE, CY=0

977 <1> ;-------------------------------------------------------------------------------

978 <1> DSK\_TYPE:

979 00003672 E8E9010000 <1> CALL XLAT\_NEW ; TRANSLATE STATE TO PRESENT ARCH.

980 00003677 8A87[CD580100] <1> MOV AL, [DSK\_STATE+eDI] ; GET PRESENT STATE INFORMATION

981 0000367D 08C0 <1> OR AL,AL ; CHECK FOR NO DRIVE

982 0000367F 7418 <1> JZ short NO\_DRV

983 00003681 B401 <1> MOV AH,NOCHGLN ; NO CHANGE LINE FOR 40 TRACK DRIVE

984 00003683 A801 <1> TEST AL,TRK\_CAPA ; IS THIS DRIVE AN 80 TRACK DRIVE?

985 00003685 7402 <1> JZ short DT\_BACK ; IF NO JUMP

986 00003687 B402 <1> MOV AH,CHGLN ; CHANGE LINE FOR 80 TRACK DRIVE

987 <1> DT\_BACK:

988 00003689 6650 <1> PUSH AX ; SAVE RETURN VALUE

989 0000368B E801020000 <1> CALL XLAT\_OLD ; TRANSLATE STATE TO COMPATIBLE MODE

990 00003690 6658 <1> POP AX ; RESTORE RETURN VALUE

991 00003692 F8 <1> CLC ; NO ERROR

992 00003693 6689F3 <1> MOV BX,SI ; GET SAVED AL TO BL

993 00003696 88D8 <1> MOV AL,BL ; PUT BACK FOR RETURN

994 00003698 C3 <1> RETn

995 <1> NO\_DRV:

996 00003699 30E4 <1> XOR AH,AH ; NO DRIVE PRESENT OR UNKNOWN

997 0000369B EBEC <1> JMP SHORT DT\_BACK

998 <1>

999 <1> ;-------------------------------------------------------------------------------

1000 <1> ; DISK\_CHANGE (AH = 16H)

1001 <1> ; THIS ROUTINE RETURNS THE STATE OF THE DISK CHANGE LINE.

1002 <1> ;

1003 <1> ; ON ENTRY: DI = DRIVE #

1004 <1> ;

1005 <1> ; ON EXIT: AH = @DSKETTE\_STATUS

1006 <1> ; 00 - DISK CHANGE LINE INACTIVE, CY = 0

1007 <1> ; 06 - DISK CHANGE LINE ACTIVE, CY = 1

1008 <1> ;-------------------------------------------------------------------------------

1009 <1> DSK\_CHANGE:

1010 0000369D E8BE010000 <1> CALL XLAT\_NEW ; TRANSLATE STATE TO PRESENT ARCH.

1011 000036A2 8A87[CD580100] <1> MOV AL, [DSK\_STATE+eDI] ; GET MEDIA STATE INFORMATION

1012 000036A8 08C0 <1> OR AL,AL ; DRIVE PRESENT ?

1013 000036AA 7422 <1> JZ short DC\_NON ; JUMP IF NO DRIVE

1014 000036AC A801 <1> TEST AL,TRK\_CAPA ; 80 TRACK DRIVE ?

1015 000036AE 7407 <1> JZ short SETIT ; IF SO , CHECK CHANGE LINE

1016 <1> DC0:

1017 000036B0 E88D0A0000 <1> CALL READ\_DSKCHNG ; GO CHECK STATE OF DISK CHANGE LINE

1018 000036B5 7407 <1> JZ short FINIS ; CHANGE LINE NOT ACTIVE

1019 <1>

1020 000036B7 C605[C0580100]06 <1> SETIT: MOV byte [DSKETTE\_STATUS], MEDIA\_CHANGE ; INDICATE MEDIA REMOVED

1021 <1>

1022 000036BE E8CE010000 <1> FINIS: CALL XLAT\_OLD ; TRANSLATE STATE TO COMPATIBLE MODE

1023 000036C3 E808070000 <1> CALL SETUP\_END ; VARIOUS CLEANUPS

1024 000036C8 6689F3 <1> MOV BX,SI ; GET SAVED AL TO BL

1025 000036CB 88D8 <1> MOV AL,BL ; PUT BACK FOR RETURN

1026 000036CD C3 <1> RETn

1027 <1> DC\_NON:

1028 000036CE 800D[C0580100]80 <1> OR byte [DSKETTE\_STATUS], TIME\_OUT ; SET TIMEOUT, NO DRIVE

1029 000036D5 EBE7 <1> JMP SHORT FINIS

1030 <1>

1031 <1> ;-------------------------------------------------------------------------------

1032 <1> ; FORMAT\_SET (AH = 17H)

1033 <1> ; THIS ROUTINE IS USED TO ESTABLISH THE TYPE OF MEDIA TO BE USED

1034 <1> ; FOR THE FOLLOWING FORMAT OPERATION.

1035 <1> ;

1036 <1> ; ON ENTRY: SI LOW = DASD TYPE FOR FORMAT

1037 <1> ; DI = DRIVE #

1038 <1> ;

1039 <1> ; ON EXIT: @DSKETTE\_STATUS REFLECTS STATUS

1040 <1> ; AH = @DSKETTE\_STATUS

1041 <1> ; CY = 1 IF ERROR

1042 <1> ;-------------------------------------------------------------------------------

1043 <1> FORMAT\_SET:

1044 000036D7 E884010000 <1> CALL XLAT\_NEW ; TRANSLATE STATE TO PRESENT ARCH.

1045 000036DC 6656 <1> PUSH SI ; SAVE DASD TYPE

1046 000036DE 6689F0 <1> MOV AX,SI ; AH = ? , AL , DASD TYPE

1047 000036E1 30E4 <1> XOR AH,AH ; AH , 0 , AL , DASD TYPE

1048 000036E3 6689C6 <1> MOV SI,AX ; SI = DASD TYPE

1049 000036E6 80A7[CD580100]0F <1> AND byte [DSK\_STATE+eDI], ~(MED\_DET+DBL\_STEP+RATE\_MSK) ; CLEAR STATE

1050 000036ED 664E <1> DEC SI ; CHECK FOR 320/360K MEDIA & DRIVE

1051 000036EF 7509 <1> JNZ short NOT\_320 ; BYPASS IF NOT

1052 000036F1 808F[CD580100]90 <1> OR byte [DSK\_STATE+eDI], MED\_DET+RATE\_250 ; SET TO 320/360

1053 000036F8 EB48 <1> JMP SHORT S0

1054 <1>

1055 <1> NOT\_320:

1056 000036FA E8B6030000 <1> CALL MED\_CHANGE ; CHECK FOR TIME\_OUT

1057 000036FF 803D[C0580100]80 <1> CMP byte [DSKETTE\_STATUS], TIME\_OUT

1058 00003706 743A <1> JZ short S0 ; IF TIME OUT TELL CALLER

1059 <1> S3:

1060 00003708 664E <1> DEC SI ; CHECK FOR 320/360K IN 1.2M DRIVE

1061 0000370A 7509 <1> JNZ short NOT\_320\_12 ; BYPASS IF NOT

1062 0000370C 808F[CD580100]70 <1> OR byte [DSK\_STATE+eDI], MED\_DET+DBL\_STEP+RATE\_300 ; SET STATE

1063 00003713 EB2D <1> JMP SHORT S0

1064 <1>

1065 <1> NOT\_320\_12:

1066 00003715 664E <1> DEC SI ; CHECK FOR 1.2M MEDIA IN 1.2M DRIVE

1067 00003717 7509 <1> JNZ short NOT\_12 ; BYPASS IF NOT

1068 00003719 808F[CD580100]10 <1> OR byte [DSK\_STATE+eDI], MED\_DET+RATE\_500 ; SET STATE VARIABLE

1069 00003720 EB20 <1> JMP SHORT S0 ; RETURN TO CALLER

1070 <1>

1071 <1> NOT\_12:

1072 00003722 664E <1> DEC SI ; CHECK FOR SET DASD TYPE 04

1073 00003724 752B <1> JNZ short FS\_ERR ; BAD COMMAND EXIT IF NOT VALID TYPE

1074 <1>

1075 00003726 F687[CD580100]04 <1> TEST byte [DSK\_STATE+eDI], DRV\_DET ; DRIVE DETERMINED ?

1076 0000372D 740B <1> JZ short ASSUME ; IF STILL NOT DETERMINED ASSUME

1077 0000372F B050 <1> MOV AL,MED\_DET+RATE\_300

1078 00003731 F687[CD580100]02 <1> TEST byte [DSK\_STATE+eDI], FMT\_CAPA ; MULTIPLE FORMAT CAPABILITY ?

1079 00003738 7502 <1> JNZ short OR\_IT\_IN ; IF 1.2 M THEN DATA RATE 300

1080 <1>

1081 <1> ASSUME:

1082 0000373A B090 <1> MOV AL,MED\_DET+RATE\_250 ; SET UP

1083 <1>

1084 <1> OR\_IT\_IN:

1085 0000373C 0887[CD580100] <1> OR [DSK\_STATE+eDI], AL ; OR IN THE CORRECT STATE

1086 <1> S0:

1087 00003742 E84A010000 <1> CALL XLAT\_OLD ; TRANSLATE STATE TO COMPATIBLE MODE

1088 00003747 E884060000 <1> CALL SETUP\_END ; VARIOUS CLEANUPS

1089 0000374C 665B <1> POP BX ; GET SAVED AL TO BL

1090 0000374E 88D8 <1> MOV AL,BL ; PUT BACK FOR RETURN

1091 00003750 C3 <1> RETn

1092 <1>

1093 <1> FS\_ERR:

1094 00003751 C605[C0580100]01 <1> MOV byte [DSKETTE\_STATUS], BAD\_CMD ; UNKNOWN STATE,BAD COMMAND

1095 00003758 EBE8 <1> JMP SHORT S0

1096 <1>

1097 <1> ;-------------------------------------------------------------------------------

1098 <1> ; SET\_MEDIA (AH = 18H)

1099 <1> ; THIS ROUTINE SETS THE TYPE OF MEDIA AND DATA RATE

1100 <1> ; TO BE USED FOR THE FOLLOWING FORMAT OPERATION.

1101 <1> ;

1102 <1> ; ON ENTRY:

1103 <1> ; [BP] = SECTOR PER TRACK

1104 <1> ; [BP+1] = TRACK #

1105 <1> ; DI = DRIVE #

1106 <1> ;

1107 <1> ; ON EXIT:

1108 <1> ; @DSKETTE\_STATUS REFLECTS STATUS

1109 <1> ; IF NO ERROR:

1110 <1> ; AH = 0

1111 <1> ; CY = 0

1112 <1> ; ES = SEGMENT OF MEDIA/DRIVE PARAMETER TABLE

1113 <1> ; DI/[BP+6] = OFFSET OF MEDIA/DRIVE PARAMETER TABLE

1114 <1> ; IF ERROR:

1115 <1> ; AH = @DSKETTE\_STATUS

1116 <1> ; CY = 1

1117 <1> ;-------------------------------------------------------------------------------

1118 <1> SET\_MEDIA:

1119 0000375A E801010000 <1> CALL XLAT\_NEW ; TRANSLATE STATE TO PRESENT ARCH.

1120 0000375F F687[CD580100]01 <1> TEST byte [DSK\_STATE+eDI], TRK\_CAPA ; CHECK FOR CHANGE LINE AVAILABLE

1121 00003766 7415 <1> JZ short SM\_CMOS ; JUMP IF 40 TRACK DRIVE

1122 00003768 E848030000 <1> CALL MED\_CHANGE ; RESET CHANGE LINE

1123 0000376D 803D[C0580100]80 <1> CMP byte [DSKETTE\_STATUS], TIME\_OUT ; IF TIME OUT TELL CALLER

1124 00003774 746B <1> JE short SM\_RTN

1125 00003776 C605[C0580100]00 <1> MOV byte [DSKETTE\_STATUS], 0 ; CLEAR STATUS

1126 <1> SM\_CMOS:

1127 0000377D E819070000 <1> CALL CMOS\_TYPE ; RETURN DRIVE TYPE IN (AL)

1128 <1> ;;20/02/2015

1129 <1> ;;JC short MD\_NOT\_FND ; ERROR IN CMOS

1130 <1> ;;OR AL,AL ; TEST FOR NO DRIVE

1131 00003782 745D <1> JZ short SM\_RTN ; RETURN IF SO

1132 00003784 E863000000 <1> CALL DR\_TYPE\_CHECK ; RTN CS:BX = MEDIA/DRIVE PARAM TBL

1133 00003789 7231 <1> JC short MD\_NOT\_FND ; TYPE NOT IN TABLE (BAD CMOS)

1134 0000378B 57 <1> PUSH eDI ; SAVE REG.

1135 0000378C 31DB <1> XOR eBX,eBX ; BX = INDEX TO DR. TYPE TABLE

1136 0000378E B906000000 <1> MOV eCX,DR\_CNT ; CX = LOOP COUNT

1137 <1> DR\_SEARCH:

1138 00003793 8AA3[705C0000] <1> MOV AH, [DR\_TYPE+eBX] ; GET DRIVE TYPE

1139 00003799 80E47F <1> AND AH,BIT7OFF ; MASK OUT MSB

1140 0000379C 38E0 <1> CMP AL,AH ; DRIVE TYPE MATCH ?

1141 0000379E 7516 <1> JNE short NXT\_MD ; NO, CHECK NEXT DRIVE TYPE

1142 <1> DR\_FND:

1143 000037A0 8BBB[715C0000] <1> MOV eDI, [DR\_TYPE+eBX+1] ; DI = MEDIA/DRIVE PARAM TABLE

1144 <1> MD\_SEARCH:

1145 000037A6 8A6704 <1> MOV AH, [eDI+MD.SEC\_TRK] ; GET SECTOR/TRACK

1146 000037A9 386500 <1> CMP [eBP],AH ; MATCH?

1147 000037AC 7508 <1> JNE short NXT\_MD ; NO, CHECK NEXT MEDIA

1148 000037AE 8A670B <1> MOV AH, [eDI+MD.MAX\_TRK] ; GET MAX. TRACK #

1149 000037B1 386501 <1> CMP [eBP+1],AH ; MATCH?

1150 000037B4 740F <1> JE short MD\_FND ; YES, GO GET RATE

1151 <1> NXT\_MD:

1152 <1> ;ADD BX,3 ; CHECK NEXT DRIVE TYPE

1153 000037B6 83C305 <1> add ebx, 5 ; 18/02/2015

1154 000037B9 E2D8 <1> LOOP DR\_SEARCH

1155 000037BB 5F <1> POP eDI ; RESTORE REG.

1156 <1> MD\_NOT\_FND:

1157 000037BC C605[C0580100]0C <1> MOV byte [DSKETTE\_STATUS], MED\_NOT\_FND ; ERROR, MEDIA TYPE NOT FOUND

1158 000037C3 EB1C <1> JMP SHORT SM\_RTN ; RETURN

1159 <1> MD\_FND:

1160 000037C5 8A470C <1> MOV AL, [eDI+MD.RATE] ; GET RATE

1161 000037C8 3C40 <1> CMP AL,RATE\_300 ; DOUBLE STEP REQUIRED FOR RATE 300

1162 000037CA 7502 <1> JNE short MD\_SET

1163 000037CC 0C20 <1> OR AL,DBL\_STEP

1164 <1> MD\_SET:

1165 <1> ;MOV [BP+6],DI ; SAVE TABLE POINTER IN STACK

1166 000037CE 897D0C <1> mov [ebp+12], edi ; 18/02/2015

1167 000037D1 0C10 <1> OR AL,MED\_DET ; SET MEDIA ESTABLISHED

1168 000037D3 5F <1> POP eDI

1169 000037D4 80A7[CD580100]0F <1> AND byte [DSK\_STATE+eDI], ~(MED\_DET+DBL\_STEP+RATE\_MSK) ; CLEAR STATE

1170 000037DB 0887[CD580100] <1> OR [DSK\_STATE+eDI], AL

1171 <1> ;MOV AX, CS ; SEGMENT OF MEDIA/DRIVE PARAMETER TABLE

1172 <1> ;MOV ES, AX ; ES IS SEGMENT OF TABLE

1173 <1> SM\_RTN:

1174 000037E1 E8AB000000 <1> CALL XLAT\_OLD ; TRANSLATE STATE TO COMPATIBLE MODE

1175 000037E6 E8E5050000 <1> CALL SETUP\_END ; VARIOUS CLEANUPS

1176 000037EB C3 <1> RETn

1177 <1>

1178 <1> ;----------------------------------------------------------------

1179 <1> ; DR\_TYPE\_CHECK :

1180 <1> ; CHECK IF THE GIVEN DRIVE TYPE IN REGISTER (AL) :

1181 <1> ; IS SUPPORTED IN BIOS DRIVE TYPE TABLE :

1182 <1> ; ON ENTRY: :

1183 <1> ; AL = DRIVE TYPE :

1184 <1> ; ON EXIT: :

1185 <1> ; CS = SEGMENT MEDIA/DRIVE PARAMETER TABLE (CODE) :

1186 <1> ; CY = 0 DRIVE TYPE SUPPORTED :

1187 <1> ; BX = OFFSET TO MEDIA/DRIVE PARAMETER TABLE :

1188 <1> ; CY = 1 DRIVE TYPE NOT SUPPORTED :

1189 <1> ; REGISTERS ALTERED: eBX :

1190 <1> ;----------------------------------------------------------------

1191 <1> DR\_TYPE\_CHECK:

1192 000037EC 6650 <1> PUSH AX

1193 000037EE 51 <1> PUSH eCX

1194 000037EF 31DB <1> XOR eBX,eBX ; BX = INDEX TO DR\_TYPE TABLE

1195 000037F1 B906000000 <1> MOV eCX,DR\_CNT ; CX = LOOP COUNT

1196 <1> TYPE\_CHK:

1197 000037F6 8AA3[705C0000] <1> MOV AH,[DR\_TYPE+eBX] ; GET DRIVE TYPE

1198 000037FC 38E0 <1> CMP AL,AH ; DRIVE TYPE MATCH?

1199 000037FE 740D <1> JE short DR\_TYPE\_VALID ; YES, RETURN WITH CARRY RESET

1200 <1> ;ADD BX,3 ; CHECK NEXT DRIVE TYPE

1201 00003800 83C305 <1> add ebx, 5 ; 16/02/2015 (32 bit address modification)

1202 00003803 E2F1 <1> LOOP TYPE\_CHK

1203 <1> ;

1204 00003805 BB[CF5C0000] <1> mov ebx, MD\_TBL6 ; 1.44MB fd parameter table

1205 <1> ; Default for GET\_PARM (11/12/2014)

1206 <1> ;

1207 0000380A F9 <1> STC ; DRIVE TYPE NOT FOUND IN TABLE

1208 0000380B EB06 <1> JMP SHORT TYPE\_RTN

1209 <1> DR\_TYPE\_VALID:

1210 0000380D 8B9B[715C0000] <1> MOV eBX,[DR\_TYPE+eBX+1] ; BX = MEDIA TABLE

1211 <1> TYPE\_RTN:

1212 00003813 59 <1> POP eCX

1213 00003814 6658 <1> POP AX

1214 00003816 C3 <1> RETn

1215 <1>

1216 <1> ;----------------------------------------------------------------

1217 <1> ; SEND\_SPEC :

1218 <1> ; SEND THE SPECIFY COMMAND TO CONTROLLER USING DATA FROM :

1219 <1> ; THE DRIVE PARAMETER TABLE POINTED BY @DISK\_POINTER :

1220 <1> ; ON ENTRY: @DISK\_POINTER = DRIVE PARAMETER TABLE :

1221 <1> ; ON EXIT: NONE :

1222 <1> ; REGISTERS ALTERED: CX, DX :

1223 <1> ;----------------------------------------------------------------

1224 <1> SEND\_SPEC:

1225 00003817 50 <1> PUSH eAX ; SAVE AX

1226 00003818 B8[3E380000] <1> MOV eAX, SPECBAC ; LOAD ERROR ADDRESS

1227 0000381D 50 <1> PUSH eAX ; PUSH NEC\_OUT ERROR RETURN

1228 0000381E B403 <1> MOV AH,03H ; SPECIFY COMMAND

1229 00003820 E885070000 <1> CALL NEC\_OUTPUT ; OUTPUT THE COMMAND

1230 00003825 28D2 <1> SUB DL,DL ; FIRST SPECIFY BYTE

1231 00003827 E878060000 <1> CALL GET\_PARM ; GET PARAMETER TO AH

1232 0000382C E879070000 <1> CALL NEC\_OUTPUT ; OUTPUT THE COMMAND

1233 00003831 B201 <1> MOV DL,1 ; SECOND SPECIFY BYTE

1234 00003833 E86C060000 <1> CALL GET\_PARM ; GET PARAMETER TO AH

1235 00003838 E86D070000 <1> CALL NEC\_OUTPUT ; OUTPUT THE COMMAND

1236 0000383D 58 <1> POP eAX ; POP ERROR RETURN

1237 <1> SPECBAC:

1238 0000383E 58 <1> POP eAX ; RESTORE ORIGINAL AX VALUE

1239 0000383F C3 <1> RETn

1240 <1>

1241 <1> ;----------------------------------------------------------------

1242 <1> ; SEND\_SPEC\_MD :

1243 <1> ; SEND THE SPECIFY COMMAND TO CONTROLLER USING DATA FROM :

1244 <1> ; THE MEDIA/DRIVE PARAMETER TABLE POINTED BY (CS:BX) :

1245 <1> ; ON ENTRY: CS:BX = MEDIA/DRIVE PARAMETER TABLE :

1246 <1> ; ON EXIT: NONE :

1247 <1> ; REGISTERS ALTERED: AX :

1248 <1> ;----------------------------------------------------------------

1249 <1> SEND\_SPEC\_MD:

1250 00003840 50 <1> PUSH eAX ; SAVE RATE DATA

1251 00003841 B8[5E380000] <1> MOV eAX, SPEC\_ESBAC ; LOAD ERROR ADDRESS

1252 00003846 50 <1> PUSH eAX ; PUSH NEC\_OUT ERROR RETURN

1253 00003847 B403 <1> MOV AH,03H ; SPECIFY COMMAND

1254 00003849 E85C070000 <1> CALL NEC\_OUTPUT ; OUTPUT THE COMMAND

1255 0000384E 8A23 <1> MOV AH, [eBX+MD.SPEC1] ; GET 1ST SPECIFY BYTE

1256 00003850 E855070000 <1> CALL NEC\_OUTPUT ; OUTPUT THE COMMAND

1257 00003855 8A6301 <1> MOV AH, [eBX+MD.SPEC2] ; GET SECOND SPECIFY BYTE

1258 00003858 E84D070000 <1> CALL NEC\_OUTPUT ; OUTPUT THE COMMAND

1259 0000385D 58 <1> POP eAX ; POP ERROR RETURN

1260 <1> SPEC\_ESBAC:

1261 0000385E 58 <1> POP eAX ; RESTORE ORIGINAL AX VALUE

1262 0000385F C3 <1> RETn

1263 <1>

1264 <1> ;-------------------------------------------------------------------------------

1265 <1> ; XLAT\_NEW

1266 <1> ; TRANSLATES DISKETTE STATE LOCATIONS FROM COMPATIBLE

1267 <1> ; MODE TO NEW ARCHITECTURE.

1268 <1> ;

1269 <1> ; ON ENTRY: DI = DRIVE #

1270 <1> ;-------------------------------------------------------------------------------

1271 <1> XLAT\_NEW:

1272 00003860 83FF01 <1> CMP eDI,1 ; VALID DRIVE

1273 00003863 7725 <1> JA short XN\_OUT ; IF INVALID BACK

1274 00003865 80BF[CD580100]00 <1> CMP byte [DSK\_STATE+eDI], 0 ; NO DRIVE ?

1275 0000386C 741D <1> JZ short DO\_DET ; IF NO DRIVE ATTEMPT DETERMINE

1276 0000386E 6689F9 <1> MOV CX,DI ; CX = DRIVE NUMBER

1277 00003871 C0E102 <1> SHL CL,2 ; CL = SHIFT COUNT, A=0, B=4

1278 00003874 A0[CC580100] <1> MOV AL, [HF\_CNTRL] ; DRIVE INFORMATION

1279 00003879 D2C8 <1> ROR AL,CL ; TO LOW NIBBLE

1280 0000387B 2407 <1> AND AL,DRV\_DET+FMT\_CAPA+TRK\_CAPA ; KEEP DRIVE BITS

1281 0000387D 80A7[CD580100]F8 <1> AND byte [DSK\_STATE+eDI], ~(DRV\_DET+FMT\_CAPA+TRK\_CAPA)

1282 00003884 0887[CD580100] <1> OR [DSK\_STATE+eDI], AL ; UPDATE DRIVE STATE

1283 <1> XN\_OUT:

1284 0000388A C3 <1> RETn

1285 <1> DO\_DET:

1286 0000388B E8BF080000 <1> CALL DRIVE\_DET ; TRY TO DETERMINE

1287 00003890 C3 <1> RETn

1288 <1>

1289 <1> ;-------------------------------------------------------------------------------

1290 <1> ; XLAT\_OLD

1291 <1> ; TRANSLATES DISKETTE STATE LOCATIONS FROM NEW

1292 <1> ; ARCHITECTURE TO COMPATIBLE MODE.

1293 <1> ;

1294 <1> ; ON ENTRY: DI = DRIVE

1295 <1> ;-------------------------------------------------------------------------------

1296 <1> XLAT\_OLD:

1297 00003891 83FF01 <1> CMP eDI,1 ; VALID DRIVE ?

1298 <1> ;JA short XO\_OUT ; IF INVALID BACK

1299 00003894 0F8786000000 <1> ja XO\_OUT

1300 0000389A 80BF[CD580100]00 <1> CMP byte [DSK\_STATE+eDI],0 ; NO DRIVE ?

1301 000038A1 747D <1> JZ short XO\_OUT ; IF NO DRIVE TRANSLATE DONE

1302 <1>

1303 <1> ;----- TEST FOR SAVED DRIVE INFORMATION ALREADY SET

1304 <1>

1305 000038A3 6689F9 <1> MOV CX,DI ; CX = DRIVE NUMBER

1306 000038A6 C0E102 <1> SHL CL,2 ; CL = SHIFT COUNT, A=0, B=4

1307 000038A9 B402 <1> MOV AH,FMT\_CAPA ; LOAD MULTIPLE DATA RATE BIT MASK

1308 000038AB D2CC <1> ROR AH,CL ; ROTATE BY MASK

1309 000038AD 8425[CC580100] <1> TEST [HF\_CNTRL], AH ; MULTIPLE-DATA RATE DETERMINED ?

1310 000038B3 751C <1> JNZ short SAVE\_SET ; IF SO, NO NEED TO RE-SAVE

1311 <1>

1312 <1> ;----- ERASE DRIVE BITS IN @HF\_CNTRL FOR THIS DRIVE

1313 <1>

1314 000038B5 B407 <1> MOV AH,DRV\_DET+FMT\_CAPA+TRK\_CAPA ; MASK TO KEEP

1315 000038B7 D2CC <1> ROR AH,CL ; FIX MASK TO KEEP

1316 000038B9 F6D4 <1> NOT AH ; TRANSLATE MASK

1317 000038BB 2025[CC580100] <1> AND [HF\_CNTRL], AH ; KEEP BITS FROM OTHER DRIVE INTACT

1318 <1>

1319 <1> ;----- ACCESS CURRENT DRIVE BITS AND STORE IN @HF\_CNTRL

1320 <1>

1321 000038C1 8A87[CD580100] <1> MOV AL, [DSK\_STATE+eDI] ; ACCESS STATE

1322 000038C7 2407 <1> AND AL,DRV\_DET+FMT\_CAPA+TRK\_CAPA ; KEEP DRIVE BITS

1323 000038C9 D2C8 <1> ROR AL,CL ; FIX FOR THIS DRIVE

1324 000038CB 0805[CC580100] <1> OR [HF\_CNTRL], AL ; UPDATE SAVED DRIVE STATE

1325 <1>

1326 <1> ;----- TRANSLATE TO COMPATIBILITY MODE

1327 <1>

1328 <1> SAVE\_SET:

1329 000038D1 8AA7[CD580100] <1> MOV AH, [DSK\_STATE+eDI] ; ACCESS STATE

1330 000038D7 88E7 <1> MOV BH,AH ; TO BH FOR LATER

1331 000038D9 80E4C0 <1> AND AH,RATE\_MSK ; KEEP ONLY RATE

1332 000038DC 80FC00 <1> CMP AH,RATE\_500 ; RATE 500 ?

1333 000038DF 7410 <1> JZ short CHK\_144 ; YES 1.2/1.2 OR 1.44/1.44

1334 000038E1 B001 <1> MOV AL,M3D1U ; AL = 360 IN 1.2 UNESTABLISHED

1335 000038E3 80FC40 <1> CMP AH,RATE\_300 ; RATE 300 ?

1336 000038E6 7518 <1> JNZ short CHK\_250 ; NO, 360/360, 720/720 OR 720/1.44

1337 000038E8 F6C720 <1> TEST BH,DBL\_STEP ; CHECK FOR DOUBLE STEP

1338 000038EB 751F <1> JNZ short TST\_DET ; MUST BE 360 IN 1.2

1339 <1> UNKNO:

1340 000038ED B007 <1> MOV AL,MED\_UNK ; NONE OF THE ABOVE

1341 000038EF EB22 <1> JMP SHORT AL\_SET ; PROCESS COMPLETE

1342 <1> CHK\_144:

1343 000038F1 E8A5050000 <1> CALL CMOS\_TYPE ; RETURN DRIVE TYPE IN (AL)

1344 <1> ;;20/02/2015

1345 <1> ;;JC short UNKNO ; ERROR, SET 'NONE OF ABOVE'

1346 000038F6 74F5 <1> jz short UNKNO ;; 20/02/2015

1347 000038F8 3C02 <1> CMP AL,2 ; 1.2MB DRIVE ?

1348 000038FA 75F1 <1> JNE short UNKNO ; NO, GO SET 'NONE OF ABOVE'

1349 000038FC B002 <1> MOV AL,M1D1U ; AL = 1.2 IN 1.2 UNESTABLISHED

1350 000038FE EB0C <1> JMP SHORT TST\_DET

1351 <1> CHK\_250:

1352 00003900 B000 <1> MOV AL,M3D3U ; AL = 360 IN 360 UNESTABLISHED

1353 00003902 80FC80 <1> CMP AH,RATE\_250 ; RATE 250 ?

1354 00003905 75E6 <1> JNZ short UNKNO ; IF SO FALL IHRU

1355 00003907 F6C701 <1> TEST BH,TRK\_CAPA ; 80 TRACK CAPABILITY ?

1356 0000390A 75E1 <1> JNZ short UNKNO ; IF SO JUMP, FALL THRU TEST DET

1357 <1> TST\_DET:

1358 0000390C F6C710 <1> TEST BH,MED\_DET ; DETERMINED ?

1359 0000390F 7402 <1> JZ short AL\_SET ; IF NOT THEN SET

1360 00003911 0403 <1> ADD AL,3 ; MAKE DETERMINED/ESTABLISHED

1361 <1> AL\_SET:

1362 00003913 80A7[CD580100]F8 <1> AND byte [DSK\_STATE+eDI], ~(DRV\_DET+FMT\_CAPA+TRK\_CAPA) ; CLEAR DRIVE

1363 0000391A 0887[CD580100] <1> OR [DSK\_STATE+eDI], AL ; REPLACE WITH COMPATIBLE MODE

1364 <1> XO\_OUT:

1365 00003920 C3 <1> RETn

1366 <1>

1367 <1> ;-------------------------------------------------------------------------------

1368 <1> ; RD\_WR\_VF

1369 <1> ; COMMON READ, WRITE AND VERIFY:

1370 <1> ; MAIN LOOP FOR STATE RETRIES.

1371 <1> ;

1372 <1> ; ON ENTRY: AH = READ/WRITE/VERIFY NEC PARAMETER

1373 <1> ; AL = READ/WRITE/VERIFY DMA PARAMETER

1374 <1> ;

1375 <1> ; ON EXIT: @DSKETTE\_STATUS, CY REFLECT STATUS OF OPERATION

1376 <1> ;-------------------------------------------------------------------------------

1377 <1> RD\_WR\_VF:

1378 00003921 6650 <1> PUSH AX ; SAVE DMA, NEC PARAMETERS

1379 00003923 E838FFFFFF <1> CALL XLAT\_NEW ; TRANSLATE STATE TO PRESENT ARCH.

1380 00003928 E8F3000000 <1> CALL SETUP\_STATE ; INITIALIZE START AND END RATE

1381 0000392D 6658 <1> POP AX ; RESTORE READ/WRITE/VERIFY

1382 <1> DO\_AGAIN:

1383 0000392F 6650 <1> PUSH AX ; SAVE READ/WRITE/VERIFY PARAMETER

1384 00003931 E87F010000 <1> CALL MED\_CHANGE ; MEDIA CHANGE AND RESET IF CHANGED

1385 00003936 6658 <1> POP AX ; RESTORE READ/WRITE/VERIFY

1386 00003938 0F82C9000000 <1> JC RWV\_END ; MEDIA CHANGE ERROR OR TIME-OUT

1387 <1> RWV:

1388 0000393E 6650 <1> PUSH AX ; SAVE READ/WRITE/VERIFY PARAMETER

1389 00003940 8AB7[CD580100] <1> MOV DH, [DSK\_STATE+eDI] ; GET RATE STATE OF THIS DRIVE

1390 00003946 80E6C0 <1> AND DH,RATE\_MSK ; KEEP ONLY RATE

1391 00003949 E84D050000 <1> CALL CMOS\_TYPE ; RETURN DRIVE TYPE IN AL (AL)

1392 <1> ;;20/02/2015

1393 <1> ;;JC short RWV\_ASSUME ; ERROR IN CMOS

1394 0000394E 7451 <1> jz short RWV\_ASSUME ; 20/02/2015

1395 00003950 3C01 <1> CMP AL,1 ; 40 TRACK DRIVE?

1396 00003952 750D <1> JNE short RWV\_1 ; NO, BYPASS CMOS VALIDITY CHECK

1397 00003954 F687[CD580100]01 <1> TEST byte [DSK\_STATE+eDI], TRK\_CAPA ; CHECK FOR 40 TRACK DRIVE

1398 0000395B 7413 <1> JZ short RWV\_2 ; YES, CMOS IS CORRECT

1399 0000395D B002 <1> MOV AL,2 ; CHANGE TO 1.2M

1400 0000395F EB0F <1> JMP SHORT RWV\_2

1401 <1> RWV\_1:

1402 00003961 720D <1> JB short RWV\_2 ; NO DRIVE SPECIFIED, CONTINUE

1403 00003963 F687[CD580100]01 <1> TEST byte [DSK\_STATE+eDI], TRK\_CAPA ; IS IT REALLY 40 TRACK?

1404 0000396A 7504 <1> JNZ short RWV\_2 ; NO, 80 TRACK

1405 0000396C B001 <1> MOV AL,1 ; IT IS 40 TRACK, FIX CMOS VALUE

1406 0000396E EB04 <1> jmp short rwv\_3

1407 <1> RWV\_2:

1408 00003970 08C0 <1> OR AL,AL ; TEST FOR NO DRIVE

1409 00003972 742D <1> JZ short RWV\_ASSUME ; ASSUME TYPE, USE MAX TRACK

1410 <1> rwv\_3:

1411 00003974 E873FEFFFF <1> CALL DR\_TYPE\_CHECK ; RTN CS:BX = MEDIA/DRIVE PARAM TBL.

1412 00003979 7226 <1> JC short RWV\_ASSUME ; TYPE NOT IN TABLE (BAD CMOS)

1413 <1>

1414 <1> ;----- SEARCH FOR MEDIA/DRIVE PARAMETER TABLE

1415 <1>

1416 0000397B 57 <1> PUSH eDI ; SAVE DRIVE #

1417 0000397C 31DB <1> XOR eBX,eBX ; BX = INDEX TO DR\_TYPE TABLE

1418 0000397E B906000000 <1> MOV eCX,DR\_CNT ; CX = LOOP COUNT

1419 <1> RWV\_DR\_SEARCH:

1420 00003983 8AA3[705C0000] <1> MOV AH, [DR\_TYPE+eBX] ; GET DRIVE TYPE

1421 00003989 80E47F <1> AND AH,BIT7OFF ; MASK OUT MSB

1422 0000398C 38E0 <1> CMP AL,AH ; DRIVE TYPE MATCH?

1423 0000398E 750B <1> JNE short RWV\_NXT\_MD ; NO, CHECK NEXT DRIVE TYPE

1424 <1> RWV\_DR\_FND:

1425 00003990 8BBB[715C0000] <1> MOV eDI, [DR\_TYPE+eBX+1] ; DI = MEDIA/DRIVE PARAMETER TABLE

1426 <1> RWV\_MD\_SEARH:

1427 00003996 3A770C <1> CMP DH, [eDI+MD.RATE] ; MATCH?

1428 00003999 741B <1> JE short RWV\_MD\_FND ; YES, GO GET 1ST SPECIFY BYTE

1429 <1> RWV\_NXT\_MD:

1430 <1> ;ADD BX,3 ; CHECK NEXT DRIVE TYPE

1431 0000399B 83C305 <1> add eBX, 5

1432 0000399E E2E3 <1> LOOP RWV\_DR\_SEARCH

1433 000039A0 5F <1> POP eDI ; RESTORE DRIVE #

1434 <1>

1435 <1> ;----- ASSUME PRIMARY DRIVE IS INSTALLED AS SHIPPED

1436 <1>

1437 <1> RWV\_ASSUME:

1438 000039A1 BB[8E5C0000] <1> MOV eBX, MD\_TBL1 ; POINT TO 40 TRACK 250 KBS

1439 000039A6 F687[CD580100]01 <1> TEST byte [DSK\_STATE+eDI], TRK\_CAPA ; TEST FOR 80 TRACK

1440 000039AD 740A <1> JZ short RWV\_MD\_FND1 ; MUST BE 40 TRACK

1441 000039AF BB[A85C0000] <1> MOV eBX, MD\_TBL3 ; POINT TO 80 TRACK 500 KBS

1442 000039B4 EB03 <1> JMP short RWV\_MD\_FND1 ; GO SPECIFY PARAMTERS

1443 <1>

1444 <1> ;----- CS:BX POINTS TO MEDIA/DRIVE PARAMETER TABLE

1445 <1>

1446 <1> RWV\_MD\_FND:

1447 000039B6 89FB <1> MOV eBX,eDI ; BX = MEDIA/DRIVE PARAMETER TABLE

1448 000039B8 5F <1> POP eDI ; RESTORE DRIVE #

1449 <1>

1450 <1> ;----- SEND THE SPECIFY COMMAND TO THE CONTROLLER

1451 <1>

1452 <1> RWV\_MD\_FND1:

1453 000039B9 E882FEFFFF <1> CALL SEND\_SPEC\_MD

1454 000039BE E864010000 <1> CALL CHK\_LASTRATE ; ZF=1 ATTEMP RATE IS SAME AS LAST RATE

1455 000039C3 7405 <1> JZ short RWV\_DBL ; YES,SKIP SEND RATE COMMAND

1456 000039C5 E83B010000 <1> CALL SEND\_RATE ; SEND DATA RATE TO NEC

1457 <1> RWV\_DBL:

1458 000039CA 53 <1> PUSH eBX ; SAVE MEDIA/DRIVE PARAM TBL ADDRESS

1459 000039CB E822040000 <1> CALL SETUP\_DBL ; CHECK FOR DOUBLE STEP

1460 000039D0 5B <1> POP eBX ; RESTORE ADDRESS

1461 000039D1 7226 <1> JC short CHK\_RET ; ERROR FROM READ ID, POSSIBLE RETRY

1462 000039D3 6658 <1> POP AX ; RESTORE NEC, DMA COMMAND

1463 000039D5 6650 <1> PUSH AX ; SAVE NEC COMMAND

1464 000039D7 53 <1> PUSH eBX ; SAVE MEDIA/DRIVE PARAM TBL ADDRESS

1465 000039D8 E861010000 <1> CALL DMA\_SETUP ; SET UP THE DMA

1466 000039DD 5B <1> POP eBX

1467 000039DE 6658 <1> POP AX ; RESTORE NEC COMMAND

1468 000039E0 722F <1> JC short RWV\_BAC ; CHECK FOR DMA BOUNDARY ERROR

1469 000039E2 6650 <1> PUSH AX ; SAVE NEC COMMAND

1470 000039E4 53 <1> PUSH eBX ; SAVE MEDIA/DRIVE PARAM TBL ADDRESS

1471 000039E5 E83C020000 <1> CALL NEC\_INIT ; INITIALIZE NEC

1472 000039EA 5B <1> POP eBX ; RESTORE ADDRESS

1473 000039EB 720C <1> JC short CHK\_RET ; ERROR - EXIT

1474 000039ED E866020000 <1> CALL RWV\_COM ; OP CODE COMMON TO READ/WRITE/VERIFY

1475 000039F2 7205 <1> JC short CHK\_RET ; ERROR - EXIT

1476 000039F4 E8AB020000 <1> CALL NEC\_TERM ; TERMINATE, GET STATUS, ETC.

1477 <1> CHK\_RET:

1478 000039F9 E84A030000 <1> CALL RETRY ; CHECK FOR, SETUP RETRY

1479 000039FE 6658 <1> POP AX ; RESTORE READ/WRITE/VERIFY PARAMETER

1480 00003A00 7305 <1> JNC short RWV\_END ; CY = 0 NO RETRY

1481 00003A02 E928FFFFFF <1> JMP DO\_AGAIN ; CY = 1 MEANS RETRY

1482 <1> RWV\_END:

1483 00003A07 E8F4020000 <1> CALL DSTATE ; ESTABLISH STATE IF SUCCESSFUL

1484 00003A0C E887030000 <1> CALL NUM\_TRANS ; AL = NUMBER TRANSFERRED

1485 <1> RWV\_BAC: ; BAD DMA ERROR ENTRY

1486 00003A11 6650 <1> PUSH AX ; SAVE NUMBER TRANSFERRED

1487 00003A13 E879FEFFFF <1> CALL XLAT\_OLD ; TRANSLATE STATE TO COMPATIBLE MODE

1488 00003A18 6658 <1> POP AX ; RESTORE NUMBER TRANSFERRED

1489 00003A1A E8B1030000 <1> CALL SETUP\_END ; VARIOUS CLEANUPS

1490 00003A1F C3 <1> RETn

1491 <1>

1492 <1> ;-------------------------------------------------------------------------------

1493 <1> ; SETUP\_STATE: INITIALIZES START AND END RATES.

1494 <1> ;-------------------------------------------------------------------------------

1495 <1> SETUP\_STATE:

1496 00003A20 F687[CD580100]10 <1> TEST byte [DSK\_STATE+eDI], MED\_DET ; MEDIA DETERMINED ?

1497 00003A27 7537 <1> JNZ short J1C ; NO STATES IF DETERMINED

1498 00003A29 66B84000 <1> MOV AX,(RATE\_500\*256)+RATE\_300 ; AH = START RATE, AL = END RATE

1499 00003A2D F687[CD580100]04 <1> TEST byte [DSK\_STATE+eDI],DRV\_DET ; DRIVE ?

1500 00003A34 740D <1> JZ short AX\_SET ; DO NOT KNOW DRIVE

1501 00003A36 F687[CD580100]02 <1> TEST byte [DSK\_STATE+eDI], FMT\_CAPA ; MULTI-RATE?

1502 00003A3D 7504 <1> JNZ short AX\_SET ; JUMP IF YES

1503 00003A3F 66B88080 <1> MOV AX,RATE\_250\*257 ; START A END RATE 250 FOR 360 DRIVE

1504 <1> AX\_SET:

1505 00003A43 80A7[CD580100]1F <1> AND byte [DSK\_STATE+eDI], ~(RATE\_MSK+DBL\_STEP) ; TURN OFF THE RATE

1506 00003A4A 08A7[CD580100] <1> OR [DSK\_STATE+eDI], AH ; RATE FIRST TO TRY

1507 00003A50 8025[C8580100]F3 <1> AND byte [LASTRATE], ~STRT\_MSK ; ERASE LAST TO TRY RATE BITS

1508 00003A57 C0C804 <1> ROR AL,4 ; TO OPERATION LAST RATE LOCATION

1509 00003A5A 0805[C8580100] <1> OR [LASTRATE], AL ; LAST RATE

1510 <1> J1C:

1511 00003A60 C3 <1> RETn

1512 <1>

1513 <1> ;-------------------------------------------------------------------------------

1514 <1> ; FMT\_INIT: ESTABLISH STATE IF UNESTABLISHED AT FORMAT TIME.

1515 <1> ;-------------------------------------------------------------------------------

1516 <1> FMT\_INIT:

1517 00003A61 F687[CD580100]10 <1> TEST byte [DSK\_STATE+eDI], MED\_DET ; IS MEDIA ESTABLISHED

1518 00003A68 7546 <1> JNZ short F1\_OUT ; IF SO RETURN

1519 00003A6A E82C040000 <1> CALL CMOS\_TYPE ; RETURN DRIVE TYPE IN AL

1520 <1> ;; 20/02/2015

1521 <1> ;;JC short CL\_DRV ; ERROR IN CMOS ASSUME NO DRIVE

1522 00003A6F 7440 <1> jz short CL\_DRV ;; 20/02/2015

1523 00003A71 FEC8 <1> DEC AL ; MAKE ZERO ORIGIN

1524 <1> ;;JS short CL\_DRV ; NO DRIVE IF AL 0

1525 00003A73 8AA7[CD580100] <1> MOV AH, [DSK\_STATE+eDI] ; AH = CURRENT STATE

1526 00003A79 80E40F <1> AND AH, ~(MED\_DET+DBL\_STEP+RATE\_MSK) ; CLEAR

1527 00003A7C 08C0 <1> OR AL,AL ; CHECK FOR 360

1528 00003A7E 7505 <1> JNZ short N\_360 ; IF 360 WILL BE 0

1529 00003A80 80CC90 <1> OR AH,MED\_DET+RATE\_250 ; ESTABLISH MEDIA

1530 00003A83 EB25 <1> JMP SHORT SKP\_STATE ; SKIP OTHER STATE PROCESSING

1531 <1> N\_360:

1532 00003A85 FEC8 <1> DEC AL ; 1.2 M DRIVE

1533 00003A87 7505 <1> JNZ short N\_12 ; JUMP IF NOT

1534 <1> F1\_RATE:

1535 00003A89 80CC10 <1> OR AH,MED\_DET+RATE\_500 ; SET FORMAT RATE

1536 00003A8C EB1C <1> JMP SHORT SKP\_STATE ; SKIP OTHER STATE PROCESSING

1537 <1> N\_12:

1538 00003A8E FEC8 <1> DEC AL ; CHECK FOR TYPE 3

1539 00003A90 750F <1> JNZ short N\_720 ; JUMP IF NOT

1540 00003A92 F6C404 <1> TEST AH,DRV\_DET ; IS DRIVE DETERMINED

1541 00003A95 7410 <1> JZ short ISNT\_12 ; TREAT AS NON 1.2 DRIVE

1542 00003A97 F6C402 <1> TEST AH,FMT\_CAPA ; IS 1.2M

1543 00003A9A 740B <1> JZ short ISNT\_12 ; JUMP IF NOT

1544 00003A9C 80CC50 <1> OR AH,MED\_DET+RATE\_300 ; RATE 300

1545 00003A9F EB09 <1> JMP SHORT SKP\_STATE ; CONTINUE

1546 <1> N\_720:

1547 00003AA1 FEC8 <1> DEC AL ; CHECK FOR TYPE 4

1548 00003AA3 750C <1> JNZ short CL\_DRV ; NO DRIVE, CMOS BAD

1549 00003AA5 EBE2 <1> JMP SHORT F1\_RATE

1550 <1> ISNT\_12:

1551 00003AA7 80CC90 <1> OR AH,MED\_DET+RATE\_250 ; MUST BE RATE 250

1552 <1>

1553 <1> SKP\_STATE:

1554 00003AAA 88A7[CD580100] <1> MOV [DSK\_STATE+eDI], AH ; STORE AWAY

1555 <1> F1\_OUT:

1556 00003AB0 C3 <1> RETn

1557 <1> CL\_DRV:

1558 00003AB1 30E4 <1> XOR AH,AH ; CLEAR STATE

1559 00003AB3 EBF5 <1> JMP SHORT SKP\_STATE ; SAVE IT

1560 <1>

1561 <1> ;-------------------------------------------------------------------------------

1562 <1> ; MED\_CHANGE

1563 <1> ; CHECKS FOR MEDIA CHANGE, RESETS MEDIA CHANGE,

1564 <1> ; CHECKS MEDIA CHANGE AGAIN.

1565 <1> ;

1566 <1> ; ON EXIT: CY = 1 MEANS MEDIA CHANGE OR TIMEOUT

1567 <1> ; @DSKETTE\_STATUS = ERROR CODE

1568 <1> ;-------------------------------------------------------------------------------

1569 <1> MED\_CHANGE:

1570 00003AB5 E888060000 <1> CALL READ\_DSKCHNG ; READ DISK CHANCE LINE STATE

1571 00003ABA 7447 <1> JZ short MC\_OUT ; BYPASS HANDLING DISK CHANGE LINE

1572 00003ABC 80A7[CD580100]EF <1> AND byte [DSK\_STATE+eDI], ~MED\_DET ; CLEAR STATE FOR THIS DRIVE

1573 <1>

1574 <1> ; THIS SEQUENCE ENSURES WHENEVER A DISKETTE IS CHANGED THAT

1575 <1> ; ON THE NEXT OPERATION THE REQUIRED MOTOR START UP TIME WILL

1576 <1> ; BE WAITED. (DRIVE MOTOR MAY GO OFF UPON DOOR OPENING).

1577 <1>

1578 00003AC3 6689F9 <1> MOV CX,DI ; CL = DRIVE 0

1579 00003AC6 B001 <1> MOV AL,1 ; MOTOR ON BIT MASK

1580 00003AC8 D2E0 <1> SHL AL,CL ; TO APPROPRIATE POSITION

1581 00003ACA F6D0 <1> NOT AL ; KEEP ALL BUT MOTOR ON

1582 00003ACC FA <1> CLI ; NO INTERRUPTS

1583 00003ACD 2005[BE580100] <1> AND [MOTOR\_STATUS], AL ; TURN MOTOR OFF INDICATOR

1584 00003AD3 FB <1> STI ; INTERRUPTS ENABLED

1585 00003AD4 E810040000 <1> CALL MOTOR\_ON ; TURN MOTOR ON

1586 <1>

1587 <1> ;----- THIS SEQUENCE OF SEEKS IS USED TO RESET DISKETTE CHANGE SIGNAL

1588 <1>

1589 00003AD9 E86DF9FFFF <1> CALL DSK\_RESET ; RESET NEC

1590 00003ADE B501 <1> MOV CH,01H ; MOVE TO CYLINDER 1

1591 00003AE0 E8FF040000 <1> CALL SEEK ; ISSUE SEEK

1592 00003AE5 30ED <1> XOR CH,CH ; MOVE TO CYLINDER 0

1593 00003AE7 E8F8040000 <1> CALL SEEK ; ISSUE SEEK

1594 00003AEC C605[C0580100]06 <1> MOV byte [DSKETTE\_STATUS], MEDIA\_CHANGE ; STORE IN STATUS

1595 <1> OK1:

1596 00003AF3 E84A060000 <1> CALL READ\_DSKCHNG ; CHECK MEDIA CHANGED AGAIN

1597 00003AF8 7407 <1> JZ short OK2 ; IF ACTIVE, NO DISKETTE, TIMEOUT

1598 <1> OK4:

1599 00003AFA C605[C0580100]80 <1> MOV byte [DSKETTE\_STATUS], TIME\_OUT ; TIMEOUT IF DRIVE EMPTY

1600 <1> OK2:

1601 00003B01 F9 <1> STC ; MEDIA CHANGED, SET CY

1602 00003B02 C3 <1> RETn

1603 <1> MC\_OUT:

1604 00003B03 F8 <1> CLC ; NO MEDIA CHANGED, CLEAR CY

1605 00003B04 C3 <1> RETn

1606 <1>

1607 <1> ;-------------------------------------------------------------------------------

1608 <1> ; SEND\_RATE

1609 <1> ; SENDS DATA RATE COMMAND TO NEC

1610 <1> ; ON ENTRY: DI = DRIVE #

1611 <1> ; ON EXIT: NONE

1612 <1> ; REGISTERS ALTERED: DX

1613 <1> ;-------------------------------------------------------------------------------

1614 <1> SEND\_RATE:

1615 00003B05 6650 <1> PUSH AX ; SAVE REG.

1616 00003B07 8025[C8580100]3F <1> AND byte [LASTRATE], ~SEND\_MSK ; ELSE CLEAR LAST RATE ATTEMPTED

1617 00003B0E 8A87[CD580100] <1> MOV AL, [DSK\_STATE+eDI] ; GET RATE STATE OF THIS DRIVE

1618 00003B14 24C0 <1> AND AL,SEND\_MSK ; KEEP ONLY RATE BITS

1619 00003B16 0805[C8580100] <1> OR [LASTRATE], AL ; SAVE NEW RATE FOR NEXT CHECK

1620 00003B1C C0C002 <1> ROL AL,2 ; MOVE TO BIT OUTPUT POSITIONS

1621 00003B1F 66BAF703 <1> MOV DX,03F7H ; OUTPUT NEW DATA RATE

1622 00003B23 EE <1> OUT DX,AL

1623 00003B24 6658 <1> POP AX ; RESTORE REG.

1624 00003B26 C3 <1> RETn

1625 <1>

1626 <1> ;-------------------------------------------------------------------------------

1627 <1> ; CHK\_LASTRATE

1628 <1> ; CHECK PREVIOUS DATE RATE SNT TO THE CONTROLLER.

1629 <1> ; ON ENTRY:

1630 <1> ; DI = DRIVE #

1631 <1> ; ON EXIT:

1632 <1> ; ZF = 1 DATA RATE IS THE SAME AS THE LAST RATE SENT TO NEC

1633 <1> ; ZF = 0 DATA RATE IS DIFFERENT FROM LAST RATE

1634 <1> ; REGISTERS ALTERED: DX

1635 <1> ;-------------------------------------------------------------------------------

1636 <1> CHK\_LASTRATE:

1637 00003B27 6650 <1> PUSH AX ; SAVE REG

1638 00003B29 2225[C8580100] <1> AND AH, [LASTRATE] ; GET LAST DATA RATE SELECTED

1639 00003B2F 8A87[CD580100] <1> MOV AL, [DSK\_STATE+eDI] ; GET RATE STATE OF THIS DRIVE

1640 00003B35 6625C0C0 <1> AND AX, SEND\_MSK\*257 ; KEEP ONLY RATE BITS OF BOTH

1641 00003B39 38E0 <1> CMP AL, AH ; COMPARE TO PREVIOUSLY TRIED

1642 <1> ; ZF = 1 RATE IS THE SAME

1643 00003B3B 6658 <1> POP AX ; RESTORE REG.

1644 00003B3D C3 <1> RETn

1645 <1>

1646 <1> ;-------------------------------------------------------------------------------

1647 <1> ; DMA\_SETUP

1648 <1> ; THIS ROUTINE SETS UP THE DMA FOR READ/WRITE/VERIFY OPERATIONS.

1649 <1> ;

1650 <1> ; ON ENTRY: AL = DMA COMMAND

1651 <1> ;

1652 <1> ; ON EXIT: @DSKETTE\_STATUS, CY REFLECT STATUS OF OPERATION

1653 <1> ;-------------------------------------------------------------------------------

1654 <1>

1655 <1> ; SI = Head #, # of Sectors or DASD Type

1656 <1>

1657 <1> ; 22/08/2015

1658 <1> ; 08/02/2015 - Protected Mode Modification

1659 <1> ; 06/02/2015 - 07/02/2015

1660 <1> ; NOTE: Buffer address must be in 1st 16MB of Physical Memory (24 bit limit).

1661 <1> ; (DMA Addres = Physical Address)

1662 <1> ; (Retro UNIX 386 v1 Kernel/System Mode Virtual Address = Physical Address)

1663 <1> ;

1664 <1>

1665 <1>

1666 <1> ; 04/02/2016 (clc)

1667 <1> ; 20/02/2015 modification (source: AWARD BIOS 1999, DMA\_SETUP)

1668 <1> ; 16/12/2014 (IODELAY)

1669 <1>

1670 <1> DMA\_SETUP:

1671 <1>

1672 <1> ;; 20/02/2015

1673 00003B3E 8B5504 <1> mov edx, [ebp+4] ; Buffer address

1674 00003B41 F7C2000000FF <1> test edx, 0FF000000h ; 16 MB limit (22/08/2015, bugfix)

1675 00003B47 756E <1> jnz short dma\_bnd\_err\_stc

1676 <1> ;

1677 00003B49 6650 <1> push ax ; DMA command

1678 00003B4B 52 <1> push edx ; \*

1679 00003B4C B203 <1> mov dl, 3 ; GET BYTES/SECTOR PARAMETER

1680 00003B4E E851030000 <1> call GET\_PARM ;

1681 00003B53 88E1 <1> mov cl, ah ; SHIFT COUNT (0=128, 1=256, 2=512 ETC)

1682 00003B55 6689F0 <1> mov ax, si ; Sector count

1683 00003B58 88C4 <1> mov ah, al ; AH = # OF SECTORS

1684 00003B5A 28C0 <1> sub al, al ; AL = 0, AX = # SECTORS \* 256

1685 00003B5C 66D1E8 <1> shr ax, 1 ; AX = # SECTORS \* 128

1686 00003B5F 66D3E0 <1> shl ax, cl ; SHIFT BY PARAMETER VALUE

1687 00003B62 6648 <1> dec ax ; -1 FOR DMA VALUE

1688 00003B64 6689C1 <1> mov cx, ax

1689 00003B67 5A <1> pop edx ; \*

1690 00003B68 6658 <1> pop ax

1691 00003B6A 3C42 <1> cmp al, 42h

1692 00003B6C 7507 <1> jne short NOT\_VERF

1693 00003B6E BA0000FF00 <1> mov edx, 0FF0000h

1694 00003B73 EB08 <1> jmp short J33

1695 <1> NOT\_VERF:

1696 00003B75 6601CA <1> add dx, cx ; check for overflow

1697 00003B78 723E <1> jc short dma\_bnd\_err

1698 <1> ;

1699 00003B7A 6629CA <1> sub dx, cx ; Restore start address

1700 <1> J33:

1701 00003B7D FA <1> CLI ; DISABLE INTERRUPTS DURING DMA SET-UP

1702 00003B7E E60C <1> OUT DMA+12,AL ; SET THE FIRST/LA5T F/F

1703 <1> IODELAY ; WAIT FOR I/O

1703 00003B80 EB00 <2> jmp short $+2

1703 00003B82 EB00 <2> jmp short $+2

1704 00003B84 E60B <1> OUT DMA+11,AL ; OUTPUT THE MODE BYTE

1705 00003B86 89D0 <1> mov eax, edx ; Buffer address

1706 00003B88 E604 <1> OUT DMA+4,AL ; OUTPUT LOW ADDRESS

1707 <1> IODELAY ; WAIT FOR I/O

1707 00003B8A EB00 <2> jmp short $+2

1707 00003B8C EB00 <2> jmp short $+2

1708 00003B8E 88E0 <1> MOV AL,AH

1709 00003B90 E604 <1> OUT DMA+4,AL ; OUTPUT HIGH ADDRESS

1710 00003B92 C1E810 <1> shr eax, 16

1711 <1> IODELAY ; I/O WAIT STATE

1711 00003B95 EB00 <2> jmp short $+2

1711 00003B97 EB00 <2> jmp short $+2

1712 00003B99 E681 <1> OUT 081H,AL ; OUTPUT highest BITS TO PAGE REGISTER

1713 <1> IODELAY

1713 00003B9B EB00 <2> jmp short $+2

1713 00003B9D EB00 <2> jmp short $+2

1714 00003B9F 6689C8 <1> mov ax, cx ; Byte count - 1

1715 00003BA2 E605 <1> OUT DMA+5,AL ; LOW BYTE OF COUNT

1716 <1> IODELAY ; WAIT FOR I/O

1716 00003BA4 EB00 <2> jmp short $+2

1716 00003BA6 EB00 <2> jmp short $+2

1717 00003BA8 88E0 <1> MOV AL, AH

1718 00003BAA E605 <1> OUT DMA+5,AL ; HIGH BYTE OF COUNT

1719 <1> IODELAY

1719 00003BAC EB00 <2> jmp short $+2

1719 00003BAE EB00 <2> jmp short $+2

1720 00003BB0 FB <1> STI ; RE-ENABLE INTERRUPTS

1721 00003BB1 B002 <1> MOV AL, 2 ; MODE FOR 8237

1722 00003BB3 E60A <1> OUT DMA+10, AL ; INITIALIZE THE DISKETTE CHANNEL

1723 <1>

1724 00003BB5 F8 <1> clc ; 04/02/2016

1725 00003BB6 C3 <1> retn

1726 <1>

1727 <1> dma\_bnd\_err\_stc:

1728 00003BB7 F9 <1> stc

1729 <1> dma\_bnd\_err:

1730 00003BB8 C605[C0580100]09 <1> MOV byte [DSKETTE\_STATUS], DMA\_BOUNDARY ; SET ERROR

1731 00003BBF C3 <1> RETn ; CY SET BY ABOVE IF ERROR

1732 <1>

1733 <1> ;; 16/12/2014

1734 <1> ;; CLI ; DISABLE INTERRUPTS DURING DMA SET-UP

1735 <1> ;; OUT DMA+12,AL ; SET THE FIRST/LA5T F/F

1736 <1> ;; ;JMP $+2 ; WAIT FOR I/O

1737 <1> ;; IODELAY

1738 <1> ;; OUT DMA+11,AL ; OUTPUT THE MODE BYTE

1739 <1> ;; ;SIODELAY

1740 <1> ;; ;CMP AL, 42H ; DMA VERIFY COMMAND

1741 <1> ;; ;JNE short NOT\_VERF ; NO

1742 <1> ;; ;XOR AX, AX ; START ADDRESS

1743 <1> ;; ;JMP SHORT J33

1744 <1> ;;;NOT\_VERF:

1745 <1> ;; ;MOV AX,ES ; GET THE ES VALUE

1746 <1> ;; ;ROL AX,4 ; ROTATE LEFT

1747 <1> ;; ;MOV CH,AL ; GET HIGHEST NIBBLE OF ES TO CH

1748 <1> ;; ;AND AL,11110000B ; ZERO THE LOW NIBBLE FROM SEGMENT

1749 <1> ;; ;ADD AX,[BP+2] ; TEST FOR CARRY FROM ADDITION

1750 <1> ;; mov eax, [ebp+4] ; 06/02/2015

1751 <1> ;; ;JNC short J33

1752 <1> ;; ;INC CH ; CARRY MEANS HIGH 4 BITS MUST BE INC

1753 <1> ;;;J33:

1754 <1> ;; PUSH eAX ; SAVE START ADDRESS

1755 <1> ;; OUT DMA+4,AL ; OUTPUT LOW ADDRESS

1756 <1> ;; ;JMP $+2 ; WAIT FOR I/O

1757 <1> ;; IODELAY

1758 <1> ;; MOV AL,AH

1759 <1> ;; OUT DMA+4,AL ; OUTPUT HIGH ADDRESS

1760 <1> ;; shr eax, 16 ; 07/02/2015

1761 <1> ;; ;MOV AL,CH ; GET HIGH 4 BITS

1762 <1> ;; ;JMP $+2 ; I/O WAIT STATE

1763 <1> ;; IODELAY

1764 <1> ;; ;AND AL,00001111B

1765 <1> ;; OUT 081H,AL ; OUTPUT HIGH 4 BITS TO PAGE REGISTER

1766 <1> ;; ;SIODELAY

1767 <1> ;;

1768 <1> ;;;----- DETERMINE COUNT

1769 <1> ;; sub eax, eax ; 08/02/2015

1770 <1> ;; MOV AX, SI ; AL = # OF SECTORS

1771 <1> ;; XCHG AL, AH ; AH = # OF SECTORS

1772 <1> ;; SUB AL, AL ; AL = 0, AX = # SECTORS \* 256

1773 <1> ;; SHR AX, 1 ; AX = # SECTORS \* 128

1774 <1> ;; PUSH AX ; SAVE # OF SECTORS \* 128

1775 <1> ;; MOV DL, 3 ; GET BYTES/SECTOR PARAMETER

1776 <1> ;; CALL GET\_PARM ; "

1777 <1> ;; MOV CL,AH ; SHIFT COUNT (0=128, 1=256, 2=512 ETC)

1778 <1> ;; POP AX ; AX = # SECTORS \* 128

1779 <1> ;; SHL AX,CL ; SHIFT BY PARAMETER VALUE

1780 <1> ;; DEC AX ; -1 FOR DMA VALUE

1781 <1> ;; PUSH eAX ; 08/02/2015 ; SAVE COUNT VALUE

1782 <1> ;; OUT DMA+5,AL ; LOW BYTE OF COUNT

1783 <1> ;; ;JMP $+2 ; WAIT FOR I/O

1784 <1> ;; IODELAY

1785 <1> ;; MOV AL, AH

1786 <1> ;; OUT DMA+5,AL ; HIGH BYTE OF COUNT

1787 <1> ;; ;IODELAY

1788 <1> ;; STI ; RE-ENABLE INTERRUPTS

1789 <1> ;; POP eCX ; 08/02/2015 ; RECOVER COUNT VALUE

1790 <1> ;; POP eAX ; 08/02/2015 ; RECOVER ADDRESS VALUE

1791 <1> ;; ;ADD AX, CX ; ADD, TEST FOR 64K OVERFLOW

1792 <1> ;; add ecx, eax ; 08/02/2015

1793 <1> ;; MOV AL, 2 ; MODE FOR 8237

1794 <1> ;; ;JMP $+2 ; WAIT FOR I/O

1795 <1> ;; SIODELAY

1796 <1> ;; OUT DMA+10, AL ; INITIALIZE THE DISKETTE CHANNEL

1797 <1> ;; ;JNC short NO\_BAD ; CHECK FOR ERROR

1798 <1> ;; jc short dma\_bnd\_err ; 08/02/2015

1799 <1> ;; and ecx, 0FFF00000h ; 16 MB limit

1800 <1> ;; jz short NO\_BAD

1801 <1> ;;dma\_bnd\_err:

1802 <1> ;; MOV byte [DSKETTE\_STATUS], DMA\_BOUNDARY ; SET ERROR

1803 <1> ;;NO\_BAD:

1804 <1> ;; RETn ; CY SET BY ABOVE IF ERROR

1805 <1>

1806 <1> ;-------------------------------------------------------------------------------

1807 <1> ; FMTDMA\_SET

1808 <1> ; THIS ROUTINE SETS UP THE DMA CONTROLLER FOR A FORMAT OPERATION.

1809 <1> ;

1810 <1> ; ON ENTRY: NOTHING REQUIRED

1811 <1> ;

1812 <1> ; ON EXIT: @DSKETTE\_STATUS, CY REFLECT STATUS OF OPERATION

1813 <1> ;-------------------------------------------------------------------------------

1814 <1>

1815 <1> FMTDMA\_SET:

1816 <1> ;; 20/02/2015 modification

1817 00003BC0 8B5504 <1> mov edx, [ebp+4] ; Buffer address

1818 00003BC3 F7C20000F0FF <1> test edx, 0FFF00000h ; 16 MB limit

1819 00003BC9 75EC <1> jnz short dma\_bnd\_err\_stc

1820 <1> ;

1821 00003BCB 6652 <1> push dx ; \*

1822 00003BCD B204 <1> mov DL, 4 ; SECTORS/TRACK VALUE IN PARM TABLE

1823 00003BCF E8D0020000 <1> call GET\_PARM ; "

1824 00003BD4 88E0 <1> mov al, ah ; AL = SECTORS/TRACK VALUE

1825 00003BD6 28E4 <1> sub ah, ah ; AX = SECTORS/TRACK VALUE

1826 00003BD8 66C1E002 <1> shl ax, 2 ; AX = SEC/TRK \* 4 (OFFSET C,H,R,N)

1827 00003BDC 6648 <1> dec ax ; -1 FOR DMA VALUE

1828 00003BDE 6689C1 <1> mov cx, ax

1829 00003BE1 665A <1> pop dx ; \*

1830 00003BE3 6601CA <1> add dx, cx ; check for overflow

1831 00003BE6 72D0 <1> jc short dma\_bnd\_err

1832 <1> ;

1833 00003BE8 6629CA <1> sub dx, cx ; Restore start address

1834 <1> ;

1835 00003BEB B04A <1> MOV AL, 04AH ; WILL WRITE TO THE DISKETTE

1836 00003BED FA <1> CLI ; DISABLE INTERRUPTS DURING DMA SET-UP

1837 00003BEE E60C <1> OUT DMA+12,AL ; SET THE FIRST/LA5T F/F

1838 <1> IODELAY ; WAIT FOR I/O

1838 00003BF0 EB00 <2> jmp short $+2

1838 00003BF2 EB00 <2> jmp short $+2

1839 00003BF4 E60B <1> OUT DMA+11,AL ; OUTPUT THE MODE BYTE

1840 00003BF6 89D0 <1> mov eax, edx ; Buffer address

1841 00003BF8 E604 <1> OUT DMA+4,AL ; OUTPUT LOW ADDRESS

1842 <1> IODELAY ; WAIT FOR I/O

1842 00003BFA EB00 <2> jmp short $+2

1842 00003BFC EB00 <2> jmp short $+2

1843 00003BFE 88E0 <1> MOV AL,AH

1844 00003C00 E604 <1> OUT DMA+4,AL ; OUTPUT HIGH ADDRESS

1845 00003C02 C1E810 <1> shr eax, 16

1846 <1> IODELAY ; I/O WAIT STATE

1846 00003C05 EB00 <2> jmp short $+2

1846 00003C07 EB00 <2> jmp short $+2

1847 00003C09 E681 <1> OUT 081H,AL ; OUTPUT highest BITS TO PAGE REGISTER

1848 <1> IODELAY

1848 00003C0B EB00 <2> jmp short $+2

1848 00003C0D EB00 <2> jmp short $+2

1849 00003C0F 6689C8 <1> mov ax, cx ; Byte count - 1

1850 00003C12 E605 <1> OUT DMA+5,AL ; LOW BYTE OF COUNT

1851 <1> IODELAY ; WAIT FOR I/O

1851 00003C14 EB00 <2> jmp short $+2

1851 00003C16 EB00 <2> jmp short $+2

1852 00003C18 88E0 <1> MOV AL, AH

1853 00003C1A E605 <1> OUT DMA+5,AL ; HIGH BYTE OF COUNT

1854 <1> IODELAY

1854 00003C1C EB00 <2> jmp short $+2

1854 00003C1E EB00 <2> jmp short $+2

1855 00003C20 FB <1> STI ; RE-ENABLE INTERRUPTS

1856 00003C21 B002 <1> MOV AL, 2 ; MODE FOR 8237

1857 00003C23 E60A <1> OUT DMA+10, AL ; INITIALIZE THE DISKETTE CHANNEL

1858 00003C25 C3 <1> retn

1859 <1>

1860 <1> ;; 08/02/2015 - Protected Mode Modification

1861 <1> ;; MOV AL, 04AH ; WILL WRITE TO THE DISKETTE

1862 <1> ;; CLI ; DISABLE INTERRUPTS DURING DMA SET-UP

1863 <1> ;; OUT DMA+12,AL ; SET THE FIRST/LA5T F/F

1864 <1> ;; ;JMP $+2 ; WAIT FOR I/O

1865 <1> ;; IODELAY

1866 <1> ;; OUT DMA+11,AL ; OUTPUT THE MODE BYTE

1867 <1> ;; ;MOV AX,ES ; GET THE ES VALUE

1868 <1> ;; ;ROL AX,4 ; ROTATE LEFT

1869 <1> ;; ;MOV CH,AL ; GET HIGHEST NIBBLE OF ES TO CH

1870 <1> ;; ;AND AL,11110000B ; ZERO THE LOW NIBBLE FROM SEGMENT

1871 <1> ;; ;ADD AX,[BP+2] ; TEST FOR CARRY FROM ADDITION

1872 <1> ;; ;JNC short J33A

1873 <1> ;; ;INC CH ; CARRY MEANS HIGH 4 BITS MUST BE INC

1874 <1> ;; mov eax, [ebp+4] ; 08/02/2015

1875 <1> ;;;J33A:

1876 <1> ;; PUSH eAX ; 08/02/2015 ; SAVE START ADDRESS

1877 <1> ;; OUT DMA+4,AL ; OUTPUT LOW ADDRESS

1878 <1> ;; ;JMP $+2 ; WAIT FOR I/O

1879 <1> ;; IODELAY

1880 <1> ;; MOV AL,AH

1881 <1> ;; OUT DMA+4,AL ; OUTPUT HIGH ADDRESS

1882 <1> ;; shr eax, 16 ; 08/02/2015

1883 <1> ;; ;MOV AL,CH ; GET HIGH 4 BITS

1884 <1> ;; ;JMP $+2 ; I/O WAIT STATE

1885 <1> ;; IODELAY

1886 <1> ;; ;AND AL,00001111B

1887 <1> ;; OUT 081H,AL ; OUTPUT HIGH 4 BITS TO PAGE REGISTER

1888 <1> ;;

1889 <1> ;;;----- DETERMINE COUNT

1890 <1> ;; sub eax, eax ; 08/02/2015

1891 <1> ;; MOV DL, 4 ; SECTORS/TRACK VALUE IN PARM TABLE

1892 <1> ;; CALL GET\_PARM ; "

1893 <1> ;; XCHG AL, AH ; AL = SECTORS/TRACK VALUE

1894 <1> ;; SUB AH, AH ; AX = SECTORS/TRACK VALUE

1895 <1> ;; SHL AX, 2 ; AX = SEC/TRK \* 4 (OFFSET C,H,R,N)

1896 <1> ;; DEC AX ; -1 FOR DMA VALUE

1897 <1> ;; PUSH eAX ; 08/02/2015 ; SAVE # OF BYTES TO BE TRANSFERED

1898 <1> ;; OUT DMA+5,AL ; LOW BYTE OF COUNT

1899 <1> ;; ;JMP $+2 ; WAIT FOR I/O

1900 <1> ;; IODELAY

1901 <1> ;; MOV AL, AH

1902 <1> ;; OUT DMA+5,AL ; HIGH BYTE OF COUNT

1903 <1> ;; STI ; RE-ENABLE INTERRUPTS

1904 <1> ;; POP eCX ; 08/02/2015 ; RECOVER COUNT VALUE

1905 <1> ;; POP eAX ; 08/02/2015 ; RECOVER ADDRESS VALUE

1906 <1> ;; ;ADD AX, CX ; ADD, TEST FOR 64K OVERFLOW

1907 <1> ;; add ecx, eax ; 08/02/2015

1908 <1> ;; MOV AL, 2 ; MODE FOR 8237

1909 <1> ;; ;JMP $+2 ; WAIT FOR I/O

1910 <1> ;; SIODELAY

1911 <1> ;; OUT DMA+10, AL ; INITIALIZE THE DISKETTE CHANNEL

1912 <1> ;; ;JNC short FMTDMA\_OK ; CHECK FOR ERROR

1913 <1> ;; jc short fmtdma\_bnd\_err ; 08/02/2015

1914 <1> ;; and ecx, 0FFF00000h ; 16 MB limit

1915 <1> ;; jz short FMTDMA\_OK

1916 <1> ;; stc ; 20/02/2015

1917 <1> ;;fmtdma\_bnd\_err:

1918 <1> ;; MOV byte [DSKETTE\_STATUS], DMA\_BOUNDARY ; SET ERROR

1919 <1> ;;FMTDMA\_OK:

1920 <1> ;; RETn ; CY SET BY ABOVE IF ERROR

1921 <1>

1922 <1> ;-------------------------------------------------------------------------------

1923 <1> ; NEC\_INIT

1924 <1> ; THIS ROUTINE SEEKS TO THE REQUESTED TRACK AND INITIALIZES

1925 <1> ; THE NEC FOR THE READ/WRITE/VERIFY/FORMAT OPERATION.

1926 <1> ;

1927 <1> ; ON ENTRY: AH = NEC COMMAND TO BE PERFORMED

1928 <1> ;

1929 <1> ; ON EXIT: @DSKETTE\_STATUS, CY REFLECT STATUS OF OPERATION

1930 <1> ;-------------------------------------------------------------------------------

1931 <1> NEC\_INIT:

1932 00003C26 6650 <1> PUSH AX ; SAVE NEC COMMAND

1933 00003C28 E8BC020000 <1> CALL MOTOR\_ON ; TURN MOTOR ON FOR SPECIFIC DRIVE

1934 <1>

1935 <1> ;----- DO THE SEEK OPERATION

1936 <1>

1937 00003C2D 8A6D01 <1> MOV CH,[eBP+1] ; CH = TRACK #

1938 00003C30 E8AF030000 <1> CALL SEEK ; MOVE TO CORRECT TRACK

1939 00003C35 6658 <1> POP AX ; RECOVER COMMAND

1940 00003C37 721E <1> JC short ER\_1 ; ERROR ON SEEK

1941 00003C39 BB[573C0000] <1> MOV eBX, ER\_1 ; LOAD ERROR ADDRESS

1942 00003C3E 53 <1> PUSH eBX ; PUSH NEC\_OUT ERROR RETURN

1943 <1>

1944 <1> ;----- SEND OUT THE PARAMETERS TO THE CONTROLLER

1945 <1>

1946 00003C3F E866030000 <1> CALL NEC\_OUTPUT ; OUTPUT THE OPERATION COMMAND

1947 00003C44 6689F0 <1> MOV AX,SI ; AH = HEAD #

1948 00003C47 89FB <1> MOV eBX,eDI ; BL = DRIVE #

1949 00003C49 C0E402 <1> SAL AH,2 ; MOVE IT TO BIT 2

1950 00003C4C 80E404 <1> AND AH,00000100B ; ISOLATE THAT BIT

1951 00003C4F 08DC <1> OR AH,BL ; OR IN THE DRIVE NUMBER

1952 00003C51 E854030000 <1> CALL NEC\_OUTPUT ; FALL THRU CY SET IF ERROR

1953 00003C56 5B <1> POP eBX ; THROW AWAY ERROR RETURN

1954 <1> ER\_1:

1955 00003C57 C3 <1> RETn

1956 <1>

1957 <1> ;-------------------------------------------------------------------------------

1958 <1> ; RWV\_COM

1959 <1> ; THIS ROUTINE SENDS PARAMETERS TO THE NEC SPECIFIC TO THE

1960 <1> ; READ/WRITE/VERIFY OPERATIONS.

1961 <1> ;

1962 <1> ; ON ENTRY: CS:BX = ADDRESS OF MEDIA/DRIVE PARAMETER TABLE

1963 <1> ; ON EXIT: @DSKETTE\_STATUS, CY REFLECT STATUS OF OPERATION

1964 <1> ;-------------------------------------------------------------------------------

1965 <1> RWV\_COM:

1966 00003C58 B8[A33C0000] <1> MOV eAX, ER\_2 ; LOAD ERROR ADDRESS

1967 00003C5D 50 <1> PUSH eAX ; PUSH NEC\_OUT ERROR RETURN

1968 00003C5E 8A6501 <1> MOV AH,[eBP+1] ; OUTPUT TRACK #

1969 00003C61 E844030000 <1> CALL NEC\_OUTPUT

1970 00003C66 6689F0 <1> MOV AX,SI ; OUTPUT HEAD #

1971 00003C69 E83C030000 <1> CALL NEC\_OUTPUT

1972 00003C6E 8A6500 <1> MOV AH,[eBP] ; OUTPUT SECTOR #

1973 00003C71 E834030000 <1> CALL NEC\_OUTPUT

1974 00003C76 B203 <1> MOV DL,3 ; BYTES/SECTOR PARAMETER FROM BLOCK

1975 00003C78 E827020000 <1> CALL GET\_PARM ; ... TO THE NEC

1976 00003C7D E828030000 <1> CALL NEC\_OUTPUT ; OUTPUT TO CONTROLLER

1977 00003C82 B204 <1> MOV DL,4 ; EOT PARAMETER FROM BLOCK

1978 00003C84 E81B020000 <1> CALL GET\_PARM ; ... TO THE NEC

1979 00003C89 E81C030000 <1> CALL NEC\_OUTPUT ; OUTPUT TO CONTROLLER

1980 00003C8E 8A6305 <1> MOV AH, [eBX+MD.GAP] ; GET GAP LENGTH

1981 <1> \_R15:

1982 00003C91 E814030000 <1> CALL NEC\_OUTPUT

1983 00003C96 B206 <1> MOV DL,6 ; DTL PARAMETER PROM BLOCK

1984 00003C98 E807020000 <1> CALL GET\_PARM ; TO THE NEC

1985 00003C9D E808030000 <1> CALL NEC\_OUTPUT ; OUTPUT TO CONTROLLER

1986 00003CA2 58 <1> POP eAX ; THROW AWAY ERROR EXIT

1987 <1> ER\_2:

1988 00003CA3 C3 <1> RETn

1989 <1>

1990 <1> ;-------------------------------------------------------------------------------

1991 <1> ; NEC\_TERM

1992 <1> ; THIS ROUTINE WAITS FOR THE OPERATION THEN ACCEPTS THE STATUS

1993 <1> ; FROM THE NEC FOR THE READ/WRITE/VERIFY/FORWAT OPERATION.

1994 <1> ;

1995 <1> ; ON EXIT: @DSKETTE\_STATUS, CY REFLECT STATUS OF OPERATION

1996 <1> ;-------------------------------------------------------------------------------

1997 <1> NEC\_TERM:

1998 <1>

1999 <1> ;----- LET THE OPERATION HAPPEN

2000 <1>

2001 00003CA4 56 <1> PUSH eSI ; SAVE HEAD #, # OF SECTORS

2002 00003CA5 E80D040000 <1> CALL WAIT\_INT ; WAIT FOR THE INTERRUPT

2003 00003CAA 9C <1> PUSHF

2004 00003CAB E837040000 <1> CALL RESULTS ; GET THE NEC STATUS

2005 00003CB0 724B <1> JC short SET\_END\_POP

2006 00003CB2 9D <1> POPF

2007 00003CB3 723E <1> JC short SET\_END ; LOOK FOR ERROR

2008 <1>

2009 <1> ;----- CHECK THE RESULTS RETURNED BY THE CONTROLLER

2010 <1>

2011 00003CB5 FC <1> CLD ; SET THE CORRECT DIRECTION

2012 00003CB6 BE[C1580100] <1> MOV eSI, NEC\_STATUS ; POINT TO STATUS FIELD

2013 00003CBB AC <1> lodsb ; GET ST0

2014 00003CBC 24C0 <1> AND AL,11000000B ; TEST FOR NORMAL TERMINATION

2015 00003CBE 7433 <1> JZ short SET\_END

2016 00003CC0 3C40 <1> CMP AL,01000000B ; TEST FOR ABNORMAL TERMINATION

2017 00003CC2 7527 <1> JNZ short J18 ; NOT ABNORMAL, BAD NEC

2018 <1>

2019 <1> ;----- ABNORMAL TERMINATION, FIND OUT WHY

2020 <1>

2021 00003CC4 AC <1> lodsb ; GET ST1

2022 00003CC5 D0E0 <1> SAL AL,1 ; TEST FOR EDT FOUND

2023 00003CC7 B404 <1> MOV AH,RECORD\_NOT\_FND

2024 00003CC9 7222 <1> JC short J19

2025 00003CCB C0E002 <1> SAL AL,2

2026 00003CCE B410 <1> MOV AH,BAD\_CRC

2027 00003CD0 721B <1> JC short J19

2028 00003CD2 D0E0 <1> SAL AL,1 ; TEST FOR DMA OVERRUN

2029 00003CD4 B408 <1> MOV AH,BAD\_DMA

2030 00003CD6 7215 <1> JC short J19

2031 00003CD8 C0E002 <1> SAL AL,2 ; TEST FOR RECORD NOT FOUND

2032 00003CDB B404 <1> MOV AH,RECORD\_NOT\_FND

2033 00003CDD 720E <1> JC short J19

2034 00003CDF D0E0 <1> SAL AL,1

2035 00003CE1 B403 <1> MOV AH,WRITE\_PROTECT ; TEST FOR WRITE\_PROTECT

2036 00003CE3 7208 <1> JC short J19

2037 00003CE5 D0E0 <1> SAL AL,1 ; TEST MISSING ADDRESS MARK

2038 00003CE7 B402 <1> MOV AH,BAD\_ADDR\_MARK

2039 00003CE9 7202 <1> JC short J19

2040 <1>

2041 <1> ;----- NEC MUST HAVE FAILED

2042 <1> J18:

2043 00003CEB B420 <1> MOV AH,BAD\_NEC

2044 <1> J19:

2045 00003CED 0825[C0580100] <1> OR [DSKETTE\_STATUS], AH

2046 <1> SET\_END:

2047 00003CF3 803D[C0580100]01 <1> CMP byte [DSKETTE\_STATUS], 1 ; SET ERROR CONDITION

2048 00003CFA F5 <1> CMC

2049 00003CFB 5E <1> POP eSI

2050 00003CFC C3 <1> RETn ; RESTORE HEAD #, # OF SECTORS

2051 <1>

2052 <1> SET\_END\_POP:

2053 00003CFD 9D <1> POPF

2054 00003CFE EBF3 <1> JMP SHORT SET\_END

2055 <1>

2056 <1> ;-------------------------------------------------------------------------------

2057 <1> ; DSTATE: ESTABLISH STATE UPON SUCCESSFUL OPERATION.

2058 <1> ;-------------------------------------------------------------------------------

2059 <1> DSTATE:

2060 00003D00 803D[C0580100]00 <1> CMP byte [DSKETTE\_STATUS],0 ; CHECK FOR ERROR

2061 00003D07 753E <1> JNZ short SETBAC ; IF ERROR JUMP

2062 00003D09 808F[CD580100]10 <1> OR byte [DSK\_STATE+eDI],MED\_DET ; NO ERROR, MARK MEDIA AS DETERMINED

2063 00003D10 F687[CD580100]04 <1> TEST byte [DSK\_STATE+eDI],DRV\_DET ; DRIVE DETERMINED ?

2064 00003D17 752E <1> JNZ short SETBAC ; IF DETERMINED NO TRY TO DETERMINE

2065 00003D19 8A87[CD580100] <1> MOV AL,[DSK\_STATE+eDI] ; LOAD STATE

2066 00003D1F 24C0 <1> AND AL,RATE\_MSK ; KEEP ONLY RATE

2067 00003D21 3C80 <1> CMP AL,RATE\_250 ; RATE 250 ?

2068 00003D23 751B <1> JNE short M\_12 ; NO, MUST BE 1.2M OR 1.44M DRIVE

2069 <1>

2070 <1> ;----- CHECK IF IT IS 1.44M

2071 <1>

2072 00003D25 E871010000 <1> CALL CMOS\_TYPE ; RETURN DRIVE TYPE IN (AL)

2073 <1> ;;20/02/2015

2074 <1> ;;JC short M\_12 ; CMOS BAD

2075 00003D2A 7414 <1> jz short M\_12 ;; 20/02/2015

2076 00003D2C 3C04 <1> CMP AL, 4 ; 1.44MB DRIVE ?

2077 00003D2E 7410 <1> JE short M\_12 ; YES

2078 <1> M\_720:

2079 00003D30 80A7[CD580100]FD <1> AND byte [DSK\_STATE+eDI], ~FMT\_CAPA ; TURN OFF FORMAT CAPABILITY

2080 00003D37 808F[CD580100]04 <1> OR byte [DSK\_STATE+eDI],DRV\_DET ; MARK DRIVE DETERMINED

2081 00003D3E EB07 <1> JMP SHORT SETBAC ; BACK

2082 <1> M\_12:

2083 00003D40 808F[CD580100]06 <1> OR byte [DSK\_STATE+eDI],DRV\_DET+FMT\_CAPA

2084 <1> ; TURN ON DETERMINED & FMT CAPA

2085 <1> SETBAC:

2086 00003D47 C3 <1> RETn

2087 <1>

2088 <1> ;-------------------------------------------------------------------------------

2089 <1> ; RETRY

2090 <1> ; DETERMINES WHETHER A RETRY IS NECESSARY.

2091 <1> ; IF RETRY IS REQUIRED THEN STATE INFORMATION IS UPDATED FOR RETRY.

2092 <1> ;

2093 <1> ; ON EXIT: CY = 1 FOR RETRY, CY = 0 FOR NO RETRY

2094 <1> ;-------------------------------------------------------------------------------

2095 <1> RETRY:

2096 00003D48 803D[C0580100]00 <1> CMP byte [DSKETTE\_STATUS],0 ; GET STATUS OF OPERATION

2097 00003D4F 7445 <1> JZ short NO\_RETRY ; SUCCESSFUL OPERATION

2098 00003D51 803D[C0580100]80 <1> CMP byte [DSKETTE\_STATUS],TIME\_OUT ; IF TIME OUT NO RETRY

2099 00003D58 743C <1> JZ short NO\_RETRY

2100 00003D5A 8AA7[CD580100] <1> MOV AH,[DSK\_STATE+eDI] ; GET MEDIA STATE OF DRIVE

2101 00003D60 F6C410 <1> TEST AH,MED\_DET ; ESTABLISHED/DETERMINED ?

2102 00003D63 7531 <1> JNZ short NO\_RETRY ; IF ESTABLISHED STATE THEN TRUE ERROR

2103 00003D65 80E4C0 <1> AND AH,RATE\_MSK ; ISOLATE RATE

2104 00003D68 8A2D[C8580100] <1> MOV CH,[LASTRATE] ; GET START OPERATION STATE

2105 00003D6E C0C504 <1> ROL CH,4 ; TO CORRESPONDING BITS

2106 00003D71 80E5C0 <1> AND CH,RATE\_MSK ; ISOLATE RATE BITS

2107 00003D74 38E5 <1> CMP CH,AH ; ALL RATES TRIED

2108 00003D76 741E <1> JE short NO\_RETRY ; IF YES, THEN TRUE ERROR

2109 <1>

2110 <1> ; SETUP STATE INDICATOR FOR RETRY ATTEMPT TO NEXT RATE

2111 <1> ; 00000000B (500) -> 10000000B (250)

2112 <1> ; 10000000B (250) -> 01000000B (300)

2113 <1> ; 01000000B (300) -> 00000000B (500)

2114 <1>

2115 00003D78 80FC01 <1> CMP AH,RATE\_500+1 ; SET CY FOR RATE 500

2116 00003D7B D0DC <1> RCR AH,1 ; TO NEXT STATE

2117 00003D7D 80E4C0 <1> AND AH,RATE\_MSK ; KEEP ONLY RATE BITS

2118 00003D80 80A7[CD580100]1F <1> AND byte [DSK\_STATE+eDI], ~(RATE\_MSK+DBL\_STEP)

2119 <1> ; RATE, DBL STEP OFF

2120 00003D87 08A7[CD580100] <1> OR [DSK\_STATE+eDI],AH ; TURN ON NEW RATE

2121 00003D8D C605[C0580100]00 <1> MOV byte [DSKETTE\_STATUS],0 ; RESET STATUS FOR RETRY

2122 00003D94 F9 <1> STC ; SET CARRY FOR RETRY

2123 00003D95 C3 <1> RETn ; RETRY RETURN

2124 <1>

2125 <1> NO\_RETRY:

2126 00003D96 F8 <1> CLC ; CLEAR CARRY NO RETRY

2127 00003D97 C3 <1> RETn ; NO RETRY RETURN

2128 <1>

2129 <1> ;-------------------------------------------------------------------------------

2130 <1> ; NUM\_TRANS

2131 <1> ; THIS ROUTINE CALCULATES THE NUMBER OF SECTORS THAT WERE

2132 <1> ; ACTUALLY TRANSFERRED TO/FROM THE DISKETTE.

2133 <1> ;

2134 <1> ; ON ENTRY: [BP+1] = TRACK

2135 <1> ; SI-HI = HEAD

2136 <1> ; [BP] = START SECTOR

2137 <1> ;

2138 <1> ; ON EXIT: AL = NUMBER ACTUALLY TRANSFERRED

2139 <1> ;-------------------------------------------------------------------------------

2140 <1> NUM\_TRANS:

2141 00003D98 30C0 <1> XOR AL,AL ; CLEAR FOR ERROR

2142 00003D9A 803D[C0580100]00 <1> CMP byte [DSKETTE\_STATUS],0 ; CHECK FOR ERROR

2143 00003DA1 752C <1> JNZ NT\_OUT ; IF ERROR 0 TRANSFERRED

2144 00003DA3 B204 <1> MOV DL,4 ; SECTORS/TRACK OFFSET TO DL

2145 00003DA5 E8FA000000 <1> CALL GET\_PARM ; AH = SECTORS/TRACK

2146 00003DAA 8A1D[C6580100] <1> MOV BL, [NEC\_STATUS+5] ; GET ENDING SECTOR

2147 00003DB0 6689F1 <1> MOV CX,SI ; CH = HEAD # STARTED

2148 00003DB3 3A2D[C5580100] <1> CMP CH, [NEC\_STATUS+4] ; GET HEAD ENDED UP ON

2149 00003DB9 750D <1> JNZ DIF\_HD ; IF ON SAME HEAD, THEN NO ADJUST

2150 00003DBB 8A2D[C4580100] <1> MOV CH, [NEC\_STATUS+3] ; GET TRACK ENDED UP ON

2151 00003DC1 3A6D01 <1> CMP CH,[eBP+1] ; IS IT ASKED FOR TRACK

2152 00003DC4 7404 <1> JZ short SAME\_TRK ; IF SAME TRACK NO INCREASE

2153 00003DC6 00E3 <1> ADD BL,AH ; ADD SECTORS/TRACK

2154 <1> DIF\_HD:

2155 00003DC8 00E3 <1> ADD BL,AH ; ADD SECTORS/TRACK

2156 <1> SAME\_TRK:

2157 00003DCA 2A5D00 <1> SUB BL,[eBP] ; SUBTRACT START FROM END

2158 00003DCD 88D8 <1> MOV AL,BL ; TO AL

2159 <1> NT\_OUT:

2160 00003DCF C3 <1> RETn

2161 <1>

2162 <1> ;-------------------------------------------------------------------------------

2163 <1> ; SETUP\_END

2164 <1> ; RESTORES @MOTOR\_COUNT TO PARAMETER PROVIDED IN TABLE

2165 <1> ; AND LOADS @DSKETTE\_STATUS TO AH, AND SETS CY.

2166 <1> ;

2167 <1> ; ON EXIT:

2168 <1> ; AH, @DSKETTE\_STATUS, CY REFLECT STATUS OF OPERATION

2169 <1> ;-------------------------------------------------------------------------------

2170 <1> SETUP\_END:

2171 00003DD0 B202 <1> MOV DL,2 ; GET THE MOTOR WAIT PARAMETER

2172 00003DD2 6650 <1> PUSH AX ; SAVE NUMBER TRANSFERRED

2173 00003DD4 E8CB000000 <1> CALL GET\_PARM

2174 00003DD9 8825[BF580100] <1> MOV [MOTOR\_COUNT],AH ; STORE UPON RETURN

2175 00003DDF 6658 <1> POP AX ; RESTORE NUMBER TRANSFERRED

2176 00003DE1 8A25[C0580100] <1> MOV AH, [DSKETTE\_STATUS] ; GET STATUS OF OPERATION

2177 00003DE7 08E4 <1> OR AH,AH ; CHECK FOR ERROR

2178 00003DE9 7402 <1> JZ short NUN\_ERR ; NO ERROR

2179 00003DEB 30C0 <1> XOR AL,AL ; CLEAR NUMBER RETURNED

2180 <1> NUN\_ERR:

2181 00003DED 80FC01 <1> CMP AH,1 ; SET THE CARRY FLAG TO INDICATE

2182 00003DF0 F5 <1> CMC ; SUCCESS OR FAILURE

2183 00003DF1 C3 <1> RETn

2184 <1>

2185 <1> ;-------------------------------------------------------------------------------

2186 <1> ; SETUP\_DBL

2187 <1> ; CHECK DOUBLE STEP.

2188 <1> ;

2189 <1> ; ON ENTRY : DI = DRIVE

2190 <1> ;

2191 <1> ; ON EXIT : CY = 1 MEANS ERROR

2192 <1> ;-------------------------------------------------------------------------------

2193 <1> SETUP\_DBL:

2194 00003DF2 8AA7[CD580100] <1> MOV AH, [DSK\_STATE+eDI] ; ACCESS STATE

2195 00003DF8 F6C410 <1> TEST AH,MED\_DET ; ESTABLISHED STATE ?

2196 00003DFB 757E <1> JNZ short NO\_DBL ; IF ESTABLISHED THEN DOUBLE DONE

2197 <1>

2198 <1> ;----- CHECK FOR TRACK 0 TO SPEED UP ACKNOWLEDGE OF UNFORMATTED DISKETTE

2199 <1>

2200 00003DFD C605[BD580100]00 <1> MOV byte [SEEK\_STATUS],0 ; SET RECALIBRATE REQUIRED ON ALL DRIVES

2201 00003E04 E8E0000000 <1> CALL MOTOR\_ON ; ENSURE MOTOR STAY ON

2202 00003E09 B500 <1> MOV CH,0 ; LOAD TRACK 0

2203 00003E0B E8D4010000 <1> CALL SEEK ; SEEK TO TRACK 0

2204 00003E10 E868000000 <1> CALL READ\_ID ; READ ID FUNCTION

2205 00003E15 7249 <1> JC short SD\_ERR ; IF ERROR NO TRACK 0

2206 <1>

2207 <1> ;----- INITIALIZE START AND MAX TRACKS (TIMES 2 FOR BOTH HEADS)

2208 <1>

2209 00003E17 66B95004 <1> MOV CX,0450H ; START, MAX TRACKS

2210 00003E1B F687[CD580100]01 <1> TEST byte [DSK\_STATE+eDI],TRK\_CAPA ; TEST FOR 80 TRACK CAPABILITY

2211 00003E22 7402 <1> JZ short CNT\_OK ; IF NOT COUNT IS SETUP

2212 00003E24 B1A0 <1> MOV CL,0A0H ; MAXIMUM TRACK 1.2 MB

2213 <1>

2214 <1> ; ATTEMPT READ ID OF ALL TRACKS, ALL HEADS UNTIL SUCCESS; UPON SUCCESS,

2215 <1> ; MUST SEE IF ASKED FOR TRACK IN SINGLE STEP MODE = TRACK ID READ; IF NOT

2216 <1> ; THEN SET DOUBLE STEP ON.

2217 <1>

2218 <1> CNT\_OK:

2219 00003E26 C605[BF580100]FF <1> MOV byte [MOTOR\_COUNT], 0FFH ; ENSURE MOTOR STAYS ON FOR OPERATION

2220 00003E2D 6651 <1> PUSH CX ; SAVE TRACK, COUNT

2221 00003E2F C605[C0580100]00 <1> MOV byte [DSKETTE\_STATUS],0 ; CLEAR STATUS, EXPECT ERRORS

2222 00003E36 6631C0 <1> XOR AX,AX ; CLEAR AX

2223 00003E39 D0ED <1> SHR CH,1 ; HALVE TRACK, CY = HEAD

2224 00003E3B C0D003 <1> RCL AL,3 ; AX = HEAD IN CORRECT BIT

2225 00003E3E 6650 <1> PUSH AX ; SAVE HEAD

2226 00003E40 E89F010000 <1> CALL SEEK ; SEEK TO TRACK

2227 00003E45 6658 <1> POP AX ; RESTORE HEAD

2228 00003E47 6609C7 <1> OR DI,AX ; DI = HEAD OR'ED DRIVE

2229 00003E4A E82E000000 <1> CALL READ\_ID ; READ ID HEAD 0

2230 00003E4F 9C <1> PUSHF ; SAVE RETURN FROM READ\_ID

2231 00003E50 6681E7FB00 <1> AND DI,11111011B ; TURN OFF HEAD 1 BIT

2232 00003E55 9D <1> POPF ; RESTORE ERROR RETURN

2233 00003E56 6659 <1> POP CX ; RESTORE COUNT

2234 00003E58 7308 <1> JNC short DO\_CHK ; IF OK, ASKED = RETURNED TRACK ?

2235 00003E5A FEC5 <1> INC CH ; INC FOR NEXT TRACK

2236 00003E5C 38CD <1> CMP CH,CL ; REACHED MAXIMUM YET

2237 00003E5E 75C6 <1> JNZ short CNT\_OK ; CONTINUE TILL ALL TRIED

2238 <1>

2239 <1> ;----- FALL THRU, READ ID FAILED FOR ALL TRACKS

2240 <1>

2241 <1> SD\_ERR:

2242 00003E60 F9 <1> STC ; SET CARRY FOR ERROR

2243 00003E61 C3 <1> RETn ; SETUP\_DBL ERROR EXIT

2244 <1>

2245 <1> DO\_CHK:

2246 00003E62 8A0D[C4580100] <1> MOV CL, [NEC\_STATUS+3] ; LOAD RETURNED TRACK

2247 00003E68 888F[D1580100] <1> MOV [DSK\_TRK+eDI], CL ; STORE TRACK NUMBER

2248 00003E6E D0ED <1> SHR CH,1 ; HALVE TRACK

2249 00003E70 38CD <1> CMP CH,CL ; IS IT THE SAME AS ASKED FOR TRACK

2250 00003E72 7407 <1> JZ short NO\_DBL ; IF SAME THEN NO DOUBLE STEP

2251 00003E74 808F[CD580100]20 <1> OR byte [DSK\_STATE+eDI],DBL\_STEP ; TURN ON DOUBLE STEP REQUIRED

2252 <1> NO\_DBL:

2253 00003E7B F8 <1> CLC ; CLEAR ERROR FLAG

2254 00003E7C C3 <1> RETn

2255 <1>

2256 <1> ;-------------------------------------------------------------------------------

2257 <1> ; READ\_ID

2258 <1> ; READ ID FUNCTION.

2259 <1> ;

2260 <1> ; ON ENTRY: DI : BIT 2 = HEAD; BITS 1,0 = DRIVE

2261 <1> ;

2262 <1> ; ON EXIT: DI : BIT 2 IS RESET, BITS 1,0 = DRIVE

2263 <1> ; @DSKETTE\_STATUS, CY REFLECT STATUS OF OPERATION

2264 <1> ;-------------------------------------------------------------------------------

2265 <1> READ\_ID:

2266 00003E7D B8[9A3E0000] <1> MOV eAX, ER\_3 ; MOVE NEC OUTPUT ERROR ADDRESS

2267 00003E82 50 <1> PUSH eAX

2268 00003E83 B44A <1> MOV AH,4AH ; READ ID COMMAND

2269 00003E85 E820010000 <1> CALL NEC\_OUTPUT ; TO CONTROLLER

2270 00003E8A 6689F8 <1> MOV AX,DI ; DRIVE # TO AH, HEAD 0

2271 00003E8D 88C4 <1> MOV AH,AL

2272 00003E8F E816010000 <1> CALL NEC\_OUTPUT ; TO CONTROLLER

2273 00003E94 E80BFEFFFF <1> CALL NEC\_TERM ; WAIT FOR OPERATION, GET STATUS

2274 00003E99 58 <1> POP eAX ; THROW AWAY ERROR ADDRESS

2275 <1> ER\_3:

2276 00003E9A C3 <1> RETn

2277 <1>

2278 <1> ;-------------------------------------------------------------------------------

2279 <1> ; CMOS\_TYPE

2280 <1> ; RETURNS DISKETTE TYPE FROM CMOS

2281 <1> ;

2282 <1> ; ON ENTRY: DI = DRIVE #

2283 <1> ;

2284 <1> ; ON EXIT: AL = TYPE; CY REFLECTS STATUS

2285 <1> ;-------------------------------------------------------------------------------

2286 <1>

2287 <1> CMOS\_TYPE: ; 11/12/2014

2288 00003E9B 8A87[F65C0000] <1> mov al, [eDI+fd0\_type]

2289 00003EA1 20C0 <1> and al, al ; 18/12/2014

2290 00003EA3 C3 <1> retn

2291 <1>

2292 <1> ;CMOS\_TYPE:

2293 <1> ; MOV AL, CMOS\_DIAG ; CMOS DIAGNOSTIC STATUS BYTE ADDRESS

2294 <1> ; CALL CMOS\_READ ; GET CMOS STATUS

2295 <1> ; TEST AL,BAD\_BAT+BAD\_CKSUM ; BATTERY GOOD AND CHECKSUM VALID

2296 <1> ; STC ; SET CY = 1 INDICATING ERROR FOR RETURN

2297 <1> ; JNZ short BAD\_CM ; ERROR IF EITHER BIT ON

2298 <1> ; MOV AL,CMOS\_DISKETTE ; ADDRESS OF DISKETTE BYTE IN CMOS

2299 <1> ; CALL CMOS\_READ ; GET DISKETTE BYTE

2300 <1> ; OR DI,DI ; SEE WHICH DRIVE IN QUESTION

2301 <1> ; JNZ short TB ; IF DRIVE 1, DATA IN LOW NIBBLE

2302 <1> ; ROR AL,4 ; EXCHANGE NIBBLES IF SECOND DRIVE

2303 <1> ;TB:

2304 <1> ; AND AL,0FH ; KEEP ONLY DRIVE DATA, RESET CY, 0

2305 <1> ;BAD\_CM:

2306 <1> ; RETn ; CY, STATUS OF READ

2307 <1>

2308 <1> ;-------------------------------------------------------------------------------

2309 <1> ; GET\_PARM

2310 <1> ; THIS ROUTINE FETCHES THE INDEXED POINTER FROM THE DISK\_BASE

2311 <1> ; BLOCK POINTED TO BY THE DATA VARIABLE @DISK\_POINTER. A BYTE FROM

2312 <1> ; THAT TABLE IS THEN MOVED INTO AH, THE INDEX OF THAT BYTE BEING

2313 <1> ; THE PARAMETER IN DL.

2314 <1> ;

2315 <1> ; ON ENTRY: DL = INDEX OF BYTE TO BE FETCHED

2316 <1> ;

2317 <1> ; ON EXIT: AH = THAT BYTE FROM BLOCK

2318 <1> ; AL,DH DESTROYED

2319 <1> ;-------------------------------------------------------------------------------

2320 <1> GET\_PARM:

2321 <1> ;PUSH DS

2322 00003EA4 56 <1> PUSH eSI

2323 <1> ;SUB AX,AX ; DS = 0, BIOS DATA AREA

2324 <1> ;MOV DS,AX

2325 <1> ;;mov ax, cs

2326 <1> ;;mov ds, ax

2327 <1> ; 08/02/2015 (protected mode modifications, bx -> ebx)

2328 00003EA5 87D3 <1> XCHG eDX,eBX ; BL = INDEX

2329 <1> ;SUB BH,BH ; BX = INDEX

2330 00003EA7 81E3FF000000 <1> and ebx, 0FFh

2331 <1> ;LDS SI, [DISK\_POINTER] ; POINT TO BLOCK

2332 <1> ;

2333 <1> ; 17/12/2014

2334 00003EAD 66A1[E55C0000] <1> mov ax, [cfd] ; current (AL) and previous fd (AH)

2335 00003EB3 38E0 <1> cmp al, ah

2336 00003EB5 7425 <1> je short gpndc

2337 00003EB7 A2[E65C0000] <1> mov [pfd], al ; current drive -> previous drive

2338 00003EBC 53 <1> push ebx ; 08/02/2015

2339 00003EBD 88C3 <1> mov bl, al

2340 <1> ; 11/12/2014

2341 00003EBF 8A83[F65C0000] <1> mov al, [eBX+fd0\_type] ; Drive type (0,1,2,3,4)

2342 <1> ; 18/12/2014

2343 00003EC5 20C0 <1> and al, al

2344 00003EC7 7507 <1> jnz short gpdtc

2345 00003EC9 BB[CF5C0000] <1> mov ebx, MD\_TBL6 ; 1.44 MB param. tbl. (default)

2346 00003ECE EB05 <1> jmp short gpdpu

2347 <1> gpdtc:

2348 00003ED0 E817F9FFFF <1> call DR\_TYPE\_CHECK

2349 <1> ; cf = 1 -> eBX points to 1.44MB fd parameter table (default)

2350 <1> gpdpu:

2351 00003ED5 891D[6C5C0000] <1> mov [DISK\_POINTER], ebx

2352 00003EDB 5B <1> pop ebx

2353 <1> gpndc:

2354 00003EDC 8B35[6C5C0000] <1> mov esi, [DISK\_POINTER] ; 08/02/2015, si -> esi

2355 00003EE2 8A241E <1> MOV AH, [eSI+eBX] ; GET THE WORD

2356 00003EE5 87D3 <1> XCHG eDX,eBX ; RESTORE BX

2357 00003EE7 5E <1> POP eSI

2358 <1> ;POP DS

2359 00003EE8 C3 <1> RETn

2360 <1>

2361 <1> ;-------------------------------------------------------------------------------

2362 <1> ; MOTOR\_ON

2363 <1> ; TURN MOTOR ON AND WAIT FOR MOTOR START UP TIME. THE @MOTOR\_COUNT

2364 <1> ; IS REPLACED WITH A SUFFICIENTLY HIGH NUMBER (0FFH) TO ENSURE

2365 <1> ; THAT THE MOTOR DOES NOT GO OFF DURING THE OPERATION. IF THE

2366 <1> ; MOTOR NEEDED TO BE TURNED ON, THE MULTI-TASKING HOOK FUNCTION

2367 <1> ; (AX=90FDH, INT 15) IS CALLED TELLING THE OPERATING SYSTEM

2368 <1> ; THAT THE BIOS IS ABOUT TO WAIT FOR MOTOR START UP. IF THIS

2369 <1> ; FUNCTION RETURNS WITH CY = 1, IT MEANS THAT THE MINIMUM WAIT

2370 <1> ; HAS BEEN COMPLETED. AT THIS POINT A CHECK IS MADE TO ENSURE

2371 <1> ; THAT THE MOTOR WASN'T TURNED OFF BY THE TIMER. IF THE HOOK DID

2372 <1> ; NOT WAIT, THE WAIT FUNCTION (AH=086H) IS CALLED TO WAIT THE

2373 <1> ; PRESCRIBED AMOUNT OF TIME. IF THE CARRY FLAG IS SET ON RETURN,

2374 <1> ; IT MEANS THAT THE FUNCTION IS IN USE AND DID NOT PERFORM THE

2375 <1> ; WAIT. A TIMER 1 WAIT LOOP WILL THEN DO THE WAIT.

2376 <1> ;

2377 <1> ; ON ENTRY: DI = DRIVE #

2378 <1> ; ON EXIT: AX,CX,DX DESTROYED

2379 <1> ;-------------------------------------------------------------------------------

2380 <1> MOTOR\_ON:

2381 00003EE9 53 <1> PUSH eBX ; SAVE REG.

2382 00003EEA E82A000000 <1> CALL TURN\_ON ; TURN ON MOTOR

2383 00003EEF 7226 <1> JC short MOT\_IS\_ON ; IF CY=1 NO WAIT

2384 00003EF1 E89BF9FFFF <1> CALL XLAT\_OLD ; TRANSLATE STATE TO COMPATIBLE MODE

2385 00003EF6 E865F9FFFF <1> CALL XLAT\_NEW ; TRANSLATE STATE TO PRESENT ARCH,

2386 <1> ;CALL TURN\_ON ; CHECK AGAIN IF MOTOR ON

2387 <1> ;JC MOT\_IS\_ON ; IF NO WAIT MEANS IT IS ON

2388 <1> M\_WAIT:

2389 00003EFB B20A <1> MOV DL,10 ; GET THE MOTOR WAIT PARAMETER

2390 00003EFD E8A2FFFFFF <1> CALL GET\_PARM

2391 <1> ;MOV AL,AH ; AL = MOTOR WAIT PARAMETER

2392 <1> ;XOR AH,AH ; AX = MOTOR WAIT PARAMETER

2393 <1> ;CMP AL,8 ; SEE IF AT LEAST A SECOND IS SPECIFIED

2394 00003F02 80FC08 <1> cmp ah, 8

2395 <1> ;JAE short GP2 ; IF YES, CONTINUE

2396 00003F05 7702 <1> ja short J13

2397 <1> ;MOV AL,8 ; ONE SECOND WAIT FOR MOTOR START UP

2398 00003F07 B408 <1> mov ah, 8

2399 <1>

2400 <1> ;----- AS CONTAINS NUMBER OF 1/8 SECONDS (125000 MICROSECONDS) TO WAIT

2401 <1> GP2:

2402 <1> ;----- FOLLOWING LOOPS REQUIRED WHEN RTC WAIT FUNCTION IS ALREADY IN USE

2403 <1> J13: ; WAIT FOR 1/8 SECOND PER (AL)

2404 00003F09 B95E200000 <1> MOV eCX,8286 ; COUNT FOR 1/8 SECOND AT 15.085737 US

2405 00003F0E E8DADEFFFF <1> CALL WAITF ; GO TO FIXED WAIT ROUTINE

2406 <1> ;DEC AL ; DECREMENT TIME VALUE

2407 00003F13 FECC <1> dec ah

2408 00003F15 75F2 <1> JNZ short J13 ; ARE WE DONE YET

2409 <1> MOT\_IS\_ON:

2410 00003F17 5B <1> POP eBX ; RESTORE REG.

2411 00003F18 C3 <1> RETn

2412 <1>

2413 <1> ;-------------------------------------------------------------------------------

2414 <1> ; TURN\_ON

2415 <1> ; TURN MOTOR ON AND RETURN WAIT STATE.

2416 <1> ;

2417 <1> ; ON ENTRY: DI = DRIVE #

2418 <1> ;

2419 <1> ; ON EXIT: CY = 0 MEANS WAIT REQUIRED

2420 <1> ; CY = 1 MEANS NO WAIT REQUIRED

2421 <1> ; AX,BX,CX,DX DESTROYED

2422 <1> ;-------------------------------------------------------------------------------

2423 <1> TURN\_ON:

2424 00003F19 89FB <1> MOV eBX,eDI ; BX = DRIVE #

2425 00003F1B 88D9 <1> MOV CL,BL ; CL = DRIVE #

2426 00003F1D C0C304 <1> ROL BL,4 ; BL = DRIVE SELECT

2427 00003F20 FA <1> CLI ; NO INTERRUPTS WHILE DETERMINING STATUS

2428 00003F21 C605[BF580100]FF <1> MOV byte [MOTOR\_COUNT],0FFH ; ENSURE MOTOR STAYS ON FOR OPERATION

2429 00003F28 A0[BE580100] <1> MOV AL, [MOTOR\_STATUS] ; GET DIGITAL OUTPUT REGISTER REFLECTION

2430 00003F2D 2430 <1> AND AL,00110000B ; KEEP ONLY DRIVE SELECT BITS

2431 00003F2F B401 <1> MOV AH,1 ; MASK FOR DETERMINING MOTOR BIT

2432 00003F31 D2E4 <1> SHL AH,CL ; AH = MOTOR ON, A=00000001, B=00000010

2433 <1>

2434 <1> ; AL = DRIVE SELECT FROM @MOTOR\_STATUS

2435 <1> ; BL = DRIVE SELECT DESIRED

2436 <1> ; AH = MOTOR ON MASK DESIRED

2437 <1>

2438 00003F33 38D8 <1> CMP AL,BL ; REQUESTED DRIVE ALREADY SELECTED ?

2439 00003F35 7508 <1> JNZ short TURN\_IT\_ON ; IF NOT SELECTED JUMP

2440 00003F37 8425[BE580100] <1> TEST AH, [MOTOR\_STATUS] ; TEST MOTOR ON BIT

2441 00003F3D 7535 <1> JNZ short NO\_MOT\_WAIT ; JUMP IF MOTOR ON AND SELECTED

2442 <1>

2443 <1> TURN\_IT\_ON:

2444 00003F3F 08DC <1> OR AH,BL ; AH = DRIVE SELECT AND MOTOR ON

2445 00003F41 8A3D[BE580100] <1> MOV BH,[MOTOR\_STATUS] ; SAVE COPY OF @MOTOR\_STATUS BEFORE

2446 00003F47 80E70F <1> AND BH,00001111B ; KEEP ONLY MOTOR BITS

2447 00003F4A 8025[BE580100]CF <1> AND byte [MOTOR\_STATUS],11001111B ; CLEAR OUT DRIVE SELECT

2448 00003F51 0825[BE580100] <1> OR [MOTOR\_STATUS],AH ; OR IN DRIVE SELECTED AND MOTOR ON

2449 00003F57 A0[BE580100] <1> MOV AL,[MOTOR\_STATUS] ; GET DIGITAL OUTPUT REGISTER REFLECTION

2450 00003F5C 88C3 <1> MOV BL,AL ; BL=@MOTOR\_STATUS AFTER, BH=BEFORE

2451 00003F5E 80E30F <1> AND BL,00001111B ; KEEP ONLY MOTOR BITS

2452 00003F61 FB <1> STI ; ENABLE INTERRUPTS AGAIN

2453 00003F62 243F <1> AND AL,00111111B ; STRIP AWAY UNWANTED BITS

2454 00003F64 C0C004 <1> ROL AL,4 ; PUT BITS IN DESIRED POSITIONS

2455 00003F67 0C0C <1> OR AL,00001100B ; NO RESET, ENABLE DMA/INTERRUPT

2456 00003F69 66BAF203 <1> MOV DX,03F2H ; SELECT DRIVE AND TURN ON MOTOR

2457 00003F6D EE <1> OUT DX,AL

2458 00003F6E 38FB <1> CMP BL,BH ; NEW MOTOR TURNED ON ?

2459 <1> ;JZ short NO\_MOT\_WAIT ; NO WAIT REQUIRED IF JUST SELECT

2460 00003F70 7403 <1> je short no\_mot\_w1 ; 27/02/2015

2461 00003F72 F8 <1> CLC ; (re)SET CARRY MEANING WAIT

2462 00003F73 C3 <1> RETn

2463 <1>

2464 <1> NO\_MOT\_WAIT:

2465 00003F74 FB <1> sti

2466 <1> no\_mot\_w1: ; 27/02/2015

2467 00003F75 F9 <1> STC ; SET NO WAIT REQUIRED

2468 <1> ;STI ; INTERRUPTS BACK ON

2469 00003F76 C3 <1> RETn

2470 <1>

2471 <1> ;-------------------------------------------------------------------------------

2472 <1> ; HD\_WAIT

2473 <1> ; WAIT FOR HEAD SETTLE TIME.

2474 <1> ;

2475 <1> ; ON ENTRY: DI = DRIVE #

2476 <1> ;

2477 <1> ; ON EXIT: AX,BX,CX,DX DESTROYED

2478 <1> ;-------------------------------------------------------------------------------

2479 <1> HD\_WAIT:

2480 00003F77 B209 <1> MOV DL,9 ; GET HEAD SETTLE PARAMETER

2481 00003F79 E826FFFFFF <1> CALL GET\_PARM

2482 00003F7E 08E4 <1> or ah, ah ; 17/12/2014

2483 00003F80 7519 <1> jnz short DO\_WAT

2484 00003F82 F605[BE580100]80 <1> TEST byte [MOTOR\_STATUS],10000000B ; SEE IF A WRITE OPERATION

2485 <1> ;JZ short ISNT\_WRITE ; IF NOT, DO NOT ENFORCE ANY VALUES

2486 <1> ;OR AH,AH ; CHECK FOR ANY WAIT?

2487 <1> ;JNZ short DO\_WAT ; IF THERE DO NOT ENFORCE

2488 00003F89 741E <1> jz short HW\_DONE

2489 00003F8B B40F <1> MOV AH,HD12\_SETTLE ; LOAD 1.2M HEAD SETTLE MINIMUM

2490 00003F8D 8A87[CD580100] <1> MOV AL,[DSK\_STATE+eDI] ; LOAD STATE

2491 00003F93 24C0 <1> AND AL,RATE\_MSK ; KEEP ONLY RATE

2492 00003F95 3C80 <1> CMP AL,RATE\_250 ; 1.2 M DRIVE ?

2493 00003F97 7502 <1> JNZ short DO\_WAT ; DEFAULT HEAD SETTLE LOADED

2494 <1> ;GP3:

2495 00003F99 B414 <1> MOV AH,HD320\_SETTLE ; USE 320/360 HEAD SETTLE

2496 <1> ; JMP SHORT DO\_WAT

2497 <1>

2498 <1> ;ISNT\_WRITE:

2499 <1> ; OR AH,AH ; CHECK FOR NO WAIT

2500 <1> ; JZ short HW\_DONE ; IF NOT WRITE AND 0 ITS OK

2501 <1>

2502 <1> ;----- AH CONTAINS NUMBER OF MILLISECONDS TO WAIT

2503 <1> DO\_WAT:

2504 <1> ; MOV AL,AH ; AL = # MILLISECONDS

2505 <1> ; ;XOR AH,AH ; AX = # MILLISECONDS

2506 <1> J29: ; 1 MILLISECOND LOOP

2507 <1> ;mov cx, WAIT\_FDU\_HEAD\_SETTLE ; 33 ; 1 ms in 30 micro units.

2508 00003F9B B942000000 <1> MOV eCX,66 ; COUNT AT 15.085737 US PER COUNT

2509 00003FA0 E848DEFFFF <1> CALL WAITF ; DELAY FOR 1 MILLISECOND

2510 <1> ;DEC AL ; DECREMENT THE COUNT

2511 00003FA5 FECC <1> dec ah

2512 00003FA7 75F2 <1> JNZ short J29 ; DO AL MILLISECOND # OF TIMES

2513 <1> HW\_DONE:

2514 00003FA9 C3 <1> RETn

2515 <1>

2516 <1> ;-------------------------------------------------------------------------------

2517 <1> ; NEC\_OUTPUT

2518 <1> ; THIS ROUTINE SENDS A BYTE TO THE NEC CONTROLLER AFTER TESTING

2519 <1> ; FOR CORRECT DIRECTION AND CONTROLLER READY THIS ROUTINE WILL

2520 <1> ; TIME OUT IF THE BYTE IS NOT ACCEPTED WITHIN A REASONABLE AMOUNT

2521 <1> ; OF TIME, SETTING THE DISKETTE STATUS ON COMPLETION.

2522 <1> ;

2523 <1> ; ON ENTRY: AH = BYTE TO BE OUTPUT

2524 <1> ;

2525 <1> ; ON EXIT: CY = 0 SUCCESS

2526 <1> ; CY = 1 FAILURE -- DISKETTE STATUS UPDATED

2527 <1> ; IF A FAILURE HAS OCCURRED, THE RETURN IS MADE ONE LEVEL

2528 <1> ; HIGHER THAN THE CALLER OF NEC OUTPUT. THIS REMOVES THE

2529 <1> ; REQUIREMENT OF TESTING AFTER EVERY CALL OF NEC\_OUTPUT.

2530 <1> ; AX,CX,DX DESTROYED

2531 <1> ;-------------------------------------------------------------------------------

2532 <1>

2533 <1> ; 09/12/2014 [Erdogan Tan]

2534 <1> ; (from 'PS2 Hardware Interface Tech. Ref. May 88', Page 09-05.)

2535 <1> ; Diskette Drive Controller Status Register (3F4h)

2536 <1> ; This read only register facilitates the transfer of data between

2537 <1> ; the system microprocessor and the controller.

2538 <1> ; Bit 7 - When set to 1, the Data register is ready to transfer data

2539 <1> ; with the system micrprocessor.

2540 <1> ; Bit 6 - The direction of data transfer. If this bit is set to 0,

2541 <1> ; the transfer is to the controller.

2542 <1> ; Bit 5 - When this bit is set to 1, the controller is in the non-DMA mode.

2543 <1> ; Bit 4 - When this bit is set to 1, a Read or Write command is being executed.

2544 <1> ; Bit 3 - Reserved.

2545 <1> ; Bit 2 - Reserved.

2546 <1> ; Bit 1 - When this bit is set to 1, dskette drive 1 is in the seek mode.

2547 <1> ; Bit 0 - When this bit is set to 1, dskette drive 1 is in the seek mode.

2548 <1>

2549 <1> ; Data Register (3F5h)

2550 <1> ; This read/write register passes data, commands and parameters, and provides

2551 <1> ; diskette status information.

2552 <1>

2553 <1> NEC\_OUTPUT:

2554 <1> ;PUSH BX ; SAVE REG.

2555 00003FAA 66BAF403 <1> MOV DX,03F4H ; STATUS PORT

2556 <1> ;MOV BL,2 ; HIGH ORDER COUNTER

2557 <1> ;XOR CX,CX ; COUNT FOR TIME OUT

2558 <1> ; 16/12/2014

2559 <1> ; waiting for (max.) 0.5 seconds

2560 <1> ;;mov byte [wait\_count], 0 ;; 27/02/2015

2561 <1> ;

2562 <1> ; 17/12/2014

2563 <1> ; Modified from AWARD BIOS 1999 - ADISK.ASM - SEND\_COMMAND

2564 <1> ;

2565 <1> ;WAIT\_FOR\_PORT: Waits for a bit at a port pointed to by DX to

2566 <1> ; go on.

2567 <1> ;INPUT:

2568 <1> ; AH=Mask for isolation bits.

2569 <1> ; AL=pattern to look for.

2570 <1> ; DX=Port to test for

2571 <1> ; BH:CX=Number of memory refresh periods to delay.

2572 <1> ; (normally 30 microseconds per period.)

2573 <1> ;

2574 <1> ;WFP\_SHORT:

2575 <1> ; Wait for port if refresh cycle is short (15-80 Us range).

2576 <1> ;

2577 <1>

2578 <1> ; mov bl, WAIT\_FDU\_SEND\_HI+1 ; 0+1

2579 <1> ; mov cx, WAIT\_FDU\_SEND\_LO ; 16667

2580 00003FAE B91B410000 <1> mov ecx, WAIT\_FDU\_SEND\_LH ; 16667 (27/02/2015)

2581 <1> ;

2582 <1> ;WFPS\_OUTER\_LP:

2583 <1> ; ;

2584 <1> ;WFPS\_CHECK\_PORT:

2585 <1> J23:

2586 00003FB3 EC <1> IN AL,DX ; GET STATUS

2587 00003FB4 24C0 <1> AND AL,11000000B ; KEEP STATUS AND DIRECTION

2588 00003FB6 3C80 <1> CMP AL,10000000B ; STATUS 1 AND DIRECTION 0 ?

2589 00003FB8 7418 <1> JZ short J27 ; STATUS AND DIRECTION OK

2590 <1> WFPS\_HI:

2591 00003FBA E461 <1> IN AL, PORT\_B ;061h ; SYS1 ; wait for hi to lo

2592 00003FBC A810 <1> TEST AL,010H ; transition on memory

2593 00003FBE 75FA <1> JNZ SHORT WFPS\_HI ; refresh.

2594 <1> WFPS\_LO:

2595 00003FC0 E461 <1> IN AL, PORT\_B ; SYS1

2596 00003FC2 A810 <1> TEST AL,010H

2597 00003FC4 74FA <1> JZ SHORT WFPS\_LO

2598 <1> ;LOOP SHORT WFPS\_CHECK\_PORT

2599 00003FC6 E2EB <1> loop J23 ; 27/02/2015

2600 <1> ; ;

2601 <1> ; dec bl

2602 <1> ; jnz short WFPS\_OUTER\_LP

2603 <1> ; jmp short WFPS\_TIMEOUT ; fail

2604 <1> ;J23:

2605 <1> ; IN AL,DX ; GET STATUS

2606 <1> ; AND AL,11000000B ; KEEP STATUS AND DIRECTION

2607 <1> ; CMP AL,10000000B ; STATUS 1 AND DIRECTION 0 ?

2608 <1> ; JZ short J27 ; STATUS AND DIRECTION OK

2609 <1> ;LOOP J23 ; CONTINUE TILL CX EXHAUSTED

2610 <1> ;DEC BL ; DECREMENT COUNTER

2611 <1> ;JNZ short J23 ; REPEAT TILL DELAY FINISHED, CX = 0

2612 <1>

2613 <1> ;;27/02/2015

2614 <1> ;16/12/2014

2615 <1> ;;cmp byte [wait\_count], 10 ; (10/18.2 seconds)

2616 <1> ;;jb short J23

2617 <1>

2618 <1> ;WFPS\_TIMEOUT:

2619 <1>

2620 <1> ;----- FALL THRU TO ERROR RETURN

2621 <1>

2622 00003FC8 800D[C0580100]80 <1> OR byte [DSKETTE\_STATUS],TIME\_OUT

2623 <1> ;POP BX ; RESTORE REG.

2624 00003FCF 58 <1> POP eAX ; 08/02/2015 ; DISCARD THE RETURN ADDRESS

2625 00003FD0 F9 <1> STC ; INDICATE ERROR TO CALLER

2626 00003FD1 C3 <1> RETn

2627 <1>

2628 <1> ;----- DIRECTION AND STATUS OK; OUTPUT BYTE

2629 <1>

2630 <1> J27:

2631 00003FD2 88E0 <1> MOV AL,AH ; GET BYTE TO OUTPUT

2632 00003FD4 6642 <1> INC DX ; DATA PORT = STATUS PORT + 1

2633 00003FD6 EE <1> OUT DX,AL ; OUTPUT THE BYTE

2634 <1> ;;NEWIODELAY ;; 27/02/2015

2635 <1> ; 27/02/2015

2636 00003FD7 9C <1> PUSHF ; SAVE FLAGS

2637 00003FD8 B903000000 <1> MOV eCX, 3 ; 30 TO 45 MICROSECONDS WAIT FOR

2638 00003FDD E80BDEFFFF <1> CALL WAITF ; NEC FLAGS UPDATE CYCLE

2639 00003FE2 9D <1> POPF ; RESTORE FLAGS FOR EXIT

2640 <1> ;POP BX ; RESTORE REG

2641 00003FE3 C3 <1> RETn ; CY = 0 FROM TEST INSTRUCTION

2642 <1>

2643 <1> ;-------------------------------------------------------------------------------

2644 <1> ; SEEK

2645 <1> ; THIS ROUTINE WILL MOVE THE HEAD ON THE NAMED DRIVE TO THE NAMED

2646 <1> ; TRACK. IF THE DRIVE HAS NOT BEEN ACCESSED SINCE THE DRIVE

2647 <1> ; RESET COMMAND WAS ISSUED, THE DRIVE WILL BE RECALIBRATED.

2648 <1> ;

2649 <1> ; ON ENTRY: DI = DRIVE #

2650 <1> ; CH = TRACK #

2651 <1> ;

2652 <1> ; ON EXIT: @DSKETTE\_STATUS, CY REFLECT STATUS OF OPERATION.

2653 <1> ; AX,BX,CX DX DESTROYED

2654 <1> ;-------------------------------------------------------------------------------

2655 <1> SEEK:

2656 00003FE4 89FB <1> MOV eBX,eDI ; BX = DRIVE #

2657 00003FE6 B001 <1> MOV AL,1 ; ESTABLISH MASK FOR RECALIBRATE TEST

2658 00003FE8 86CB <1> XCHG CL,BL ; SET DRIVE VALULE INTO CL

2659 00003FEA D2C0 <1> ROL AL,CL ; SHIFT MASK BY THE DRIVE VALUE

2660 00003FEC 86CB <1> XCHG CL,BL ; RECOVER TRACK VALUE

2661 00003FEE 8405[BD580100] <1> TEST AL,[SEEK\_STATUS] ; TEST FOR RECALIBRATE REQUIRED

2662 00003FF4 7526 <1> JNZ short J28A ; JUMP IF RECALIBRATE NOT REQUIRED

2663 <1>

2664 00003FF6 0805[BD580100] <1> OR [SEEK\_STATUS],AL ; TURN ON THE NO RECALIBRATE BIT IN FLAG

2665 00003FFC E862000000 <1> CALL RECAL ; RECALIBRATE DRIVE

2666 00004001 730E <1> JNC short AFT\_RECAL ; RECALIBRATE DONE

2667 <1>

2668 <1> ;----- ISSUE RECALIBRATE FOR 80 TRACK DISKETTES

2669 <1>

2670 00004003 C605[C0580100]00 <1> MOV byte [DSKETTE\_STATUS],0 ; CLEAR OUT INVALID STATUS

2671 0000400A E854000000 <1> CALL RECAL ; RECALIBRATE DRIVE

2672 0000400F 7251 <1> JC short RB ; IF RECALIBRATE FAILS TWICE THEN ERROR

2673 <1>

2674 <1> AFT\_RECAL:

2675 00004011 C687[D1580100]00 <1> MOV byte [DSK\_TRK+eDI],0 ; SAVE NEW CYLINDER AS PRESENT POSITION

2676 00004018 08ED <1> OR CH,CH ; CHECK FOR SEEK TO TRACK 0

2677 0000401A 743F <1> JZ short DO\_WAIT ; HEAD SETTLE, CY = 0 IF JUMP

2678 <1>

2679 <1> ;----- DRIVE IS IN SYNCHRONIZATION WITH CONTROLLER, SEEK TO TRACK

2680 <1>

2681 0000401C F687[CD580100]20 <1> J28A: TEST byte [DSK\_STATE+eDI],DBL\_STEP ; CHECK FOR DOUBLE STEP REQUIRED

2682 00004023 7402 <1> JZ short \_R7 ; SINGLE STEP REQUIRED BYPASS DOUBLE

2683 00004025 D0E5 <1> SHL CH,1 ; DOUBLE NUMBER OF STEP TO TAKE

2684 <1>

2685 00004027 3AAF[D1580100] <1> \_R7: CMP CH, [DSK\_TRK+eDI] ; SEE IF ALREADY AT THE DESIRED TRACK

2686 0000402D 7433 <1> JE short RB ; IF YES, DO NOT NEED TO SEEK

2687 <1>

2688 0000402F BA[62400000] <1> MOV eDX, NEC\_ERR ; LOAD RETURN ADDRESS

2689 00004034 52 <1> PUSH eDX ; (\*) ; ON STACK FOR NEC OUTPUT ERROR

2690 00004035 88AF[D1580100] <1> MOV [DSK\_TRK+eDI],CH ; SAVE NEW CYLINDER AS PRESENT POSITION

2691 0000403B B40F <1> MOV AH,0FH ; SEEK COMMAND TO NEC

2692 0000403D E868FFFFFF <1> CALL NEC\_OUTPUT

2693 00004042 89FB <1> MOV eBX,eDI ; BX = DRIVE #

2694 00004044 88DC <1> MOV AH,BL ; OUTPUT DRIVE NUMBER

2695 00004046 E85FFFFFFF <1> CALL NEC\_OUTPUT

2696 0000404B 8AA7[D1580100] <1> MOV AH, [DSK\_TRK+eDI] ; GET CYLINDER NUMBER

2697 00004051 E854FFFFFF <1> CALL NEC\_OUTPUT

2698 00004056 E829000000 <1> CALL CHK\_STAT\_2 ; ENDING INTERRUPT AND SENSE STATUS

2699 <1>

2700 <1> ;----- WAIT FOR HEAD SETTLE

2701 <1>

2702 <1> DO\_WAIT:

2703 0000405B 9C <1> PUSHF ; SAVE STATUS

2704 0000405C E816FFFFFF <1> CALL HD\_WAIT ; WAIT FOR HEAD SETTLE TIME

2705 00004061 9D <1> POPF ; RESTORE STATUS

2706 <1> RB:

2707 <1> NEC\_ERR:

2708 <1> ; 08/02/2015 (code trick here from original IBM PC/AT DISKETTE.ASM)

2709 <1> ; (\*) nec\_err -> retn (push edx -> pop edx) -> nec\_err -> retn

2710 00004062 C3 <1> RETn ; RETURN TO CALLER

2711 <1>

2712 <1> ;-------------------------------------------------------------------------------

2713 <1> ; RECAL

2714 <1> ; RECALIBRATE DRIVE

2715 <1> ;

2716 <1> ; ON ENTRY: DI = DRIVE #

2717 <1> ;

2718 <1> ; ON EXIT: CY REFLECTS STATUS OF OPERATION.

2719 <1> ;-------------------------------------------------------------------------------

2720 <1> RECAL:

2721 00004063 6651 <1> PUSH CX

2722 00004065 B8[81400000] <1> MOV eAX, RC\_BACK ; LOAD NEC\_OUTPUT ERROR

2723 0000406A 50 <1> PUSH eAX

2724 0000406B B407 <1> MOV AH,07H ; RECALIBRATE COMMAND

2725 0000406D E838FFFFFF <1> CALL NEC\_OUTPUT

2726 00004072 89FB <1> MOV eBX,eDI ; BX = DRIVE #

2727 00004074 88DC <1> MOV AH,BL

2728 00004076 E82FFFFFFF <1> CALL NEC\_OUTPUT ; OUTPUT THE DRIVE NUMBER

2729 0000407B E804000000 <1> CALL CHK\_STAT\_2 ; GET THE INTERRUPT AND SENSE INT STATUS

2730 00004080 58 <1> POP eAX ; THROW AWAY ERROR

2731 <1> RC\_BACK:

2732 00004081 6659 <1> POP CX

2733 00004083 C3 <1> RETn

2734 <1>

2735 <1> ;-------------------------------------------------------------------------------

2736 <1> ; CHK\_STAT\_2

2737 <1> ; THIS ROUTINE HANDLES THE INTERRUPT RECEIVED AFTER RECALIBRATE,

2738 <1> ; OR SEEK TO THE ADAPTER. THE INTERRUPT IS WAITED FOR, THE

2739 <1> ; INTERRUPT STATUS SENSED, AND THE RESULT RETURNED TO THE CALLER.

2740 <1> ;

2741 <1> ; ON EXIT: @DSKETTE\_STATUS, CY REFLECT STATUS OF OPERATION.

2742 <1> ;-------------------------------------------------------------------------------

2743 <1> CHK\_STAT\_2:

2744 00004084 B8[AC400000] <1> MOV eAX, CS\_BACK ; LOAD NEC\_OUTPUT ERROR ADDRESS

2745 00004089 50 <1> PUSH eAX

2746 0000408A E828000000 <1> CALL WAIT\_INT ; WAIT FOR THE INTERRUPT

2747 0000408F 721A <1> JC short J34 ; IF ERROR, RETURN IT

2748 00004091 B408 <1> MOV AH,08H ; SENSE INTERRUPT STATUS COMMAND

2749 00004093 E812FFFFFF <1> CALL NEC\_OUTPUT

2750 00004098 E84A000000 <1> CALL RESULTS ; READ IN THE RESULTS

2751 0000409D 720C <1> JC short J34

2752 0000409F A0[C1580100] <1> MOV AL,[NEC\_STATUS] ; GET THE FIRST STATUS BYTE

2753 000040A4 2460 <1> AND AL,01100000B ; ISOLATE THE BITS

2754 000040A6 3C60 <1> CMP AL,01100000B ; TEST FOR CORRECT VALUE

2755 000040A8 7403 <1> JZ short J35 ; IF ERROR, GO MARK IT

2756 000040AA F8 <1> CLC ; GOOD RETURN

2757 <1> J34:

2758 000040AB 58 <1> POP eAX ; THROW AWAY ERROR RETURN

2759 <1> CS\_BACK:

2760 000040AC C3 <1> RETn

2761 <1> J35:

2762 000040AD 800D[C0580100]40 <1> OR byte [DSKETTE\_STATUS], BAD\_SEEK

2763 000040B4 F9 <1> STC ; ERROR RETURN CODE

2764 000040B5 EBF4 <1> JMP SHORT J34

2765 <1>

2766 <1> ;-------------------------------------------------------------------------------

2767 <1> ; WAIT\_INT

2768 <1> ; THIS ROUTINE WAITS FOR AN INTERRUPT TO OCCUR A TIME OUT ROUTINE

2769 <1> ; TAKES PLACE DURING THE WAIT, SO THAT AN ERROR MAY BE RETURNED

2770 <1> ; IF THE DRIVE IS NOT READY.

2771 <1> ;

2772 <1> ; ON EXIT: @DSKETTE\_STATUS, CY REFLECT STATUS OF OPERATION.

2773 <1> ;-------------------------------------------------------------------------------

2774 <1>

2775 <1> ; 17/12/2014

2776 <1> ; 2.5 seconds waiting !

2777 <1> ;(AWARD BIOS - 1999, WAIT\_FDU\_INT\_LOW, WAIT\_FDU\_INT\_HI)

2778 <1> ; amount of time to wait for completion interrupt from NEC.

2779 <1>

2780 <1>

2781 <1> WAIT\_INT:

2782 000040B7 FB <1> STI ; TURN ON INTERRUPTS, JUST IN CASE

2783 000040B8 F8 <1> CLC ; CLEAR TIMEOUT INDICATOR

2784 <1> ;MOV BL,10 ; CLEAR THE COUNTERS

2785 <1> ;XOR CX,CX ; FOR 2 SECOND WAIT

2786 <1>

2787 <1> ; Modification from AWARD BIOS - 1999 (ATORGS.ASM, WAIT

2788 <1> ;

2789 <1> ;WAIT\_FOR\_MEM:

2790 <1> ; Waits for a bit at a specified memory location pointed

2791 <1> ; to by ES:[DI] to become set.

2792 <1> ;INPUT:

2793 <1> ; AH=Mask to test with.

2794 <1> ; ES:[DI] = memory location to watch.

2795 <1> ; BH:CX=Number of memory refresh periods to delay.

2796 <1> ; (normally 30 microseconds per period.)

2797 <1>

2798 <1> ; waiting for (max.) 2.5 secs in 30 micro units.

2799 <1> ; mov cx, WAIT\_FDU\_INT\_LO ; 017798

2800 <1> ;; mov bl, WAIT\_FDU\_INT\_HI

2801 <1> ; mov bl, WAIT\_FDU\_INT\_HI + 1

2802 <1> ; 27/02/2015

2803 000040B9 B986450100 <1> mov ecx, WAIT\_FDU\_INT\_LH ; 83334 (2.5 seconds)

2804 <1> WFMS\_CHECK\_MEM:

2805 000040BE F605[BD580100]80 <1> test byte [SEEK\_STATUS],INT\_FLAG ; TEST FOR INTERRUPT OCCURRING

2806 000040C5 7516 <1> jnz short J37

2807 <1> WFMS\_HI:

2808 000040C7 E461 <1> IN AL,PORT\_B ; 061h ; SYS1, wait for lo to hi

2809 000040C9 A810 <1> TEST AL,010H ; transition on memory

2810 000040CB 75FA <1> JNZ SHORT WFMS\_HI ; refresh.

2811 <1> WFMS\_LO:

2812 000040CD E461 <1> IN AL,PORT\_B ;SYS1

2813 000040CF A810 <1> TEST AL,010H

2814 000040D1 74FA <1> JZ SHORT WFMS\_LO

2815 000040D3 E2E9 <1> LOOP WFMS\_CHECK\_MEM

2816 <1> ;WFMS\_OUTER\_LP:

2817 <1> ;; or bl, bl ; check outer counter

2818 <1> ;; jz short J36A ; WFMS\_TIMEOUT

2819 <1> ; dec bl

2820 <1> ; jz short J36A

2821 <1> ; jmp short WFMS\_CHECK\_MEM

2822 <1>

2823 <1> ;17/12/2014

2824 <1> ;16/12/2014

2825 <1> ; mov byte [wait\_count], 0 ; Reset (INT 08H) counter

2826 <1> ;J36:

2827 <1> ; TEST byte [SEEK\_STATUS],INT\_FLAG ; TEST FOR INTERRUPT OCCURRING

2828 <1> ; JNZ short J37

2829 <1> ;16/12/2014

2830 <1> ;LOOP J36 ; COUNT DOWN WHILE WAITING

2831 <1> ;DEC BL ; SECOND LEVEL COUNTER

2832 <1> ;JNZ short J36

2833 <1> ; cmp byte [wait\_count], 46 ; (46/18.2 seconds)

2834 <1> ; jb short J36

2835 <1>

2836 <1> ;WFMS\_TIMEOUT:

2837 <1> ;J36A:

2838 000040D5 800D[C0580100]80 <1> OR byte [DSKETTE\_STATUS], TIME\_OUT ; NOTHING HAPPENED

2839 000040DC F9 <1> STC ; ERROR RETURN

2840 <1> J37:

2841 000040DD 9C <1> PUSHF ; SAVE CURRENT CARRY

2842 000040DE 8025[BD580100]7F <1> AND byte [SEEK\_STATUS], ~INT\_FLAG ; TURN OFF INTERRUPT FLAG

2843 000040E5 9D <1> POPF ; RECOVER CARRY

2844 000040E6 C3 <1> RETn ; GOOD RETURN CODE

2845 <1>

2846 <1> ;-------------------------------------------------------------------------------

2847 <1> ; RESULTS

2848 <1> ; THIS ROUTINE WILL READ ANYTHING THAT THE NEC CONTROLLER RETURNS

2849 <1> ; FOLLOWING AN INTERRUPT.

2850 <1> ;

2851 <1> ; ON EXIT: @DSKETTE\_STATUS, CY REFLECT STATUS OF OPERATION.

2852 <1> ; AX,BX,CX,DX DESTROYED

2853 <1> ;-------------------------------------------------------------------------------

2854 <1> RESULTS:

2855 000040E7 57 <1> PUSH eDI

2856 000040E8 BF[C1580100] <1> MOV eDI, NEC\_STATUS ; POINTER TO DATA AREA

2857 000040ED B307 <1> MOV BL,7 ; MAX STATUS BYTES

2858 000040EF 66BAF403 <1> MOV DX,03F4H ; STATUS PORT

2859 <1>

2860 <1> ;----- WAIT FOR REQUEST FOR MASTER

2861 <1>

2862 <1> \_R10:

2863 <1> ; 16/12/2014

2864 <1> ; wait for (max) 0.5 seconds

2865 <1> ;MOV BH,2 ; HIGH ORDER COUNTER

2866 <1> ;XOR CX,CX ; COUNTER

2867 <1>

2868 <1> ;Time to wait while waiting for each byte of NEC results = .5

2869 <1> ;seconds. .5 seconds = 500,000 micros. 500,000/30 = 16,667.

2870 <1> ; 27/02/2015

2871 000040F3 B91B410000 <1> mov ecx, WAIT\_FDU\_RESULTS\_LH ; 16667

2872 <1> ;mov cx, WAIT\_FDU\_RESULTS\_LO ; 16667

2873 <1> ;mov bh, WAIT\_FDU\_RESULTS\_HI+1 ; 0+1

2874 <1>

2875 <1> WFPSR\_OUTER\_LP:

2876 <1> ;

2877 <1> WFPSR\_CHECK\_PORT:

2878 <1> J39: ; WAIT FOR MASTER

2879 000040F8 EC <1> IN AL,DX ; GET STATUS

2880 000040F9 24C0 <1> AND AL,11000000B ; KEEP ONLY STATUS AND DIRECTION

2881 000040FB 3CC0 <1> CMP AL,11000000B ; STATUS 1 AND DIRECTION 1 ?

2882 000040FD 7418 <1> JZ short J42 ; STATUS AND DIRECTION OK

2883 <1> WFPSR\_HI:

2884 000040FF E461 <1> IN AL, PORT\_B ;061h ; SYS1 ; wait for hi to lo

2885 00004101 A810 <1> TEST AL,010H ; transition on memory

2886 00004103 75FA <1> JNZ SHORT WFPSR\_HI ; refresh.

2887 <1> WFPSR\_LO:

2888 00004105 E461 <1> IN AL, PORT\_B ; SYS1

2889 00004107 A810 <1> TEST AL,010H

2890 00004109 74FA <1> JZ SHORT WFPSR\_LO

2891 0000410B E2EB <1> LOOP WFPSR\_CHECK\_PORT

2892 <1> ;; 27/02/2015

2893 <1> ;;dec bh

2894 <1> ;;jnz short WFPSR\_OUTER\_LP

2895 <1> ;jmp short WFPSR\_TIMEOUT ; fail

2896 <1>

2897 <1> ;;mov byte [wait\_count], 0

2898 <1> ;J39: ; WAIT FOR MASTER

2899 <1> ; IN AL,DX ; GET STATUS

2900 <1> ; AND AL,11000000B ; KEEP ONLY STATUS AND DIRECTION

2901 <1> ; CMP AL,11000000B ; STATUS 1 AND DIRECTION 1 ?

2902 <1> ; JZ short J42 ; STATUS AND DIRECTION OK

2903 <1> ;LOOP J39 ; LOOP TILL TIMEOUT

2904 <1> ;DEC BH ; DECREMENT HIGH ORDER COUNTER

2905 <1> ;JNZ short J39 ; REPEAT TILL DELAY DONE

2906 <1> ;

2907 <1> ;;cmp byte [wait\_count], 10 ; (10/18.2 seconds)

2908 <1> ;;jb short J39

2909 <1>

2910 <1> ;WFPSR\_TIMEOUT:

2911 0000410D 800D[C0580100]80 <1> OR byte [DSKETTE\_STATUS],TIME\_OUT

2912 00004114 F9 <1> STC ; SET ERROR RETURN

2913 00004115 EB29 <1> JMP SHORT POPRES ; POP REGISTERS AND RETURN

2914 <1>

2915 <1> ;----- READ IN THE STATUS

2916 <1>

2917 <1> J42:

2918 00004117 EB00 <1> JMP $+2 ; I/O DELAY

2919 00004119 6642 <1> INC DX ; POINT AT DATA PORT

2920 0000411B EC <1> IN AL,DX ; GET THE DATA

2921 <1> ; 16/12/2014

2922 <1> NEWIODELAY

2922 0000411C E6EB <2> out 0ebh,al

2923 0000411E 8807 <1> MOV [eDI],AL ; STORE THE BYTE

2924 00004120 47 <1> INC eDI ; INCREMENT THE POINTER

2925 <1> ; 16/12/2014

2926 <1> ; push cx

2927 <1> ; mov cx, 30

2928 <1> ;wdw2:

2929 <1> ; NEWIODELAY

2930 <1> ; loop wdw2

2931 <1> ; pop cx

2932 <1>

2933 00004121 B903000000 <1> MOV eCX,3 ; MINIMUM 24 MICROSECONDS FOR NEC

2934 00004126 E8C2DCFFFF <1> CALL WAITF ; WAIT 30 TO 45 MICROSECONDS

2935 0000412B 664A <1> DEC DX ; POINT AT STATUS PORT

2936 0000412D EC <1> IN AL,DX ; GET STATUS

2937 <1> ; 16/12/2014

2938 <1> NEWIODELAY

2938 0000412E E6EB <2> out 0ebh,al

2939 <1> ;

2940 00004130 A810 <1> TEST AL,00010000B ; TEST FOR NEC STILL BUSY

2941 00004132 740C <1> JZ short POPRES ; RESULTS DONE ?

2942 <1>

2943 00004134 FECB <1> DEC BL ; DECREMENT THE STATUS COUNTER

2944 00004136 75BB <1> JNZ short \_R10 ; GO BACK FOR MORE

2945 00004138 800D[C0580100]20 <1> OR byte [DSKETTE\_STATUS],BAD\_NEC ; TOO MANY STATUS BYTES

2946 0000413F F9 <1> STC ; SET ERROR FLAG

2947 <1>

2948 <1> ;----- RESULT OPERATION IS DONE

2949 <1> POPRES:

2950 00004140 5F <1> POP eDI

2951 00004141 C3 <1> RETn ; RETURN WITH CARRY SET

2952 <1>

2953 <1> ;-------------------------------------------------------------------------------

2954 <1> ; READ\_DSKCHNG

2955 <1> ; READS THE STATE OF THE DISK CHANGE LINE.

2956 <1> ;

2957 <1> ; ON ENTRY: DI = DRIVE #

2958 <1> ;

2959 <1> ; ON EXIT: DI = DRIVE #

2960 <1> ; ZF = 0 : DISK CHANGE LINE INACTIVE

2961 <1> ; ZF = 1 : DISK CHANGE LINE ACTIVE

2962 <1> ; AX,CX,DX DESTROYED

2963 <1> ;-------------------------------------------------------------------------------

2964 <1> READ\_DSKCHNG:

2965 00004142 E8A2FDFFFF <1> CALL MOTOR\_ON ; TURN ON THE MOTOR IF OFF

2966 00004147 66BAF703 <1> MOV DX,03F7H ; ADDRESS DIGITAL INPUT REGISTER

2967 0000414B EC <1> IN AL,DX ; INPUT DIGITAL INPUT REGISTER

2968 0000414C A880 <1> TEST AL,DSK\_CHG ; CHECK FOR DISK CHANGE LINE ACTIVE

2969 0000414E C3 <1> RETn ; RETURN TO CALLER WITH ZERO FLAG SET

2970 <1>

2971 <1> ;-------------------------------------------------------------------------------

2972 <1> ; DRIVE\_DET

2973 <1> ; DETERMINES WHETHER DRIVE IS 80 OR 40 TRACKS AND

2974 <1> ; UPDATES STATE INFORMATION ACCORDINGLY.

2975 <1> ; ON ENTRY: DI = DRIVE #

2976 <1> ;-------------------------------------------------------------------------------

2977 <1> DRIVE\_DET:

2978 0000414F E895FDFFFF <1> CALL MOTOR\_ON ; TURN ON MOTOR IF NOT ALREADY ON

2979 00004154 E80AFFFFFF <1> CALL RECAL ; RECALIBRATE DRIVE

2980 00004159 7251 <1> JC short DD\_BAC ; ASSUME NO DRIVE PRESENT

2981 0000415B B530 <1> MOV CH,TRK\_SLAP ; SEEK TO TRACK 48

2982 0000415D E882FEFFFF <1> CALL SEEK

2983 00004162 7248 <1> JC short DD\_BAC ; ERROR NO DRIVE

2984 00004164 B50B <1> MOV CH,QUIET\_SEEK+1 ; SEEK TO TRACK 10

2985 <1> SK\_GIN:

2986 00004166 FECD <1> DEC CH ; DECREMENT TO NEXT TRACK

2987 00004168 6651 <1> PUSH CX ; SAVE TRACK

2988 0000416A E875FEFFFF <1> CALL SEEK

2989 0000416F 723C <1> JC short POP\_BAC ; POP AND RETURN

2990 00004171 B8[AD410000] <1> MOV eAX, POP\_BAC ; LOAD NEC OUTPUT ERROR ADDRESS

2991 00004176 50 <1> PUSH eAX

2992 00004177 B404 <1> MOV AH,SENSE\_DRV\_ST ; SENSE DRIVE STATUS COMMAND BYTE

2993 00004179 E82CFEFFFF <1> CALL NEC\_OUTPUT ; OUTPUT TO NEC

2994 0000417E 6689F8 <1> MOV AX,DI ; AL = DRIVE

2995 00004181 88C4 <1> MOV AH,AL ; AH = DRIVE

2996 00004183 E822FEFFFF <1> CALL NEC\_OUTPUT ; OUTPUT TO NEC

2997 00004188 E85AFFFFFF <1> CALL RESULTS ; GO GET STATUS

2998 0000418D 58 <1> POP eAX ; THROW AWAY ERROR ADDRESS

2999 0000418E 6659 <1> POP CX ; RESTORE TRACK

3000 00004190 F605[C1580100]10 <1> TEST byte [NEC\_STATUS], HOME ; TRACK 0 ?

3001 00004197 74CD <1> JZ short SK\_GIN ; GO TILL TRACK 0

3002 00004199 08ED <1> OR CH,CH ; IS HOME AT TRACK 0

3003 0000419B 7408 <1> JZ short IS\_80 ; MUST BE 80 TRACK DRIVE

3004 <1>

3005 <1> ; DRIVE IS A 360; SET DRIVE TO DETERMINED;

3006 <1> ; SET MEDIA TO DETERMINED AT RATE 250.

3007 <1>

3008 0000419D 808F[CD580100]94 <1> OR byte [DSK\_STATE+eDI], DRV\_DET+MED\_DET+RATE\_250

3009 000041A4 C3 <1> RETn ; ALL INFORMATION SET

3010 <1> IS\_80:

3011 000041A5 808F[CD580100]01 <1> OR byte [DSK\_STATE+eDI], TRK\_CAPA ; SETUP 80 TRACK CAPABILITY

3012 <1> DD\_BAC:

3013 000041AC C3 <1> RETn

3014 <1> POP\_BAC:

3015 000041AD 6659 <1> POP CX ; THROW AWAY

3016 000041AF C3 <1> RETn

3017 <1>

3018 <1> fdc\_int:

3019 <1> ; 30/07/2015

3020 <1> ; 16/02/2015

3021 <1> ;int\_0Eh: ; 11/12/2014

3022 <1>

3023 <1> ;--- HARDWARE INT 0EH -- ( IRQ LEVEL 6 ) --------------------------------------

3024 <1> ; DISK\_INT

3025 <1> ; THIS ROUTINE HANDLES THE DISKETTE INTERRUPT.

3026 <1> ;

3027 <1> ; ON EXIT: THE INTERRUPT FLAG IS SET IN @SEEK\_STATUS.

3028 <1> ;-------------------------------------------------------------------------------

3029 <1> DISK\_INT\_1:

3030 <1>

3031 000041B0 6650 <1> PUSH AX ; SAVE WORK REGISTER

3032 000041B2 1E <1> push ds

3033 000041B3 66B81000 <1> mov ax, KDATA

3034 000041B7 8ED8 <1> mov ds, ax

3035 000041B9 800D[BD580100]80 <1> OR byte [SEEK\_STATUS], INT\_FLAG ; TURN ON INTERRUPT OCCURRED

3036 000041C0 B020 <1> MOV AL,EOI ; END OF INTERRUPT MARKER

3037 000041C2 E620 <1> OUT INTA00,AL ; INTERRUPT CONTROL PORT

3038 000041C4 1F <1> pop ds

3039 000041C5 6658 <1> POP AX ; RECOVER REGISTER

3040 000041C7 CF <1> IRETd ; RETURN FROM INTERRUPT

3041 <1>

3042 <1> ;-------------------------------------------------------------------------------

3043 <1> ; DSKETTE\_SETUP

3044 <1> ; THIS ROUTINE DOES A PRELIMINARY CHECK TO SEE WHAT TYPE OF

3045 <1> ; DISKETTE DRIVES ARE ATTACH TO THE SYSTEM.

3046 <1> ;-------------------------------------------------------------------------------

3047 <1>

3048 <1> ; 29/05/2016 - TRDOS 386 (TRDOS v2.0)

3049 <1>

3050 <1> DSKETTE\_SETUP:

3051 <1> ;PUSH AX ; SAVE REGISTERS

3052 <1> ;PUSH BX

3053 <1> ;PUSH CX

3054 000041C8 52 <1> PUSH eDX

3055 <1> ;PUSH DI

3056 <1> ;;PUSH DS

3057 <1> ; 14/12/2014

3058 <1> ;mov word [DISK\_POINTER], MD\_TBL6

3059 <1> ;mov [DISK\_POINTER+2], cs

3060 <1> ;

3061 <1> ;OR byte [RTC\_WAIT\_FLAG], 1 ; NO RTC WAIT, FORCE USE OF LOOP

3062 000041C9 31FF <1> XOR eDI,eDI ; INITIALIZE DRIVE POINTER

3063 000041CB 66C705[CD580100]00- <1> MOV WORD [DSK\_STATE],0 ; INITIALIZE STATES

3063 000041D3 00 <1>

3064 000041D4 8025[C8580100]33 <1> AND byte [LASTRATE],~(STRT\_MSK+SEND\_MSK) ; CLEAR START & SEND

3065 000041DB 800D[C8580100]C0 <1> OR byte [LASTRATE],SEND\_MSK ; INITIALIZE SENT TO IMPOSSIBLE

3066 000041E2 C605[BD580100]00 <1> MOV byte [SEEK\_STATUS],0 ; INDICATE RECALIBRATE NEEDED

3067 000041E9 C605[BF580100]00 <1> MOV byte [MOTOR\_COUNT],0 ; INITIALIZE MOTOR COUNT

3068 000041F0 C605[BE580100]00 <1> MOV byte [MOTOR\_STATUS],0 ; INITIALIZE DRIVES TO OFF STATE

3069 000041F7 C605[C0580100]00 <1> MOV byte [DSKETTE\_STATUS],0 ; NO ERRORS

3070 <1> ;

3071 <1> ; 28/02/2015

3072 <1> ;mov word [cfd], 100h

3073 000041FE E848F2FFFF <1> call DSK\_RESET

3074 00004203 5A <1> pop edx

3075 00004204 F8 <1> clc ; 29/05/2016

3076 00004205 C3 <1> retn

3077 <1>

3078 <1> ;SUP0:

3079 <1> ; CALL DRIVE\_DET ; DETERMINE DRIVE

3080 <1> ; CALL XLAT\_OLD ; TRANSLATE STATE TO COMPATIBLE MODE

3081 <1> ; ; 02/01/2015

3082 <1> ; ;INC DI ; POINT TO NEXT DRIVE

3083 <1> ; ;CMP DI,MAX\_DRV ; SEE IF DONE

3084 <1> ; ;JNZ short SUP0 ; REPEAT FOR EACH ORIVE

3085 <1> ; cmp byte [fd1\_type], 0

3086 <1> ; jna short sup1

3087 <1> ; or di, di

3088 <1> ; jnz short sup1

3089 <1> ; inc di

3090 <1> ; jmp short SUP0

3091 <1> ;sup1:

3092 <1> ; MOV byte [SEEK\_STATUS],0 ; FORCE RECALIBRATE

3093 <1> ; ;AND byte [RTC\_WAIT\_FLAG],0FEH ; ALLOW FOR RTC WAIT

3094 <1> ; CALL SETUP\_END ; VARIOUS CLEANUPS

3095 <1> ; ;;POP DS ; RESTORE CALLERS REGISTERS

3096 <1> ; ;POP DI

3097 <1> ; POP eDX

3098 <1> ; ;POP CX

3099 <1> ; ;POP BX

3100 <1> ; ;POP AX

3101 <1> ; RETn

3102 <1>

3103 <1> ;//////////////////////////////////////////////////////

3104 <1> ;; END OF DISKETTE I/O ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

3105 <1> ;

3106 <1>

3107 <1> int13h: ; 21/02/2015

3108 00004206 9C <1> pushfd

3109 00004207 0E <1> push cs

3110 00004208 E843010000 <1> call DISK\_IO

3111 0000420D C3 <1> retn

3112 <1>

3113 <1> ;;;;;; DISK I/O ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; 21/02/2015 ;;;

3114 <1> ;/////////////////////////////////////////////////////////////////////

3115 <1>

3116 <1> ; DISK I/O - Erdogan Tan (Retro UNIX 386 v1 project)

3117 <1> ; 18/02/2016

3118 <1> ; 17/02/2016

3119 <1> ; 23/02/2015

3120 <1> ; 21/02/2015 (unix386.s)

3121 <1> ; 22/12/2014 - 14/02/2015 (dsectrm2.s)

3122 <1> ;

3123 <1> ; Original Source Code:

3124 <1> ; DISK ----- 09/25/85 FIXED DISK BIOS

3125 <1> ; (IBM PC XT Model 286 System BIOS Source Code, 04-21-86)

3126 <1> ;

3127 <1> ; Modifications: by reference of AWARD BIOS 1999 (D1A0622)

3128 <1> ; Source Code - ATORGS.ASM, AHDSK.ASM

3129 <1> ;

3130 <1>

3131 <1>

3132 <1> ;The wait for controller to be not busy is 10 seconds.

3133 <1> ;10,000,000 / 30 = 333,333. 333,333 decimal = 051615h

3134 <1> ;;WAIT\_HDU\_CTLR\_BUSY\_LO equ 1615h

3135 <1> ;;WAIT\_HDU\_CTLR\_BUSY\_HI equ 05h

3136 <1> WAIT\_HDU\_CTRL\_BUSY\_LH equ 51615h ;21/02/2015

3137 <1>

3138 <1> ;The wait for controller to issue completion interrupt is 10 seconds.

3139 <1> ;10,000,000 / 30 = 333,333. 333,333 decimal = 051615h

3140 <1> ;;WAIT\_HDU\_INT\_LO equ 1615h

3141 <1> ;;WAIT\_HDU\_INT\_HI equ 05h

3142 <1> WAIT\_HDU\_INT\_LH equ 51615h ; 21/02/2015

3143 <1>

3144 <1> ;The wait for Data request on read and write longs is

3145 <1> ;2000 us. (?)

3146 <1> ;;WAIT\_HDU\_DRQ\_LO equ 1000 ; 03E8h

3147 <1> ;;WAIT\_HDU\_DRQ\_HI equ 0

3148 <1> WAIT\_HDU\_DRQ\_LH equ 1000 ; 21/02/2015

3149 <1>

3150 <1> ; Port 61h (PORT\_B)

3151 <1> SYS1 equ 61h ; PORT\_B (diskette.inc)

3152 <1>

3153 <1> ; 23/12/2014

3154 <1> %define CMD\_BLOCK eBP-8 ; 21/02/2015

3155 <1>

3156 <1>

3157 <1> ;--- INT 13H -------------------------------------------------------------------

3158 <1> ; :

3159 <1> ; FIXED DISK I/O INTERFACE :

3160 <1> ; :

3161 <1> ; THIS INTERFACE PROVIDES ACCESS TO 5 1/4" FIXED DISKS THROUGH :

3162 <1> ; THE IBM FIXED DISK CONTROLLER. :

3163 <1> ; :

3164 <1> ; THE BIOS ROUTINES ARE MEANT TO BE ACCESSED THROUGH :

3165 <1> ; SOFTWARE INTERRUPTS ONLY. ANY ADDRESSES PRESENT IN :

3166 <1> ; THESE LISTINGS ARE INCLUDED ONLY FOR COMPLETENESS, :

3167 <1> ; NOT FOR REFERENCE. APPLICATIONS WHICH REFERENCE ANY :

3168 <1> ; ABSOLUTE ADDRESSES WITHIN THE CODE SEGMENTS OF BIOS :

3169 <1> ; VIOLATE THE STRUCTURE AND DESIGN OF BIOS. :

3170 <1> ; :

3171 <1> ;------------------------------------------------------------------------------:

3172 <1> ; :

3173 <1> ; INPUT (AH)= HEX COMMAND VALUE :

3174 <1> ; :

3175 <1> ; (AH)= 00H RESET DISK (DL = 80H,81H) / DISKETTE :

3176 <1> ; (AH)= 01H READ THE STATUS OF THE LAST DISK OPERATION INTO (AL) :

3177 <1> ; NOTE: DL < 80H - DISKETTE :

3178 <1> ; DL > 80H - DISK :

3179 <1> ; (AH)= 02H READ THE DESIRED SECTORS INTO MEMORY :

3180 <1> ; (AH)= 03H WRITE THE DESIRED SECTORS FROM MEMORY :

3181 <1> ; (AH)= 04H VERIFY THE DESIRED SECTORS :

3182 <1> ; (AH)= 05H FORMAT THE DESIRED TRACK :

3183 <1> ; (AH)= 06H UNUSED :

3184 <1> ; (AH)= 07H UNUSED :

3185 <1> ; (AH)= 08H RETURN THE CURRENT DRIVE PARAMETERS :

3186 <1> ; (AH)= 09H INITIALIZE DRIVE PAIR CHARACTERISTICS :

3187 <1> ; INTERRUPT 41 POINTS TO DATA BLOCK FOR DRIVE 0 :

3188 <1> ; INTERRUPT 46 POINTS TO DATA BLOCK FOR DRIVE 1 :

3189 <1> ; (AH)= 0AH READ LONG :

3190 <1> ; (AH)= 0BH WRITE LONG (READ & WRITE LONG ENCOMPASS 512 + 4 BYTES ECC) :

3191 <1> ; (AH)= 0CH SEEK :

3192 <1> ; (AH)= 0DH ALTERNATE DISK RESET (SEE DL) :

3193 <1> ; (AH)= 0EH UNUSED :

3194 <1> ; (AH)= 0FH UNUSED :

3195 <1> ; (AH)= 10H TEST DRIVE READY :

3196 <1> ; (AH)= 11H RECALIBRATE :

3197 <1> ; (AH)= 12H UNUSED :

3198 <1> ; (AH)= 13H UNUSED :

3199 <1> ; (AH)= 14H CONTROLLER INTERNAL DIAGNOSTIC :

3200 <1> ; (AH)= 15H READ DASD TYPE :

3201 <1> ; :

3202 <1> ;-------------------------------------------------------------------------------

3203 <1> ; :

3204 <1> ; REGISTERS USED FOR FIXED DISK OPERATIONS :

3205 <1> ; :

3206 <1> ; (DL) - DRIVE NUMBER (80H-81H FOR DISK. VALUE CHECKED) :

3207 <1> ; (DH) - HEAD NUMBER (0-15 ALLOWED, NOT VALUE CHECKED) :

3208 <1> ; (CH) - CYLINDER NUMBER (0-1023, NOT VALUE CHECKED)(SEE CL):

3209 <1> ; (CL) - SECTOR NUMBER (1-17, NOT VALUE CHECKED) :

3210 <1> ; :

3211 <1> ; NOTE: HIGH 2 BITS OF CYLINDER NUMBER ARE PLACED :

3212 <1> ; IN THE HIGH 2 BITS OF THE CL REGISTER :

3213 <1> ; (10 BITS TOTAL) :

3214 <1> ; :

3215 <1> ; (AL) - NUMBER OF SECTORS (MAXIMUM POSSIBLE RANGE 1-80H, :

3216 <1> ; FOR READ/WRITE LONG 1-79H) :

3217 <1> ; :

3218 <1> ; (ES:BX) - ADDRESS OF BUFFER FOR READS AND WRITES, :

3219 <1> ; (NOT REQUIRED FOR VERIFY) :

3220 <1> ; :

3221 <1> ; FORMAT (AH=5) ES:BX POINTS TO A 512 BYTE BUFFER. THE FIRST :

3222 <1> ; 2\*(SECTORS/TRACK) BYTES CONTAIN F,N FOR EACH SECTOR.:

3223 <1> ; F = 00H FOR A GOOD SECTOR :

3224 <1> ; 80H FOR A BAD SECTOR :

3225 <1> ; N = SECTOR NUMBER :

3226 <1> ; FOR AN INTERLEAVE OF 2 AND 17 SECTORS/TRACK :

3227 <1> ; THE TABLE SHOULD BE: :

3228 <1> ; :

3229 <1> ; DB 00H,01H,00H,0AH,00H,02H,00H,0BH,00H,03H,00H,0CH :

3230 <1> ; DB 00H,04H,00H,0DH,00H,05H,00H,0EH,00H,06H,00H,0FH :

3231 <1> ; DB 00H,07H,00H,10H,00H,08H,00H,11H,00H,09H :

3232 <1> ; :

3233 <1> ;-------------------------------------------------------------------------------

3234 <1>

3235 <1> ;-------------------------------------------------------------------------------

3236 <1> ; OUTPUT :

3237 <1> ; AH = STATUS OF CURRENT OPERATION :

3238 <1> ; STATUS BITS ARE DEFINED IN THE EQUATES BELOW :

3239 <1> ; CY = 0 SUCCESSFUL OPERATION (AH=0 ON RETURN) :

3240 <1> ; CY = 1 FAILED OPERATION (AH HAS ERROR REASON) :

3241 <1> ; :

3242 <1> ; NOTE: ERROR 11H INDICATES THAT THE DATA READ HAD A RECOVERABLE :

3243 <1> ; ERROR WHICH WAS CORRECTED BY THE ECC ALGORITHM. THE DATA :

3244 <1> ; IS PROBABLY GOOD, HOWEVER THE BIOS ROUTINE INDICATES AN :

3245 <1> ; ERROR TO ALLOW THE CONTROLLING PROGRAM A CHANCE TO DECIDE :

3246 <1> ; FOR ITSELF. THE ERROR MAY NOT RECUR IF THE DATA IS :

3247 <1> ; REWRITTEN. :

3248 <1> ; :

3249 <1> ; IF DRIVE PARAMETERS WERE REQUESTED (DL >= 80H), :

3250 <1> ; INPUT: :

3251 <1> ; (DL) = DRIVE NUMBER :

3252 <1> ; ; 27/05/2016 - TRDOS 386 (TRDOS v2.0) :

3253 <1> ; EBX = Buffer address for fixed disk parameters table (32 bytes) :

3254 <1> ; OUTPUT: :

3255 <1> ; (DL) = NUMBER OF CONSECUTIVE ACKNOWLEDGING DRIVES ATTACHED (1-2) :

3256 <1> ; (CONTROLLER CARD ZERO TALLY ONLY) :

3257 <1> ; (DH) = MAXIMUM USEABLE VALUE FOR HEAD NUMBER :

3258 <1> ; (CH) = MAXIMUM USEABLE VALUE FOR CYLINDER NUMBER :

3259 <1> ; (CL) = MAXIMUM USEABLE VALUE FOR SECTOR NUMBER :

3260 <1> ; AND CYLINDER NUMBER HIGH BITS :

3261 <1> ; :

3262 <1> ; IF READ DASD TYPE WAS REQUESTED, :

3263 <1> ; :

3264 <1> ; AH = 0 - NOT PRESENT :

3265 <1> ; 1 - DISKETTE - NO CHANGE LINE AVAILABLE :

3266 <1> ; 2 - DISKETTE - CHANGE LINE AVAILABLE :

3267 <1> ; 3 - FIXED DISK :

3268 <1> ; :

3269 <1> ; CX,DX = NUMBER OF 512 BYTE BLOCKS WHEN AH = 3 :

3270 <1> ; :

3271 <1> ; REGISTERS WILL BE PRESERVED EXCEPT WHEN THEY ARE USED TO RETURN :

3272 <1> ; INFORMATION. :

3273 <1> ; :

3274 <1> ; NOTE: IF AN ERROR IS REPORTED BY THE DISK CODE, THE APPROPRIATE :

3275 <1> ; ACTION IS TO RESET THE DISK, THEN RETRY THE OPERATION. :

3276 <1> ; :

3277 <1> ;-------------------------------------------------------------------------------

3278 <1>

3279 <1> SENSE\_FAIL EQU 0FFH ; NOT IMPLEMENTED

3280 <1> NO\_ERR EQU 0E0H ; STATUS ERROR/ERROR REGISTER=0

3281 <1> WRITE\_FAULT EQU 0CCH ; WRITE FAULT ON SELECTED DRIVE

3282 <1> UNDEF\_ERR EQU 0BBH ; UNDEFINED ERROR OCCURRED

3283 <1> NOT\_RDY EQU 0AAH ; DRIVE NOT READY

3284 <1> TIME\_OUT EQU 80H ; ATTACHMENT FAILED TO RESPOND

3285 <1> BAD\_SEEK EQU 40H ; SEEK OPERATION FAILED

3286 <1> BAD\_CNTLR EQU 20H ; CONTROLLER HAS FAILED

3287 <1> DATA\_CORRECTED EQU 11H ; ECC CORRECTED DATA ERROR

3288 <1> BAD\_ECC EQU 10H ; BAD ECC ON DISK READ

3289 <1> BAD\_TRACK EQU 0BH ; NOT IMPLEMENTED

3290 <1> BAD\_SECTOR EQU 0AH ; BAD SECTOR FLAG DETECTED

3291 <1> ;DMA\_BOUNDARY EQU 09H ; DATA EXTENDS TOO FAR

3292 <1> INIT\_FAIL EQU 07H ; DRIVE PARAMETER ACTIVITY FAILED

3293 <1> BAD\_RESET EQU 05H ; RESET FAILED

3294 <1> ;RECORD\_NOT\_FND EQU 04H ; REQUESTED SECTOR NOT FOUND

3295 <1> ;BAD\_ADDR\_MARK EQU 02H ; ADDRESS MARK NOT FOUND

3296 <1> ;BAD\_CMD EQU 01H ; BAD COMMAND PASSED TO DISK I/O

3297 <1>

3298 <1> ;--------------------------------------------------------

3299 <1> ; :

3300 <1> ; FIXED DISK PARAMETER TABLE :

3301 <1> ; - THE TABLE IS COMPOSED OF A BLOCK DEFINED AS: :

3302 <1> ; :

3303 <1> ; +0 (1 WORD) - MAXIMUM NUMBER OF CYLINDERS :

3304 <1> ; +2 (1 BYTE) - MAXIMUM NUMBER OF HEADS :

3305 <1> ; +3 (1 WORD) - NOT USED/SEE PC-XT :

3306 <1> ; +5 (1 WORD) - STARTING WRITE PRECOMPENSATION CYL :

3307 <1> ; +7 (1 BYTE) - MAXIMUM ECC DATA BURST LENGTH :

3308 <1> ; +8 (1 BYTE) - CONTROL BYTE :

3309 <1> ; BIT 7 DISABLE RETRIES -OR- :

3310 <1> ; BIT 6 DISABLE RETRIES :

3311 <1> ; BIT 3 MORE THAN 8 HEADS :

3312 <1> ; +9 (3 BYTES)- NOT USED/SEE PC-XT :

3313 <1> ; +12 (1 WORD) - LANDING ZONE :

3314 <1> ; +14 (1 BYTE) - NUMBER OF SECTORS/TRACK :

3315 <1> ; +15 (1 BYTE) - RESERVED FOR FUTURE USE :

3316 <1> ; :

3317 <1> ; - TO DYNAMICALLY DEFINE A SET OF PARAMETERS :

3318 <1> ; BUILD A TABLE FOR UP TO 15 TYPES AND PLACE :

3319 <1> ; THE CORRESPONDING VECTOR INTO INTERRUPT 41 :

3320 <1> ; FOR DRIVE 0 AND INTERRUPT 46 FOR DRIVE 1. :

3321 <1> ; :

3322 <1> ;--------------------------------------------------------

3323 <1>

3324 <1> ;--------------------------------------------------------

3325 <1> ; :

3326 <1> ; HARDWARE SPECIFIC VALUES :

3327 <1> ; :

3328 <1> ; - CONTROLLER I/O PORT :

3329 <1> ; :

3330 <1> ; > WHEN READ FROM: :

3331 <1> ; HF\_PORT+0 - READ DATA (FROM CONTROLLER TO CPU) :

3332 <1> ; HF\_PORT+1 - GET ERROR REGISTER :

3333 <1> ; HF\_PORT+2 - GET SECTOR COUNT :

3334 <1> ; HF\_PORT+3 - GET SECTOR NUMBER :

3335 <1> ; HF\_PORT+4 - GET CYLINDER LOW :

3336 <1> ; HF\_PORT+5 - GET CYLINDER HIGH (2 BITS) :

3337 <1> ; HF\_PORT+6 - GET SIZE/DRIVE/HEAD :

3338 <1> ; HF\_PORT+7 - GET STATUS REGISTER :

3339 <1> ; :

3340 <1> ; > WHEN WRITTEN TO: :

3341 <1> ; HF\_PORT+0 - WRITE DATA (FROM CPU TO CONTROLLER) :

3342 <1> ; HF\_PORT+1 - SET PRECOMPENSATION CYLINDER :

3343 <1> ; HF\_PORT+2 - SET SECTOR COUNT :

3344 <1> ; HF\_PORT+3 - SET SECTOR NUMBER :

3345 <1> ; HF\_PORT+4 - SET CYLINDER LOW :

3346 <1> ; HF\_PORT+5 - SET CYLINDER HIGH (2 BITS) :

3347 <1> ; HF\_PORT+6 - SET SIZE/DRIVE/HEAD :

3348 <1> ; HF\_PORT+7 - SET COMMAND REGISTER :

3349 <1> ; :

3350 <1> ;--------------------------------------------------------

3351 <1>

3352 <1> ;HF\_PORT EQU 01F0H ; DISK PORT

3353 <1> ;HF1\_PORT equ 0170h

3354 <1> ;HF\_REG\_PORT EQU 03F6H

3355 <1> ;HF1\_REG\_PORT equ 0376h

3356 <1>

3357 <1> HDC1\_BASEPORT equ 1F0h

3358 <1> HDC2\_BASEPORT equ 170h

3359 <1>

3360 <1> align 2

3361 <1>

3362 <1> ;----- STATUS REGISTER

3363 <1>

3364 <1> ST\_ERROR EQU 00000001B ;

3365 <1> ST\_INDEX EQU 00000010B ;

3366 <1> ST\_CORRCTD EQU 00000100B ; ECC CORRECTION SUCCESSFUL

3367 <1> ST\_DRQ EQU 00001000B ;

3368 <1> ST\_SEEK\_COMPL EQU 00010000B ; SEEK COMPLETE

3369 <1> ST\_WRT\_FLT EQU 00100000B ; WRITE FAULT

3370 <1> ST\_READY EQU 01000000B ;

3371 <1> ST\_BUSY EQU 10000000B ;

3372 <1>

3373 <1> ;----- ERROR REGISTER

3374 <1>

3375 <1> ERR\_DAM EQU 00000001B ; DATA ADDRESS MARK NOT FOUND

3376 <1> ERR\_TRK\_0 EQU 00000010B ; TRACK 0 NOT FOUND ON RECAL

3377 <1> ERR\_ABORT EQU 00000100B ; ABORTED COMMAND

3378 <1> ; EQU 00001000B ; NOT USED

3379 <1> ERR\_ID EQU 00010000B ; ID NOT FOUND

3380 <1> ; EQU 00100000B ; NOT USED

3381 <1> ERR\_DATA\_ECC EQU 01000000B

3382 <1> ERR\_BAD\_BLOCK EQU 10000000B

3383 <1>

3384 <1>

3385 <1> RECAL\_CMD EQU 00010000B ; DRIVE RECAL (10H)

3386 <1> READ\_CMD EQU 00100000B ; READ (20H)

3387 <1> WRITE\_CMD EQU 00110000B ; WRITE (30H)

3388 <1> VERIFY\_CMD EQU 01000000B ; VERIFY (40H)

3389 <1> FMTTRK\_CMD EQU 01010000B ; FORMAT TRACK (50H)

3390 <1> INIT\_CMD EQU 01100000B ; INITIALIZE (60H)

3391 <1> SEEK\_CMD EQU 01110000B ; SEEK (70H)

3392 <1> DIAG\_CMD EQU 10010000B ; DIAGNOSTIC (90H)

3393 <1> SET\_PARM\_CMD EQU 10010001B ; DRIVE PARMS (91H)

3394 <1> NO\_RETRIES EQU 00000001B ; CHD MODIFIER (01H)

3395 <1> ECC\_MODE EQU 00000010B ; CMD MODIFIER (02H)

3396 <1> BUFFER\_MODE EQU 00001000B ; CMD MODIFIER (08H)

3397 <1>

3398 <1> ;MAX\_FILE EQU 2

3399 <1> ;S\_MAX\_FILE EQU 2

3400 <1> MAX\_FILE equ 4 ; 22/12/2014

3401 <1> S\_MAX\_FILE equ 4 ; 22/12/2014

3402 <1>

3403 <1> DELAY\_1 EQU 25H ; DELAY FOR OPERATION COMPLETE

3404 <1> DELAY\_2 EQU 0600H ; DELAY FOR READY

3405 <1> DELAY\_3 EQU 0100H ; DELAY FOR DATA REQUEST

3406 <1>

3407 <1> HF\_FAIL EQU 08H ; CMOS FLAG IN BYTE 0EH

3408 <1>

3409 <1> ;----- COMMAND BLOCK REFERENCE

3410 <1>

3411 <1> ;CMD\_BLOCK EQU BP-8 ; @CMD\_BLOCK REFERENCES BLOCK HEAD IN SS

3412 <1> ; (BP) POINTS TO COMMAND BLOCK TAIL

3413 <1> ; AS DEFINED BY THE "ENTER" PARMS

3414 <1> ; 19/12/2014

3415 <1> ORG\_VECTOR equ 4\*13h ; INT 13h vector

3416 <1> DISK\_VECTOR equ 4\*40h ; INT 40h vector (for floppy disks)

3417 <1> ;HDISK\_INT equ 4\*76h ; Primary HDC - Hardware interrupt (IRQ14)

3418 <1> ;HDISK\_INT1 equ 4\*76h ; Primary HDC - Hardware interrupt (IRQ14)

3419 <1> ;HDISK\_INT2 equ 4\*77h ; Secondary HDC - Hardware interrupt (IRQ15)

3420 <1> ;HF\_TBL\_VEC equ 4\*41h ; Pointer to 1st fixed disk parameter table

3421 <1> ;HF1\_TBL\_VEC equ 4\*46h ; Pointer to 2nd fixed disk parameter table

3422 <1>

3423 <1> align 2

3424 <1>

3425 <1> ;----------------------------------------------------------------

3426 <1> ; FIXED DISK I/O SETUP :

3427 <1> ; :

3428 <1> ; - ESTABLISH TRANSFER VECTORS FOR THE FIXED DISK :

3429 <1> ; - PERFORM POWER ON DIAGNOSTICS :

3430 <1> ; SHOULD AN ERROR OCCUR A "1701" MESSAGE IS DISPLAYED :

3431 <1> ; :

3432 <1> ;----------------------------------------------------------------

3433 <1>

3434 <1> ; 29/05/2016 - TRDOS 386 (TRDOS v2.0)

3435 <1>

3436 <1> DISK\_SETUP:

3437 <1> ;CLI

3438 <1> ;;MOV AX,ABS0 ; GET ABSOLUTE SEGMENT

3439 <1> ;xor ax,ax

3440 <1> ;MOV DS,AX ; SET SEGMENT REGISTER

3441 <1> ;MOV AX, [ORG\_VECTOR] ; GET DISKETTE VECTOR

3442 <1> ;MOV [DISK\_VECTOR],AX ; INTO INT 40H

3443 <1> ;MOV AX, [ORG\_VECTOR+2]

3444 <1> ;MOV [DISK\_VECTOR+2],AX

3445 <1> ;MOV word [ORG\_VECTOR],DISK\_IO ; FIXED DISK HANDLER

3446 <1> ;MOV [ORG\_VECTOR+2],CS

3447 <1> ; 1st controller (primary master, slave) - IRQ 14

3448 <1> ;;MOV word [HDISK\_INT],HD\_INT ; FIXED DISK INTERRUPT

3449 <1> ;mov word [HDISK\_INT1],HD\_INT ;

3450 <1> ;;MOV [HDISK\_INT+2],CS

3451 <1> ;mov [HDISK\_INT1+2],CS

3452 <1> ; 2nd controller (secondary master, slave) - IRQ 15

3453 <1> ;mov word [HDISK\_INT2],HD1\_INT ;

3454 <1> ;mov [HDISK\_INT2+2],CS

3455 <1> ;

3456 <1> ;;MOV word [HF\_TBL\_VEC],HD0\_DPT ; PARM TABLE DRIVE 80

3457 <1> ;;MOV word [HF\_TBL\_VEC+2],DPT\_SEGM

3458 <1> ;;MOV word [HF1\_TBL\_VEC],HD1\_DPT ; PARM TABLE DRIVE 81

3459 <1> ;;MOV word [HF1\_TBL\_VEC+2],DPT\_SEGM

3460 <1> ;push cs

3461 <1> ;pop ds

3462 <1> ;mov word [HDPM\_TBL\_VEC],HD0\_DPT ; PARM TABLE DRIVE 80h

3463 <1> ;mov word [HDPM\_TBL\_VEC+2],DPT\_SEGM

3464 0000420E C705[D8580100]0000- <1> mov dword [HDPM\_TBL\_VEC], (DPT\_SEGM\*16)+HD0\_DPT

3464 00004216 0900 <1>

3465 <1> ;mov word [HDPS\_TBL\_VEC],HD1\_DPT ; PARM TABLE DRIVE 81h

3466 <1> ;mov word [HDPS\_TBL\_VEC+2],DPT\_SEGM

3467 00004218 C705[DC580100]2000- <1> mov dword [HDPS\_TBL\_VEC], (DPT\_SEGM\*16)+HD1\_DPT

3467 00004220 0900 <1>

3468 <1> ;mov word [HDSM\_TBL\_VEC],HD2\_DPT ; PARM TABLE DRIVE 82h

3469 <1> ;mov word [HDSM\_TBL\_VEC+2],DPT\_SEGM

3470 00004222 C705[E0580100]4000- <1> mov dword [HDSM\_TBL\_VEC], (DPT\_SEGM\*16)+HD2\_DPT

3470 0000422A 0900 <1>

3471 <1> ;mov word [HDSS\_TBL\_VEC],HD3\_DPT ; PARM TABLE DRIVE 83h

3472 <1> ;mov word [HDSS\_TBL\_VEC+2],DPT\_SEGM

3473 0000422C C705[E4580100]6000- <1> mov dword [HDSS\_TBL\_VEC], (DPT\_SEGM\*16)+HD3\_DPT

3473 00004234 0900 <1>

3474 <1> ;

3475 <1> ;;IN AL,INTB01 ; TURN ON SECOND INTERRUPT CHIP

3476 <1> ;;;AND AL,0BFH

3477 <1> ;;and al, 3Fh ; enable IRQ 14 and IRQ 15

3478 <1> ;;;JMP $+2

3479 <1> ;;IODELAY

3480 <1> ;;OUT INTB01,AL

3481 <1> ;;IODELAY

3482 <1> ;;IN AL,INTA01 ; LET INTERRUPTS PASS THRU TO

3483 <1> ;;AND AL,0FBH ; SECOND CHIP

3484 <1> ;;;JMP $+2

3485 <1> ;;IODELAY

3486 <1> ;;OUT INTA01,AL

3487 <1> ;

3488 <1> ;STI

3489 <1> ;;PUSH DS ; MOVE ABS0 POINTER TO

3490 <1> ;;POP ES ; EXTRA SEGMENT POINTER

3491 <1> ;;;CALL DDS ; ESTABLISH DATA SEGMENT

3492 <1> ;;MOV byte [DISK\_STATUS1],0 ; RESET THE STATUS INDICATOR

3493 <1> ;;MOV byte [HF\_NUM],0 ; ZERO NUMBER OF FIXED DISKS

3494 <1> ;;MOV byte [CONTROL\_BYTE],0

3495 <1> ;;MOV byte [PORT\_OFF],0 ; ZERO CARD OFFSET

3496 <1> ; 20/12/2014 - private code by Erdogan Tan

3497 <1> ; (out of original PC-AT, PC-XT BIOS code)

3498 <1> ;mov si, hd0\_type

3499 00004236 BE[F85C0000] <1> mov esi, hd0\_type

3500 <1> ;mov cx, 4

3501 0000423B B904000000 <1> mov ecx, 4

3502 <1> hde\_l:

3503 00004240 AC <1> lodsb

3504 00004241 3C80 <1> cmp al, 80h ; 8?h = existing

3505 00004243 7206 <1> jb short \_L4

3506 00004245 FE05[D4580100] <1> inc byte [HF\_NUM] ; + 1 hard (fixed) disk drives

3507 <1> \_L4: ; 26/02/2015

3508 0000424B E2F3 <1> loop hde\_l

3509 <1> ;\_L4: ; 0 <= [HF\_NUM] =< 4

3510 <1> ;L4:

3511 <1> ;

3512 <1> ;; 31/12/2014 - cancel controller diagnostics here

3513 <1> ;;;mov cx, 3 ; 26/12/2014 (Award BIOS 1999)

3514 <1> ;;mov cl, 3

3515 <1> ;;

3516 <1> ;;MOV DL,80H ; CHECK THE CONTROLLER

3517 <1> ;;hdc\_dl:

3518 <1> ;;MOV AH,14H ; USE CONTROLLER DIAGNOSTIC COMMAND

3519 <1> ;;INT 13H ; CALL BIOS WITH DIAGNOSTIC COMMAND

3520 <1> ;;;JC short CTL\_ERRX ; DISPLAY ERROR MESSAGE IF BAD RETURN

3521 <1> ;;;jc short POD\_DONE ;22/12/2014

3522 <1> ;;jnc short hdc\_reset0

3523 <1> ;;loop hdc\_dl

3524 <1> ;;; 27/12/2014

3525 <1> ;;stc

3526 <1> ;;retn

3527 <1> ;

3528 <1> ;;hdc\_reset0:

3529 <1> ; 18/01/2015

3530 0000424D 8A0D[D4580100] <1> mov cl, [HF\_NUM]

3531 00004253 20C9 <1> and cl, cl

3532 00004255 740E <1> jz short POD\_DONE

3533 <1> ;

3534 00004257 B27F <1> mov dl, 7Fh

3535 <1> hdc\_reset1:

3536 00004259 FEC2 <1> inc dl

3537 <1> ;; 31/12/2015

3538 <1> ;;push dx

3539 <1> ;;push cx

3540 <1> ;;push ds

3541 <1> ;;sub ax, ax

3542 <1> ;;mov ds, ax

3543 <1> ;;MOV AX, [TIMER\_LOW] ; GET START TIMER COUNTS

3544 <1> ;;pop ds

3545 <1> ;;MOV BX,AX

3546 <1> ;;ADD AX,6\*182 ; 60 SECONDS\* 18.2

3547 <1> ;;MOV CX,AX

3548 <1> ;;mov word [wait\_count], 0 ; 22/12/2014 (reset wait counter)

3549 <1> ;;

3550 <1> ;; 31/12/2014 - cancel HD\_RESET\_1

3551 <1> ;;CALL HD\_RESET\_1 ; SET UP DRIVE 0, (1,2,3)

3552 <1> ;;pop cx

3553 <1> ;;pop dx

3554 <1> ;;

3555 <1> ; 18/01/2015

3556 0000425B B40D <1> mov ah, 0Dh ; ALTERNATE RESET

3557 <1> ;int 13h

3558 0000425D E8A4FFFFFF <1> call int13h

3559 00004262 E2F5 <1> loop hdc\_reset1

3560 00004264 F8 <1> clc ; 29/05/2016

3561 <1> POD\_DONE:

3562 00004265 C3 <1> RETn

3563 <1>

3564 <1> ;;----- POD\_ERROR

3565 <1>

3566 <1> ;;CTL\_ERRX:

3567 <1> ; ;MOV SI,OFFSET F1782 ; CONTROLLER ERROR

3568 <1> ; ;CALL SET\_FAIL ; DO NOT IPL FROM DISK

3569 <1> ; ;CALL E\_MSG ; DISPLAY ERROR AND SET (BP) ERROR FLAG

3570 <1> ; ;JMP short POD\_DONE

3571 <1>

3572 <1> ;;HD\_RESET\_1:

3573 <1> ;; ;PUSH BX ; SAVE TIMER LIMITS

3574 <1> ;; ;PUSH CX

3575 <1> ;;RES\_1: MOV AH,09H ; SET DRIVE PARAMETERS

3576 <1> ;; INT 13H

3577 <1> ;; JC short RES\_2

3578 <1> ;; MOV AH,11H ; RECALIBRATE DRIVE

3579 <1> ;; INT 13H

3580 <1> ;; JNC short RES\_CK ; DRIVE OK

3581 <1> ;;RES\_2: ;CALL POD\_TCHK ; CHECK TIME OUT

3582 <1> ;; cmp word [wait\_count], 6\*182 ; waiting time (in timer ticks)

3583 <1> ;; ; (30 seconds)

3584 <1> ;; ;cmc

3585 <1> ;; ;JNC short RES\_1

3586 <1> ;; jb short RES\_1

3587 <1> ;;;RES\_FL: ;MOV SI,OFFSET F1781 ; INDICATE DISK 1 FAILURE;

3588 <1> ;; ;TEST DL,1

3589 <1> ;; ;JNZ RES\_E1

3590 <1> ;; ;MOV SI,OFFSET F1780 ; INDICATE DISK 0 FAILURE

3591 <1> ;; ;CALL SET\_FAIL ; DO NOT TRY TO IPL DISK 0

3592 <1> ;; ;JMP SHORT RES\_E1

3593 <1> ;;RES\_ER: ; 22/12/2014

3594 <1> ;;RES\_OK:

3595 <1> ;; ;POP CX ; RESTORE TIMER LIMITS

3596 <1> ;; ;POP BX

3597 <1> ;; RETn

3598 <1> ;;

3599 <1> ;;RES\_RS: MOV AH,00H ; RESET THE DRIVE

3600 <1> ;; INT 13H

3601 <1> ;;RES\_CK: MOV AH,08H ; GET MAX CYLINDER,HEAD,SECTOR

3602 <1> ;; MOV BL,DL ; SAVE DRIVE CODE

3603 <1> ;; INT 13H

3604 <1> ;; JC short RES\_ER

3605 <1> ;; MOV [NEC\_STATUS],CX ; SAVE MAX CYLINDER, SECTOR

3606 <1> ;; MOV DL,BL ; RESTORE DRIVE CODE

3607 <1> ;;RES\_3: MOV AX,0401H ; VERIFY THE LAST SECTOR

3608 <1> ;; INT 13H

3609 <1> ;; JNC short RES\_OK ; VERIFY OK

3610 <1> ;; CMP AH,BAD\_SECTOR ; OK ALSO IF JUST ID READ

3611 <1> ;; JE short RES\_OK

3612 <1> ;; CMP AH,DATA\_CORRECTED

3613 <1> ;; JE short RES\_OK

3614 <1> ;; CMP AH,BAD\_ECC

3615 <1> ;; JE short RES\_OK

3616 <1> ;; ;CALL POD\_TCHK ; CHECK FOR TIME OUT

3617 <1> ;; cmp word [wait\_count], 6\*182 ; waiting time (in timer ticks)

3618 <1> ;; ; (60 seconds)

3619 <1> ;; cmc

3620 <1> ;; JC short RES\_ER ; FAILED

3621 <1> ;; MOV CX,[NEC\_STATUS] ; GET SECTOR ADDRESS, AND CYLINDER

3622 <1> ;; MOV AL,CL ; SEPARATE OUT SECTOR NUMBER

3623 <1> ;; AND AL,3FH

3624 <1> ;; DEC AL ; TRY PREVIOUS ONE

3625 <1> ;; JZ short RES\_RS ; WE'VE TRIED ALL SECTORS ON TRACK

3626 <1> ;; AND CL,0C0H ; KEEP CYLINDER BITS

3627 <1> ;; OR CL,AL ; MERGE SECTOR WITH CYLINDER BITS

3628 <1> ;; MOV [NEC\_STATUS],CX ; SAVE CYLINDER, NEW SECTOR NUMBER

3629 <1> ;; JMP short RES\_3 ; TRY AGAIN

3630 <1> ;;;RES\_ER: MOV SI,OFFSET F1791 ; INDICATE DISK 1 ERROR

3631 <1> ;; ;TEST DL,1

3632 <1> ;; ;JNZ short RES\_E1

3633 <1> ;; ;MOV SI,OFFSET F1790 ; INDICATE DISK 0 ERROR

3634 <1> ;;;RES\_E1:

3635 <1> ;; ;CALL E\_MSG ; DISPLAY ERROR AND SET (BP) ERROR FLAG

3636 <1> ;;;RES\_OK:

3637 <1> ;; ;POP CX ; RESTORE TIMER LIMITS

3638 <1> ;; ;POP BX

3639 <1> ;; ;RETn

3640 <1> ;

3641 <1> ;;SET\_FAIL:

3642 <1> ; ;MOV AX,X\*(CMOS\_DIAG+NMI) ; GET CMOS ERROR BYTE

3643 <1> ; ;CALL CMOS\_READ

3644 <1> ; ;OR AL,HF\_FAIL ; SET DO NOT IPL FROM DISK FLAG

3645 <1> ; ;XCHG AH,AL ; SAVE IT

3646 <1> ; ;CALL CMOS\_WRITE ; PUT IT OUT

3647 <1> ; ;RETn

3648 <1> ;

3649 <1> ;;POD\_TCHK: ; CHECK FOR 30 SECOND TIME OUT

3650 <1> ; ;POP AX ; SAVE RETURN

3651 <1> ; ;POP CX ; GET TIME OUT LIMITS

3652 <1> ; ;POP BX

3653 <1> ; ;PUSH BX ; AND SAVE THEM AGAIN

3654 <1> ; ;PUSH CX

3655 <1> ; ;PUSH AX

3656 <1> ; ;push ds

3657 <1> ; ;xor ax, ax

3658 <1> ; ;mov ds, ax ; RESTORE RETURN

3659 <1> ; ;MOV AX, [TIMER\_LOW] ; AX = CURRENT TIME

3660 <1> ; ; ; BX = START TIME

3661 <1> ; ; ; CX = END TIME

3662 <1> ; ;pop ds

3663 <1> ; ;CMP BX,CX

3664 <1> ; ;JB short TCHK1 ; START < END

3665 <1> ; ;CMP BX,AX

3666 <1> ; ;JB short TCHKG ; END < START < CURRENT

3667 <1> ; ;JMP SHORT TCHK2 ; END, CURRENT < START

3668 <1> ;;TCHK1: CMP AX,BX

3669 <1> ;; JB short TCHKNG ; CURRENT < START < END

3670 <1> ;;TCHK2: CMP AX,CX

3671 <1> ;; JB short TCHKG ; START < CURRENT < END

3672 <1> ;; ; OR CURRENT < END < START

3673 <1> ;;TCHKNG: STC ; CARRY SET INDICATES TIME OUT

3674 <1> ;; RETn

3675 <1> ;;TCHKG: CLC ; INDICATE STILL TIME

3676 <1> ;; RETn

3677 <1> ;;

3678 <1> ;;int\_13h:

3679 <1>

3680 <1> ;----------------------------------------

3681 <1> ; FIXED DISK BIOS ENTRY POINT :

3682 <1> ;----------------------------------------

3683 <1>

3684 <1> ; 15/01/2017

3685 <1> ; 14/01/2017

3686 <1> ; 07/01/2017

3687 <1> ; 02/01/2017

3688 <1> ; 01/06/2016

3689 <1> ; 16/05/2016, 27/05/2016, 28/05/2016, 29/05/2016

3690 <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)

3691 <1> int33h: ; DISK I/O

3692 <1> ; 29/05/2016

3693 00004266 80642408FE <1> and byte [esp+8], 11111110b ; clear carry bit of eflags register

3694 <1> ; 16/05/2016

3695 0000426B 1E <1> push ds

3696 0000426C 53 <1> push ebx ; user's buffer address (virtual)

3697 0000426D 66BB1000 <1> mov bx, KDATA ; System (Kernel's) data segment

3698 00004271 8EDB <1> mov ds, bx

3699 <1>

3700 <1> ;;15/01/2017

3701 <1> ; 14/01/2017

3702 <1> ; 02/01/2017

3703 <1> ;;mov byte [intflg], 33h ; disk io interrupt

3704 <1> ;pop ebx

3705 <1> ;mov [user\_buffer], ebx

3706 <1>

3707 00004273 8F05[C8650100] <1> pop dword [user\_buffer] ; 01/06/2016

3708 <1>

3709 00004279 C605[FE5E0100]00 <1> mov byte [scount], 0 ; sector count for transfer

3710 00004280 80FC03 <1> cmp ah, 03h ; chs write

3711 00004283 7744 <1> ja short int33h\_2

3712 00004285 7407 <1> je short int33h\_0

3713 00004287 80FC02 <1> cmp ah, 02h ; chs read

3714 0000428A 726A <1> jb short int33h\_5

3715 0000428C EB63 <1> jmp short int33h\_4

3716 <1> int33h\_0:

3717 <1> ; transfer user's buffer content to sector buffer

3718 0000428E 51 <1> push ecx

3719 0000428F 0FB6C8 <1> movzx ecx, al

3720 <1> int33h\_1:

3721 00004292 56 <1> push esi

3722 00004293 8B35[C8650100] <1> mov esi, [user\_buffer]

3723 <1> ; esi = user's buffer address (virtual, ebx)

3724 00004299 57 <1> push edi

3725 0000429A 06 <1> push es

3726 0000429B 50 <1> push eax

3727 0000429C 66B81000 <1> mov ax, KDATA

3728 000042A0 8EC0 <1> mov es, ax

3729 000042A2 BF00000700 <1> mov edi, Cluster\_Buffer

3730 000042A7 C1E109 <1> shl ecx, 9 ; \* 512

3731 000042AA E814A50000 <1> call transfer\_from\_user\_buffer

3732 000042AF 58 <1> pop eax

3733 000042B0 07 <1> pop es

3734 000042B1 5F <1> pop edi

3735 000042B2 5E <1> pop esi

3736 000042B3 59 <1> pop ecx

3737 000042B4 7340 <1> jnc short int33h\_5

3738 000042B6 8B1D[C8650100] <1> mov ebx, [user\_buffer] ; 01/06/2016

3739 000042BC 1F <1> pop ds

3740 <1>

3741 <1> ;;15/01/2017

3742 <1> ; 02/01/2017

3743 <1> ;cli

3744 <1> ;;mov byte [ss:intflg], 0 ; 07/01/2017

3745 <1> ;

3746 <1> ; (\*) 29/05/2016

3747 <1> ; (\*) retf 4 ; skip eflags on stack

3748 <1>

3749 <1> ; 29/05/2016 -set carry flag on stack-

3750 <1> ; [esp] = EIP

3751 <1> ; [esp+4] = CS

3752 <1> ; [esp+8] = E-FLAGS

3753 000042BD 804C240801 <1> or byte [esp+8], 1 ; set carry bit of eflags register

3754 <1> ; [esp+12] = ESP (user)

3755 <1> ; [esp+16] = SS (User)

3756 000042C2 B8FF000000 <1> mov eax, 0FFh ; Unknown error !?

3757 <1> ;iretd

3758 000042C7 EB79 <1> jmp short int33h\_7 ; 07/01/2017

3759 <1>

3760 <1> ; (\*) 29/05/2016 - 'ref 4' intruction causes to stack fault

3761 <1> ; (OUTER-PRIVILEGE-LEVEL)

3762 <1> ; INTEL 80386 PROGRAMMER'S REFERENCE MANUAL 1986

3763 <1> ; // RETF instruction:

3764 <1> ;

3765 <1> ; IF OperandMode=32 THEN

3766 <1> ; Load CS:EIP from stack;

3767 <1> ; Set CS RPL to CPL;

3768 <1> ; Increment eSP by 8 plus the immediate offset if it exists;

3769 <1> ; Load SS:eSP from stack;

3770 <1> ; ELSE (\* OperandMode=16 \*)

3771 <1> ; Load CS:IP from stack;

3772 <1> ; Set CS RPL to CPL;

3773 <1> ; Increment eSP by 4 plus the immediate offset if it exists;

3774 <1> ; Load SS:eSP from stack;

3775 <1> ; FI;

3776 <1> ;

3777 <1> ; //

3778 <1>

3779 <1> int33h\_2:

3780 000042C9 80FC05 <1> cmp ah, 05h ; format track

3781 000042CC 770A <1> ja short int33h\_3

3782 000042CE 7226 <1> jb short int33h\_5

3783 000042D0 51 <1> push ecx

3784 000042D1 B901000000 <1> mov ecx, 1

3785 000042D6 EBBA <1> jmp short int33h\_1

3786 <1> int33h\_3:

3787 000042D8 80FC1C <1> cmp ah, 1Ch ; LBA write

3788 000042DB 7719 <1> ja short int33h\_5

3789 000042DD 74AF <1> je short int33h\_0

3790 000042DF 80FC1B <1> cmp ah, 1Bh ; LBA read

3791 000042E2 740D <1> je short int33h\_4

3792 000042E4 80FC08 <1> cmp ah, 08h ; get disk parameters

3793 000042E7 750D <1> jne short int33h\_5

3794 <1> ; 01/06/2016

3795 000042E9 8B1D[C8650100] <1> mov ebx, [user\_buffer] ; user's buffer address

3796 000042EF EB0A <1> jmp short int33h\_6

3797 <1> int33h\_4:

3798 000042F1 A2[FE5E0100] <1> mov byte [scount], al ; <= 128 sectors

3799 <1> int33h\_5:

3800 000042F6 BB00000700 <1> mov ebx, Cluster\_Buffer ; max. 65536 bytes

3801 <1> ; buf. addr: 70000h

3802 <1> ;mov byte [ClusterBuffer\_Valid], 0

3803 <1> int33h\_6:

3804 000042FB 1F <1> pop ds

3805 000042FC 9C <1> pushfd

3806 000042FD 0E <1> push cs

3807 000042FE E84D000000 <1> call DISK\_IO

3808 00004303 2E8B1D[C8650100] <1> mov ebx, [CS:user\_buffer] ; 01/06/2016

3809 0000430A 723D <1> jc short int33h\_9

3810 <1> ;

3811 0000430C 2E803D[FE5E0100]00 <1> cmp byte [CS:scount], 0

3812 00004314 762C <1> jna short int33h\_7

3813 <1> ; transfer sector buffer content to user's buffer

3814 00004316 06 <1> push es

3815 00004317 1E <1> push ds

3816 00004318 50 <1> push eax

3817 00004319 66B81000 <1> mov ax, KDATA

3818 0000431D 8ED8 <1> mov ds, ax

3819 0000431F 8EC0 <1> mov es, ax

3820 00004321 51 <1> push ecx

3821 00004322 56 <1> push esi

3822 00004323 57 <1> push edi

3823 00004324 0FB60D[FE5E0100] <1> movzx ecx, byte [scount]

3824 0000432B C1E109 <1> shl ecx, 9 ; \* 512 bytes

3825 0000432E 89DF <1> mov edi, ebx ; user's buffer address

3826 00004330 BE00000700 <1> mov esi, Cluster\_Buffer

3827 00004335 E83FA40000 <1> call transfer\_to\_user\_buffer

3828 0000433A 5F <1> pop edi

3829 0000433B 5E <1> pop esi

3830 0000433C 59 <1> pop ecx

3831 0000433D 58 <1> pop eax

3832 0000433E 1F <1> pop ds

3833 0000433F 07 <1> pop es

3834 00004340 7202 <1> jc short int33h\_8

3835 <1> int33h\_7:

3836 00004342 FA <1> cli

3837 <1> ;;15/01/2017

3838 <1> ;;mov byte [ss:intflg], 0 ; 07/01/2017

3839 <1> ; cf = 0 ; use eflags which is in stack

3840 00004343 CF <1> iretd

3841 <1> int33h\_8:

3842 00004344 B8FF000000 <1> mov eax, 0FFh ; Unknown error !?

3843 <1> int33h\_9:

3844 <1> ; cf = 1

3845 <1>

3846 <1> ; (\*) 29/05/2016

3847 <1> ; (\*) retf 4 ; skip eflags on stack

3848 <1> ; Note: This 'retf 4' was wrong, -it was causing

3849 <1> ; to stack errors in ring 3-

3850 <1> ; POP sequence of 'retf 4' is as

3851 <1> ; "eip, cs, eflags, esp, ss, +4 bytes"

3852 <1> ; it is not as "eip, cs, +4 bytes, esp, ss" !

3853 <1>

3854 <1> ; 29/05/2016 -set carry flag on stack-

3855 00004349 804C240801 <1> or byte [esp+8], 1 ; set carry bit of eflags register

3856 <1> ;iretd

3857 0000434E EBF2 <1> jmp short int33h\_7 ; 07/01/2017

3858 <1>

3859 <1> ; 09/12/2017

3860 <1> ; 29/05/2016

3861 <1> ; 27/05/2016 - TRDOS 386 (TRDOS v2.0)

3862 <1>

3863 <1> DISK\_IO:

3864 00004350 80FA80 <1> CMP DL,80H ; TEST FOR FIXED DISK DRIVE

3865 <1> ;JAE short A1 ; YES, HANDLE HERE

3866 <1> ;;;INT 40H ; DISKETTE HANDLER

3867 <1> ;;call int40h

3868 00004353 0F8222F0FFFF <1> jb DISKETTE\_IO\_1

3869 <1> ;RET\_2:

3870 <1> ;RETf 2 ; BACK TO CALLER

3871 <1> ; retf 4

3872 <1> A1:

3873 00004359 FB <1> STI ; ENABLE INTERRUPTS

3874 <1> ;; 04/01/2015

3875 <1> ;;OR AH,AH

3876 <1> ;;JNZ short A2

3877 <1> ;;INT 40H ; RESET NEC WHEN AH=0

3878 <1> ;;SUB AH,AH

3879 0000435A 80FA83 <1> CMP DL,(80H + S\_MAX\_FILE - 1)

3880 <1> ;JA short RET\_2

3881 0000435D 7616 <1> jna short \_A0

3882 <1> ; 29/05/2016

3883 0000435F 1E <1> push ds

3884 00004360 6650 <1> push ax

3885 00004362 66B81000 <1> mov ax, KDATA

3886 00004366 8ED8 <1> mov ds, ax

3887 00004368 6658 <1> pop ax

3888 0000436A B4AA <1> mov ah, 0AAh ; Hard disk drive not ready !

3889 <1> ; (Programmer's guide to AMIBIOS, 1992)

3890 0000436C 8825[D3580100] <1> mov byte [DISK\_STATUS1], ah

3891 00004372 1F <1> pop ds

3892 00004373 EB38 <1> jmp short RET\_2

3893 <1> \_A0:

3894 <1> ; 18/01/2015

3895 00004375 08E4 <1> or ah,ah

3896 00004377 743A <1> jz short A4

3897 00004379 80FC0D <1> cmp ah, 0Dh ; Alternate reset

3898 0000437C 7504 <1> jne short A2

3899 0000437E 28E4 <1> sub ah,ah ; Reset

3900 00004380 EB31 <1> jmp short A4

3901 <1> A2:

3902 00004382 80FC08 <1> CMP AH,08H ; GET PARAMETERS IS A SPECIAL CASE

3903 <1> ;JNZ short A3

3904 <1> ;JMP GET\_PARM\_N

3905 00004385 0F8432030000 <1> je GET\_PARM\_N

3906 0000438B 80FC15 <1> A3: CMP AH,15H ; READ DASD TYPE IS ALSO

3907 <1> ;JNZ short A4

3908 <1> ;JMP READ\_DASD\_TYPE

3909 0000438E 0F84DB020000 <1> je READ\_DASD\_TYPE

3910 <1> ; 02/02/2015

3911 00004394 80FC1D <1> cmp ah, 1Dh ;(Temporary for Retro UNIX 386 v1)

3912 <1> ; 12/01/2015

3913 00004397 F5 <1> cmc

3914 00004398 7319 <1> jnc short A4

3915 <1> int33h\_bad\_cmd:

3916 <1> ; 16/05/2016

3917 <1> ; 30/01/2015

3918 <1> ; 29/05/2016

3919 0000439A 1E <1> push ds

3920 0000439B 6650 <1> push ax

3921 0000439D 66B81000 <1> mov ax, KDATA

3922 000043A1 8ED8 <1> mov ds, ax

3923 000043A3 6658 <1> pop ax

3924 000043A5 B401 <1> mov ah, BAD\_CMD

3925 000043A7 8825[D3580100] <1> mov [DISK\_STATUS1], ah ; BAD\_CMD ; COMMAND ERROR

3926 <1> ;jmp short RET\_2

3927 <1> RET\_2:

3928 <1> ; (\*) 29/05/2016

3929 <1> ; (\*) retf 4

3930 000043AD 804C240801 <1> or byte [esp+8], 1 ; set carry bit of eflags register

3931 000043B2 CF <1> iretd

3932 <1> A4: ; SAVE REGISTERS DURING OPERATION

3933 000043B3 C8080000 <1> ENTER 8,0 ; SAVE (BP) AND MAKE ROOM FOR @CMD\_BLOCK

3934 000043B7 53 <1> PUSH eBX ; IN THE STACK, THE COMMAND BLOCK IS:

3935 000043B8 51 <1> PUSH eCX ; @CMD\_BLOCK == BYTE PTR [BP]-8

3936 000043B9 52 <1> PUSH eDX

3937 000043BA 1E <1> PUSH DS

3938 000043BB 06 <1> PUSH ES

3939 000043BC 56 <1> PUSH eSI

3940 000043BD 57 <1> PUSH eDI

3941 <1> ;;04/01/2015

3942 <1> ;;OR AH,AH ; CHECK FOR RESET

3943 <1> ;;JNZ short A5

3944 <1> ;;MOV DL,80H ; FORCE DRIVE 80 FOR RESET

3945 <1> ;;A5:

3946 <1> ;push cs

3947 <1> ;pop ds

3948 <1> ; 21/02/2015

3949 000043BE 6650 <1> push ax

3950 000043C0 66B81000 <1> mov ax, KDATA

3951 000043C4 8ED8 <1> mov ds, ax

3952 000043C6 8EC0 <1> mov es, ax

3953 000043C8 6658 <1> pop ax

3954 000043CA E88D000000 <1> CALL DISK\_IO\_CONT ; PERFORM THE OPERATION

3955 <1> ;;CALL DDS ; ESTABLISH SEGMENT

3956 000043CF 8A25[D3580100] <1> MOV AH,[DISK\_STATUS1] ; GET STATUS FROM OPERATION

3957 <1> ;(\*) CMP AH,1 ; SET THE CARRY FLAG TO INDICATE

3958 <1> ;(\*) CMC ; SUCCESS OR FAILURE

3959 000043D5 5F <1> POP eDI ; RESTORE REGISTERS

3960 000043D6 5E <1> POP eSI

3961 000043D7 07 <1> POP ES

3962 000043D8 1F <1> POP DS

3963 000043D9 5A <1> POP eDX

3964 000043DA 59 <1> POP eCX

3965 000043DB 5B <1> POP eBX

3966 000043DC C9 <1> LEAVE ; ADJUST (SP) AND RESTORE (BP)

3967 <1> ;RETf 2 ; THROW AWAY SAVED FLAGS

3968 <1> ; (\*) 29/05/2016

3969 <1> ; (\*) retf 4

3970 000043DD 80FC01 <1> cmp ah, 1

3971 000043E0 7205 <1> jc short \_A5

3972 000043E2 804C240801 <1> or byte [esp+8], 1 ; set carry bit of eflags register

3973 <1> \_A5:

3974 000043E7 CF <1> iretd

3975 <1>

3976 <1> ; 21/02/2015

3977 <1> ; dw --> dd

3978 <1> D1: ; FUNCTION TRANSFER TABLE

3979 000043E8 [AB450000] <1> dd DISK\_RESET ; 000H

3980 000043EC [22460000] <1> dd RETURN\_STATUS ; 001H

3981 000043F0 [2F460000] <1> dd DISK\_READ ; 002H

3982 000043F4 [38460000] <1> dd DISK\_WRITE ; 003H

3983 000043F8 [41460000] <1> dd DISK\_VERF ; 004H

3984 000043FC [59460000] <1> dd FMT\_TRK ; 005H

3985 00004400 [A1450000] <1> dd BAD\_COMMAND ; 006H FORMAT BAD SECTORS

3986 00004404 [A1450000] <1> dd BAD\_COMMAND ; 007H FORMAT DRIVE

3987 00004408 [A1450000] <1> dd BAD\_COMMAND ; 008H RETURN PARAMETERS

3988 0000440C [44470000] <1> dd INIT\_DRV ; 009H

3989 00004410 [A3470000] <1> dd RD\_LONG ; 00AH

3990 00004414 [AC470000] <1> dd WR\_LONG ; 00BH

3991 00004418 [B5470000] <1> dd DISK\_SEEK ; 00CH

3992 0000441C [AB450000] <1> dd DISK\_RESET ; 00DH

3993 00004420 [A1450000] <1> dd BAD\_COMMAND ; 00EH READ BUFFER

3994 00004424 [A1450000] <1> dd BAD\_COMMAND ; 00FH WRITE BUFFER

3995 00004428 [DD470000] <1> dd TST\_RDY ; 010H

3996 0000442C [01480000] <1> dd HDISK\_RECAL ; 011H

3997 00004430 [A1450000] <1> dd BAD\_COMMAND ; 012H MEMORY DIAGNOSTIC

3998 00004434 [A1450000] <1> dd BAD\_COMMAND ; 013H DRIVE DIAGNOSTIC

3999 00004438 [37480000] <1> dd CTLR\_DIAGNOSTIC ; 014H CONTROLLER DIAGNOSTIC

4000 <1> ; 02/02/2015 (Temporary - Retro UNIX 386 v1 - DISK I/O test)

4001 0000443C [A1450000] <1> dd BAD\_COMMAND ; 015h

4002 00004440 [A1450000] <1> dd BAD\_COMMAND ; 016h

4003 00004444 [A1450000] <1> dd BAD\_COMMAND ; 017h

4004 00004448 [A1450000] <1> dd BAD\_COMMAND ; 018h

4005 0000444C [A1450000] <1> dd BAD\_COMMAND ; 019h

4006 00004450 [A1450000] <1> dd BAD\_COMMAND ; 01Ah

4007 00004454 [2F460000] <1> dd DISK\_READ ; 01Bh ; LBA read

4008 00004458 [38460000] <1> dd DISK\_WRITE ; 01Ch ; LBA write

4009 <1> D1L EQU $ - D1

4010 <1>

4011 <1> DISK\_IO\_CONT:

4012 <1> ;;CALL DDS ; ESTABLISH SEGMENT

4013 0000445C 80FC01 <1> CMP AH,01H ; RETURN STATUS

4014 <1> ;;JNZ short SU0

4015 <1> ;;JMP RETURN\_STATUS

4016 0000445F 0F84BD010000 <1> je RETURN\_STATUS

4017 <1> SU0:

4018 00004465 C605[D3580100]00 <1> MOV byte [DISK\_STATUS1],0 ; RESET THE STATUS INDICATOR

4019 <1> ;;PUSH BX ; SAVE DATA ADDRESS

4020 <1> ;mov si, bx ;; 14/02/2015

4021 0000446C 89DE <1> mov esi, ebx ; 21/02/2015

4022 0000446E 8A1D[D4580100] <1> MOV BL,[HF\_NUM] ; GET NUMBER OF DRIVES

4023 <1> ;; 04/01/2015

4024 <1> ;;PUSH AX

4025 00004474 80E27F <1> AND DL,7FH ; GET DRIVE AS 0 OR 1

4026 <1> ; (get drive number as 0 to 3)

4027 00004477 38D3 <1> CMP BL,DL

4028 <1> ;;JBE BAD\_COMMAND\_POP ; INVALID DRIVE

4029 00004479 0F8622010000 <1> jbe BAD\_COMMAND ;; 14/02/2015

4030 <1> ;

4031 <1> ;;03/01/2015

4032 0000447F 29DB <1> sub ebx, ebx

4033 00004481 88D3 <1> mov bl, dl

4034 <1> ;sub bh, bh

4035 00004483 883D[E8580100] <1> mov [LBAMode], bh ; 0

4036 <1> ;;test byte [bx+hd0\_type], 1 ; LBA ready ?

4037 <1> ;test byte [ebx+hd0\_type], 1

4038 <1> ;jz short su1 ; no

4039 <1> ;inc byte [LBAMode]

4040 <1> ;su1:

4041 <1> ; 21/02/2015 (32 bit modification)

4042 <1> ;04/01/2015

4043 00004489 6650 <1> push ax ; \*\*\*

4044 <1> ;PUSH ES ; \*\*

4045 0000448B 6652 <1> PUSH DX ; \*

4046 0000448D 6650 <1> push ax

4047 0000448F E889060000 <1> CALL GET\_VEC ; GET DISK PARAMETERS

4048 <1> ; 02/02/2015

4049 <1> ;mov ax, [ES:BX+16] ; I/O port base address (1F0h, 170h)

4050 00004494 668B4310 <1> mov ax, [ebx+16]

4051 00004498 66A3[E85C0000] <1> mov [HF\_PORT], ax

4052 <1> ;mov dx, [ES:BX+18] ; control port address (3F6h, 376h)

4053 0000449E 668B5312 <1> mov dx, [ebx+18]

4054 000044A2 668915[EA5C0000] <1> mov [HF\_REG\_PORT], dx

4055 <1> ;mov al, [ES:BX+20] ; head register upper nibble (A0h,B0h,E0h,F0h)

4056 000044A9 8A4314 <1> mov al, [ebx+20]

4057 <1> ; 23/02/2015

4058 000044AC A840 <1> test al, 40h ; LBA bit (bit 6)

4059 000044AE 7406 <1> jz short su1

4060 000044B0 FE05[E8580100] <1> inc byte [LBAMode] ; 1

4061 <1> su1:

4062 000044B6 C0E804 <1> shr al, 4

4063 000044B9 2401 <1> and al, 1

4064 000044BB A2[EC5C0000] <1> mov [hf\_m\_s], al

4065 <1> ;

4066 <1> ; 03/01/2015

4067 <1> ;MOV AL,byte [ES:BX+8] ; GET CONTROL BYTE MODIFIER

4068 000044C0 8A4308 <1> mov al, [ebx+8]

4069 <1> ;MOV DX,[HF\_REG\_PORT] ; Device Control register

4070 000044C3 EE <1> OUT DX,AL ; SET EXTRA HEAD OPTION

4071 <1> ; Control Byte: (= 08h, here)

4072 <1> ; bit 0 - 0

4073 <1> ; bit 1 - nIEN (1 = disable irq)

4074 <1> ; bit 2 - SRST (software RESET)

4075 <1> ; bit 3 - use extra heads (8 to 15)

4076 <1> ; -always set to 1-

4077 <1> ; (bits 3 to 7 are reserved

4078 <1> ; for ATA devices)

4079 000044C4 8A25[D5580100] <1> MOV AH,[CONTROL\_BYTE] ; SET EXTRA HEAD OPTION IN

4080 000044CA 80E4C0 <1> AND AH,0C0H ; CONTROL BYTE

4081 000044CD 08C4 <1> OR AH,AL

4082 000044CF 8825[D5580100] <1> MOV [CONTROL\_BYTE],AH

4083 <1> ; 04/01/2015

4084 000044D5 6658 <1> pop ax

4085 000044D7 665A <1> pop dx ; \* ;; 14/02/2015

4086 000044D9 20E4 <1> and ah, ah ; Reset function ?

4087 000044DB 7507 <1> jnz short su2

4088 <1> ;;pop dx ; \* ;; 14/02/2015

4089 <1> ;pop es ; \*\*

4090 000044DD 6658 <1> pop ax ; \*\*\*

4091 <1> ;;pop bx

4092 000044DF E9C7000000 <1> jmp DISK\_RESET

4093 <1> su2:

4094 000044E4 803D[E8580100]00 <1> cmp byte [LBAMode], 0

4095 000044EB 7662 <1> jna short su3

4096 <1> ;

4097 <1> ; 02/02/2015 (LBA read/write function calls)

4098 000044ED 80FC1B <1> cmp ah, 1Bh

4099 000044F0 720B <1> jb short lbarw1

4100 000044F2 80FC1C <1> cmp ah, 1Ch

4101 000044F5 775D <1> ja short invldfnc

4102 <1> ;;pop dx ; \* ; 14/02/2015

4103 <1> ;mov ax, cx ; Lower word of LBA address (bits 0-15)

4104 000044F7 89C8 <1> mov eax, ecx ; LBA address (21/02/2015)

4105 <1> ;; 14/02/2015

4106 000044F9 88D1 <1> mov cl, dl ; 14/02/2015

4107 <1> ;;mov dx, bx

4108 <1> ;mov dx, si ; higher word of LBA address (bits 16-23)

4109 <1> ;;mov bx, di

4110 <1> ;mov si, di ; Buffer offset

4111 000044FB EB32 <1> jmp short lbarw2

4112 <1> lbarw1:

4113 <1> ; convert CHS to LBA

4114 <1> ;

4115 <1> ; LBA calculation - AWARD BIOS - 1999 - AHDSK.ASM

4116 <1> ; LBA = "# of Heads" \* Sectors/Track \* Cylinder + Head \* Sectors/Track

4117 <1> ; + Sector - 1

4118 000044FD 6652 <1> push dx ; \* ;; 14/02/2015

4119 <1> ;xor dh, dh

4120 000044FF 31D2 <1> xor edx, edx

4121 <1> ;mov dl, [ES:BX+14] ; sectors per track (logical)

4122 00004501 8A530E <1> mov dl, [ebx+14]

4123 <1> ;xor ah, ah

4124 00004504 31C0 <1> xor eax, eax

4125 <1> ;mov al, [ES:BX+2] ; heads (logical)

4126 00004506 8A4302 <1> mov al, [ebx+2]

4127 00004509 FEC8 <1> dec al

4128 0000450B 6640 <1> inc ax ; 0 = 256

4129 0000450D 66F7E2 <1> mul dx

4130 <1> ; AX = # of Heads" \* Sectors/Track

4131 00004510 6689CA <1> mov dx, cx

4132 <1> ;and cx, 3Fh ; sector (1 to 63)

4133 00004513 83E13F <1> and ecx, 3fh

4134 00004516 86D6 <1> xchg dl, dh

4135 00004518 C0EE06 <1> shr dh, 6

4136 <1> ; DX = cylinder (0 to 1023)

4137 <1> ;mul dx

4138 <1> ; DX:AX = # of Heads" \* Sectors/Track \* Cylinder

4139 0000451B F7E2 <1> mul edx

4140 0000451D FEC9 <1> dec cl ; sector - 1

4141 <1> ;add ax, cx

4142 <1> ;adc dx, 0

4143 <1> ; DX:AX = # of Heads" \* Sectors/Track \* Cylinder + Sector -1

4144 0000451F 01C8 <1> add eax, ecx

4145 00004521 6659 <1> pop cx ; \* ; ch = head, cl = drive number (zero based)

4146 <1> ;push dx

4147 <1> ;push ax

4148 00004523 50 <1> push eax

4149 <1> ;mov al, [ES:BX+14] ; sectors per track (logical)

4150 00004524 8A430E <1> mov al, [ebx+14]

4151 00004527 F6E5 <1> mul ch

4152 <1> ; AX = Head \* Sectors/Track

4153 00004529 0FB7C0 <1> movzx eax, ax ; 09/12/2017

4154 <1> ;pop dx

4155 0000452C 5A <1> pop edx

4156 <1> ;add ax, dx

4157 <1> ;pop dx

4158 <1> ;adc dx, 0 ; add carry bit

4159 0000452D 01D0 <1> add eax, edx

4160 <1> lbarw2:

4161 0000452F 29D2 <1> sub edx, edx ; 21/02/2015

4162 00004531 88CA <1> mov dl, cl ; 21/02/2015

4163 00004533 C645F800 <1> mov byte [CMD\_BLOCK], 0 ; Features Register

4164 <1> ; NOTE: Features register (1F1h, 171h)

4165 <1> ; is not used for ATA device R/W functions.

4166 <1> ; It is old/obsolete 'write precompensation'

4167 <1> ; register and error register

4168 <1> ; for old ATA/IDE devices.

4169 <1> ; 18/01/2014

4170 <1> ;mov ch, [hf\_m\_s] ; Drive 0 (master) or 1 (slave)

4171 00004537 8A0D[EC5C0000] <1> mov cl, [hf\_m\_s]

4172 <1> ;shl ch, 4 ; bit 4 (drive bit)

4173 <1> ;or ch, 0E0h ; bit 5 = 1

4174 <1> ; bit 6 = 1 = LBA mode

4175 <1> ; bit 7 = 1

4176 0000453D 80C90E <1> or cl, 0Eh ; 1110b

4177 <1> ;and dh, 0Fh ; LBA byte 4 (bits 24 to 27)

4178 00004540 25FFFFFF0F <1> and eax, 0FFFFFFFh

4179 00004545 C1E11C <1> shl ecx, 28 ; 21/02/2015

4180 <1> ;or dh, ch

4181 00004548 09C8 <1> or eax, ecx

4182 <1> ;;mov [CMD\_BLOCK+2], al ; LBA byte 1 (bits 0 to 7)

4183 <1> ; (Sector Number Register)

4184 <1> ;;mov [CMD\_BLOCK+3], ah ; LBA byte 2 (bits 8 to 15)

4185 <1> ; (Cylinder Low Register)

4186 <1> ;mov [CMD\_BLOCK+2], ax ; LBA byte 1, 2

4187 <1> ;mov [CMD\_BLOCK+4], dl ; LBA byte 3 (bits 16 to 23)

4188 <1> ; (Cylinder High Register)

4189 <1> ;;mov [CMD\_BLOCK+5], dh ; LBA byte 4 (bits 24 to 27)

4190 <1> ; (Drive/Head Register)

4191 <1>

4192 <1> ;mov [CMD\_BLOCK+4], dx ; LBA byte 4, LBA & DEV select bits

4193 0000454A 8945FA <1> mov [CMD\_BLOCK+2], eax ; 21/02/2015

4194 <1> ;14/02/2015

4195 <1> ;mov dl, cl ; Drive number (INIT\_DRV)

4196 0000454D EB38 <1> jmp short su4

4197 <1> su3:

4198 <1> ; 02/02/2015

4199 <1> ; (Temporary functions 1Bh & 1Ch are not valid for CHS mode)

4200 0000454F 80FC14 <1> cmp ah, 14h

4201 00004552 7604 <1> jna short chsfnc

4202 <1> invldfnc:

4203 <1> ; 14/02/2015

4204 <1> ;pop es ; \*\*

4205 00004554 6658 <1> pop ax ; \*\*\*

4206 <1> ;jmp short BAD\_COMMAND\_POP

4207 00004556 EB49 <1> jmp short BAD\_COMMAND

4208 <1> chsfnc:

4209 <1> ;MOV AX,[ES:BX+5] ; GET WRITE PRE-COMPENSATION CYLINDER

4210 00004558 668B4305 <1> mov ax, [ebx+5]

4211 0000455C 66C1E802 <1> SHR AX,2

4212 00004560 8845F8 <1> MOV [CMD\_BLOCK],AL

4213 <1> ;;MOV AL,[ES:BX+8] ; GET CONTROL BYTE MODIFIER

4214 <1> ;;PUSH DX

4215 <1> ;;MOV DX,[HF\_REG\_PORT]

4216 <1> ;;OUT DX,AL ; SET EXTRA HEAD OPTION

4217 <1> ;;POP DX ; \*

4218 <1> ;;POP ES ; \*\*

4219 <1> ;;MOV AH,[CONTROL\_BYTE] ; SET EXTRA HEAD OPTION IN

4220 <1> ;;AND AH,0C0H ; CONTROL BYTE

4221 <1> ;;OR AH,AL

4222 <1> ;;MOV [CONTROL\_BYTE],AH

4223 <1> ;

4224 00004563 88C8 <1> MOV AL,CL ; GET SECTOR NUMBER

4225 00004565 243F <1> AND AL,3FH

4226 00004567 8845FA <1> MOV [CMD\_BLOCK+2],AL

4227 0000456A 886DFB <1> MOV [CMD\_BLOCK+3],CH ; GET CYLINDER NUMBER

4228 0000456D 88C8 <1> MOV AL,CL

4229 0000456F C0E806 <1> SHR AL,6

4230 00004572 8845FC <1> MOV [CMD\_BLOCK+4],AL ; CYLINDER HIGH ORDER 2 BITS

4231 <1> ;;05/01/2015

4232 <1> ;;MOV AL,DL ; DRIVE NUMBER

4233 00004575 A0[EC5C0000] <1> mov al, [hf\_m\_s]

4234 0000457A C0E004 <1> SHL AL,4

4235 0000457D 80E60F <1> AND DH,0FH ; HEAD NUMBER

4236 00004580 08F0 <1> OR AL,DH

4237 <1> ;OR AL,80H or 20H

4238 00004582 0CA0 <1> OR AL,80h+20h ; ECC AND 512 BYTE SECTORS

4239 00004584 8845FD <1> MOV [CMD\_BLOCK+5],AL ; ECC/SIZE/DRIVE/HEAD

4240 <1> su4:

4241 <1> ;POP ES ; \*\*

4242 <1> ;; 14/02/2015

4243 <1> ;;POP AX

4244 <1> ;;MOV [CMD\_BLOCK+1],AL ; SECTOR COUNT

4245 <1> ;;PUSH AX

4246 <1> ;;MOV AL,AH ; GET INTO LOW BYTE

4247 <1> ;;XOR AH,AH ; ZERO HIGH BYTE

4248 <1> ;;SAL AX,1 ; \*2 FOR TABLE LOOKUP

4249 00004587 6658 <1> pop ax ; \*\*\*

4250 00004589 8845F9 <1> mov [CMD\_BLOCK+1], al

4251 0000458C 29DB <1> sub ebx, ebx

4252 0000458E 88E3 <1> mov bl, ah

4253 <1> ;xor bh, bh

4254 <1> ;sal bx, 1

4255 00004590 66C1E302 <1> sal bx, 2 ; 32 bit offset (21/02/2015)

4256 <1> ;;MOV SI,AX ; PUT INTO SI FOR BRANCH

4257 <1> ;;CMP AX,D1L ; TEST WITHIN RANGE

4258 <1> ;;JNB short BAD\_COMMAND\_POP

4259 <1> ;cmp bx, D1L

4260 00004594 83FB74 <1> cmp ebx, D1L

4261 00004597 7308 <1> jnb short BAD\_COMMAND

4262 <1> ;xchg bx, si

4263 00004599 87DE <1> xchg ebx, esi

4264 <1> ;;;POP AX ; RESTORE AX

4265 <1> ;;;POP BX ; AND DATA ADDRESS

4266 <1>

4267 <1> ;;PUSH CX

4268 <1> ;;PUSH AX ; ADJUST ES:BX

4269 <1> ;MOV CX,BX ; GET 3 HIGH ORDER NIBBLES OF BX

4270 <1> ;SHR CX,4

4271 <1> ;MOV AX,ES

4272 <1> ;ADD AX,CX

4273 <1> ;MOV ES,AX

4274 <1> ;AND BX,000FH ; ES:BX CHANGED TO ES:000X

4275 <1> ;;POP AX

4276 <1> ;;POP CX

4277 <1> ;;JMP word [CS:SI+D1]

4278 <1> ;jmp word [SI+D1]

4279 0000459B FFA6[E8430000] <1> jmp dword [esi+D1]

4280 <1> ;;BAD\_COMMAND\_POP:

4281 <1> ;; POP AX

4282 <1> ;; POP BX

4283 <1> BAD\_COMMAND:

4284 000045A1 C605[D3580100]01 <1> MOV byte [DISK\_STATUS1],BAD\_CMD ; COMMAND ERROR

4285 000045A8 B000 <1> MOV AL,0

4286 000045AA C3 <1> RETn

4287 <1>

4288 <1> ;----------------------------------------

4289 <1> ; RESET THE DISK SYSTEM (AH=00H) :

4290 <1> ;----------------------------------------

4291 <1>

4292 <1> ; 18-1-2015 : one controller reset (not other one)

4293 <1>

4294 <1> DISK\_RESET:

4295 000045AB FA <1> CLI

4296 000045AC E4A1 <1> IN AL,INTB01 ; GET THE MASK REGISTER

4297 <1> ;JMP $+2

4298 <1> IODELAY

4298 000045AE EB00 <2> jmp short $+2

4298 000045B0 EB00 <2> jmp short $+2

4299 <1> ;AND AL,0BFH ; ENABLE FIXED DISK INTERRUPT

4300 000045B2 243F <1> and al,3Fh ; 22/12/2014 (IRQ 14 & IRQ 15)

4301 000045B4 E6A1 <1> OUT INTB01,AL

4302 000045B6 FB <1> STI ; START INTERRUPTS

4303 <1> ; 14/02/2015

4304 000045B7 6689D7 <1> mov di, dx

4305 <1> ; 04/01/2015

4306 <1> ;xor di,di

4307 <1> drst0:

4308 000045BA B004 <1> MOV AL,04H ; bit 2 - SRST

4309 <1> ;MOV DX,HF\_REG\_PORT

4310 000045BC 668B15[EA5C0000] <1> MOV DX,[HF\_REG\_PORT]

4311 000045C3 EE <1> OUT DX,AL ; RESET

4312 <1> ; MOV CX,10 ; DELAY COUNT

4313 <1> ;DRD: DEC CX

4314 <1> ; JNZ short DRD ; WAIT 4.8 MICRO-SEC

4315 <1> ;mov cx,2 ; wait for 30 micro seconds

4316 000045C4 B902000000 <1> mov ecx, 2 ; 21/02/2015

4317 000045C9 E81FD8FFFF <1> call WAITF ; (Award Bios 1999 - WAIT\_REFRESH,

4318 <1> ; 40 micro seconds)

4319 000045CE A0[D5580100] <1> mov al,[CONTROL\_BYTE]

4320 000045D3 240F <1> AND AL,0FH ; SET HEAD OPTION

4321 000045D5 EE <1> OUT DX,AL ; TURN RESET OFF

4322 000045D6 E838040000 <1> CALL NOT\_BUSY

4323 000045DB 7515 <1> JNZ short DRERR ; TIME OUT ON RESET

4324 000045DD 668B15[E85C0000] <1> MOV DX,[HF\_PORT]

4325 000045E4 FEC2 <1> inc dl ; HF\_PORT+1

4326 <1> ; 02/01/2015 - Award BIOS 1999 - AHDSK.ASM

4327 <1> ;mov cl, 10

4328 000045E6 B90A000000 <1> mov ecx, 10 ; 21/02/2015

4329 <1> drst1:

4330 000045EB EC <1> IN AL,DX ; GET RESET STATUS

4331 000045EC 3C01 <1> CMP AL,1

4332 <1> ; 04/01/2015

4333 000045EE 740A <1> jz short drst2

4334 <1> ;JNZ short DRERR ; BAD RESET STATUS

4335 <1> ; Drive/Head Register - bit 4

4336 000045F0 E2F9 <1> loop drst1

4337 <1> DRERR:

4338 000045F2 C605[D3580100]05 <1> MOV byte [DISK\_STATUS1],BAD\_RESET ; CARD FAILED

4339 000045F9 C3 <1> RETn

4340 <1> drst2:

4341 <1> ; 14/02/2015

4342 000045FA 6689FA <1> mov dx,di

4343 <1> ;drst3:

4344 <1> ; ; 05/01/2015

4345 <1> ; shl di,1

4346 <1> ; ; 04/01/2015

4347 <1> ; mov ax,[di+hd\_cports]

4348 <1> ; cmp ax,[HF\_REG\_PORT]

4349 <1> ; je short drst4

4350 <1> ; mov [HF\_REG\_PORT], ax

4351 <1> ; ; 03/01/2015

4352 <1> ; mov ax,[di+hd\_ports]

4353 <1> ; mov [HF\_PORT], ax

4354 <1> ; ; 05/01/2014

4355 <1> ; shr di,1

4356 <1> ; ; 04/01/2015

4357 <1> ; jmp short drst0 ; reset other controller

4358 <1> ;drst4:

4359 <1> ; ; 05/01/2015

4360 <1> ; shr di,1

4361 <1> ; mov al,[di+hd\_dregs]

4362 <1> ; and al,10h ; bit 4 only

4363 <1> ; shr al,4 ; bit 4 -> bit 0

4364 <1> ; mov [hf\_m\_s], al ; (0 = master, 1 = slave)

4365 <1> ;

4366 000045FD A0[EC5C0000] <1> mov al, [hf\_m\_s] ; 18/01/2015

4367 00004602 A801 <1> test al,1

4368 <1> ; jnz short drst6

4369 00004604 7516 <1> jnz short drst4

4370 00004606 8065FDEF <1> AND byte [CMD\_BLOCK+5],0EFH ; SET TO DRIVE 0

4371 <1> ;drst5:

4372 <1> drst3:

4373 0000460A E835010000 <1> CALL INIT\_DRV ; SET MAX HEADS

4374 <1> ;mov dx,di

4375 0000460F E8ED010000 <1> CALL HDISK\_RECAL ; RECAL TO RESET SEEK SPEED

4376 <1> ; 04/01/2014

4377 <1> ; inc di

4378 <1> ; mov dx,di

4379 <1> ; cmp dl,[HF\_NUM]

4380 <1> ; jb short drst3

4381 <1> ;DRE:

4382 00004614 C605[D3580100]00 <1> MOV byte [DISK\_STATUS1],0 ; IGNORE ANY SET UP ERRORS

4383 0000461B C3 <1> RETn

4384 <1> ;drst6:

4385 <1> drst4: ; Drive/Head Register - bit 4

4386 0000461C 804DFD10 <1> OR byte [CMD\_BLOCK+5],010H ; SET TO DRIVE 1

4387 <1> ;jmp short drst5

4388 00004620 EBE8 <1> jmp short drst3

4389 <1>

4390 <1> ;----------------------------------------

4391 <1> ; DISK STATUS ROUTINE (AH = 01H) :

4392 <1> ;----------------------------------------

4393 <1>

4394 <1> RETURN\_STATUS:

4395 00004622 A0[D3580100] <1> MOV AL,[DISK\_STATUS1] ; OBTAIN PREVIOUS STATUS

4396 00004627 C605[D3580100]00 <1> MOV byte [DISK\_STATUS1],0 ; RESET STATUS

4397 0000462E C3 <1> RETn

4398 <1>

4399 <1> ;----------------------------------------

4400 <1> ; DISK READ ROUTINE (AH = 02H) :

4401 <1> ;----------------------------------------

4402 <1>

4403 <1> DISK\_READ:

4404 0000462F C645FE20 <1> MOV byte [CMD\_BLOCK+6],READ\_CMD

4405 00004633 E954020000 <1> JMP COMMANDI

4406 <1>

4407 <1> ;----------------------------------------

4408 <1> ; DISK WRITE ROUTINE (AH = 03H) :

4409 <1> ;----------------------------------------

4410 <1>

4411 <1> DISK\_WRITE:

4412 00004638 C645FE30 <1> MOV byte [CMD\_BLOCK+6],WRITE\_CMD

4413 0000463C E9A6020000 <1> JMP COMMANDO

4414 <1>

4415 <1> ;----------------------------------------

4416 <1> ; DISK VERIFY (AH = 04H) :

4417 <1> ;----------------------------------------

4418 <1>

4419 <1> DISK\_VERF:

4420 00004641 C645FE40 <1> MOV byte [CMD\_BLOCK+6],VERIFY\_CMD

4421 00004645 E814030000 <1> CALL COMMAND

4422 0000464A 750C <1> JNZ short VERF\_EXIT ; CONTROLLER STILL BUSY

4423 0000464C E886030000 <1> CALL \_WAIT ; (Original: CALL WAIT)

4424 00004651 7505 <1> JNZ short VERF\_EXIT ; TIME OUT

4425 00004653 E813040000 <1> CALL CHECK\_STATUS

4426 <1> VERF\_EXIT:

4427 00004658 C3 <1> RETn

4428 <1>

4429 <1> ;----------------------------------------

4430 <1> ; FORMATTING (AH = 05H) :

4431 <1> ;----------------------------------------

4432 <1>

4433 <1> FMT\_TRK: ; FORMAT TRACK (AH = 005H)

4434 00004659 C645FE50 <1> MOV byte [CMD\_BLOCK+6],FMTTRK\_CMD

4435 <1> ;PUSH ES

4436 <1> ;PUSH BX

4437 0000465D 53 <1> push ebx

4438 0000465E E8BA040000 <1> CALL GET\_VEC ; GET DISK PARAMETERS ADDRESS

4439 <1> ;MOV AL,[ES:BX+14] ; GET SECTORS/TRACK

4440 00004663 8A430E <1> mov al, [ebx+14]

4441 00004666 8845F9 <1> MOV [CMD\_BLOCK+1],AL ; SET SECTOR COUNT IN COMMAND

4442 00004669 5B <1> pop ebx

4443 <1> ;POP BX

4444 <1> ;POP ES

4445 0000466A E97F020000 <1> JMP CMD\_OF ; GO EXECUTE THE COMMAND

4446 <1>

4447 <1> ;----------------------------------------

4448 <1> ; READ DASD TYPE (AH = 15H) :

4449 <1> ;----------------------------------------

4450 <1>

4451 <1> READ\_DASD\_TYPE:

4452 <1> READ\_D\_T: ; GET DRIVE PARAMETERS

4453 0000466F 1E <1> PUSH DS ; SAVE REGISTERS

4454 <1> ;PUSH ES

4455 00004670 53 <1> PUSH eBX

4456 <1> ;CALL DDS ; ESTABLISH ADDRESSING

4457 <1> ;push cs

4458 <1> ;pop ds

4459 00004671 66BB1000 <1> mov bx, KDATA

4460 00004675 8EDB <1> mov ds, bx

4461 <1> ;mov es, bx

4462 00004677 C605[D3580100]00 <1> MOV byte [DISK\_STATUS1],0

4463 0000467E 8A1D[D4580100] <1> MOV BL,[HF\_NUM] ; GET NUMBER OF DRIVES

4464 00004684 80E27F <1> AND DL,7FH ; GET DRIVE NUMBER

4465 00004687 38D3 <1> CMP BL,DL

4466 00004689 7627 <1> JBE short RDT\_NOT\_PRESENT ; RETURN DRIVE NOT PRESENT

4467 0000468B E88D040000 <1> CALL GET\_VEC ; GET DISK PARAMETER ADDRESS

4468 <1> ;MOV AL,[ES:BX+2] ; HEADS

4469 00004690 8A4302 <1> mov al, [ebx+2]

4470 <1> ;MOV CL,[ES:BX+14]

4471 00004693 8A4B0E <1> mov cl, [ebx+14]

4472 00004696 F6E9 <1> IMUL CL ; \* NUMBER OF SECTORS

4473 <1> ;MOV CX,[ES:BX] ; MAX NUMBER OF CYLINDERS

4474 00004698 668B0B <1> mov cx ,[ebx]

4475 <1> ;

4476 <1> ; 02/01/2015

4477 <1> ; \*\* leave the last cylinder as reserved for diagnostics \*\*

4478 <1> ; (Also in Award BIOS - 1999, AHDSK.ASM, FUN15 -> sub ax, 1)

4479 0000469B 6649 <1> DEC CX ; LEAVE ONE FOR DIAGNOSTICS

4480 <1> ;

4481 0000469D 66F7E9 <1> IMUL CX ; NUMBER OF SECTORS

4482 000046A0 6689D1 <1> MOV CX,DX ; HIGH ORDER HALF

4483 000046A3 6689C2 <1> MOV DX,AX ; LOW ORDER HALF

4484 <1> ;SUB AX,AX

4485 000046A6 28C0 <1> sub al, al

4486 000046A8 B403 <1> MOV AH,03H ; INDICATE FIXED DISK

4487 000046AA 5B <1> RDT2: POP eBX ; RESTORE REGISTERS

4488 <1> ;POP ES

4489 000046AB 1F <1> POP DS

4490 <1> ; (\*) CLC ; CLEAR CARRY

4491 <1> ;RETf 2

4492 <1> ; (\*) 29/05/2016

4493 <1> ; (\*) retf 4

4494 000046AC 80642408FE <1> and byte [esp+8], 0FEh ; clear carry bit of eflags register

4495 000046B1 CF <1> iretd

4496 <1>

4497 <1> RDT\_NOT\_PRESENT:

4498 000046B2 6629C0 <1> SUB AX,AX ; DRIVE NOT PRESENT RETURN

4499 000046B5 6689C1 <1> MOV CX,AX ; ZERO BLOCK COUNT

4500 000046B8 6689C2 <1> MOV DX,AX

4501 000046BB EBED <1> JMP short RDT2

4502 <1>

4503 <1> ; 28/05/2016

4504 <1> ; 27/05/2016 - TRDOS 386 (TRDOS v2.0)

4505 <1>

4506 <1> ;----------------------------------------

4507 <1> ; GET PARAMETERS (AH = 08H) :

4508 <1> ;----------------------------------------

4509 <1>

4510 <1> GET\_PARM\_N:

4511 <1> ; ebx = user's buffer address for parameters table

4512 <1> ;GET\_PARM: ; GET DRIVE PARAMETERS

4513 000046BD 1E <1> PUSH DS ; SAVE REGISTERS

4514 000046BE 06 <1> PUSH ES

4515 000046BF 53 <1> PUSH eBX

4516 <1> ;MOV AX,ABS0 ; ESTABLISH ADDRESSING

4517 <1> ;MOV DS,AX

4518 <1> ;TEST DL,1 ; CHECK FOR DRIVE 1

4519 <1> ;JZ short G0

4520 <1> ;LES BX,@HF1\_TBL\_VEC

4521 <1> ;JMP SHORT G1

4522 <1> ;G0: LES BX,@HF\_TBL\_VEC

4523 <1> ;G1:

4524 <1> ;CALL DDS ; ESTABLISH SEGMENT

4525 <1> ; 22/12/2014

4526 <1> ;push cs

4527 <1> ;pop ds

4528 000046C0 66BB1000 <1> mov bx, KDATA

4529 000046C4 8EDB <1> mov ds, bx

4530 000046C6 8EC3 <1> mov es, bx ; 27/05/2016

4531 <1> ;

4532 000046C8 80EA80 <1> SUB DL,80H

4533 000046CB 80FA04 <1> CMP DL,MAX\_FILE ; TEST WITHIN RANGE

4534 000046CE 7361 <1> JAE short G4

4535 <1> ;

4536 000046D0 31DB <1> xor ebx, ebx ; 21/02/2015

4537 <1> ; 22/12/2014

4538 000046D2 88D3 <1> mov bl, dl

4539 <1> ;xor bh, bh

4540 000046D4 C0E302 <1> shl bl, 2 ; convert index to offset

4541 <1> ;add bx, HF\_TBL\_VEC

4542 000046D7 81C3[D8580100] <1> add ebx, HF\_TBL\_VEC

4543 <1> ;mov ax, [bx+2]

4544 <1> ;mov es, ax ; dpt segment

4545 <1> ;mov bx, [bx] ; dpt offset

4546 000046DD 8B1B <1> mov ebx, [ebx] ; 32 bit offset

4547 <1>

4548 000046DF C605[D3580100]00 <1> MOV byte [DISK\_STATUS1],0

4549 <1> ;MOV AX,[ES:BX] ; MAX NUMBER OF CYLINDERS

4550 000046E6 668B03 <1> mov ax, [ebx]

4551 <1> ;;SUB AX,2 ; ADJUST FOR 0-N

4552 000046E9 6648 <1> dec ax ; max. cylinder number

4553 000046EB 88C5 <1> MOV CH,AL

4554 000046ED 66250003 <1> AND AX,0300H ; HIGH TWO BITS OF CYLINDER

4555 000046F1 66D1E8 <1> SHR AX,1

4556 000046F4 66D1E8 <1> SHR AX,1

4557 <1> ;OR AL,[ES:BX+14] ; SECTORS

4558 000046F7 0A430E <1> or al, [ebx+14]

4559 000046FA 88C1 <1> MOV CL,AL

4560 <1> ;MOV DH,[ES:BX+2] ; HEADS

4561 000046FC 8A7302 <1> mov dh, [ebx+2]

4562 000046FF FECE <1> DEC DH ; 0-N RANGE

4563 00004701 8A15[D4580100] <1> MOV DL,[HF\_NUM] ; DRIVE COUNT

4564 00004707 6629C0 <1> SUB AX,AX

4565 <1> ;27/12/2014

4566 <1> ;mov di, bx ; HDPT offset

4567 <1>

4568 <1> ; 27/05/2016

4569 <1> ; return fixed disk parameters table to user

4570 <1> ; in user's buffer, which is pointed by EBX

4571 <1> ;

4572 0000470A 873C24 <1> xchg edi, [esp] ; ebx (input)-> edi, edi -> [esp]

4573 0000470D 56 <1> push esi

4574 0000470E 89DE <1> mov esi, ebx ; hard disk parameter table (32 bytes)

4575 00004710 89FB <1> mov ebx, edi ; ebx = user's buffer address

4576 00004712 51 <1> push ecx

4577 00004713 50 <1> push eax

4578 00004714 B920000000 <1> mov ecx, 32 ; 32 bytes

4579 00004719 E85BA00000 <1> call transfer\_to\_user\_buffer ; trdosk6.s (16/05/2016)

4580 0000471E 58 <1> pop eax

4581 0000471F 59 <1> pop ecx

4582 00004720 5E <1> pop esi

4583 00004721 5F <1> pop edi

4584 00004722 730A <1> jnc short G5

4585 <1> ; 29/05/2016 (\*)

4586 00004724 B8FF000000 <1> mov eax, 0FFh ; unknown error !

4587 <1> \_G6:

4588 00004729 804C241001 <1> or byte [esp+16], 1 ; set carry bit of eflags register

4589 <1> G5:

4590 <1> ; 27/05/2016

4591 <1> ;POP eBX ; RESTORE REGISTERS

4592 0000472E 07 <1> POP ES

4593 0000472F 1F <1> POP DS

4594 <1> ;RETf 2

4595 <1> ; (\*) 29/05/2016

4596 <1> ; (\*) retf 4

4597 <1> ; (\*) or byte [esp+8], 1 ; set carry bit of eflags register

4598 00004730 CF <1> iretd

4599 <1> G4:

4600 00004731 C605[D3580100]07 <1> MOV byte [DISK\_STATUS1],INIT\_FAIL ; OPERATION FAILED

4601 00004738 B407 <1> MOV AH,INIT\_FAIL

4602 0000473A 28C0 <1> SUB AL,AL

4603 0000473C 6629D2 <1> SUB DX,DX

4604 0000473F 6629C9 <1> SUB CX,CX

4605 <1> ; 29/05/2016 (\*)

4606 <1> ;STC ; SET ERROR FLAG

4607 <1> ;JMP short G5

4608 00004742 EBE5 <1> jmp short \_G6

4609 <1>

4610 <1> ;----------------------------------------

4611 <1> ; INITIALIZE DRIVE (AH = 09H) :

4612 <1> ;----------------------------------------

4613 <1> ; 03/01/2015

4614 <1> ; According to ATA-ATAPI specification v2.0 to v5.0

4615 <1> ; logical sector per logical track

4616 <1> ; and logical heads - 1 would be set but

4617 <1> ; it is seen as it will be good

4618 <1> ; if physical parameters will be set here

4619 <1> ; because, number of heads <= 16.

4620 <1> ; (logical heads usually more than 16)

4621 <1> ; NOTE: ATA logical parameters (software C, H, S)

4622 <1> ; == INT 13h physical parameters

4623 <1>

4624 <1> ;INIT\_DRV:

4625 <1> ; MOV byte [CMD\_BLOCK+6],SET\_PARM\_CMD

4626 <1> ; CALL GET\_VEC ; ES:BX -> PARAMETER BLOCK

4627 <1> ; MOV AL,[ES:BX+2] ; GET NUMBER OF HEADS

4628 <1> ; DEC AL ; CONVERT TO 0-INDEX

4629 <1> ; MOV AH,[CMD\_BLOCK+5] ; GET SDH REGISTER

4630 <1> ; AND AH,0F0H ; CHANGE HEAD NUMBER

4631 <1> ; OR AH,AL ; TO MAX HEAD

4632 <1> ; MOV [CMD\_BLOCK+5],AH

4633 <1> ; MOV AL,[ES:BX+14] ; MAX SECTOR NUMBER

4634 <1> ; MOV [CMD\_BLOCK+1],AL

4635 <1> ; SUB AX,AX

4636 <1> ; MOV [CMD\_BLOCK+3],AL ; ZERO FLAGS

4637 <1> ; CALL COMMAND ; TELL CONTROLLER

4638 <1> ; JNZ short INIT\_EXIT ; CONTROLLER BUSY ERROR

4639 <1> ; CALL NOT\_BUSY ; WAIT FOR IT TO BE DONE

4640 <1> ; JNZ short INIT\_EXIT ; TIME OUT

4641 <1> ; CALL CHECK\_STATUS

4642 <1> ;INIT\_EXIT:

4643 <1> ; RETn

4644 <1>

4645 <1> ; 04/01/2015

4646 <1> ; 02/01/2015 - Derived from from AWARD BIOS 1999

4647 <1> ; AHDSK.ASM - INIT\_DRIVE

4648 <1> INIT\_DRV:

4649 <1> ;xor ah,ah

4650 00004744 31C0 <1> xor eax, eax ; 21/02/2015

4651 00004746 B00B <1> mov al,11 ; Physical heads from translated HDPT

4652 00004748 3825[E8580100] <1> cmp [LBAMode], ah ; 0

4653 0000474E 7702 <1> ja short idrv0

4654 00004750 B002 <1> mov al,2 ; Physical heads from standard HDPT

4655 <1> idrv0:

4656 <1> ; DL = drive number (0 based)

4657 00004752 E8C6030000 <1> call GET\_VEC

4658 <1> ;push bx

4659 00004757 53 <1> push ebx ; 21/02/2015

4660 <1> ;add bx,ax

4661 00004758 01C3 <1> add ebx, eax

4662 <1> ;; 05/01/2015

4663 0000475A 8A25[EC5C0000] <1> mov ah, [hf\_m\_s] ; drive number (0= master, 1= slave)

4664 <1> ;;and ah,1

4665 00004760 C0E404 <1> shl ah,4

4666 00004763 80CCA0 <1> or ah,0A0h ; Drive/Head register - 10100000b (A0h)

4667 <1> ;mov al,[es:bx]

4668 00004766 8A03 <1> mov al, [ebx] ; 21/02/2015

4669 00004768 FEC8 <1> dec al ; last head number

4670 <1> ;and al,0Fh

4671 0000476A 08E0 <1> or al,ah ; lower 4 bits for head number

4672 <1> ;

4673 0000476C C645FE91 <1> mov byte [CMD\_BLOCK+6],SET\_PARM\_CMD

4674 00004770 8845FD <1> mov [CMD\_BLOCK+5],al

4675 <1> ;pop bx

4676 00004773 5B <1> pop ebx

4677 00004774 29C0 <1> sub eax, eax ; 21/02/2015

4678 00004776 B004 <1> mov al,4 ; Physical sec per track from translated HDPT

4679 00004778 803D[E8580100]00 <1> cmp byte [LBAMode], 0

4680 0000477F 7702 <1> ja short idrv1

4681 00004781 B00E <1> mov al,14 ; Physical sec per track from standard HDPT

4682 <1> idrv1:

4683 <1> ;xor ah,ah

4684 <1> ;add bx,ax

4685 00004783 01C3 <1> add ebx, eax ; 21/02/2015

4686 <1> ;mov al,[es:bx]

4687 <1> ; sector number

4688 00004785 8A03 <1> mov al, [ebx]

4689 00004787 8845F9 <1> mov [CMD\_BLOCK+1],al

4690 0000478A 28C0 <1> sub al,al

4691 0000478C 8845FB <1> mov [CMD\_BLOCK+3],al ; ZERO FLAGS

4692 0000478F E8CA010000 <1> call COMMAND ; TELL CONTROLLER

4693 00004794 750C <1> jnz short INIT\_EXIT ; CONTROLLER BUSY ERROR

4694 00004796 E878020000 <1> call NOT\_BUSY ; WAIT FOR IT TO BE DONE

4695 0000479B 7505 <1> jnz short INIT\_EXIT ; TIME OUT

4696 0000479D E8C9020000 <1> call CHECK\_STATUS

4697 <1> INIT\_EXIT:

4698 000047A2 C3 <1> RETn

4699 <1>

4700 <1> ;----------------------------------------

4701 <1> ; READ LONG (AH = 0AH) :

4702 <1> ;----------------------------------------

4703 <1>

4704 <1> RD\_LONG:

4705 <1> ;MOV @CMD\_BLOCK+6,READ\_CMD OR ECC\_MODE

4706 000047A3 C645FE22 <1> mov byte [CMD\_BLOCK+6],READ\_CMD + ECC\_MODE

4707 000047A7 E9E0000000 <1> JMP COMMANDI

4708 <1>

4709 <1> ;----------------------------------------

4710 <1> ; WRITE LONG (AH = 0BH) :

4711 <1> ;----------------------------------------

4712 <1>

4713 <1> WR\_LONG:

4714 <1> ;MOV @CMD\_BLOCK+6,WRITE\_CMD OR ECC\_MODE

4715 000047AC C645FE32 <1> MOV byte [CMD\_BLOCK+6],WRITE\_CMD + ECC\_MODE

4716 000047B0 E932010000 <1> JMP COMMANDO

4717 <1>

4718 <1> ;----------------------------------------

4719 <1> ; SEEK (AH = 0CH) :

4720 <1> ;----------------------------------------

4721 <1>

4722 <1> DISK\_SEEK:

4723 000047B5 C645FE70 <1> MOV byte [CMD\_BLOCK+6],SEEK\_CMD

4724 000047B9 E8A0010000 <1> CALL COMMAND

4725 000047BE 751C <1> JNZ short DS\_EXIT ; CONTROLLER BUSY ERROR

4726 000047C0 E812020000 <1> CALL \_WAIT

4727 000047C5 7515 <1> JNZ DS\_EXIT ; TIME OUT ON SEEK

4728 000047C7 E89F020000 <1> CALL CHECK\_STATUS

4729 000047CC 803D[D3580100]40 <1> CMP byte [DISK\_STATUS1],BAD\_SEEK

4730 000047D3 7507 <1> JNE short DS\_EXIT

4731 000047D5 C605[D3580100]00 <1> MOV byte [DISK\_STATUS1],0

4732 <1> DS\_EXIT:

4733 000047DC C3 <1> RETn

4734 <1>

4735 <1> ;----------------------------------------

4736 <1> ; TEST DISK READY (AH = 10H) :

4737 <1> ;----------------------------------------

4738 <1>

4739 <1> TST\_RDY: ; WAIT FOR CONTROLLER

4740 000047DD E831020000 <1> CALL NOT\_BUSY

4741 000047E2 751C <1> JNZ short TR\_EX

4742 000047E4 8A45FD <1> MOV AL,[CMD\_BLOCK+5] ; SELECT DRIVE

4743 000047E7 668B15[E85C0000] <1> MOV DX,[HF\_PORT]

4744 000047EE 80C206 <1> add dl,6

4745 000047F1 EE <1> OUT DX,AL

4746 000047F2 E88C020000 <1> CALL CHECK\_ST ; CHECK STATUS ONLY

4747 000047F7 7507 <1> JNZ short TR\_EX

4748 000047F9 C605[D3580100]00 <1> MOV byte [DISK\_STATUS1],0 ; WIPE OUT DATA CORRECTED ERROR

4749 <1> TR\_EX:

4750 00004800 C3 <1> RETn

4751 <1>

4752 <1> ;----------------------------------------

4753 <1> ; RECALIBRATE (AH = 11H) :

4754 <1> ;----------------------------------------

4755 <1>

4756 <1> HDISK\_RECAL:

4757 00004801 C645FE10 <1> MOV byte [CMD\_BLOCK+6],RECAL\_CMD ; 10h, 16

4758 00004805 E854010000 <1> CALL COMMAND ; START THE OPERATION

4759 0000480A 7523 <1> JNZ short RECAL\_EXIT ; ERROR

4760 0000480C E8C6010000 <1> CALL \_WAIT ; WAIT FOR COMPLETION

4761 00004811 7407 <1> JZ short RECAL\_X ; TIME OUT ONE OK ?

4762 00004813 E8BF010000 <1> CALL \_WAIT ; WAIT FOR COMPLETION LONGER

4763 00004818 7515 <1> JNZ short RECAL\_EXIT ; TIME OUT TWO TIMES IS ERROR

4764 <1> RECAL\_X:

4765 0000481A E84C020000 <1> CALL CHECK\_STATUS

4766 0000481F 803D[D3580100]40 <1> CMP byte [DISK\_STATUS1],BAD\_SEEK ; SEEK NOT COMPLETE

4767 00004826 7507 <1> JNE short RECAL\_EXIT ; IS OK

4768 00004828 C605[D3580100]00 <1> MOV byte [DISK\_STATUS1],0

4769 <1> RECAL\_EXIT:

4770 0000482F 803D[D3580100]00 <1> CMP byte [DISK\_STATUS1],0

4771 00004836 C3 <1> RETn

4772 <1>

4773 <1> ;----------------------------------------

4774 <1> ; CONTROLLER DIAGNOSTIC (AH = 14H) :

4775 <1> ;----------------------------------------

4776 <1>

4777 <1> CTLR\_DIAGNOSTIC:

4778 00004837 FA <1> CLI ; DISABLE INTERRUPTS WHILE CHANGING MASK

4779 00004838 E4A1 <1> IN AL,INTB01 ; TURN ON SECOND INTERRUPT CHIP

4780 <1> ;AND AL,0BFH

4781 0000483A 243F <1> and al, 3Fh ; enable IRQ 14 & IRQ 15

4782 <1> ;JMP $+2

4783 <1> IODELAY

4783 0000483C EB00 <2> jmp short $+2

4783 0000483E EB00 <2> jmp short $+2

4784 00004840 E6A1 <1> OUT INTB01,AL

4785 <1> IODELAY

4785 00004842 EB00 <2> jmp short $+2

4785 00004844 EB00 <2> jmp short $+2

4786 00004846 E421 <1> IN AL,INTA01 ; LET INTERRUPTS PASS THRU TO

4787 00004848 24FB <1> AND AL,0FBH ; SECOND CHIP

4788 <1> ;JMP $+2

4789 <1> IODELAY

4789 0000484A EB00 <2> jmp short $+2

4789 0000484C EB00 <2> jmp short $+2

4790 0000484E E621 <1> OUT INTA01,AL

4791 00004850 FB <1> STI

4792 00004851 E8BD010000 <1> CALL NOT\_BUSY ; WAIT FOR CARD

4793 00004856 752B <1> JNZ short CD\_ERR ; BAD CARD

4794 <1> ;MOV DX, HF\_PORT+7

4795 00004858 668B15[E85C0000] <1> mov dx, [HF\_PORT]

4796 0000485F 80C207 <1> add dl, 7

4797 00004862 B090 <1> MOV AL,DIAG\_CMD ; START DIAGNOSE

4798 00004864 EE <1> OUT DX,AL

4799 00004865 E8A9010000 <1> CALL NOT\_BUSY ; WAIT FOR IT TO COMPLETE

4800 0000486A B480 <1> MOV AH,TIME\_OUT

4801 0000486C 7517 <1> JNZ short CD\_EXIT ; TIME OUT ON DIAGNOSTIC

4802 <1> ;MOV DX,HF\_PORT+1 ; GET ERROR REGISTER

4803 0000486E 668B15[E85C0000] <1> mov dx, [HF\_PORT]

4804 00004875 FEC2 <1> inc dl

4805 00004877 EC <1> IN AL,DX

4806 00004878 A2[CA580100] <1> MOV [HF\_ERROR],AL ; SAVE IT

4807 0000487D B400 <1> MOV AH,0

4808 0000487F 3C01 <1> CMP AL,1 ; CHECK FOR ALL OK

4809 00004881 7402 <1> JE SHORT CD\_EXIT

4810 00004883 B420 <1> CD\_ERR: MOV AH,BAD\_CNTLR

4811 <1> CD\_EXIT:

4812 00004885 8825[D3580100] <1> MOV [DISK\_STATUS1],AH

4813 0000488B C3 <1> RETn

4814 <1>

4815 <1> ;----------------------------------------

4816 <1> ; COMMANDI :

4817 <1> ; REPEATEDLY INPUTS DATA TILL :

4818 <1> ; NSECTOR RETURNS ZERO :

4819 <1> ;----------------------------------------

4820 <1> COMMANDI:

4821 0000488C E862020000 <1> CALL CHECK\_DMA ; CHECK 64K BOUNDARY ERROR

4822 00004891 7253 <1> JC short CMD\_ABORT

4823 <1> ;MOV DI,BX

4824 00004893 89DF <1> mov edi, ebx ; 21/02/2015

4825 00004895 E8C4000000 <1> CALL COMMAND ; OUTPUT COMMAND

4826 0000489A 754A <1> JNZ short CMD\_ABORT

4827 <1> CMD\_I1:

4828 0000489C E836010000 <1> CALL \_WAIT ; WAIT FOR DATA REQUEST INTERRUPT

4829 000048A1 7543 <1> JNZ short TM\_OUT ; TIME OUT

4830 <1> cmd\_i1x: ; 18/02/2016

4831 <1> ;MOV CX,256 ; SECTOR SIZE IN WORDS

4832 000048A3 B900010000 <1> mov ecx, 256 ; 21/02/2015

4833 <1> ;MOV DX,HF\_PORT

4834 000048A8 668B15[E85C0000] <1> mov dx,[HF\_PORT]

4835 000048AF FA <1> CLI

4836 000048B0 FC <1> CLD

4837 000048B1 F3666D <1> REP INSW ; GET THE SECTOR

4838 000048B4 FB <1> STI

4839 000048B5 F645FE02 <1> TEST byte [CMD\_BLOCK+6],ECC\_MODE ; CHECK FOR NORMAL INPUT

4840 000048B9 7419 <1> JZ short CMD\_I3

4841 000048BB E880010000 <1> CALL WAIT\_DRQ ; WAIT FOR DATA REQUEST

4842 000048C0 7224 <1> JC short TM\_OUT

4843 <1> ;MOV DX,HF\_PORT

4844 000048C2 668B15[E85C0000] <1> mov dx,[HF\_PORT]

4845 <1> ;MOV CX,4 ; GET ECC BYTES

4846 000048C9 B904000000 <1> mov ecx, 4 ; mov cx, 4

4847 000048CE EC <1> CMD\_I2: IN AL,DX

4848 <1> ;MOV [ES:DI],AL ; GO SLOW FOR BOARD

4849 000048CF 8807 <1> mov [edi], al ; 21/02/2015

4850 000048D1 47 <1> INC eDI

4851 000048D2 E2FA <1> LOOP CMD\_I2

4852 <1> CMD\_I3:

4853 <1> ; wait for 400 ns

4854 000048D4 80C207 <1> add dl, 7

4855 000048D7 EC <1> in al, dx

4856 000048D8 EC <1> in al, dx

4857 000048D9 EC <1> in al, dx

4858 <1> ;

4859 000048DA E88C010000 <1> CALL CHECK\_STATUS

4860 000048DF 7505 <1> JNZ short CMD\_ABORT ; ERROR RETURNED

4861 000048E1 FE4DF9 <1> DEC byte [CMD\_BLOCK+1] ; CHECK FOR MORE

4862 <1> ;JNZ SHORT CMD\_I1

4863 000048E4 75BD <1> jnz short cmd\_i1x ; 18/02/2016

4864 <1> CMD\_ABORT:

4865 000048E6 C3 <1> TM\_OUT: RETn

4866 <1>

4867 <1> ;----------------------------------------

4868 <1> ; COMMANDO :

4869 <1> ; REPEATEDLY OUTPUTS DATA TILL :

4870 <1> ; NSECTOR RETURNS ZERO :

4871 <1> ;----------------------------------------

4872 <1> COMMANDO:

4873 000048E7 E807020000 <1> CALL CHECK\_DMA ; CHECK 64K BOUNDARY ERROR

4874 000048EC 72F8 <1> JC short CMD\_ABORT

4875 000048EE 89DE <1> CMD\_OF: MOV eSI,eBX ; 21/02/2015

4876 000048F0 E869000000 <1> CALL COMMAND ; OUTPUT COMMAND

4877 000048F5 75EF <1> JNZ short CMD\_ABORT

4878 000048F7 E844010000 <1> CALL WAIT\_DRQ ; WAIT FOR DATA REQUEST

4879 000048FC 72E8 <1> JC short TM\_OUT ; TOO LONG

4880 <1> CMD\_O1: ;PUSH DS

4881 <1> ;PUSH ES ; MOVE ES TO DS

4882 <1> ;POP DS

4883 <1> ;MOV CX,256 ; PUT THE DATA OUT TO THE CARD

4884 <1> ;MOV DX,HF\_PORT

4885 <1> ; 01/02/2015

4886 000048FE 668B15[E85C0000] <1> mov dx, [HF\_PORT]

4887 <1> ;push es

4888 <1> ;pop ds

4889 <1> ;mov cx, 256

4890 00004905 B900010000 <1> mov ecx, 256 ; 21/02/2015

4891 0000490A FA <1> CLI

4892 0000490B FC <1> CLD

4893 0000490C F3666F <1> REP OUTSW

4894 0000490F FB <1> STI

4895 <1> ;POP DS ; RESTORE DS

4896 00004910 F645FE02 <1> TEST byte [CMD\_BLOCK+6],ECC\_MODE ; CHECK FOR NORMAL OUTPUT

4897 00004914 7419 <1> JZ short CMD\_O3

4898 00004916 E825010000 <1> CALL WAIT\_DRQ ; WAIT FOR DATA REQUEST

4899 0000491B 72C9 <1> JC short TM\_OUT

4900 <1> ;MOV DX,HF\_PORT

4901 0000491D 668B15[E85C0000] <1> mov dx, [HF\_PORT]

4902 <1> ;MOV CX,4 ; OUTPUT THE ECC BYTES

4903 00004924 B904000000 <1> mov ecx, 4 ; mov cx, 4

4904 <1> CMD\_O2: ;MOV AL,[ES:SI]

4905 00004929 8A06 <1> mov al, [esi]

4906 0000492B EE <1> OUT DX,AL

4907 0000492C 46 <1> INC eSI

4908 0000492D E2FA <1> LOOP CMD\_O2

4909 <1> CMD\_O3:

4910 0000492F E8A3000000 <1> CALL \_WAIT ; WAIT FOR SECTOR COMPLETE INTERRUPT

4911 00004934 75B0 <1> JNZ short TM\_OUT ; ERROR RETURNED

4912 00004936 E830010000 <1> CALL CHECK\_STATUS

4913 0000493B 75A9 <1> JNZ short CMD\_ABORT

4914 0000493D F605[C9580100]08 <1> TEST byte [HF\_STATUS],ST\_DRQ ; CHECK FOR MORE

4915 00004944 75B8 <1> JNZ SHORT CMD\_O1

4916 <1> ;MOV DX,HF\_PORT+2 ; CHECK RESIDUAL SECTOR COUNT

4917 00004946 668B15[E85C0000] <1> mov dx, [HF\_PORT]

4918 <1> ;add dl, 2

4919 0000494D FEC2 <1> inc dl

4920 0000494F FEC2 <1> inc dl

4921 00004951 EC <1> IN AL,DX ;

4922 00004952 A8FF <1> TEST AL,0FFH ;

4923 00004954 7407 <1> JZ short CMD\_O4 ; COUNT = 0 OK

4924 00004956 C605[D3580100]BB <1> MOV byte [DISK\_STATUS1],UNDEF\_ERR

4925 <1> ; OPERATION ABORTED - PARTIAL TRANSFER

4926 <1> CMD\_O4:

4927 0000495D C3 <1> RETn

4928 <1>

4929 <1> ;--------------------------------------------------------

4930 <1> ; COMMAND :

4931 <1> ; THIS ROUTINE OUTPUTS THE COMMAND BLOCK :

4932 <1> ; OUTPUT :

4933 <1> ; BL = STATUS :

4934 <1> ; BH = ERROR REGISTER :

4935 <1> ;--------------------------------------------------------

4936 <1>

4937 <1> COMMAND:

4938 0000495E 53 <1> PUSH eBX ; WAIT FOR SEEK COMPLETE AND READY

4939 <1> ;;MOV CX,DELAY\_2 ; SET INITIAL DELAY BEFORE TEST

4940 <1> COMMAND1:

4941 <1> ;;PUSH CX ; SAVE LOOP COUNT

4942 0000495F E879FEFFFF <1> CALL TST\_RDY ; CHECK DRIVE READY

4943 <1> ;;POP CX

4944 00004964 7419 <1> JZ short COMMAND2 ; DRIVE IS READY

4945 00004966 803D[D3580100]80 <1> CMP byte [DISK\_STATUS1],TIME\_OUT ; TST\_RDY TIMED OUT--GIVE UP

4946 <1> ;JZ short CMD\_TIMEOUT

4947 <1> ;;LOOP COMMAND1 ; KEEP TRYING FOR A WHILE

4948 <1> ;JMP SHORT COMMAND4 ; ITS NOT GOING TO GET READY

4949 0000496D 7507 <1> jne short COMMAND4

4950 <1> CMD\_TIMEOUT:

4951 0000496F C605[D3580100]20 <1> MOV byte [DISK\_STATUS1],BAD\_CNTLR

4952 <1> COMMAND4:

4953 00004976 5B <1> POP eBX

4954 00004977 803D[D3580100]00 <1> CMP byte [DISK\_STATUS1],0 ; SET CONDITION CODE FOR CALLER

4955 0000497E C3 <1> RETn

4956 <1> COMMAND2:

4957 0000497F 5B <1> POP eBX

4958 00004980 57 <1> PUSH eDI

4959 00004981 C605[CB580100]00 <1> MOV byte [HF\_INT\_FLAG],0 ; RESET INTERRUPT FLAG

4960 00004988 FA <1> CLI ; INHIBIT INTERRUPTS WHILE CHANGING MASK

4961 00004989 E4A1 <1> IN AL,INTB01 ; TURN ON SECOND INTERRUPT CHIP

4962 <1> ;AND AL,0BFH

4963 0000498B 243F <1> and al, 3Fh ; Enable IRQ 14 & 15

4964 <1> ;JMP $+2

4965 <1> IODELAY

4965 0000498D EB00 <2> jmp short $+2

4965 0000498F EB00 <2> jmp short $+2

4966 00004991 E6A1 <1> OUT INTB01,AL

4967 00004993 E421 <1> IN AL,INTA01 ; LET INTERRUPTS PASS THRU TO

4968 00004995 24FB <1> AND AL,0FBH ; SECOND CHIP

4969 <1> ;JMP $+2

4970 <1> IODELAY

4970 00004997 EB00 <2> jmp short $+2

4970 00004999 EB00 <2> jmp short $+2

4971 0000499B E621 <1> OUT INTA01,AL

4972 0000499D FB <1> STI

4973 0000499E 31FF <1> XOR eDI,eDI ; INDEX THE COMMAND TABLE

4974 <1> ;MOV DX,HF\_PORT+1 ; DISK ADDRESS

4975 000049A0 668B15[E85C0000] <1> mov dx, [HF\_PORT]

4976 000049A7 FEC2 <1> inc dl

4977 000049A9 F605[D5580100]C0 <1> TEST byte [CONTROL\_BYTE],0C0H ; CHECK FOR RETRY SUPPRESSION

4978 000049B0 7411 <1> JZ short COMMAND3

4979 000049B2 8A45FE <1> MOV AL, [CMD\_BLOCK+6] ; YES-GET OPERATION CODE

4980 000049B5 24F0 <1> AND AL,0F0H ; GET RID OF MODIFIERS

4981 000049B7 3C20 <1> CMP AL,20H ; 20H-40H IS READ, WRITE, VERIFY

4982 000049B9 7208 <1> JB short COMMAND3

4983 000049BB 3C40 <1> CMP AL,40H

4984 000049BD 7704 <1> JA short COMMAND3

4985 000049BF 804DFE01 <1> OR byte [CMD\_BLOCK+6],NO\_RETRIES

4986 <1> ; VALID OPERATION FOR RETRY SUPPRESS

4987 <1> COMMAND3:

4988 000049C3 8A443DF8 <1> MOV AL,[CMD\_BLOCK+eDI] ; GET THE COMMAND STRING BYTE

4989 000049C7 EE <1> OUT DX,AL ; GIVE IT TO CONTROLLER

4990 <1> IODELAY

4990 000049C8 EB00 <2> jmp short $+2

4990 000049CA EB00 <2> jmp short $+2

4991 000049CC 47 <1> INC eDI ; NEXT BYTE IN COMMAND BLOCK

4992 000049CD 6642 <1> INC DX ; NEXT DISK ADAPTER REGISTER

4993 000049CF 6683FF07 <1> cmp di, 7 ; 1/1/2015 ; ALL DONE?

4994 000049D3 75EE <1> JNZ short COMMAND3 ; NO--GO DO NEXT ONE

4995 000049D5 5F <1> POP eDI

4996 000049D6 C3 <1> RETn ; ZERO FLAG IS SET

4997 <1>

4998 <1> ;CMD\_TIMEOUT:

4999 <1> ; MOV byte [DISK\_STATUS1],BAD\_CNTLR

5000 <1> ;COMMAND4:

5001 <1> ; POP BX

5002 <1> ; CMP [DISK\_STATUS1],0 ; SET CONDITION CODE FOR CALLER

5003 <1> ; RETn

5004 <1>

5005 <1> ;----------------------------------------

5006 <1> ; WAIT FOR INTERRUPT :

5007 <1> ;----------------------------------------

5008 <1> ;WAIT:

5009 <1> \_WAIT:

5010 000049D7 FB <1> STI ; MAKE SURE INTERRUPTS ARE ON

5011 <1> ;SUB CX,CX ; SET INITIAL DELAY BEFORE TEST

5012 <1> ;CLC

5013 <1> ;MOV AX,9000H ; DEVICE WAIT INTERRUPT

5014 <1> ;INT 15H

5015 <1> ;JC WT2 ; DEVICE TIMED OUT

5016 <1> ;MOV BL,DELAY\_1 ; SET DELAY COUNT

5017 <1>

5018 <1> ;mov bl, WAIT\_HDU\_INT\_HI

5019 <1> ;; 21/02/2015

5020 <1> ;;mov bl, WAIT\_HDU\_INT\_HI + 1

5021 <1> ;;mov cx, WAIT\_HDU\_INT\_LO

5022 000049D8 B915160500 <1> mov ecx, WAIT\_HDU\_INT\_LH

5023 <1> ; (AWARD BIOS -> WAIT\_FOR\_MEM)

5024 <1> ;----- WAIT LOOP

5025 <1>

5026 <1> WT1:

5027 <1> ;TEST byte [HF\_INT\_FLAG],80H ; TEST FOR INTERRUPT

5028 000049DD F605[CB580100]C0 <1> test byte [HF\_INT\_FLAG],0C0h

5029 <1> ;LOOPZ WT1

5030 000049E4 7517 <1> JNZ short WT3 ; INTERRUPT--LETS GO

5031 <1> ;DEC BL

5032 <1> ;JNZ short WT1 ; KEEP TRYING FOR A WHILE

5033 <1>

5034 <1> WT1\_hi:

5035 000049E6 E461 <1> in al, SYS1 ; 61h (PORT\_B) ; wait for lo to hi

5036 000049E8 A810 <1> test al, 10h ; transition on memory

5037 000049EA 75FA <1> jnz short WT1\_hi ; refresh.

5038 <1> WT1\_lo:

5039 000049EC E461 <1> in al, SYS1 ; 061h (PORT\_B)

5040 000049EE A810 <1> test al, 10h

5041 000049F0 74FA <1> jz short WT1\_lo

5042 000049F2 E2E9 <1> loop WT1

5043 <1> ;;or bl, bl

5044 <1> ;;jz short WT2

5045 <1> ;;dec bl

5046 <1> ;;jmp short WT1

5047 <1> ;dec bl

5048 <1> ;jnz short WT1

5049 <1>

5050 000049F4 C605[D3580100]80 <1> WT2: MOV byte [DISK\_STATUS1],TIME\_OUT ; REPORT TIME OUT ERROR

5051 000049FB EB0E <1> JMP SHORT WT4

5052 000049FD C605[D3580100]00 <1> WT3: MOV byte [DISK\_STATUS1],0

5053 00004A04 C605[CB580100]00 <1> MOV byte [HF\_INT\_FLAG],0

5054 00004A0B 803D[D3580100]00 <1> WT4: CMP byte [DISK\_STATUS1],0 ; SET CONDITION CODE FOR CALLER

5055 00004A12 C3 <1> RETn

5056 <1>

5057 <1> ;----------------------------------------

5058 <1> ; WAIT FOR CONTROLLER NOT BUSY :

5059 <1> ;----------------------------------------

5060 <1> NOT\_BUSY:

5061 00004A13 FB <1> STI ; MAKE SURE INTERRUPTS ARE ON

5062 <1> ;PUSH eBX

5063 <1> ;SUB CX,CX ; SET INITIAL DELAY BEFORE TEST

5064 00004A14 668B15[E85C0000] <1> mov DX, [HF\_PORT]

5065 00004A1B 80C207 <1> add dl, 7 ; Status port (HF\_PORT+7)

5066 <1> ;MOV BL,DELAY\_1

5067 <1> ; wait for 10 seconds

5068 <1> ;mov cx, WAIT\_HDU\_INT\_LO ; 1615h

5069 <1> ;;mov bl, WAIT\_HDU\_INT\_HI ; 05h

5070 <1> ;mov bl, WAIT\_HDU\_INT\_HI + 1

5071 00004A1E B915160500 <1> mov ecx, WAIT\_HDU\_INT\_LH ; 21/02/2015

5072 <1> ;

5073 <1> ;; mov byte [wait\_count], 0 ; Reset wait counter

5074 <1> NB1:

5075 00004A23 EC <1> IN AL,DX ; CHECK STATUS

5076 <1> ;TEST AL,ST\_BUSY

5077 00004A24 2480 <1> and al, ST\_BUSY

5078 <1> ;LOOPNZ NB1

5079 00004A26 7410 <1> JZ short NB2 ; NOT BUSY--LETS GO

5080 <1> ;DEC BL

5081 <1> ;JNZ short NB1 ; KEEP TRYING FOR A WHILE

5082 <1>

5083 00004A28 E461 <1> NB1\_hi: IN AL,SYS1 ; wait for hi to lo

5084 00004A2A A810 <1> TEST AL,010H ; transition on memory

5085 00004A2C 75FA <1> JNZ SHORT NB1\_hi ; refresh.

5086 00004A2E E461 <1> NB1\_lo: IN AL,SYS1

5087 00004A30 A810 <1> TEST AL,010H

5088 00004A32 74FA <1> JZ short NB1\_lo

5089 00004A34 E2ED <1> LOOP NB1

5090 <1> ;dec bl

5091 <1> ;jnz short NB1

5092 <1> ;

5093 <1> ;; cmp byte [wait\_count], 182 ; 10 seconds (182 timer ticks)

5094 <1> ;; jb short NB1

5095 <1> ;

5096 <1> ;MOV [DISK\_STATUS1],TIME\_OUT ; REPORT TIME OUT ERROR

5097 <1> ;JMP SHORT NB3

5098 00004A36 B080 <1> mov al, TIME\_OUT

5099 <1> NB2:

5100 <1> ;MOV byte [DISK\_STATUS1],0

5101 <1> ;NB3:

5102 <1> ;POP eBX

5103 00004A38 A2[D3580100] <1> mov [DISK\_STATUS1], al ;;; will be set after return

5104 <1> ;CMP byte [DISK\_STATUS1],0 ; SET CONDITION CODE FOR CALLER

5105 00004A3D 08C0 <1> or al, al ; (zf = 0 --> timeout)

5106 00004A3F C3 <1> RETn

5107 <1>

5108 <1> ;----------------------------------------

5109 <1> ; WAIT FOR DATA REQUEST :

5110 <1> ;----------------------------------------

5111 <1> WAIT\_DRQ:

5112 <1> ;MOV CX,DELAY\_3

5113 <1> ;MOV DX,HF\_PORT+7

5114 00004A40 668B15[E85C0000] <1> mov dx, [HF\_PORT]

5115 00004A47 80C207 <1> add dl, 7

5116 <1> ;;MOV bl, WAIT\_HDU\_DRQ\_HI ; 0

5117 <1> ;MOV cx, WAIT\_HDU\_DRQ\_LO ; 1000 (30 milli seconds)

5118 <1> ; (but it is written as 2000

5119 <1> ; micro seconds in ATORGS.ASM file

5120 <1> ; of Award Bios - 1999, D1A0622)

5121 00004A4A B9E8030000 <1> mov ecx, WAIT\_HDU\_DRQ\_LH ; 21/02/2015

5122 00004A4F EC <1> WQ\_1: IN AL,DX ; GET STATUS

5123 00004A50 A808 <1> TEST AL,ST\_DRQ ; WAIT FOR DRQ

5124 00004A52 7516 <1> JNZ short WQ\_OK

5125 <1> ;LOOP WQ\_1 ; KEEP TRYING FOR A SHORT WHILE

5126 <1> WQ\_hi:

5127 00004A54 E461 <1> IN AL,SYS1 ; wait for hi to lo

5128 00004A56 A810 <1> TEST AL,010H ; transition on memory

5129 00004A58 75FA <1> JNZ SHORT WQ\_hi ; refresh.

5130 00004A5A E461 <1> WQ\_lo: IN AL,SYS1

5131 00004A5C A810 <1> TEST AL,010H

5132 00004A5E 74FA <1> JZ SHORT WQ\_lo

5133 00004A60 E2ED <1> LOOP WQ\_1

5134 <1>

5135 00004A62 C605[D3580100]80 <1> MOV byte [DISK\_STATUS1],TIME\_OUT ; ERROR

5136 00004A69 F9 <1> STC

5137 <1> WQ\_OK:

5138 00004A6A C3 <1> RETn

5139 <1> ;WQ\_OK: ;CLC

5140 <1> ; RETn

5141 <1>

5142 <1> ;----------------------------------------

5143 <1> ; CHECK FIXED DISK STATUS :

5144 <1> ;----------------------------------------

5145 <1> CHECK\_STATUS:

5146 00004A6B E813000000 <1> CALL CHECK\_ST ; CHECK THE STATUS BYTE

5147 00004A70 7509 <1> JNZ short CHECK\_S1 ; AN ERROR WAS FOUND

5148 00004A72 A801 <1> TEST AL,ST\_ERROR ; WERE THERE ANY OTHER ERRORS

5149 00004A74 7405 <1> JZ short CHECK\_S1 ; NO ERROR REPORTED

5150 00004A76 E849000000 <1> CALL CHECK\_ER ; ERROR REPORTED

5151 <1> CHECK\_S1:

5152 00004A7B 803D[D3580100]00 <1> CMP byte [DISK\_STATUS1],0 ; SET STATUS FOR CALLER

5153 00004A82 C3 <1> RETn

5154 <1>

5155 <1> ;----------------------------------------

5156 <1> ; CHECK FIXED DISK STATUS BYTE :

5157 <1> ;----------------------------------------

5158 <1> CHECK\_ST:

5159 <1> ;MOV DX,HF\_PORT+7 ; GET THE STATUS

5160 00004A83 668B15[E85C0000] <1> mov dx, [HF\_PORT]

5161 00004A8A 80C207 <1> add dl, 7

5162 <1>

5163 <1> ; 17/02/2016

5164 <1> ;(http://wiki.osdev.org/ATA\_PIO\_Mode)

5165 <1> ;"delay 400ns to allow drive to set new values of BSY and DRQ"

5166 00004A8D EC <1> IN AL,DX

5167 <1> ;in al, dx ; 100ns

5168 <1> ;in al, dx ; 100ns

5169 <1> ;in al, dx ; 100ns

5170 <1> NEWIODELAY ; 18/02/2016 (AWARD BIOS - 1999, 'CKST' in AHSDK.ASM)

5170 00004A8E E6EB <2> out 0ebh,al

5171 <1> ;

5172 00004A90 A2[C9580100] <1> MOV [HF\_STATUS],AL

5173 00004A95 B400 <1> MOV AH,0

5174 00004A97 A880 <1> TEST AL,ST\_BUSY ; IF STILL BUSY

5175 00004A99 751A <1> JNZ short CKST\_EXIT ; REPORT OK

5176 00004A9B B4CC <1> MOV AH,WRITE\_FAULT

5177 00004A9D A820 <1> TEST AL,ST\_WRT\_FLT ; CHECK FOR WRITE FAULT

5178 00004A9F 7514 <1> JNZ short CKST\_EXIT

5179 00004AA1 B4AA <1> MOV AH,NOT\_RDY

5180 00004AA3 A840 <1> TEST AL,ST\_READY ; CHECK FOR NOT READY

5181 00004AA5 740E <1> JZ short CKST\_EXIT

5182 00004AA7 B440 <1> MOV AH,BAD\_SEEK

5183 00004AA9 A810 <1> TEST AL,ST\_SEEK\_COMPL ; CHECK FOR SEEK NOT COMPLETE

5184 00004AAB 7408 <1> JZ short CKST\_EXIT

5185 00004AAD B411 <1> MOV AH,DATA\_CORRECTED

5186 00004AAF A804 <1> TEST AL,ST\_CORRCTD ; CHECK FOR CORRECTED ECC

5187 00004AB1 7502 <1> JNZ short CKST\_EXIT

5188 00004AB3 B400 <1> MOV AH,0

5189 <1> CKST\_EXIT:

5190 00004AB5 8825[D3580100] <1> MOV [DISK\_STATUS1],AH ; SET ERROR FLAG

5191 00004ABB 80FC11 <1> CMP AH,DATA\_CORRECTED ; KEEP GOING WITH DATA CORRECTED

5192 00004ABE 7403 <1> JZ short CKST\_EX1

5193 00004AC0 80FC00 <1> CMP AH,0

5194 <1> CKST\_EX1:

5195 00004AC3 C3 <1> RETn

5196 <1>

5197 <1> ;----------------------------------------

5198 <1> ; CHECK FIXED DISK ERROR REGISTER :

5199 <1> ;----------------------------------------

5200 <1> CHECK\_ER:

5201 <1> ;MOV DX, HF\_PORT+1 ; GET THE ERROR REGISTER

5202 00004AC4 668B15[E85C0000] <1> mov dx, [HF\_PORT] ;

5203 00004ACB FEC2 <1> inc dl

5204 00004ACD EC <1> IN AL,DX

5205 00004ACE A2[CA580100] <1> MOV [HF\_ERROR],AL

5206 00004AD3 53 <1> PUSH eBX ; 21/02/2015

5207 00004AD4 B908000000 <1> MOV eCX,8 ; TEST ALL 8 BITS

5208 00004AD9 D0E0 <1> CK1: SHL AL,1 ; MOVE NEXT ERROR BIT TO CARRY

5209 00004ADB 7202 <1> JC short CK2 ; FOUND THE ERROR

5210 00004ADD E2FA <1> LOOP CK1 ; KEEP TRYING

5211 00004ADF BB[DC5C0000] <1> CK2: MOV eBX, ERR\_TBL ; COMPUTE ADDRESS OF

5212 00004AE4 01CB <1> ADD eBX,eCX ; ERROR CODE

5213 <1> ;;MOV AH,BYTE [CS:BX] ; GET ERROR CODE

5214 <1> ;mov ah, [bx]

5215 00004AE6 8A23 <1> mov ah, [ebx] ; 21/02/2015

5216 00004AE8 8825[D3580100] <1> CKEX: MOV [DISK\_STATUS1],AH ; SAVE ERROR CODE

5217 00004AEE 5B <1> POP eBX

5218 00004AEF 80FC00 <1> CMP AH,0

5219 00004AF2 C3 <1> RETn

5220 <1>

5221 <1> ;--------------------------------------------------------

5222 <1> ; CHECK\_DMA :

5223 <1> ; -CHECK ES:BX AND # SECTORS TO MAKE SURE THAT IT WILL :

5224 <1> ; FIT WITHOUT SEGMENT OVERFLOW. :

5225 <1> ; -ES:BX HAS BEEN REVISED TO THE FORMAT SSSS:000X :

5226 <1> ; -OK IF # SECTORS < 80H (7FH IF LONG READ OR WRITE) :

5227 <1> ; -OK IF # SECTORS = 80H (7FH) AND BX <= 00H (04H) :

5228 <1> ; -ERROR OTHERWISE :

5229 <1> ;--------------------------------------------------------

5230 <1> CHECK\_DMA:

5231 00004AF3 6650 <1> PUSH AX ; SAVE REGISTERS

5232 00004AF5 66B80080 <1> MOV AX,8000H ; AH = MAX # SECTORS AL = MAX OFFSET

5233 00004AF9 F645FE02 <1> TEST byte [CMD\_BLOCK+6],ECC\_MODE

5234 00004AFD 7404 <1> JZ short CKD1

5235 00004AFF 66B8047F <1> MOV AX,7F04H ; ECC IS 4 MORE BYTES

5236 00004B03 3A65F9 <1> CKD1: CMP AH, [CMD\_BLOCK+1] ; NUMBER OF SECTORS

5237 00004B06 7706 <1> JA short CKDOK ; IT WILL FIT

5238 00004B08 7208 <1> JB short CKDERR ; TOO MANY

5239 00004B0A 38D8 <1> CMP AL,BL ; CHECK OFFSET ON MAX SECTORS

5240 00004B0C 7204 <1> JB short CKDERR ; ERROR

5241 00004B0E F8 <1> CKDOK: CLC ; CLEAR CARRY

5242 00004B0F 6658 <1> POP AX

5243 00004B11 C3 <1> RETn ; NORMAL RETURN

5244 00004B12 F9 <1> CKDERR: STC ; INDICATE ERROR

5245 00004B13 C605[D3580100]09 <1> MOV byte [DISK\_STATUS1],DMA\_BOUNDARY

5246 00004B1A 6658 <1> POP AX

5247 00004B1C C3 <1> RETn

5248 <1>

5249 <1> ;----------------------------------------

5250 <1> ; SET UP ES:BX-> DISK PARMS :

5251 <1> ;----------------------------------------

5252 <1>

5253 <1> ; INPUT -> DL = 0 based drive number

5254 <1> ; OUTPUT -> ES:BX = disk parameter table address

5255 <1>

5256 <1> GET\_VEC:

5257 <1> ;SUB AX,AX ; GET DISK PARAMETER ADDRESS

5258 <1> ;MOV ES,AX

5259 <1> ;TEST DL,1

5260 <1> ;JZ short GV\_0

5261 <1> ; LES BX,[HF1\_TBL\_VEC] ; ES:BX -> DRIVE PARAMETERS

5262 <1> ; JMP SHORT GV\_EXIT

5263 <1> ;GV\_0:

5264 <1> ; LES BX,[HF\_TBL\_VEC] ; ES:BX -> DRIVE PARAMETERS

5265 <1> ;

5266 <1> ;xor bh, bh

5267 00004B1D 31DB <1> xor ebx, ebx

5268 00004B1F 88D3 <1> mov bl, dl

5269 <1> ;;02/01/2015

5270 <1> ;;shl bl, 1 ; port address offset

5271 <1> ;;mov ax, [bx+hd\_ports] ; Base port address (1F0h, 170h)

5272 <1> ;;shl bl, 1 ; dpt pointer offset

5273 00004B21 C0E302 <1> shl bl, 2 ;;

5274 <1> ;add bx, HF\_TBL\_VEC ; Disk parameter table pointer

5275 00004B24 81C3[D8580100] <1> add ebx, HF\_TBL\_VEC ; 21/02/2015

5276 <1> ;push word [bx+2] ; dpt segment

5277 <1> ;pop es

5278 <1> ;mov bx, [bx] ; dpt offset

5279 00004B2A 8B1B <1> mov ebx, [ebx]

5280 <1> ;GV\_EXIT:

5281 00004B2C C3 <1> RETn

5282 <1>

5283 <1> hdc1\_int: ; 21/02/2015

5284 <1> ;--- HARDWARE INT 76H -- ( IRQ LEVEL 14 ) ----------------------

5285 <1> ; :

5286 <1> ; FIXED DISK INTERRUPT ROUTINE :

5287 <1> ; :

5288 <1> ;----------------------------------------------------------------

5289 <1>

5290 <1> ; 22/12/2014

5291 <1> ; IBM PC-XT Model 286 System BIOS Source Code - DISK.ASM (HD\_INT)

5292 <1> ; '11/15/85'

5293 <1> ; AWARD BIOS 1999 (D1A0622)

5294 <1> ; Source Code - ATORGS.ASM (INT\_HDISK, INT\_HDISK1)

5295 <1>

5296 <1> ;int\_76h:

5297 <1> HD\_INT:

5298 00004B2D 6650 <1> PUSH AX

5299 00004B2F 1E <1> PUSH DS

5300 <1> ;CALL DDS

5301 <1> ; 21/02/2015 (32 bit, 386 pm modification)

5302 00004B30 66B81000 <1> mov ax, KDATA

5303 00004B34 8ED8 <1> mov ds, ax

5304 <1> ;

5305 <1> ;;MOV @HF\_INT\_FLAG,0FFH ; ALL DONE

5306 <1> ;mov byte [CS:HF\_INT\_FLAG], 0FFh

5307 00004B36 C605[CB580100]FF <1> mov byte [HF\_INT\_FLAG], 0FFh

5308 <1> ;

5309 00004B3D 6652 <1> push dx

5310 00004B3F 66BAF701 <1> mov dx, HDC1\_BASEPORT+7 ; Status Register (1F7h)

5311 <1> ; Clear Controller

5312 <1> Clear\_IRQ1415: ; (Award BIOS - 1999)

5313 00004B43 EC <1> in al, dx ;

5314 00004B44 665A <1> pop dx

5315 <1> NEWIODELAY

5315 00004B46 E6EB <2> out 0ebh,al

5316 <1> ;

5317 00004B48 B020 <1> MOV AL,EOI ; NON-SPECIFIC END OF INTERRUPT

5318 00004B4A E6A0 <1> OUT INTB00,AL ; FOR CONTROLLER #2

5319 <1> ;JMP $+2 ; WAIT

5320 <1> NEWIODELAY

5320 00004B4C E6EB <2> out 0ebh,al

5321 00004B4E E620 <1> OUT INTA00,AL ; FOR CONTROLLER #1

5322 00004B50 1F <1> POP DS

5323 <1> ;STI ; RE-ENABLE INTERRUPTS

5324 <1> ;MOV AX,9100H ; DEVICE POST

5325 <1> ;INT 15H ; INTERRUPT

5326 <1> irq15\_iret: ; 25/02/2015

5327 00004B51 6658 <1> POP AX

5328 00004B53 CF <1> IRETd ; RETURN FROM INTERRUPT

5329 <1>

5330 <1> hdc2\_int: ; 21/02/2015

5331 <1> ;++++ HARDWARE INT 77H ++ ( IRQ LEVEL 15 ) +++++++++++++++++++++

5332 <1> ; :

5333 <1> ; FIXED DISK INTERRUPT ROUTINE :

5334 <1> ; :

5335 <1> ;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

5336 <1>

5337 <1> ;int\_77h:

5338 <1> HD1\_INT:

5339 00004B54 6650 <1> PUSH AX

5340 <1> ; Check if that is a spurious IRQ (from slave PIC)

5341 <1> ; 25/02/2015 (source: http://wiki.osdev.org/8259\_PIC)

5342 00004B56 B00B <1> mov al, 0Bh ; In-Service Register

5343 00004B58 E6A0 <1> out 0A0h, al

5344 00004B5A EB00 <1> jmp short $+2

5345 00004B5C EB00 <1> jmp short $+2

5346 00004B5E E4A0 <1> in al, 0A0h

5347 00004B60 2480 <1> and al, 80h ; bit 7 (is it real IRQ 15 or fake?)

5348 00004B62 74ED <1> jz short irq15\_iret ; Fake (spurious)IRQ, do not send EOI)

5349 <1> ;

5350 00004B64 1E <1> PUSH DS

5351 <1> ;CALL DDS

5352 <1> ; 21/02/2015 (32 bit, 386 pm modification)

5353 00004B65 66B81000 <1> mov ax, KDATA

5354 00004B69 8ED8 <1> mov ds, ax

5355 <1> ;

5356 <1> ;;MOV @HF\_INT\_FLAG,0FFH ; ALL DONE

5357 <1> ;or byte [CS:HF\_INT\_FLAG],0C0h

5358 00004B6B 800D[CB580100]C0 <1> or byte [HF\_INT\_FLAG], 0C0h

5359 <1> ;

5360 00004B72 6652 <1> push dx

5361 00004B74 66BA7701 <1> mov dx, HDC2\_BASEPORT+7 ; Status Register (177h)

5362 <1> ; Clear Controller (Award BIOS 1999)

5363 00004B78 EBC9 <1> jmp short Clear\_IRQ1415

5364 <1>

5365 <1>

5366 <1> ;%include 'diskdata.inc' ; 11/03/2015

5367 <1> ;%include 'diskbss.inc' ; 11/03/2015

5368 <1>

5369 <1>

5370 <1> ;////////////////////////////////////////////////////////////////////

5371 <1> ;; END OF DISK I/O SYTEM ///

2159 %include 'memory.s' ; 09/03/2015

1 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2 <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0 - memory.s

3 <1> ; ----------------------------------------------------------------------------

4 <1> ; Last Update: 22/07/2017

5 <1> ; ----------------------------------------------------------------------------

6 <1> ; Beginning: 24/01/2016

7 <1> ; ----------------------------------------------------------------------------

8 <1> ; Assembler: NASM version 2.11 (trdos386.s)

9 <1> ; ----------------------------------------------------------------------------

10 <1> ; Turkish Rational DOS

11 <1> ; Operating System Project v2.0 by ERDOGAN TAN (Beginning: 04/01/2016)

12 <1> ;

13 <1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan

14 <1> ; memory.inc (18/10/2015)

15 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

16 <1>

17 <1> ; MEMORY.ASM - Retro UNIX 386 v1 MEMORY MANAGEMENT FUNCTIONS (PROCEDURES)

18 <1> ; Retro UNIX 386 v1 Kernel (unix386.s, v0.2.0.14) - MEMORY.INC

19 <1> ; Last Modification: 18/10/2015

20 <1>

21 <1> ; ///////// MEMORY MANAGEMENT FUNCTIONS (PROCEDURES) ///////////////

22 <1>

23 <1> ;;04/11/2014 (unix386.s)

24 <1> ;PDE\_A\_PRESENT equ 1 ; Present flag for PDE

25 <1> ;PDE\_A\_WRITE equ 2 ; Writable (write permission) flag

26 <1> ;PDE\_A\_USER equ 4 ; User (non-system/kernel) page flag

27 <1> ;;

28 <1> ;PTE\_A\_PRESENT equ 1 ; Present flag for PTE (bit 0)

29 <1> ;PTE\_A\_WRITE equ 2 ; Writable (write permission) flag (bit 1)

30 <1> ;PTE\_A\_USER equ 4 ; User (non-system/kernel) page flag (bit 2)

31 <1> ;PTE\_A\_ACCESS equ 32 ; Accessed flag (bit 5) ; 09/03/2015

32 <1>

33 <1> ; 27/04/2015

34 <1> ; 09/03/2015

35 <1> PAGE\_SIZE equ 4096 ; page size in bytes

36 <1> PAGE\_SHIFT equ 12 ; page table shift count

37 <1> PAGE\_D\_SHIFT equ 22 ; 12 + 10 ; page directory shift count

38 <1> PAGE\_OFF equ 0FFFh ; 12 bit byte offset in page frame

39 <1> PTE\_MASK equ 03FFh ; page table entry mask

40 <1> PTE\_DUPLICATED equ 200h ; duplicated page sign (AVL bit 0)

41 <1> PDE\_A\_CLEAR equ 0F000h ; to clear PDE attribute bits

42 <1> PTE\_A\_CLEAR equ 0F000h ; to clear PTE attribute bits

43 <1> LOGIC\_SECT\_SIZE equ 512 ; logical sector size

44 <1> ERR\_MAJOR\_PF equ 0E0h ; major error: page fault

45 <1> ERR\_MINOR\_IM equ 4 ;15/10/2016 (1->4); insufficient (out of) memory

46 <1> ERR\_MINOR\_PV equ 6 ;15/10/2016 (1->4); protection violation

47 <1> SWP\_DISK\_READ\_ERR equ 40

48 <1> SWP\_DISK\_NOT\_PRESENT\_ERR equ 41

49 <1> SWP\_SECTOR\_NOT\_PRESENT\_ERR equ 42

50 <1> SWP\_NO\_FREE\_SPACE\_ERR equ 43

51 <1> SWP\_DISK\_WRITE\_ERR equ 44

52 <1> SWP\_NO\_PAGE\_TO\_SWAP\_ERR equ 45

53 <1> PTE\_A\_ACCESS\_BIT equ 5 ; Bit 5 (accessed flag)

54 <1> SECTOR\_SHIFT equ 3 ; sector shift (to convert page block number)

55 <1> ; 12/07/2016

56 <1> PTE\_SHARED equ 400h ; AVL bit 1, direct memory access bit

57 <1> ; (Indicates that the page is not allocated

58 <1> ; for the process, it is a shared or system

59 <1> ; page, it must not be deallocated!)

60 <1> ;

61 <1> ;; Retro Unix 386 v1 - paging method/principles

62 <1> ;;

63 <1> ;; 10/10/2014

64 <1> ;; RETRO UNIX 386 v1 - PAGING METHOD/PRINCIPLES

65 <1> ;;

66 <1> ;; KERNEL PAGE MAP: 1 to 1 physical memory page map

67 <1> ;; (virtual address = physical address)

68 <1> ;; KERNEL PAGE TABLES:

69 <1> ;; Kernel page directory and all page tables are

70 <1> ;; on memory as initialized, as equal to physical memory

71 <1> ;; layout. Kernel pages can/must not be swapped out/in.

72 <1> ;;

73 <1> ;; what for: User pages may be swapped out, when accessing

74 <1> ;; a page in kernel/system mode, if it would be swapped out,

75 <1> ;; kernel would have to swap it in! But it is also may be

76 <1> ;; in use by a user process. (In system/kernel mode

77 <1> ;; kernel can access all memory pages even if they are

78 <1> ;; reserved/allocated for user processes. Swap out/in would

79 <1> ;; cause conflicts.)

80 <1> ;;

81 <1> ;; As result of these conditions,

82 <1> ;; all kernel pages must be initialized as equal to

83 <1> ;; physical layout for preventing page faults.

84 <1> ;; Also, calling "allocate page" procedure after

85 <1> ;; a page fault can cause another page fault (double fault)

86 <1> ;; if all kernel page tables would not be initialized.

87 <1> ;;

88 <1> ;; [first\_page] = Beginning of users space, as offset to

89 <1> ;; memory allocation table. (double word aligned)

90 <1> ;;

91 <1> ;; [next\_page] = first/next free space to be searched

92 <1> ;; as offset to memory allocation table. (dw aligned)

93 <1> ;;

94 <1> ;; [last\_page] = End of memory (users space), as offset

95 <1> ;; to memory allocation table. (double word aligned)

96 <1> ;;

97 <1> ;; USER PAGE TABLES:

98 <1> ;; Demand paging (& 'copy on write' allocation method) ...

99 <1> ;; 'ready only' marked copies of the

100 <1> ;; parent process's page table entries (for

101 <1> ;; same physical memory).

102 <1> ;; (A page will be copied to a new page after

103 <1> ;; if it causes R/W page fault.)

104 <1> ;;

105 <1> ;; Every user process has own (different)

106 <1> ;; page directory and page tables.

107 <1> ;;

108 <1> ;; Code starts at virtual address 0, always.

109 <1> ;; (Initial value of EIP is 0 in user mode.)

110 <1> ;; (Programs can be written/developed as simple

111 <1> ;; flat memory programs.)

112 <1> ;;

113 <1> ;; MEMORY ALLOCATION STRATEGY:

114 <1> ;; Memory page will be allocated by kernel only

115 <1> ;; (in kernel/system mode only).

116 <1> ;; \* After a

117 <1> ;; - 'not present' page fault

118 <1> ;; - 'writing attempt on read only page' page fault

119 <1> ;; \* For loading (opening, reading) a file or disk/drive

120 <1> ;; \* As responce to 'allocate additional memory blocks'

121 <1> ;; request by running process.

122 <1> ;; \* While creating a process, allocating a new buffer,

123 <1> ;; new page tables etc.

124 <1> ;;

125 <1> ;; At first,

126 <1> ;; - 'allocate page' procedure will be called;

127 <1> ;, if it will return with a valid (>0) physical address

128 <1> ;; (that means the relevant M.A.T. bit has been RESET)

129 <1> ;; relevant memory page/block will be cleared (zeroed).

130 <1> ;; - 'allocate page' will be called for allocating page

131 <1> ;; directory, page table and running space (data/code).

132 <1> ;; - every successful 'allocate page' call will decrease

133 <1> ;; 'free\_pages' count (pointer).

134 <1> ;; - 'out of (insufficient) memory error' will be returned

135 <1> ;; if 'free\_pages' points to a ZERO.

136 <1> ;; - swapping out and swapping in (if it is not a new page)

137 <1> ;; procedures will be called as responce to 'out of memory'

138 <1> ;; error except errors caused by attribute conflicts.

139 <1> ;; (swapper functions)

140 <1> ;;

141 <1> ;; At second,

142 <1> ;; - page directory entry will be updated then page table

143 <1> ;; entry will be updated.

144 <1> ;;

145 <1> ;; MEMORY ALLOCATION TABLE FORMAT:

146 <1> ;; - M.A.T. has a size according to available memory as

147 <1> ;; follows:

148 <1> ;; - 1 (allocation) bit per 1 page (4096 bytes)

149 <1> ;; - a bit with value of 0 means allocated page

150 <1> ;; - a bit with value of 1 means a free page

151 <1> ;, - 'free\_pages' pointer holds count of free pages

152 <1> ;; depending on M.A.T.

153 <1> ;; (NOTE: Free page count will not be checked

154 <1> ;; again -on M.A.T.- after initialization.

155 <1> ;; Kernel will trust on initial count.)

156 <1> ;, - 'free\_pages' count will be decreased by allocation

157 <1> ;; and it will be increased by deallocation procedures.

158 <1> ;;

159 <1> ;; - Available memory will be calculated during

160 <1> ;; the kernel's initialization stage (in real mode).

161 <1> ;; Memory allocation table and kernel page tables

162 <1> ;; will be formatted/sized as result of available

163 <1> ;; memory calculation before paging is enabled.

164 <1> ;;

165 <1> ;; For 4GB Available/Present Memory: (max. possible memory size)

166 <1> ;; - Memory Allocation Table size will be 128 KB.

167 <1> ;; - Memory allocation for kernel page directory size

168 <1> ;; is always 4 KB. (in addition to total allocation size

169 <1> ;; for page tables)

170 <1> ;; - Memory allocation for kernel page tables (1024 tables)

171 <1> ;; is 4 MB (1024\*4\*1024 bytes).

172 <1> ;; - User (available) space will be started

173 <1> ;; at 6th MB of the memory (after 1MB+4MB).

174 <1> ;; - The first 640 KB is for kernel's itself plus

175 <1> ;; memory allocation table and kernel's page directory

176 <1> ;; (D0000h-EFFFFh may be used as kernel space...)

177 <1> ;; - B0000h to B7FFFh address space (32 KB) will be used

178 <1> ;; for buffers.

179 <1> ;; - ROMBIOS, VIDEO BUFFER and VIDEO ROM space are reserved.

180 <1> ;, (A0000h-AFFFFh, C0000h-CFFFFh, F0000h-FFFFFh)

181 <1> ;; - Kernel page tables start at 100000h (2nd MB)

182 <1> ;;

183 <1> ;; For 1GB Available Memory:

184 <1> ;; - Memory Allocation Table size will be 32 KB.

185 <1> ;; - Memory allocation for kernel page directory size

186 <1> ;; is always 4 KB. (in addition to total allocation size

187 <1> ;; for page tables)

188 <1> ;; - Memory allocation for kernel page tables (256 tables)

189 <1> ;; is 1 MB (256\*4\*1024 bytes).

190 <1> ;; - User (available) space will be started

191 <1> ;; at 3th MB of the memory (after 1MB+1MB).

192 <1> ;; - The first 640 KB is for kernel's itself plus

193 <1> ;; memory allocation table and kernel's page directory

194 <1> ;; (D0000h-EFFFFh may be used as kernel space...)

195 <1> ;; - B0000h to B7FFFh address space (32 KB) will be used

196 <1> ;; for buffers.

197 <1> ;; - ROMBIOS, VIDEO BUFFER and VIDEO ROM space are reserved.

198 <1> ;, (A0000h-AFFFFh, C0000h-CFFFFh, F0000h-FFFFFh)

199 <1> ;; - Kernel page tables start at 100000h (2nd MB).

200 <1> ;;

201 <1> ;;

202 <1>

203 <1>

204 <1> ;;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

205 <1> ;;

206 <1> ;; RETRO UNIX 386 v1 - Paging (Method for Copy On Write paging principle)

207 <1> ;; DEMAND PAGING - PARENT&CHILD PAGE TABLE DUPLICATION PRINCIPLES (23/04/2015)

208 <1>

209 <1> ;; Main factor: "sys fork" system call

210 <1> ;;

211 <1> ;; FORK

212 <1> ;; |----> parent - duplicated PTEs, read only pages

213 <1> ;; writable pages ---->|

214 <1> ;; |----> child - duplicated PTEs, read only pages

215 <1> ;;

216 <1> ;; AVL bit (0) of Page Table Entry is used as duplication sign

217 <1> ;;

218 <1> ;; AVL Bit 0 [PTE Bit 9] = 'Duplicated PTE belongs to child' sign/flag (if it is set)

219 <1> ;; Note: Dirty bit (PTE bit 6) may be used instead of AVL bit 0 (PTE bit 9)

220 <1> ;; -while R/W bit is 0-.

221 <1> ;;

222 <1> ;; Duplicate page tables with writable pages (the 1st sys fork in the process):

223 <1> ;; # Parent's Page Table Entries are updated to point same pages as read only,

224 <1> ;; as duplicated PTE bit -AVL bit 0, PTE bit 9- are reset/clear.

225 <1> ;; # Then Parent's Page Table is copied to Child's Page Table.

226 <1> ;; # Child's Page Table Entries are updated as duplicated child bit

227 <1> ;; -AVL bit 0, PTE bit 9- is set.

228 <1> ;;

229 <1> ;; Duplicate page tables with read only pages (several sys fork system calls):

230 <1> ;; # Parent's read only pages are copied to new child pages.

231 <1> ;; Parent's PTE attributes are not changed.

232 <1> ;; (Because, there is another parent-child fork before this fork! We must not

233 <1> ;; destroy/mix previous fork result).

234 <1> ;; # Child's Page Table Entries (which are corresponding to Parent's

235 <1> ;; read only pages) are set as writable (while duplicated PTE bit is clear).

236 <1> ;; # Parent's PTEs with writable page attribute are updated to point same pages

237 <1> ;; as read only, (while) duplicated PTE bit is reset (clear).

238 <1> ;; # Parent's Page Table Entries (with writable page attribute) are duplicated

239 <1> ;; as Child's Page Table Entries without copying actual page.

240 <1> ;; # Child 's Page Table Entries (which are corresponding to Parent's writable

241 <1> ;; pages) are updated as duplicated PTE bit (AVL bit 0, PTE bit 9- is set.

242 <1> ;;

243 <1> ;; !? WHAT FOR (duplication after duplication):

244 <1> ;; In UNIX method for sys fork (a typical 'fork' application in /etc/init)

245 <1> ;; program/executable code continues from specified location as child process,

246 <1> ;; returns back previous code location as parent process, every child after

247 <1> ;; every sys fork uses last image of code and data just prior the fork.

248 <1> ;; Even if the parent code changes data, the child will not see the changed data

249 <1> ;; after the fork. In Retro UNIX 8086 v1, parent's process segment (32KB)

250 <1> ;; was copied to child's process segment (all of code and data) according to

251 <1> ;; original UNIX v1 which copies all of parent process code and data -core-

252 <1> ;; to child space -core- but swaps that core image -of child- on to disk.

253 <1> ;; If I (Erdogan Tan) would use a method of to copy parent's core

254 <1> ;; (complete running image of parent process) to the child process;

255 <1> ;; for big sizes, i would force Retro UNIX 386 v1 to spend many memory pages

256 <1> ;; and times only for a sys fork. (It would excessive reservation for sys fork,

257 <1> ;; because sys fork usually is prior to sys exec; sys exec always establishes

258 <1> ;; a new/fresh core -running space-, by clearing all code/data content).

259 <1> ;; 'Read Only' page flag ensures page fault handler is needed only for a few write

260 <1> ;; attempts between sys fork and sys exec, not more... (I say so by thinking

261 <1> ;; of "/etc/init" content, specially.) sys exec will clear page tables and

262 <1> ;; new/fresh pages will be used to load and run new executable/program.

263 <1> ;; That is what for i have preferred "copy on write", "duplication" method

264 <1> ;; for sharing same read only pages between parent and child processes.

265 <1> ;; That is a pitty i have to use new private flag (AVL bit 0, "duplicated PTE

266 <1> ;; belongs to child" sign) for cooperation on duplicated pages between a parent

267 <1> ;; and it's child processes; otherwise parent process would destroy data belongs

268 <1> ;; to its child or vice versa; or some pages would remain unclaimed

269 <1> ;; -deallocation problem-.

270 <1> ;; Note: to prevent conflicts, read only pages must not be swapped out...

271 <1> ;;

272 <1> ;; WHEN PARENT TRIES TO WRITE IT'S READ ONLY (DUPLICATED) PAGE:

273 <1> ;; # Page fault handler will do those:

274 <1> ;; - 'Duplicated PTE' flag (PTE bit 9) is checked (on the failed PTE).

275 <1> ;; - If it is reset/clear, there is a child uses same page.

276 <1> ;; - Parent's read only page -previous page- is copied to a new writable page.

277 <1> ;; - Parent's PTE is updated as writable page, as unique page (AVL=0)

278 <1> ;; - (Page fault handler whill check this PTE later, if child process causes to

279 <1> ;; page fault due to write attempt on read only page. Of course, the previous

280 <1> ;; read only page will be converted to writable and unique page which belongs

281 <1> ;; to child process.)

282 <1> ;; WHEN CHILD TRIES TO WRITE IT'S READ ONLY (DUPLICATED) PAGE:

283 <1> ;; # Page fault handler will do those:

284 <1> ;; - 'Duplicated PTE' flag (PTE bit 9) is checked (on the failed PTE).

285 <1> ;; - If it is set, there is a parent uses -or was using- same page.

286 <1> ;; - Same PTE address within parent's page table is checked if it has same page

287 <1> ;; address or not.

288 <1> ;; - If parent's PTE has same address, child will continue with a new writable page.

289 <1> ;; Parent's PTE will point to same (previous) page as writable, unique (AVL=0).

290 <1> ;; - If parent's PTE has different address, child will continue with it's

291 <1> ;; own/same page but read only flag (0) will be changed to writable flag (1) and

292 <1> ;; 'duplicated PTE (belongs to child)' flag/sign will be cleared/reset.

293 <1> ;;

294 <1> ;; NOTE: When a child process is terminated, read only flags of parent's page tables

295 <1> ;; will be set as writable (and unique) in case of child process was using

296 <1> ;; same pages with duplicated child PTE sign... Depending on sys fork and

297 <1> ;; duplication method details, it is not possible multiple child processes

298 <1> ;; were using same page with duplicated PTEs.

299 <1> ;;

300 <1> ;;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

301 <1>

302 <1> ;; 08/10/2014

303 <1> ;; 11/09/2014 - Retro UNIX 386 v1 PAGING (further) draft

304 <1> ;; by Erdogan Tan (Based on KolibriOS 'memory.inc')

305 <1>

306 <1> ;; 'allocate\_page' code is derived and modified from KolibriOS

307 <1> ;; 'alloc\_page' procedure in 'memory.inc'

308 <1> ;; (25/08/2014, Revision: 5057) file

309 <1> ;; by KolibriOS Team (2004-2012)

310 <1>

311 <1> allocate\_page:

312 <1> ; 01/07/2015

313 <1> ; 05/05/2015

314 <1> ; 30/04/2015

315 <1> ; 16/10/2014

316 <1> ; 08/10/2014

317 <1> ; 09/09/2014 (Retro UNIX 386 v1 - beginning)

318 <1> ;

319 <1> ; INPUT -> none

320 <1> ;

321 <1> ; OUTPUT ->

322 <1> ; EAX = PHYSICAL (real/flat) ADDRESS OF THE ALLOCATED PAGE

323 <1> ; (corresponding MEMORY ALLOCATION TABLE bit is RESET)

324 <1> ;

325 <1> ; CF = 1 and EAX = 0

326 <1> ; if there is not a free page to be allocated

327 <1> ;

328 <1> ; Modified Registers -> none (except EAX)

329 <1> ;

330 00004B7A A1[40580100] <1> mov eax, [free\_pages]

331 00004B7F 21C0 <1> and eax, eax

332 00004B81 7438 <1> jz short out\_of\_memory

333 <1> ;

334 00004B83 53 <1> push ebx

335 00004B84 51 <1> push ecx

336 <1> ;

337 00004B85 BB00001000 <1> mov ebx, MEM\_ALLOC\_TBL ; Memory Allocation Table offset

338 00004B8A 89D9 <1> mov ecx, ebx

339 <1> ; NOTE: 32 (first\_page) is initial

340 <1> ; value of [next\_page].

341 <1> ; It points to the first available

342 <1> ; page block for users (ring 3) ...

343 <1> ; (MAT offset 32 = 1024/32)

344 <1> ; (at the of the first 4 MB)

345 00004B8C 031D[44580100] <1> add ebx, [next\_page] ; Free page searching starts from here

346 <1> ; next\_free\_page >> 5

347 00004B92 030D[48580100] <1> add ecx, [last\_page] ; Free page searching ends here

348 <1> ; (total\_pages - 1) >> 5

349 <1> al\_p\_scan:

350 00004B98 39CB <1> cmp ebx, ecx

351 00004B9A 770A <1> ja short al\_p\_notfound

352 <1> ;

353 <1> ; 01/07/2015

354 <1> ; AMD64 Architecture Programmer’s Manual

355 <1> ; Volume 3:

356 <1> ; General-Purpose and System Instructions

357 <1> ;

358 <1> ; BSF - Bit Scan Forward

359 <1> ;

360 <1> ; Searches the value in a register or a memory location

361 <1> ; (second operand) for the least-significant set bit.

362 <1> ; If a set bit is found, the instruction clears the zero flag (ZF)

363 <1> ; and stores the index of the least-significant set bit in a destination

364 <1> ; register (first operand). If the second operand contains 0,

365 <1> ; the instruction sets ZF to 1 and does not change the contents of the

366 <1> ; destination register. The bit index is an unsigned offset from bit 0

367 <1> ; of the searched value

368 <1> ;

369 00004B9C 0FBC03 <1> bsf eax, [ebx] ; Scans source operand for first bit set (1).

370 <1> ; Clear ZF if a bit is found set (1) and

371 <1> ; loads the destination with an index to

372 <1> ; first set bit. (0 -> 31)

373 <1> ; Sets ZF to 1 if no bits are found set.

374 00004B9F 7525 <1> jnz short al\_p\_found ; ZF = 0 -> a free page has been found

375 <1> ;

376 <1> ; NOTE: a Memory Allocation Table bit

377 <1> ; with value of 1 means

378 <1> ; the corresponding page is free

379 <1> ; (Retro UNIX 386 v1 feature only!)

380 00004BA1 83C304 <1> add ebx, 4

381 <1> ; We return back for searching next page block

382 <1> ; NOTE: [free\_pages] is not ZERO; so,

383 <1> ; we always will find at least 1 free page here.

384 00004BA4 EBF2 <1> jmp short al\_p\_scan

385 <1> ;

386 <1> al\_p\_notfound:

387 00004BA6 81E900001000 <1> sub ecx, MEM\_ALLOC\_TBL

388 00004BAC 890D[44580100] <1> mov [next\_page], ecx ; next/first free page = last page

389 <1> ; (deallocate\_page procedure will change it)

390 00004BB2 31C0 <1> xor eax, eax

391 00004BB4 A3[40580100] <1> mov [free\_pages], eax ; 0

392 00004BB9 59 <1> pop ecx

393 00004BBA 5B <1> pop ebx

394 <1> ;

395 <1> out\_of\_memory:

396 00004BBB E85B040000 <1> call swap\_out

397 00004BC0 7325 <1> jnc short al\_p\_ok ; [free\_pages] = 0, re-allocation by swap\_out

398 <1> ;

399 00004BC2 29C0 <1> sub eax, eax ; 0

400 00004BC4 F9 <1> stc

401 00004BC5 C3 <1> retn

402 <1>

403 <1> al\_p\_found:

404 00004BC6 89D9 <1> mov ecx, ebx

405 00004BC8 81E900001000 <1> sub ecx, MEM\_ALLOC\_TBL

406 00004BCE 890D[44580100] <1> mov [next\_page], ecx ; Set first free page searching start

407 <1> ; address/offset (to the next)

408 00004BD4 FF0D[40580100] <1> dec dword [free\_pages] ; 1 page has been allocated (X = X-1)

409 <1> ;

410 00004BDA 0FB303 <1> btr [ebx], eax ; The destination bit indexed by the source value

411 <1> ; is copied into the Carry Flag and then cleared

412 <1> ; in the destination.

413 <1> ;

414 <1> ; Reset the bit which is corresponding to the

415 <1> ; (just) allocated page.

416 <1> ; 01/07/2015 (4\*8 = 32, 1 allocation byte = 8 pages)

417 00004BDD C1E103 <1> shl ecx, 3 ; (page block offset \* 32) + page index

418 00004BE0 01C8 <1> add eax, ecx ; = page number

419 00004BE2 C1E00C <1> shl eax, 12 ; physical address of the page (flat/real value)

420 <1> ; EAX = physical address of memory page

421 <1> ;

422 <1> ; NOTE: The relevant page directory and page table entry will be updated

423 <1> ; according to this EAX value...

424 00004BE5 59 <1> pop ecx

425 00004BE6 5B <1> pop ebx

426 <1> al\_p\_ok:

427 00004BE7 C3 <1> retn

428 <1>

429 <1>

430 <1> make\_page\_dir:

431 <1> ; 18/04/2015

432 <1> ; 12/04/2015

433 <1> ; 23/10/2014

434 <1> ; 16/10/2014

435 <1> ; 09/10/2014 ; (Retro UNIX 386 v1 - beginning)

436 <1> ;

437 <1> ; INPUT ->

438 <1> ; none

439 <1> ; OUTPUT ->

440 <1> ; (EAX = 0)

441 <1> ; cf = 1 -> insufficient (out of) memory error

442 <1> ; cf = 0 ->

443 <1> ; u.pgdir = page directory (physical) address of the current

444 <1> ; process/user.

445 <1> ;

446 <1> ; Modified Registers -> EAX

447 <1> ;

448 00004BE8 E88DFFFFFF <1> call allocate\_page

449 00004BED 7216 <1> jc short mkpd\_error

450 <1> ;

451 00004BEF A3[B8030300] <1> mov [u.pgdir], eax ; Page dir address for current user/process

452 <1> ; (Physical address)

453 <1> clear\_page:

454 <1> ; 18/04/2015

455 <1> ; 09/10/2014 ; (Retro UNIX 386 v1 - beginning)

456 <1> ;

457 <1> ; INPUT ->

458 <1> ; EAX = physical address of the page

459 <1> ; OUTPUT ->

460 <1> ; all bytes of the page will be cleared

461 <1> ;

462 <1> ; Modified Registers -> none

463 <1> ;

464 00004BF4 57 <1> push edi

465 00004BF5 51 <1> push ecx

466 00004BF6 50 <1> push eax

467 00004BF7 B900040000 <1> mov ecx, PAGE\_SIZE / 4

468 00004BFC 89C7 <1> mov edi, eax

469 00004BFE 31C0 <1> xor eax, eax

470 00004C00 F3AB <1> rep stosd

471 00004C02 58 <1> pop eax

472 00004C03 59 <1> pop ecx

473 00004C04 5F <1> pop edi

474 <1> mkpd\_error:

475 <1> mkpt\_error:

476 00004C05 C3 <1> retn

477 <1>

478 <1> make\_page\_table:

479 <1> ; 23/06/2015

480 <1> ; 18/04/2015

481 <1> ; 12/04/2015

482 <1> ; 16/10/2014

483 <1> ; 09/10/2014 ; (Retro UNIX 386 v1 - beginning)

484 <1> ;

485 <1> ; INPUT ->

486 <1> ; EBX = virtual (linear) address

487 <1> ; ECX = page table attributes (lower 12 bits)

488 <1> ; (higher 20 bits must be ZERO)

489 <1> ; (bit 0 must be 1)

490 <1> ; u.pgdir = page directory (physical) address

491 <1> ; OUTPUT ->

492 <1> ; EDX = Page directory entry address

493 <1> ; EAX = Page table address

494 <1> ; cf = 1 -> insufficient (out of) memory error

495 <1> ; cf = 0 -> page table address in the PDE (EDX)

496 <1> ;

497 <1> ; Modified Registers -> EAX, EDX

498 <1> ;

499 00004C06 E86FFFFFFF <1> call allocate\_page

500 00004C0B 72F8 <1> jc short mkpt\_error

501 00004C0D E811000000 <1> call set\_pde

502 00004C12 EBE0 <1> jmp short clear\_page

503 <1>

504 <1> make\_page:

505 <1> ; 24/07/2015

506 <1> ; 23/06/2015 ; (Retro UNIX 386 v1 - beginning)

507 <1> ;

508 <1> ; INPUT ->

509 <1> ; EBX = virtual (linear) address

510 <1> ; ECX = page attributes (lower 12 bits)

511 <1> ; (higher 20 bits must be ZERO)

512 <1> ; (bit 0 must be 1)

513 <1> ; u.pgdir = page directory (physical) address

514 <1> ; OUTPUT ->

515 <1> ; EBX = Virtual address

516 <1> ; (EDX = PTE value)

517 <1> ; EAX = Physical address

518 <1> ; cf = 1 -> insufficient (out of) memory error

519 <1> ;

520 <1> ; Modified Registers -> EAX, EDX

521 <1> ;

522 00004C14 E861FFFFFF <1> call allocate\_page

523 00004C19 7207 <1> jc short mkp\_err

524 00004C1B E821000000 <1> call set\_pte

525 00004C20 73D2 <1> jnc short clear\_page ; 18/04/2015

526 <1> mkp\_err:

527 00004C22 C3 <1> retn

528 <1>

529 <1>

530 <1> set\_pde: ; Set page directory entry (PDE)

531 <1> ; 20/07/2015

532 <1> ; 18/04/2015

533 <1> ; 12/04/2015

534 <1> ; 23/10/2014

535 <1> ; 10/10/2014 ; (Retro UNIX 386 v1 - beginning)

536 <1> ;

537 <1> ; INPUT ->

538 <1> ; EAX = physical address

539 <1> ; (use present value if EAX = 0)

540 <1> ; EBX = virtual (linear) address

541 <1> ; ECX = page table attributes (lower 12 bits)

542 <1> ; (higher 20 bits must be ZERO)

543 <1> ; (bit 0 must be 1)

544 <1> ; u.pgdir = page directory (physical) address

545 <1> ; OUTPUT ->

546 <1> ; EDX = PDE address

547 <1> ; EAX = page table address (physical)

548 <1> ; ;(CF=1 -> Invalid page address)

549 <1> ;

550 <1> ; Modified Registers -> EDX

551 <1> ;

552 00004C23 89DA <1> mov edx, ebx

553 00004C25 C1EA16 <1> shr edx, PAGE\_D\_SHIFT ; 22

554 00004C28 C1E202 <1> shl edx, 2 ; offset to page directory (1024\*4)

555 00004C2B 0315[B8030300] <1> add edx, [u.pgdir]

556 <1> ;

557 00004C31 21C0 <1> and eax, eax

558 00004C33 7506 <1> jnz short spde\_1

559 <1> ;

560 00004C35 8B02 <1> mov eax, [edx] ; old PDE value

561 <1> ;test al, 1

562 <1> ;jz short spde\_2

563 00004C37 662500F0 <1> and ax, PDE\_A\_CLEAR ; 0F000h ; clear lower 12 bits

564 <1> spde\_1:

565 <1> ;and cx, 0FFFh

566 00004C3B 8902 <1> mov [edx], eax

567 00004C3D 66090A <1> or [edx], cx

568 00004C40 C3 <1> retn

569 <1> ;spde\_2: ; error

570 <1> ; stc

571 <1> ; retn

572 <1>

573 <1> set\_pte: ; Set page table entry (PTE)

574 <1> ; 24/07/2015

575 <1> ; 20/07/2015

576 <1> ; 23/06/2015

577 <1> ; 18/04/2015

578 <1> ; 12/04/2015

579 <1> ; 10/10/2014 ; (Retro UNIX 386 v1 - beginning)

580 <1> ;

581 <1> ; INPUT ->

582 <1> ; EAX = physical page address

583 <1> ; (use present value if EAX = 0)

584 <1> ; EBX = virtual (linear) address

585 <1> ; ECX = page attributes (lower 12 bits)

586 <1> ; (higher 20 bits must be ZERO)

587 <1> ; (bit 0 must be 1)

588 <1> ; u.pgdir = page directory (physical) address

589 <1> ; OUTPUT ->

590 <1> ; EAX = physical page address

591 <1> ; (EDX = PTE value)

592 <1> ; EBX = virtual address

593 <1> ;

594 <1> ; CF = 1 -> error

595 <1> ;

596 <1> ; Modified Registers -> EAX, EDX

597 <1> ;

598 00004C41 50 <1> push eax

599 00004C42 A1[B8030300] <1> mov eax, [u.pgdir] ; 20/07/2015

600 00004C47 E837000000 <1> call get\_pde

601 <1> ; EDX = PDE address

602 <1> ; EAX = PDE value

603 00004C4C 5A <1> pop edx ; physical page address

604 00004C4D 722A <1> jc short spte\_err ; PDE not present

605 <1> ;

606 00004C4F 53 <1> push ebx ; 24/07/2015

607 00004C50 662500F0 <1> and ax, PDE\_A\_CLEAR ; 0F000h ; clear lower 12 bits

608 <1> ; EDX = PT address (physical)

609 00004C54 C1EB0C <1> shr ebx, PAGE\_SHIFT ; 12

610 00004C57 81E3FF030000 <1> and ebx, PTE\_MASK ; 03FFh

611 <1> ; clear higher 10 bits (PD bits)

612 00004C5D C1E302 <1> shl ebx, 2 ; offset to page table (1024\*4)

613 00004C60 01C3 <1> add ebx, eax

614 <1> ;

615 00004C62 8B03 <1> mov eax, [ebx] ; Old PTE value

616 00004C64 A801 <1> test al, 1

617 00004C66 740C <1> jz short spte\_0

618 00004C68 09D2 <1> or edx, edx

619 00004C6A 750F <1> jnz short spte\_1

620 00004C6C 662500F0 <1> and ax, PTE\_A\_CLEAR ; 0F000h ; clear lower 12 bits

621 00004C70 89C2 <1> mov edx, eax

622 00004C72 EB09 <1> jmp short spte\_2

623 <1> spte\_0:

624 <1> ; If this PTE contains a swap (disk) address,

625 <1> ; it can be updated by using 'swap\_in' procedure

626 <1> ; only!

627 00004C74 21C0 <1> and eax, eax

628 00004C76 7403 <1> jz short spte\_1

629 <1> ; 24/07/2015

630 <1> ; swapped page ! (on disk)

631 00004C78 5B <1> pop ebx

632 <1> spte\_err:

633 00004C79 F9 <1> stc

634 00004C7A C3 <1> retn

635 <1> spte\_1:

636 00004C7B 89D0 <1> mov eax, edx

637 <1> spte\_2:

638 00004C7D 09CA <1> or edx, ecx

639 <1> ; 23/06/2015

640 00004C7F 8913 <1> mov [ebx], edx ; PTE value in EDX

641 <1> ; 24/07/2015

642 00004C81 5B <1> pop ebx

643 00004C82 C3 <1> retn

644 <1>

645 <1> get\_pde: ; Get present value of the relevant PDE

646 <1> ; 20/07/2015

647 <1> ; 18/04/2015

648 <1> ; 12/04/2015

649 <1> ; 10/10/2014 ; (Retro UNIX 386 v1 - beginning)

650 <1> ;

651 <1> ; INPUT ->

652 <1> ; EBX = virtual (linear) address

653 <1> ; EAX = page directory (physical) address

654 <1> ; OUTPUT ->

655 <1> ; EDX = Page directory entry address

656 <1> ; EAX = Page directory entry value

657 <1> ; CF = 1 -> PDE not present or invalid ?

658 <1> ; Modified Registers -> EDX, EAX

659 <1> ;

660 00004C83 89DA <1> mov edx, ebx

661 00004C85 C1EA16 <1> shr edx, PAGE\_D\_SHIFT ; 22 (12+10)

662 00004C88 C1E202 <1> shl edx, 2 ; offset to page directory (1024\*4)

663 00004C8B 01C2 <1> add edx, eax ; page directory address (physical)

664 00004C8D 8B02 <1> mov eax, [edx]

665 00004C8F A801 <1> test al, PDE\_A\_PRESENT ; page table is present or not !

666 00004C91 751F <1> jnz short gpte\_retn

667 00004C93 F9 <1> stc

668 <1> gpde\_retn:

669 00004C94 C3 <1> retn

670 <1>

671 <1> get\_pte:

672 <1> ; Get present value of the relevant PTE

673 <1> ; 29/07/2015

674 <1> ; 20/07/2015

675 <1> ; 18/04/2015

676 <1> ; 12/04/2015

677 <1> ; 10/10/2014 ; (Retro UNIX 386 v1 - beginning)

678 <1> ;

679 <1> ; INPUT ->

680 <1> ; EBX = virtual (linear) address

681 <1> ; EAX = page directory (physical) address

682 <1> ; OUTPUT ->

683 <1> ; EDX = Page table entry address (if CF=0)

684 <1> ; Page directory entry address (if CF=1)

685 <1> ; (Bit 0 value is 0 if PT is not present)

686 <1> ; EAX = Page table entry value (page address)

687 <1> ; CF = 1 -> PDE not present or invalid ?

688 <1> ; Modified Registers -> EAX, EDX

689 <1> ;

690 00004C95 E8E9FFFFFF <1> call get\_pde

691 00004C9A 72F8 <1> jc short gpde\_retn ; page table is not present

692 <1> ;jnc short gpte\_1

693 <1> ;retn

694 <1> ;gpte\_1:

695 00004C9C 662500F0 <1> and ax, PDE\_A\_CLEAR ; 0F000h ; clear lower 12 bits

696 00004CA0 89DA <1> mov edx, ebx

697 00004CA2 C1EA0C <1> shr edx, PAGE\_SHIFT ; 12

698 00004CA5 81E2FF030000 <1> and edx, PTE\_MASK ; 03FFh

699 <1> ; clear higher 10 bits (PD bits)

700 00004CAB C1E202 <1> shl edx, 2 ; offset from start of page table (1024\*4)

701 00004CAE 01C2 <1> add edx, eax

702 00004CB0 8B02 <1> mov eax, [edx]

703 <1> gpte\_retn:

704 00004CB2 C3 <1> retn

705 <1>

706 <1> deallocate\_page\_dir:

707 <1> ; 15/09/2015

708 <1> ; 05/08/2015

709 <1> ; 30/04/2015

710 <1> ; 28/04/2015

711 <1> ; 17/10/2014

712 <1> ; 12/10/2014 (Retro UNIX 386 v1 - beginning)

713 <1> ;

714 <1> ; INPUT ->

715 <1> ; EAX = PHYSICAL ADDRESS OF THE PAGE DIRECTORY (CHILD)

716 <1> ; EBX = PHYSICAL ADDRESS OF THE PARENT'S PAGE DIRECTORY

717 <1> ; OUTPUT ->

718 <1> ; All of page tables in the page directory

719 <1> ; and page dir's itself will be deallocated

720 <1> ; except 'read only' duplicated pages (will be converted

721 <1> ; to writable pages).

722 <1> ;

723 <1> ; Modified Registers -> EAX

724 <1> ;

725 <1> ;

726 00004CB3 56 <1> push esi

727 00004CB4 51 <1> push ecx

728 00004CB5 50 <1> push eax

729 00004CB6 89C6 <1> mov esi, eax

730 00004CB8 31C9 <1> xor ecx, ecx

731 <1> ; The 1st PDE points to Kernel Page Table 0 (the 1st 4MB),

732 <1> ; it must not be deallocated

733 00004CBA 890E <1> mov [esi], ecx ; 0 ; clear PDE 0

734 <1> dapd\_0:

735 00004CBC AD <1> lodsd

736 00004CBD A801 <1> test al, PDE\_A\_PRESENT ; bit 0, present flag (must be 1)

737 00004CBF 7409 <1> jz short dapd\_1

738 00004CC1 662500F0 <1> and ax, PDE\_A\_CLEAR ; 0F000h ; clear lower 12 (attribute) bits

739 00004CC5 E812000000 <1> call deallocate\_page\_table

740 <1> dapd\_1:

741 00004CCA 41 <1> inc ecx ; page directory entry index

742 00004CCB 81F900040000 <1> cmp ecx, PAGE\_SIZE / 4 ; 1024

743 00004CD1 72E9 <1> jb short dapd\_0

744 <1> dapd\_2:

745 00004CD3 58 <1> pop eax

746 00004CD4 E87F000000 <1> call deallocate\_page ; deallocate the page dir's itself

747 00004CD9 59 <1> pop ecx

748 00004CDA 5E <1> pop esi

749 00004CDB C3 <1> retn

750 <1>

751 <1> deallocate\_page\_table:

752 <1> ; 12/07/2016

753 <1> ; 19/09/2015

754 <1> ; 15/09/2015

755 <1> ; 05/08/2015

756 <1> ; 30/04/2015

757 <1> ; 28/04/2015

758 <1> ; 24/10/2014

759 <1> ; 23/10/2014

760 <1> ; 12/10/2014 (Retro UNIX 386 v1 - beginning)

761 <1> ;

762 <1> ; INPUT ->

763 <1> ; EAX = PHYSICAL (real/flat) ADDRESS OF THE PAGE TABLE

764 <1> ; EBX = PHYSICAL ADDRESS OF THE PARENT'S PAGE DIRECTORY

765 <1> ; (ECX = page directory entry index)

766 <1> ; OUTPUT ->

767 <1> ; All of pages in the page table and page table's itself

768 <1> ; will be deallocated except 'read only' duplicated pages

769 <1> ; (will be converted to writable pages).

770 <1> ;

771 <1> ; Modified Registers -> EAX

772 <1> ;

773 00004CDC 56 <1> push esi

774 00004CDD 57 <1> push edi

775 00004CDE 52 <1> push edx

776 00004CDF 50 <1> push eax ; \*

777 00004CE0 89C6 <1> mov esi, eax

778 00004CE2 31FF <1> xor edi, edi ; 0

779 <1> dapt\_0:

780 00004CE4 AD <1> lodsd

781 00004CE5 A801 <1> test al, PTE\_A\_PRESENT ; bit 0, present flag (must be 1)

782 00004CE7 7441 <1> jz short dapt\_1

783 <1> ;

784 00004CE9 A802 <1> test al, PTE\_A\_WRITE ; bit 1, writable (r/w) flag

785 <1> ; (must be 1)

786 00004CEB 754C <1> jnz short dapt\_3

787 <1> ; Read only -duplicated- page (belongs to a parent or a child)

788 00004CED 66A90002 <1> test ax, PTE\_DUPLICATED ; Was this page duplicated

789 <1> ; as child's page ?

790 00004CF1 7451 <1> jz short dapt\_4 ; Clear PTE but don't deallocate the page!

791 <1> ; check the parent's PTE value is read only & same page or not..

792 <1> ; ECX = page directory entry index (0-1023)

793 00004CF3 53 <1> push ebx

794 00004CF4 51 <1> push ecx

795 00004CF5 66C1E102 <1> shl cx, 2 ; \*4

796 00004CF9 01CB <1> add ebx, ecx ; PDE offset (for the parent)

797 00004CFB 8B0B <1> mov ecx, [ebx]

798 00004CFD F6C101 <1> test cl, PDE\_A\_PRESENT ; present (valid) or not ?

799 00004D00 7435 <1> jz short dapt\_2 ; parent process does not use this page

800 00004D02 6681E100F0 <1> and cx, PDE\_A\_CLEAR ; 0F000h ; Clear attribute bits

801 <1> ; EDI = page table entry index (0-1023)

802 00004D07 89FA <1> mov edx, edi

803 00004D09 66C1E202 <1> shl dx, 2 ; \*4

804 00004D0D 01CA <1> add edx, ecx ; PTE offset (for the parent)

805 00004D0F 8B1A <1> mov ebx, [edx]

806 00004D11 F6C301 <1> test bl, PTE\_A\_PRESENT ; present or not ?

807 00004D14 7421 <1> jz short dapt\_2 ; parent process does not use this page

808 00004D16 662500F0 <1> and ax, PTE\_A\_CLEAR ; 0F000h ; Clear attribute bits

809 00004D1A 6681E300F0 <1> and bx, PTE\_A\_CLEAR ; 0F000h ; Clear attribute bits

810 00004D1F 39D8 <1> cmp eax, ebx ; parent's and child's pages are same ?

811 00004D21 7514 <1> jne short dapt\_2 ; not same page

812 <1> ; deallocate the child's page

813 00004D23 800A02 <1> or byte [edx], PTE\_A\_WRITE ; convert to writable page (parent)

814 00004D26 59 <1> pop ecx

815 00004D27 5B <1> pop ebx

816 00004D28 EB1A <1> jmp short dapt\_4

817 <1> dapt\_1:

818 00004D2A 09C0 <1> or eax, eax ; swapped page ?

819 00004D2C 741D <1> jz short dapt\_5 ; no

820 <1> ; yes

821 00004D2E D1E8 <1> shr eax, 1

822 00004D30 E8CA040000 <1> call unlink\_swap\_block ; Deallocate swapped page block

823 <1> ; on the swap disk (or in file)

824 00004D35 EB14 <1> jmp short dapt\_5

825 <1> dapt\_2:

826 00004D37 59 <1> pop ecx

827 00004D38 5B <1> pop ebx

828 <1> dapt\_3:

829 <1> ; 12/07/2016

830 00004D39 66A90004 <1> test ax, PTE\_SHARED ; shared or direct memory access indicator

831 00004D3D 7505 <1> jnz short dapt\_4 ; AVL bit 1 = 1, do not deallocate this page!

832 <1> ;

833 <1> ;and ax, PTE\_A\_CLEAR ; 0F000h ; clear lower 12 (attribute) bits

834 00004D3F E814000000 <1> call deallocate\_page ; set the mem allocation bit of this page

835 <1> dapt\_4:

836 00004D44 C746FC00000000 <1> mov dword [esi-4], 0 ; clear/reset PTE (child, dupl. as parent)

837 <1> dapt\_5:

838 00004D4B 47 <1> inc edi ; page table entry index

839 00004D4C 81FF00040000 <1> cmp edi, PAGE\_SIZE / 4 ; 1024

840 00004D52 7290 <1> jb short dapt\_0

841 <1> ;

842 00004D54 58 <1> pop eax ; \*

843 00004D55 5A <1> pop edx

844 00004D56 5F <1> pop edi

845 00004D57 5E <1> pop esi

846 <1> ;

847 <1> ;call deallocate\_page ; deallocate the page table's itself

848 <1> ;retn

849 <1>

850 <1> deallocate\_page:

851 <1> ; 15/09/2015

852 <1> ; 28/04/2015

853 <1> ; 10/03/2015

854 <1> ; 17/10/2014

855 <1> ; 12/10/2014 (Retro UNIX 386 v1 - beginning)

856 <1> ;

857 <1> ; INPUT ->

858 <1> ; EAX = PHYSICAL (real/flat) ADDRESS OF THE ALLOCATED PAGE

859 <1> ; OUTPUT ->

860 <1> ; [free\_pages] is increased

861 <1> ; (corresponding MEMORY ALLOCATION TABLE bit is SET)

862 <1> ; CF = 1 if the page is already deallocated

863 <1> ; (or not allocated) before.

864 <1> ;

865 <1> ; Modified Registers -> EAX

866 <1> ;

867 00004D58 53 <1> push ebx

868 00004D59 52 <1> push edx

869 <1> ;

870 00004D5A C1E80C <1> shr eax, PAGE\_SHIFT ; shift physical address to

871 <1> ; 12 bits right

872 <1> ; to get page number

873 00004D5D 89C2 <1> mov edx, eax

874 <1> ; 15/09/2015

875 00004D5F C1EA03 <1> shr edx, 3 ; to get offset to M.A.T.

876 <1> ; (1 allocation bit = 1 page)

877 <1> ; (1 allocation bytes = 8 pages)

878 00004D62 80E2FC <1> and dl, 0FCh ; clear lower 2 bits

879 <1> ; (to get 32 bit position)

880 <1> ;

881 00004D65 BB00001000 <1> mov ebx, MEM\_ALLOC\_TBL ; Memory Allocation Table address

882 00004D6A 01D3 <1> add ebx, edx

883 00004D6C 83E01F <1> and eax, 1Fh ; lower 5 bits only

884 <1> ; (allocation bit position)

885 00004D6F 3B15[44580100] <1> cmp edx, [next\_page] ; is the new free page address lower

886 <1> ; than the address in 'next\_page' ?

887 <1> ; (next/first free page value)

888 00004D75 7306 <1> jnb short dap\_1 ; no

889 00004D77 8915[44580100] <1> mov [next\_page], edx ; yes

890 <1> dap\_1:

891 00004D7D 0FAB03 <1> bts [ebx], eax ; unlink/release/deallocate page

892 <1> ; set relevant bit to 1.

893 <1> ; set CF to the previous bit value

894 <1> ;cmc ; complement carry flag

895 <1> ;jc short dap\_2 ; do not increase free\_pages count

896 <1> ; if the page is already deallocated

897 <1> ; before.

898 00004D80 FF05[40580100] <1> inc dword [free\_pages]

899 <1> dap\_2:

900 00004D86 5A <1> pop edx

901 00004D87 5B <1> pop ebx

902 00004D88 C3 <1> retn

903 <1>

904 <1> ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

905 <1> ;; ;;

906 <1> ;; Copyright (C) KolibriOS team 2004-2012. All rights reserved. ;;

907 <1> ;; Distributed under terms of the GNU General Public License ;;

908 <1> ;; ;;

909 <1> ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

910 <1>

911 <1> ;;$Revision: 5057 $

912 <1>

913 <1>

914 <1> ;;align 4

915 <1> ;;proc alloc\_page

916 <1>

917 <1> ;; pushfd

918 <1> ;; cli

919 <1> ;; push ebx

920 <1> ;;;//-

921 <1> ;; cmp [pg\_data.pages\_free], 1

922 <1> ;; jle .out\_of\_memory

923 <1> ;;;//-

924 <1> ;;

925 <1> ;; mov ebx, [page\_start]

926 <1> ;; mov ecx, [page\_end]

927 <1> ;;.l1:

928 <1> ;; bsf eax, [ebx];

929 <1> ;; jnz .found

930 <1> ;; add ebx, 4

931 <1> ;; cmp ebx, ecx

932 <1> ;; jb .l1

933 <1> ;; pop ebx

934 <1> ;; popfd

935 <1> ;; xor eax, eax

936 <1> ;; ret

937 <1> ;;.found:

938 <1> ;;;//-

939 <1> ;; dec [pg\_data.pages\_free]

940 <1> ;; jz .out\_of\_memory

941 <1> ;;;//-

942 <1> ;; btr [ebx], eax

943 <1> ;; mov [page\_start], ebx

944 <1> ;; sub ebx, sys\_pgmap

945 <1> ;; lea eax, [eax+ebx\*8]

946 <1> ;; shl eax, 12

947 <1> ;;;//- dec [pg\_data.pages\_free]

948 <1> ;; pop ebx

949 <1> ;; popfd

950 <1> ;; ret

951 <1> ;;;//-

952 <1> ;;.out\_of\_memory:

953 <1> ;; mov [pg\_data.pages\_free], 1

954 <1> ;; xor eax, eax

955 <1> ;; pop ebx

956 <1> ;; popfd

957 <1> ;; ret

958 <1> ;;;//-

959 <1> ;;endp

960 <1>

961 <1> duplicate\_page\_dir:

962 <1> ; 21/09/2015

963 <1> ; 31/08/2015

964 <1> ; 20/07/2015

965 <1> ; 28/04/2015

966 <1> ; 27/04/2015

967 <1> ; 18/04/2015

968 <1> ; 12/04/2015

969 <1> ; 18/10/2014

970 <1> ; 16/10/2014 (Retro UNIX 386 v1 - beginning)

971 <1> ;

972 <1> ; INPUT ->

973 <1> ; [u.pgdir] = PHYSICAL (real/flat) ADDRESS of the parent's

974 <1> ; page directory.

975 <1> ; OUTPUT ->

976 <1> ; EAX = PHYSICAL (real/flat) ADDRESS of the child's

977 <1> ; page directory.

978 <1> ; (New page directory with new page table entries.)

979 <1> ; (New page tables with read only copies of the parent's

980 <1> ; pages.)

981 <1> ; EAX = 0 -> Error (CF = 1)

982 <1> ;

983 <1> ; Modified Registers -> none (except EAX)

984 <1> ;

985 00004D89 E8ECFDFFFF <1> call allocate\_page

986 00004D8E 723E <1> jc short dpd\_err

987 <1> ;

988 00004D90 55 <1> push ebp ; 20/07/2015

989 00004D91 56 <1> push esi

990 00004D92 57 <1> push edi

991 00004D93 53 <1> push ebx

992 00004D94 51 <1> push ecx

993 00004D95 8B35[B8030300] <1> mov esi, [u.pgdir]

994 00004D9B 89C7 <1> mov edi, eax

995 00004D9D 50 <1> push eax ; save child's page directory address

996 <1> ; 31/08/2015

997 <1> ; copy PDE 0 from the parent's page dir to the child's page dir

998 <1> ; (use same system space for all user page tables)

999 00004D9E A5 <1> movsd

1000 00004D9F BD00004000 <1> mov ebp, 1024\*4096 ; pass the 1st 4MB (system space)

1001 00004DA4 B9FF030000 <1> mov ecx, (PAGE\_SIZE / 4) - 1 ; 1023

1002 <1> dpd\_0:

1003 00004DA9 AD <1> lodsd

1004 <1> ;or eax, eax

1005 <1> ;jnz short dpd\_1

1006 00004DAA A801 <1> test al, PDE\_A\_PRESENT ; bit 0 = 1

1007 00004DAC 7508 <1> jnz short dpd\_1

1008 <1> ; 20/07/2015 (virtual address at the end of the page table)

1009 00004DAE 81C500004000 <1> add ebp, 1024\*4096 ; page size \* PTE count

1010 00004DB4 EB0F <1> jmp short dpd\_2

1011 <1> dpd\_1:

1012 00004DB6 662500F0 <1> and ax, PDE\_A\_CLEAR ; 0F000h ; clear attribute bits

1013 00004DBA 89C3 <1> mov ebx, eax

1014 <1> ; EBX = Parent's page table address

1015 00004DBC E81F000000 <1> call duplicate\_page\_table

1016 00004DC1 720C <1> jc short dpd\_p\_err

1017 <1> ; EAX = Child's page table address

1018 00004DC3 0C07 <1> or al, PDE\_A\_PRESENT + PDE\_A\_WRITE + PDE\_A\_USER

1019 <1> ; set bit 0, bit 1 and bit 2 to 1

1020 <1> ; (present, writable, user)

1021 <1> dpd\_2:

1022 00004DC5 AB <1> stosd

1023 00004DC6 E2E1 <1> loop dpd\_0

1024 <1> ;

1025 00004DC8 58 <1> pop eax ; restore child's page directory address

1026 <1> dpd\_3:

1027 00004DC9 59 <1> pop ecx

1028 00004DCA 5B <1> pop ebx

1029 00004DCB 5F <1> pop edi

1030 00004DCC 5E <1> pop esi

1031 00004DCD 5D <1> pop ebp ; 20/07/2015

1032 <1> dpd\_err:

1033 00004DCE C3 <1> retn

1034 <1> dpd\_p\_err:

1035 <1> ; release the allocated pages missing (recover free space)

1036 00004DCF 58 <1> pop eax ; the new page directory address (physical)

1037 00004DD0 8B1D[B8030300] <1> mov ebx, [u.pgdir] ; parent's page directory address

1038 00004DD6 E8D8FEFFFF <1> call deallocate\_page\_dir

1039 00004DDB 29C0 <1> sub eax, eax ; 0

1040 00004DDD F9 <1> stc

1041 00004DDE EBE9 <1> jmp short dpd\_3

1042 <1>

1043 <1> duplicate\_page\_table:

1044 <1> ; 20/02/2017

1045 <1> ; 21/09/2015

1046 <1> ; 20/07/2015

1047 <1> ; 05/05/2015

1048 <1> ; 28/04/2015

1049 <1> ; 27/04/2015

1050 <1> ; 18/04/2015

1051 <1> ; 18/10/2014

1052 <1> ; 16/10/2014 (Retro UNIX 386 v1 - beginning)

1053 <1> ;

1054 <1> ; INPUT ->

1055 <1> ; EBX = PHYSICAL (real/flat) ADDRESS of the parent's page table.

1056 <1> ; 20/02/2017

1057 <1> ; EBP = Linear address of the page (from 'duplicate\_page\_dir')

1058 <1> ; (Linear address = CORE + user's virtual address)

1059 <1> ; OUTPUT ->

1060 <1> ; EAX = PHYSICAL (real/flat) ADDRESS of the child's page table.

1061 <1> ; (with 'read only' attribute of page table entries)

1062 <1> ; 20/02/2017

1063 <1> ; EBP = Next linear page address (for 'duplicate\_page\_dir')

1064 <1> ;

1065 <1> ; CF = 1 -> error

1066 <1> ;

1067 <1> ; Modified Registers -> EBP (except EAX)

1068 <1> ;

1069 00004DE0 E895FDFFFF <1> call allocate\_page

1070 00004DE5 726A <1> jc short dpt\_err

1071 <1> ;

1072 00004DE7 50 <1> push eax ; \*

1073 00004DE8 56 <1> push esi

1074 00004DE9 57 <1> push edi

1075 00004DEA 52 <1> push edx

1076 00004DEB 51 <1> push ecx

1077 <1> ;

1078 00004DEC 89DE <1> mov esi, ebx

1079 00004DEE 89C7 <1> mov edi, eax

1080 00004DF0 89C2 <1> mov edx, eax

1081 00004DF2 81C200100000 <1> add edx, PAGE\_SIZE

1082 <1> dpt\_0:

1083 00004DF8 AD <1> lodsd

1084 00004DF9 21C0 <1> and eax, eax

1085 00004DFB 7444 <1> jz short dpt\_3

1086 00004DFD A801 <1> test al, PTE\_A\_PRESENT ; bit 0 = 1

1087 00004DFF 7507 <1> jnz short dpt\_1

1088 <1> ; 20/07/2015

1089 <1> ; ebp = virtual (linear) address of the memory page

1090 00004E01 E83F050000 <1> call reload\_page ; 28/04/2015

1091 00004E06 7244 <1> jc short dpt\_p\_err

1092 <1> dpt\_1:

1093 <1> ; 21/09/2015

1094 00004E08 89C1 <1> mov ecx, eax

1095 00004E0A 662500F0 <1> and ax, PTE\_A\_CLEAR ; 0F000h ; clear attribute bits

1096 00004E0E F6C102 <1> test cl, PTE\_A\_WRITE ; writable page ?

1097 00004E11 7525 <1> jnz short dpt\_2

1098 <1> ; Read only (parent) page

1099 <1> ; - there is a third process which uses this page -

1100 <1> ; Allocate a new page for the child process

1101 00004E13 E862FDFFFF <1> call allocate\_page

1102 00004E18 7232 <1> jc short dpt\_p\_err

1103 00004E1A 57 <1> push edi

1104 00004E1B 56 <1> push esi

1105 00004E1C 89CE <1> mov esi, ecx

1106 00004E1E 89C7 <1> mov edi, eax

1107 00004E20 B900040000 <1> mov ecx, PAGE\_SIZE/4

1108 00004E25 F3A5 <1> rep movsd ; copy page (4096 bytes)

1109 00004E27 5E <1> pop esi

1110 00004E28 5F <1> pop edi

1111 <1> ;

1112 00004E29 53 <1> push ebx

1113 00004E2A 50 <1> push eax

1114 <1> ; 20/07/2015

1115 00004E2B 89EB <1> mov ebx, ebp

1116 <1> ; ebx = virtual (linear) address of the memory page

1117 00004E2D E887030000 <1> call add\_to\_swap\_queue

1118 00004E32 58 <1> pop eax

1119 00004E33 5B <1> pop ebx

1120 <1> ; 21/09/2015

1121 00004E34 0C07 <1> or al, PTE\_A\_USER+PTE\_A\_WRITE+PTE\_A\_PRESENT

1122 <1> ; user + writable + present page

1123 00004E36 EB09 <1> jmp short dpt\_3

1124 <1> dpt\_2:

1125 <1> ;or ax, PTE\_A\_USER+PTE\_A\_PRESENT

1126 00004E38 0C05 <1> or al, PTE\_A\_USER+PTE\_A\_PRESENT

1127 <1> ; (read only page!)

1128 00004E3A 8946FC <1> mov [esi-4], eax ; update parent's PTE

1129 00004E3D 660D0002 <1> or ax, PTE\_DUPLICATED ; (read only page & duplicated PTE!)

1130 <1> dpt\_3:

1131 00004E41 AB <1> stosd ; EDI points to child's PTE

1132 <1> ;

1133 00004E42 81C500100000 <1> add ebp, 4096 ; 20/07/2015 (next page)

1134 <1> ;

1135 00004E48 39D7 <1> cmp edi, edx

1136 00004E4A 72AC <1> jb short dpt\_0

1137 <1> dpt\_p\_err:

1138 00004E4C 59 <1> pop ecx

1139 00004E4D 5A <1> pop edx

1140 00004E4E 5F <1> pop edi

1141 00004E4F 5E <1> pop esi

1142 00004E50 58 <1> pop eax ; \*

1143 <1> dpt\_err:

1144 00004E51 C3 <1> retn

1145 <1>

1146 <1> page\_fault\_handler: ; CPU EXCEPTION 0Eh (14) : Page Fault !

1147 <1> ; 21/09/2015

1148 <1> ; 19/09/2015

1149 <1> ; 17/09/2015

1150 <1> ; 28/08/2015

1151 <1> ; 20/07/2015

1152 <1> ; 28/06/2015

1153 <1> ; 03/05/2015

1154 <1> ; 30/04/2015

1155 <1> ; 18/04/2015

1156 <1> ; 12/04/2015

1157 <1> ; 30/10/2014

1158 <1> ; 11/09/2014

1159 <1> ; 10/09/2014 (Retro UNIX 386 v1 - beginning)

1160 <1> ;

1161 <1> ; Note: This is not an interrupt/exception handler.

1162 <1> ; This is a 'page fault remedy' subroutine

1163 <1> ; which will be called by standard/uniform

1164 <1> ; exception handler.

1165 <1> ;

1166 <1> ; INPUT ->

1167 <1> ; [error\_code] = 32 bit ERROR CODE (lower 5 bits are valid)

1168 <1> ;

1169 <1> ; cr2 = the virtual (linear) address

1170 <1> ; which has caused to page fault (19/09/2015)

1171 <1> ;

1172 <1> ; OUTPUT ->

1173 <1> ; (corresponding PAGE TABLE ENTRY is mapped/set)

1174 <1> ; EAX = 0 -> no error

1175 <1> ; EAX > 0 -> error code in EAX (also CF = 1)

1176 <1> ;

1177 <1> ; Modified Registers -> none (except EAX)

1178 <1> ;

1179 <1> ;

1180 <1> ; ERROR CODE:

1181 <1> ; 31 ..... 4 3 2 1 0

1182 <1> ; +---+-- --+---+---+---+---+---+---+

1183 <1> ; | Reserved | I | R | U | W | P |

1184 <1> ; +---+-- --+---+---+---+---+---+---+

1185 <1> ;

1186 <1> ; P : PRESENT - When set, the page fault was caused by

1187 <1> ; a page-protection violation. When not set,

1188 <1> ; it was caused by a non-present page.

1189 <1> ; W : WRITE - When set, the page fault was caused by

1190 <1> ; a page write. When not set, it was caused

1191 <1> ; by a page read.

1192 <1> ; U : USER - When set, the page fault was caused

1193 <1> ; while CPL = 3.

1194 <1> ; This does not necessarily mean that

1195 <1> ; the page fault was a privilege violation.

1196 <1> ; R : RESERVD - When set, the page fault was caused by

1197 <1> ; WRITE reading a 1 in a reserved field.

1198 <1> ; I : INSTRUC - When set, the page fault was caused by

1199 <1> ; FETCH an instruction fetch

1200 <1> ;

1201 <1> ;; x86 (32 bit) VIRTUAL ADDRESS TRANSLATION

1202 <1> ; 31 22 12 11 0

1203 <1> ; +-------------------+-------------------+-----------------------+

1204 <1> ; | PAGE DIR. ENTRY # | PAGE TAB. ENTRY # | OFFSET |

1205 <1> ; +-------------------+-------------------+-----------------------+

1206 <1> ;

1207 <1>

1208 <1> ;; CR3 REGISTER (Control Register 3)

1209 <1> ; 31 12 5 4 3 2 0

1210 <1> ; +---------------------------------------+-------------+---+-----+

1211 <1> ; | | |P|P| |

1212 <1> ; | PAGE DIRECTORY TABLE BASE ADDRESS | reserved |C|W|rsvrd|

1213 <1> ; | | |D|T| |

1214 <1> ; +---------------------------------------+-------------+---+-----+

1215 <1> ;

1216 <1> ; PWT - WRITE THROUGH

1217 <1> ; PCD - CACHE DISABLE

1218 <1> ;

1219 <1> ;

1220 <1> ;; x86 PAGE DIRECTORY ENTRY (4 KByte Page)

1221 <1> ; 31 12 11 9 8 7 6 5 4 3 2 1 0

1222 <1> ; +---------------------------------------+-----+---+-+-+---+-+-+-+

1223 <1> ; | | | | | | |P|P|U|R| |

1224 <1> ; | PAGE TABLE BASE ADDRESS 31..12 | AVL |G|0|D|A|C|W|/|/|P|

1225 <1> ; | | | | | | |D|T|S|W| |

1226 <1> ; +---------------------------------------+-----+---+-+-+---+-+-+-+

1227 <1> ;

1228 <1> ; P - PRESENT

1229 <1> ; R/W - READ/WRITE

1230 <1> ; U/S - USER/SUPERVISOR

1231 <1> ; PWT - WRITE THROUGH

1232 <1> ; PCD - CACHE DISABLE

1233 <1> ; A - ACCESSED

1234 <1> ; D - DIRTY (IGNORED)

1235 <1> ; PAT - PAGE ATTRIBUTE TABLE INDEX (CACHE BEHAVIOR)

1236 <1> ; G - GLOBAL (IGNORED)

1237 <1> ; AVL - AVAILABLE FOR SYSTEMS PROGRAMMER USE

1238 <1> ;

1239 <1> ;

1240 <1> ;; x86 PAGE TABLE ENTRY (4 KByte Page)

1241 <1> ; 31 12 11 9 8 7 6 5 4 3 2 1 0

1242 <1> ; +---------------------------------------+-----+---+-+-+---+-+-+-+

1243 <1> ; | | | |P| | |P|P|U|R| |

1244 <1> ; | PAGE FRAME BASE ADDRESS 31..12 | AVL |G|A|D|A|C|W|/|/|P|

1245 <1> ; | | | |T| | |D|T|S|W| |

1246 <1> ; +---------------------------------------+-----+---+-+-+---+-+-+-+

1247 <1> ;

1248 <1> ; P - PRESENT

1249 <1> ; R/W - READ/WRITE

1250 <1> ; U/S - USER/SUPERVISOR

1251 <1> ; PWT - WRITE THROUGH

1252 <1> ; PCD - CACHE DISABLE

1253 <1> ; A - ACCESSED

1254 <1> ; D - DIRTY

1255 <1> ; PAT - PAGE ATTRIBUTE TABLE INDEX (CACHE BEHAVIOR)

1256 <1> ; G - GLOBAL

1257 <1> ; AVL - AVAILABLE FOR SYSTEMS PROGRAMMER USE

1258 <1> ;

1259 <1> ;

1260 <1> ;; 80386 PAGE TABLE ENTRY (4 KByte Page)

1261 <1> ; 31 12 11 9 8 7 6 5 4 3 2 1 0

1262 <1> ; +---------------------------------------+-----+-+-+-+-+---+-+-+-+

1263 <1> ; | | | | | | | | |U|R| |

1264 <1> ; | PAGE FRAME BASE ADDRESS 31..12 | AVL |0|0|D|A|0|0|/|/|P|

1265 <1> ; | | | | | | | | |S|W| |

1266 <1> ; +---------------------------------------+-----+-+-+-+-+---+-+-+-+

1267 <1> ;

1268 <1> ; P - PRESENT

1269 <1> ; R/W - READ/WRITE

1270 <1> ; U/S - USER/SUPERVISOR

1271 <1> ; D - DIRTY

1272 <1> ; AVL - AVAILABLE FOR SYSTEMS PROGRAMMER USE

1273 <1> ;

1274 <1> ; NOTE: 0 INDICATES INTEL RESERVED. DO NOT DEFINE.

1275 <1> ;

1276 <1> ;

1277 <1> ;; Invalid Page Table Entry

1278 <1> ; 31 1 0

1279 <1> ; +-------------------------------------------------------------+-+

1280 <1> ; | | |

1281 <1> ; | AVAILABLE |0|

1282 <1> ; | | |

1283 <1> ; +-------------------------------------------------------------+-+

1284 <1> ;

1285 <1>

1286 00004E52 53 <1> push ebx

1287 00004E53 52 <1> push edx

1288 00004E54 51 <1> push ecx

1289 <1> ;

1290 <1> ; 21/09/2015 (debugging)

1291 00004E55 FF05[CC030300] <1> inc dword [u.pfcount] ; page fault count for running process

1292 00004E5B FF05[80050300] <1> inc dword [PF\_Count] ; total page fault count

1293 <1> ; 28/06/2015

1294 <1> ;mov edx, [error\_code] ; Lower 5 bits are valid

1295 00004E61 8A15[78050300] <1> mov dl, [error\_code]

1296 <1> ;

1297 00004E67 F6C201 <1> test dl, 1 ; page fault was caused by a non-present page

1298 <1> ; sign

1299 00004E6A 7422 <1> jz short pfh\_alloc\_np

1300 <1> ;

1301 <1> ; If it is not a 'write on read only page' type page fault

1302 <1> ; major page fault error with minor reason must be returned without

1303 <1> ; fixing the problem. 'sys\_exit with error' will be needed

1304 <1> ; after return here!

1305 <1> ; Page fault will be remedied, by copying page contents

1306 <1> ; to newly allocated page with write permission;

1307 <1> ; sys\_fork -> sys\_exec -> copy on write, demand paging method is

1308 <1> ; used for working with minimum possible memory usage.

1309 <1> ; sys\_fork will duplicate page directory and tables of parent

1310 <1> ; process with 'read only' flag. If the child process attempts to

1311 <1> ; write on these read only pages, page fault will be directed here

1312 <1> ; for allocating a new page with same data/content.

1313 <1> ;

1314 <1> ; IMPORTANT : Retro UNIX 386 v1 (and SINGLIX and TR-DOS)

1315 <1> ; will not force to separate CODE and DATA space

1316 <1> ; in a process/program...

1317 <1> ; CODE segment/section may contain DATA!

1318 <1> ; It is flat, smoth and simplest programming method already as in

1319 <1> ; Retro UNIX 8086 v1 and MS-DOS programs.

1320 <1> ;

1321 00004E6C F6C202 <1> test dl, 2 ; page fault was caused by a page write

1322 <1> ; sign

1323 00004E6F 0F84AB000000 <1> jz pfh\_p\_err

1324 <1> ; 31/08/2015

1325 00004E75 F6C204 <1> test dl, 4 ; page fault was caused while CPL = 3 (user mode)

1326 <1> ; sign. (U+W+P = 4+2+1 = 7)

1327 00004E78 0F84A2000000 <1> jz pfh\_pv\_err

1328 <1> ;

1329 <1> ; make a new page and copy the parent's page content

1330 <1> ; as the child's new page content

1331 <1> ;

1332 00004E7E 0F20D3 <1> mov ebx, cr2 ; CR2 contains the linear address

1333 <1> ; which has caused to page fault

1334 00004E81 E8A2000000 <1> call copy\_page

1335 00004E86 0F828D000000 <1> jc pfh\_im\_err ; insufficient memory

1336 <1> ;

1337 00004E8C EB7D <1> jmp pfh\_cpp\_ok

1338 <1> ;

1339 <1> pfh\_alloc\_np:

1340 00004E8E E8E7FCFFFF <1> call allocate\_page ; (allocate a new page)

1341 00004E93 0F8280000000 <1> jc pfh\_im\_err ; 'insufficient memory' error

1342 <1> pfh\_chk\_cpl:

1343 <1> ; EAX = Physical (base) address of the allocated (new) page

1344 <1> ; (Lower 12 bits are ZERO, because

1345 <1> ; the address is on a page boundary)

1346 00004E99 80E204 <1> and dl, 4 ; CPL = 3 ?

1347 00004E9C 7505 <1> jnz short pfh\_um

1348 <1> ; Page fault handler for kernel/system mode (CPL=0)

1349 00004E9E 0F20DB <1> mov ebx, cr3 ; CR3 (Control Register 3) contains physical address

1350 <1> ; of the current/active page directory

1351 <1> ; (Always kernel/system mode page directory, here!)

1352 <1> ; Note: Lower 12 bits are 0. (page boundary)

1353 00004EA1 EB06 <1> jmp short pfh\_get\_pde

1354 <1> ;

1355 <1> pfh\_um: ; Page fault handler for user/appl. mode (CPL=3)

1356 00004EA3 8B1D[B8030300] <1> mov ebx, [u.pgdir] ; Page directory of current/active process

1357 <1> ; Physical address of the USER's page directory

1358 <1> ; Note: Lower 12 bits are 0. (page boundary)

1359 <1> pfh\_get\_pde:

1360 00004EA9 80CA03 <1> or dl, 3 ; USER + WRITE + PRESENT or SYSTEM + WRITE + PRESENT

1361 00004EAC 0F20D1 <1> mov ecx, cr2 ; CR2 contains the virtual address

1362 <1> ; which has been caused to page fault

1363 <1> ;

1364 00004EAF C1E914 <1> shr ecx, 20 ; shift 20 bits right

1365 00004EB2 80E1FC <1> and cl, 0FCh ; mask lower 2 bits to get PDE offset

1366 <1> ;

1367 00004EB5 01CB <1> add ebx, ecx ; now, EBX points to the relevant page dir entry

1368 00004EB7 8B0B <1> mov ecx, [ebx] ; physical (base) address of the page table

1369 00004EB9 F6C101 <1> test cl, 1 ; check bit 0 is set (1) or not (0).

1370 00004EBC 740B <1> jz short pfh\_set\_pde ; Page directory entry is not valid,

1371 <1> ; set/validate page directory entry

1372 00004EBE 6681E100F0 <1> and cx, PDE\_A\_CLEAR ; 0F000h ; Clear attribute bits

1373 00004EC3 89CB <1> mov ebx, ecx ; Physical address of the page table

1374 00004EC5 89C1 <1> mov ecx, eax ; new page address (physical)

1375 00004EC7 EB16 <1> jmp short pfh\_get\_pte

1376 <1> pfh\_set\_pde:

1377 <1> ;; NOTE: Page directories and page tables never be swapped out!

1378 <1> ;; (So, we know this PDE is empty or invalid)

1379 <1> ;

1380 00004EC9 08D0 <1> or al, dl ; lower 3 bits are used as U/S, R/W, P flags

1381 00004ECB 8903 <1> mov [ebx], eax ; Let's put the new page directory entry here !

1382 00004ECD 30C0 <1> xor al, al ; clear lower (3..8) bits

1383 00004ECF 89C3 <1> mov ebx, eax

1384 00004ED1 E8A4FCFFFF <1> call allocate\_page ; (allocate a new page)

1385 00004ED6 7241 <1> jc short pfh\_im\_err ; 'insufficient memory' error

1386 <1> pfh\_spde\_1:

1387 <1> ; EAX = Physical (base) address of the allocated (new) page

1388 00004ED8 89C1 <1> mov ecx, eax

1389 00004EDA E815FDFFFF <1> call clear\_page ; Clear page content

1390 <1> pfh\_get\_pte:

1391 00004EDF 0F20D0 <1> mov eax, cr2 ; virtual address

1392 <1> ; which has been caused to page fault

1393 00004EE2 89C7 <1> mov edi, eax ; 20/07/2015

1394 00004EE4 C1E80C <1> shr eax, 12 ; shift 12 bit right to get

1395 <1> ; higher 20 bits of the page fault address

1396 00004EE7 25FF030000 <1> and eax, 3FFh ; mask PDE# bits, the result is PTE# (0 to 1023)

1397 00004EEC C1E002 <1> shl eax, 2 ; shift 2 bits left to get PTE offset

1398 00004EEF 01C3 <1> add ebx, eax ; now, EBX points to the relevant page table entry

1399 00004EF1 8B03 <1> mov eax, [ebx] ; get previous value of pte

1400 <1> ; bit 0 of EAX is always 0 (otherwise we would not be here)

1401 00004EF3 21C0 <1> and eax, eax

1402 00004EF5 7410 <1> jz short pfh\_gpte\_1

1403 <1> ; 20/07/2015

1404 00004EF7 87D9 <1> xchg ebx, ecx ; new page address (physical)

1405 00004EF9 55 <1> push ebp ; 20/07/2015

1406 00004EFA 0F20D5 <1> mov ebp, cr2

1407 <1> ; ECX = physical address of the page table entry

1408 <1> ; EBX = Memory page address (physical!)

1409 <1> ; EAX = Swap disk (offset) address

1410 <1> ; EBP = virtual address (page fault address)

1411 00004EFD E8B7000000 <1> call swap\_in

1412 00004F02 5D <1> pop ebp

1413 00004F03 7210 <1> jc short pfh\_err\_retn

1414 00004F05 87CB <1> xchg ecx, ebx

1415 <1> ; EBX = physical address of the page table entry

1416 <1> ; ECX = new page

1417 <1> pfh\_gpte\_1:

1418 00004F07 08D1 <1> or cl, dl ; lower 3 bits are used as U/S, R/W, P flags

1419 00004F09 890B <1> mov [ebx], ecx ; Let's put the new page table entry here !

1420 <1> pfh\_cpp\_ok:

1421 <1> ; 20/07/2015

1422 00004F0B 0F20D3 <1> mov ebx, cr2

1423 00004F0E E8A6020000 <1> call add\_to\_swap\_queue

1424 <1> ;

1425 <1> ; The new PTE (which contains the new page) will be added to

1426 <1> ; the swap queue, here.

1427 <1> ; (Later, if memory will become insufficient,

1428 <1> ; one page will be swapped out which is at the head of

1429 <1> ; the swap queue by using FIFO and access check methods.)

1430 <1> ;

1431 00004F13 31C0 <1> xor eax, eax ; 0

1432 <1> ;

1433 <1> pfh\_err\_retn:

1434 00004F15 59 <1> pop ecx

1435 00004F16 5A <1> pop edx

1436 00004F17 5B <1> pop ebx

1437 00004F18 C3 <1> retn

1438 <1>

1439 <1> pfh\_im\_err:

1440 00004F19 B8E4000000 <1> mov eax, ERR\_MAJOR\_PF + ERR\_MINOR\_IM ; Error code in AX

1441 <1> ; Major (Primary) Error: Page Fault

1442 <1> ; Minor (Secondary) Error: Insufficient Memory !

1443 00004F1E EBF5 <1> jmp short pfh\_err\_retn

1444 <1>

1445 <1>

1446 <1> pfh\_p\_err: ; 09/03/2015

1447 <1> pfh\_pv\_err:

1448 <1> ; Page fault was caused by a protection-violation

1449 00004F20 B8E6000000 <1> mov eax, ERR\_MAJOR\_PF + ERR\_MINOR\_PV ; Error code in AX

1450 <1> ; Major (Primary) Error: Page Fault

1451 <1> ; Minor (Secondary) Error: Protection violation !

1452 00004F25 F9 <1> stc

1453 00004F26 EBED <1> jmp short pfh\_err\_retn

1454 <1>

1455 <1> copy\_page:

1456 <1> ; 22/09/2015

1457 <1> ; 21/09/2015

1458 <1> ; 19/09/2015

1459 <1> ; 07/09/2015

1460 <1> ; 31/08/2015

1461 <1> ; 20/07/2015

1462 <1> ; 05/05/2015

1463 <1> ; 03/05/2015

1464 <1> ; 18/04/2015

1465 <1> ; 12/04/2015

1466 <1> ; 30/10/2014

1467 <1> ; 18/10/2014 (Retro UNIX 386 v1 - beginning)

1468 <1> ;

1469 <1> ; INPUT ->

1470 <1> ; EBX = Virtual (linear) address of source page

1471 <1> ; (Page fault address)

1472 <1> ; OUTPUT ->

1473 <1> ; EAX = PHYSICAL (real/flat) ADDRESS OF THE ALLOCATED PAGE

1474 <1> ; (corresponding PAGE TABLE ENTRY is mapped/set)

1475 <1> ; EAX = 0 (CF = 1)

1476 <1> ; if there is not a free page to be allocated

1477 <1> ; (page content of the source page will be copied

1478 <1> ; onto the target/new page)

1479 <1> ;

1480 <1> ; Modified Registers -> ecx, ebx (except EAX)

1481 <1> ;

1482 00004F28 56 <1> push esi

1483 00004F29 57 <1> push edi

1484 <1> ;push ebx

1485 <1> ;push ecx

1486 00004F2A 31F6 <1> xor esi, esi

1487 00004F2C C1EB0C <1> shr ebx, 12 ; shift 12 bits right to get PDE & PTE numbers

1488 00004F2F 89D9 <1> mov ecx, ebx ; save page fault address (as 12 bit shifted)

1489 00004F31 C1EB08 <1> shr ebx, 8 ; shift 8 bits right and then

1490 00004F34 80E3FC <1> and bl, 0FCh ; mask lower 2 bits to get PDE offset

1491 00004F37 89DF <1> mov edi, ebx ; save it for the parent of current process

1492 00004F39 031D[B8030300] <1> add ebx, [u.pgdir] ; EBX points to the relevant page dir entry

1493 00004F3F 8B03 <1> mov eax, [ebx] ; physical (base) address of the page table

1494 00004F41 662500F0 <1> and ax, PTE\_A\_CLEAR ; 0F000h ; clear attribute bits

1495 00004F45 89CB <1> mov ebx, ecx ; (restore higher 20 bits of page fault address)

1496 00004F47 81E3FF030000 <1> and ebx, 3FFh ; mask PDE# bits, the result is PTE# (0 to 1023)

1497 00004F4D 66C1E302 <1> shl bx, 2 ; shift 2 bits left to get PTE offset

1498 00004F51 01C3 <1> add ebx, eax ; EBX points to the relevant page table entry

1499 <1> ; 07/09/2015

1500 00004F53 66F7030002 <1> test word [ebx], PTE\_DUPLICATED ; (Does current process share this

1501 <1> ; read only page as a child process?)

1502 00004F58 7509 <1> jnz short cpp\_0 ; yes

1503 00004F5A 8B0B <1> mov ecx, [ebx] ; PTE value

1504 00004F5C 6681E100F0 <1> and cx, PTE\_A\_CLEAR ; 0F000h ; clear page attributes

1505 00004F61 EB32 <1> jmp short cpp\_1

1506 <1> cpp\_0:

1507 00004F63 89FE <1> mov esi, edi

1508 00004F65 0335[BC030300] <1> add esi, [u.ppgdir] ; the parent's page directory entry

1509 00004F6B 8B06 <1> mov eax, [esi] ; physical (base) address of the page table

1510 00004F6D 662500F0 <1> and ax, PTE\_A\_CLEAR ; 0F000h ; clear attribute bits

1511 00004F71 89CE <1> mov esi, ecx ; (restore higher 20 bits of page fault address)

1512 00004F73 81E6FF030000 <1> and esi, 3FFh ; mask PDE# bits, the result is PTE# (0 to 1023)

1513 00004F79 66C1E602 <1> shl si, 2 ; shift 2 bits left to get PTE offset

1514 00004F7D 01C6 <1> add esi, eax ; EDX points to the relevant page table entry

1515 00004F7F 8B0E <1> mov ecx, [esi] ; PTE value of the parent process

1516 <1> ; 21/09/2015

1517 00004F81 8B03 <1> mov eax, [ebx] ; PTE value of the child process

1518 00004F83 662500F0 <1> and ax, PTE\_A\_CLEAR ; 0F000h ; clear page attributes

1519 <1> ;

1520 00004F87 F6C101 <1> test cl, PTE\_A\_PRESENT ; is it a present/valid page ?

1521 00004F8A 7424 <1> jz short cpp\_3 ; the parent's page is not same page

1522 <1> ;

1523 00004F8C 6681E100F0 <1> and cx, PTE\_A\_CLEAR ; 0F000h ; clear page attributes

1524 00004F91 39C8 <1> cmp eax, ecx ; Same page?

1525 00004F93 751B <1> jne short cpp\_3 ; Parent page and child page are not same

1526 <1> ; Convert child's page to writable page

1527 <1> cpp\_1:

1528 00004F95 E8E0FBFFFF <1> call allocate\_page

1529 00004F9A 721A <1> jc short cpp\_4 ; 'insufficient memory' error

1530 00004F9C 21F6 <1> and esi, esi ; check ESI is valid or not

1531 00004F9E 7405 <1> jz short cpp\_2

1532 <1> ; Convert read only page to writable page

1533 <1> ;(for the parent of the current process)

1534 <1> ;and word [esi], PTE\_A\_CLEAR ; 0F000h

1535 <1> ; 22/09/2015

1536 00004FA0 890E <1> mov [esi], ecx

1537 00004FA2 800E07 <1> or byte [esi], PTE\_A\_PRESENT + PTE\_A\_WRITE + PTE\_A\_USER

1538 <1> ; 1+2+4 = 7

1539 <1> cpp\_2:

1540 00004FA5 89C7 <1> mov edi, eax ; new page address of the child process

1541 <1> ; 07/09/2015

1542 00004FA7 89CE <1> mov esi, ecx ; the page address of the parent process

1543 00004FA9 B900040000 <1> mov ecx, PAGE\_SIZE / 4

1544 00004FAE F3A5 <1> rep movsd ; 31/08/2015

1545 <1> cpp\_3:

1546 00004FB0 0C07 <1> or al, PTE\_A\_PRESENT + PTE\_A\_WRITE + PTE\_A\_USER ; 1+2+4 = 7

1547 00004FB2 8903 <1> mov [ebx], eax ; Update PTE

1548 00004FB4 28C0 <1> sub al, al ; clear attributes

1549 <1> cpp\_4:

1550 <1> ;pop ecx

1551 <1> ;pop ebx

1552 00004FB6 5F <1> pop edi

1553 00004FB7 5E <1> pop esi

1554 00004FB8 C3 <1> retn

1555 <1>

1556 <1> ;; 28/04/2015

1557 <1> ;; 24/10/2014

1558 <1> ;; 21/10/2014 (Retro UNIX 386 v1 - beginning)

1559 <1> ;; SWAP\_PAGE\_QUEUE (4096 bytes)

1560 <1> ;;

1561 <1> ;; 0000 0001 0002 0003 .... 1020 1021 1022 1023

1562 <1> ;; +------+------+------+------+- -+------+------+------+------+

1563 <1> ;; | pg1 | pg2 | pg3 | pg4 | .... |pg1021|pg1022|pg1023|pg1024|

1564 <1> ;; +------+------+------+------+- -+------+------+------+------+

1565 <1> ;;

1566 <1> ;; [swpq\_last] = 0 to 4096 (step 4) -> the last position on the queue

1567 <1> ;;

1568 <1> ;; Method:

1569 <1> ;; Swap page queue is a list of allocated pages with physical

1570 <1> ;; addresses (system mode virtual adresses = physical addresses).

1571 <1> ;; It is used for 'swap\_in' and 'swap\_out' procedures.

1572 <1> ;; When a new page is being allocated, swap queue is updated

1573 <1> ;; by 'swap\_queue\_shift' procedure, header of the queue (offset 0)

1574 <1> ;; is checked for 'accessed' flag. If the 1st page on the queue

1575 <1> ;; is 'accessed' or 'read only', it is dropped from the list;

1576 <1> ;; other pages from the 2nd to the last (in [swpq\_last]) shifted

1577 <1> ;; to head then the 2nd page becomes the 1st and '[swpq\_last]'

1578 <1> ;; offset value becomes it's previous offset value - 4.

1579 <1> ;; If the 1st page of the swap page queue is not 'accessed'

1580 <1> ;; the queue/list is not shifted.

1581 <1> ;; After the queue/list shift, newly allocated page is added

1582 <1> ;; to the tail of the queue at the [swpq\_count\*4] position.

1583 <1> ;; But, if [swpq\_count] > 1023, the newly allocated page

1584 <1> ;; will not be added to the tail of swap page queue.

1585 <1> ;;

1586 <1> ;; During 'swap\_out' procedure, swap page queue is checked for

1587 <1> ;; the first non-accessed, writable page in the list,

1588 <1> ;; from the head to the tail. The list is shifted to left

1589 <1> ;; (to the head) till a non-accessed page will be found in the list.

1590 <1> ;; Then, this page is swapped out (to disk) and then it is dropped

1591 <1> ;; from the list by a final swap queue shift. [swpq\_count] value

1592 <1> ;; is changed. If all pages on the queue' are 'accessed',

1593 <1> ;; 'insufficient memory' error will be returned ('swap\_out'

1594 <1> ;; procedure will be failed)...

1595 <1> ;;

1596 <1> ;; Note: If the 1st page of the queue is an 'accessed' page,

1597 <1> ;; 'accessed' flag of the page will be reset (0) and that page

1598 <1> ;; (PTE) will be added to the tail of the queue after

1599 <1> ;; the check, if [swpq\_count] < 1023. If [swpq\_count] = 1024

1600 <1> ;; the queue will be rotated and the PTE in the head will be

1601 <1> ;; added to the tail after resetting 'accessed' bit.

1602 <1> ;;

1603 <1> ;;

1604 <1> ;;

1605 <1> ;; SWAP DISK/FILE (with 4096 bytes swapped page blocks)

1606 <1> ;;

1607 <1> ;; 00000000 00000004 00000008 0000000C ... size-8 size-4

1608 <1> ;; +---------+---------+---------+---------+-- --+---------+---------+

1609 <1> ;; |descriptr| page(1) | page(2) | page(3) | ... |page(n-1)| page(n) |

1610 <1> ;; +---------+---------+---------+---------+-- --+---------+---------+

1611 <1> ;;

1612 <1> ;; [swpd\_next] = the first free block address in swapped page records

1613 <1> ;; for next free block search by 'swap\_out' procedure.

1614 <1> ;; [swpd\_size] = swap disk/file size in sectors (512 bytes)

1615 <1> ;; NOTE: max. possible swap disk size is 1024 GB

1616 <1> ;; (entire swap space must be accessed by using

1617 <1> ;; 31 bit offset address)

1618 <1> ;; [swpd\_free] = free block (4096 bytes) count in swap disk/file space

1619 <1> ;; [swpd\_start] = absolute/start address of the swap disk/file

1620 <1> ;; 0 for file, or beginning sector of the swap partition

1621 <1> ;; [swp\_drv] = logical drive description table addr. of swap disk/file

1622 <1> ;;

1623 <1> ;;

1624 <1> ;; Method:

1625 <1> ;; When the memory (ram) becomes insufficient, page allocation

1626 <1> ;; procedure swaps out a page from memory to the swap disk

1627 <1> ;; (partition) or swap file to get a new free page at the memory.

1628 <1> ;; Swapping out is performed by using swap page queue.

1629 <1> ;;

1630 <1> ;; Allocation block size of swap disk/file is equal to page size

1631 <1> ;; (4096 bytes). Swapping address (in sectors) is recorded

1632 <1> ;; into relevant page file entry as 31 bit physical (logical)

1633 <1> ;; offset address as 1 bit shifted to left for present flag (0).

1634 <1> ;; Swapped page address is between 1 and swap disk/file size - 4.

1635 <1> ;; Absolute physical (logical) address of the swapped page is

1636 <1> ;; calculated by adding offset value to the swap partition's

1637 <1> ;; start address. If the swap device (disk) is a virtual disk

1638 <1> ;; or it is a file, start address of the swap disk/volume is 0,

1639 <1> ;; and offset value is equal to absolute (physical or logical)

1640 <1> ;; address/position. (It has not to be ZERO if the swap partition

1641 <1> ;; is in a partitioned virtual hard disk.)

1642 <1> ;;

1643 <1> ;; Note: Swap addresses are always specified/declared in sectors,

1644 <1> ;; not in bytes or in blocks/zones/clusters (4096 bytes) as unit.

1645 <1> ;;

1646 <1> ;; Swap disk/file allocation is mapped via 'Swap Allocation Table'

1647 <1> ;; at memory as similar to 'Memory Allocation Table'.

1648 <1> ;;

1649 <1> ;; Every bit of Swap Allocation Table repsesents one swap block

1650 <1> ;; (equal to page size) respectively. Bit 0 of the S.A.T. byte 0

1651 <1> ;; is reserved for swap disk/file block 0 as descriptor block

1652 <1> ;; (also for compatibility with PTE). If bit value is ZERO,

1653 <1> ;; it means relevant (respective) block is in use, and,

1654 <1> ;; of course, if bit value is 1, it means relevant (respective)

1655 <1> ;; swap disk/file block is free.

1656 <1> ;; For example: bit 1 of the byte 128 repsesents block 1025

1657 <1> ;; (128\*8+1) or sector (offset) 8200 on the swap disk or

1658 <1> ;; byte (offset/position) 4198400 in the swap file.

1659 <1> ;; 4GB swap space is represented via 128KB Swap Allocation Table.

1660 <1> ;; Initial layout of Swap Allocation Table is as follows:

1661 <1> ;; ------------------------------------------------------------

1662 <1> ;; 0111111111111111111111111 .... 11111111111111111111111111111

1663 <1> ;; ------------------------------------------------------------

1664 <1> ;; (0 is reserved block, 1s represent free blocks respectively.)

1665 <1> ;; (Note: Allocation cell/unit of the table is bit, not byte)

1666 <1> ;;

1667 <1> ;; ..............................................................

1668 <1> ;;

1669 <1> ;; 'swap\_out' procedure checks 'free\_swap\_blocks' count at first,

1670 <1> ;; then it searches Swap Allocation Table if free count is not

1671 <1> ;; zero. From begining the [swpd\_next] dword value, the first bit

1672 <1> ;; position with value of 1 on the table is converted to swap

1673 <1> ;; disk/file offset address, in sectors (not 4096 bytes block).

1674 <1> ;; 'ldrv\_write' procedure is called with ldrv (logical drive

1675 <1> ;; number of physical swap disk or virtual swap disk)

1676 <1> ;; number, sector offset (not absolute sector -LBA- number),

1677 <1> ;; and sector count (8, 512\*8 = 4096) and buffer adress

1678 <1> ;; (memory page). That will be a direct disk write procedure.

1679 <1> ;; (for preventing late memory allocation, significant waiting).

1680 <1> ;; If disk write procedure returns with error or free count of

1681 <1> ;; swap blocks is ZERO, 'swap\_out' procedure will return with

1682 <1> ;; 'insufficient memory error' (cf=1).

1683 <1> ;;

1684 <1> ;; (Note: Even if free swap disk/file blocks was not zero,

1685 <1> ;; any disk write error will not be fixed by 'swap\_out' procedure,

1686 <1> ;; in other words, 'swap\_out' will not check the table for other

1687 <1> ;; free blocks after a disk write error. It will return to

1688 <1> ;; the caller with error (CF=1) which means swapping is failed.

1689 <1> ;;

1690 <1> ;; After writing the page on to swap disk/file address/sector,

1691 <1> ;; 'swap\_out' procesure returns with that swap (offset) sector

1692 <1> ;; address (cf=0).

1693 <1> ;;

1694 <1> ;; ..............................................................

1695 <1> ;;

1696 <1> ;; 'swap\_in' procedure loads addressed (relevant) swap disk or

1697 <1> ;; file sectors at specified memory page. Then page allocation

1698 <1> ;; procedure updates relevant page table entry with 'present'

1699 <1> ;; attribute. If swap disk or file reading fails there is nothing

1700 <1> ;; to do, except to terminate the process which is the owner of

1701 <1> ;; the swapped page.

1702 <1> ;;

1703 <1> ;; 'swap\_in' procedure sets the relevant/respective bit value

1704 <1> ;; in the Swap Allocation Table (as free block). 'swap\_in' also

1705 <1> ;; updates [swpd\_first] pointer if it is required.

1706 <1> ;;

1707 <1> ;; ..............................................................

1708 <1> ;;

1709 <1> ;; Note: If [swap\_enabled] value is ZERO, that means there is not

1710 <1> ;; a swap disk or swap file in use... 'swap\_in' and 'swap\_out'

1711 <1> ;; procedures ans 'swap page que' procedures will not be active...

1712 <1> ;; 'Insufficient memory' error will be returned by 'swap\_out'

1713 <1> ;; and 'general protection fault' will be returned by 'swap\_in'

1714 <1> ;; procedure, if it is called mistakenly (a wrong value in a PTE).

1715 <1> ;;

1716 <1>

1717 <1> swap\_in:

1718 <1> ; 31/08/2015

1719 <1> ; 20/07/2015

1720 <1> ; 28/04/2015

1721 <1> ; 18/04/2015

1722 <1> ; 24/10/2014 (Retro UNIX 386 v1 - beginning)

1723 <1> ;

1724 <1> ; INPUT ->

1725 <1> ; EBX = PHYSICAL (real/flat) ADDRESS OF THE MEMORY PAGE

1726 <1> ; EBP = VIRTUAL (LINEAR) ADDRESS (page fault address)

1727 <1> ; EAX = Offset Address for the swapped page on the

1728 <1> ; swap disk or in the swap file.

1729 <1> ;

1730 <1> ; OUTPUT ->

1731 <1> ; EAX = 0 if loading at memory has been successful

1732 <1> ;

1733 <1> ; CF = 1 -> swap disk reading error (disk/file not present

1734 <1> ; or sector not present or drive not ready

1735 <1> ; EAX = Error code

1736 <1> ; [u.error] = EAX

1737 <1> ; = The last error code for the process

1738 <1> ; (will be reset after returning to user)

1739 <1> ;

1740 <1> ; Modified Registers -> EAX

1741 <1> ;

1742 <1>

1743 00004FB9 833D[62050300]00 <1> cmp dword [swp\_drv], 0

1744 00004FC0 7646 <1> jna short swpin\_dnp\_err

1745 <1>

1746 00004FC2 3B05[66050300] <1> cmp eax, [swpd\_size]

1747 00004FC8 734A <1> jnb short swpin\_snp\_err

1748 <1>

1749 00004FCA 56 <1> push esi

1750 00004FCB 53 <1> push ebx

1751 00004FCC 51 <1> push ecx

1752 00004FCD 8B35[62050300] <1> mov esi, [swp\_drv]

1753 00004FD3 B908000000 <1> mov ecx, PAGE\_SIZE / LOGIC\_SECT\_SIZE ; 8 !

1754 <1> ; Note: Even if corresponding physical disk's sector

1755 <1> ; size different than 512 bytes, logical disk sector

1756 <1> ; size is 512 bytes and disk reading procedure

1757 <1> ; will be performed for reading 4096 bytes

1758 <1> ; (2\*2048, 8\*512).

1759 <1> ; ESI = Logical disk description table address

1760 <1> ; EBX = Memory page (buffer) address (physical!)

1761 <1> ; EAX = Sector adress (offset address, logical sector number)

1762 <1> ; ECX = Sector count ; 8 sectors

1763 00004FD8 50 <1> push eax

1764 00004FD9 E8AF020000 <1> call logical\_disk\_read

1765 00004FDE 58 <1> pop eax

1766 00004FDF 730C <1> jnc short swpin\_read\_ok

1767 <1> ;

1768 00004FE1 B828000000 <1> mov eax, SWP\_DISK\_READ\_ERR ; drive not ready or read error

1769 00004FE6 A3[C8030300] <1> mov [u.error], eax

1770 00004FEB EB17 <1> jmp short swpin\_retn

1771 <1> ;

1772 <1> swpin\_read\_ok:

1773 <1> ; EAX = Offset address (logical sector number)

1774 00004FED E80D020000 <1> call unlink\_swap\_block ; Deallocate swap block

1775 <1> ;

1776 <1> ; EBX = Memory page (buffer) address (physical!)

1777 <1> ; 20/07/2015

1778 00004FF2 89EB <1> mov ebx, ebp ; virtual address (page fault address)

1779 00004FF4 6681E300F0 <1> and bx, ~PAGE\_OFF ; ~0FFFh ; reset bits, 0 to 11

1780 00004FF9 8A1D[B3030300] <1> mov bl, [u.uno] ; current process number

1781 <1> ; EBX = Virtual (Linear) address & process number combination

1782 00004FFF E8DB000000 <1> call swap\_queue\_shift

1783 <1> ; eax = 0 ; 10/06/2016 (if ebx input > 0, eax output = 0)

1784 <1> ;sub eax, eax ; 0 ; Error Code = 0 (no error)

1785 <1> ; zf = 1

1786 <1> swpin\_retn:

1787 00005004 59 <1> pop ecx

1788 00005005 5B <1> pop ebx

1789 00005006 5E <1> pop esi

1790 00005007 C3 <1> retn

1791 <1>

1792 <1> swpin\_dnp\_err:

1793 00005008 B829000000 <1> mov eax, SWP\_DISK\_NOT\_PRESENT\_ERR

1794 <1> swpin\_err\_retn:

1795 0000500D A3[C8030300] <1> mov [u.error], eax

1796 00005012 F9 <1> stc

1797 00005013 C3 <1> retn

1798 <1>

1799 <1> swpin\_snp\_err:

1800 00005014 B82A000000 <1> mov eax, SWP\_SECTOR\_NOT\_PRESENT\_ERR

1801 00005019 EBF2 <1> jmp short swpin\_err\_retn

1802 <1>

1803 <1> swap\_out:

1804 <1> ; 10/06/2016

1805 <1> ; 07/06/2016

1806 <1> ; 23/05/2016

1807 <1> ; 19/05/2016 - TRDOS 386 (TRDOS v2.0)

1808 <1> ; 24/10/2014 - 31/08/2015 (Retro UNIX 386 v1)

1809 <1> ;

1810 <1> ; INPUT ->

1811 <1> ; none

1812 <1> ;

1813 <1> ; OUTPUT ->

1814 <1> ; EAX = Physical page address (which is swapped out

1815 <1> ; for allocating a new page)

1816 <1> ; CF = 1 -> swap disk writing error (disk/file not present

1817 <1> ; or sector not present or drive not ready

1818 <1> ; EAX = Error code

1819 <1> ; [u.error] = EAX

1820 <1> ; = The last error code for the process

1821 <1> ; (will be reset after returning to user)

1822 <1> ;

1823 <1> ; Modified Registers -> none (except EAX)

1824 <1> ;

1825 0000501B 66833D[60050300]01 <1> cmp word [swpq\_count], 1

1826 00005023 0F82AF000000 <1> jc swpout\_im\_err ; 'insufficient memory'

1827 <1>

1828 <1> ;cmp dword [swp\_drv], 1

1829 <1> ;jc short swpout\_dnp\_err ; 'swap disk/file not present'

1830 <1>

1831 00005029 833D[6A050300]01 <1> cmp dword [swpd\_free], 1

1832 00005030 0F828F000000 <1> jc swpout\_nfspc\_err ; 'no free space on swap disk'

1833 <1>

1834 00005036 53 <1> push ebx ; \*

1835 <1> swpout\_1:

1836 <1> ; 10/06/2016

1837 00005037 31DB <1> xor ebx, ebx ; shift the queue and return a PTE value

1838 00005039 E8A1000000 <1> call swap\_queue\_shift

1839 0000503E 21C0 <1> and eax, eax ; 0 = empty queue (improper entries)

1840 00005040 0F848A000000 <1> jz swpout\_npts\_err ; There is not any proper PTE

1841 <1> ; pointer in the swap queue

1842 <1> ; EAX = PTE value of the page

1843 <1> ; EBX = PTE address of the page

1844 00005046 662500F0 <1> and ax, PTE\_A\_CLEAR ; 0F000h ; clear attribute bits

1845 <1> ;

1846 <1> ; 07/06/2016

1847 <1> ; 19/05/2016

1848 <1> ; check this page is in timer events or not

1849 <1>

1850 <1> swpout\_timer\_page\_0:

1851 0000504A 52 <1> push edx ; \*\*

1852 <1>

1853 <1> ; 07/06/2016

1854 0000504B 803D[CF650100]00 <1> cmp byte [timer\_events], 0

1855 00005052 762F <1> jna short swpout\_2

1856 <1> ;

1857 00005054 8A15[CF650100] <1> mov dl, [timer\_events]

1858 <1>

1859 0000505A 51 <1> push ecx ; \*\*\*

1860 0000505B 53 <1> push ebx ; \*\*\*\*

1861 0000505C BB[60040300] <1> mov ebx, timer\_set ; beginning address of timer event

1862 <1> ; structures

1863 <1> swpout\_timer\_page\_1:

1864 00005061 8A0B <1> mov cl, [ebx]

1865 00005063 08C9 <1> or cl, cl ; 0 = free, >0 = process number

1866 00005065 7415 <1> jz short swpout\_timer\_page\_3

1867 00005067 8B4B0C <1> mov ecx, [ebx+12] ; response (signal return) address

1868 0000506A 6681E100F0 <1> and cx, PTE\_A\_CLEAR ; clear offset part (right 12 bits)

1869 <1> ; of the response byte address, to

1870 <1> ; get beginning of the page address)

1871 0000506F 39C8 <1> cmp eax, ecx

1872 00005071 7505 <1> jne short swpout\_timer\_page\_2 ; not same page

1873 <1>

1874 <1> ; !same page!

1875 <1> ;

1876 <1> ; NOTE: // 19/05/2016 // - TRDOS 386 feature only ! -

1877 <1> ; This page will be used by the kernel to put timer event

1878 <1> ; response (signal return) byte at the requested address;

1879 <1> ; in order to prevent a possible wrong write (while

1880 <1> ; this page is swapped out) on physical memory,

1881 <1> ; we must protect this page against to be swapped out!

1882 <1> ;

1883 00005073 5B <1> pop ebx ; \*\*\*\*

1884 00005074 59 <1> pop ecx ; \*\*\*

1885 00005075 5A <1> pop edx ; \*\*

1886 00005076 EBBF <1> jmp short swpout\_1 ; do not swap out this page !

1887 <1>

1888 <1> swpout\_timer\_page\_2:

1889 <1> ; 07/06/2016

1890 00005078 FECA <1> dec dl

1891 0000507A 7405 <1> jz short swpout\_timer\_page\_4

1892 <1> swpout\_timer\_page\_3:

1893 <1> ;cmp ebx, timer\_set + 240 ; last timer event (15\*16)

1894 <1> ;jnb short swpout\_timer\_page\_4

1895 0000507C 83C310 <1> add ebx, 16

1896 0000507F EBE0 <1> jmp short swpout\_timer\_page\_1

1897 <1>

1898 <1> swpout\_timer\_page\_4:

1899 00005081 5B <1> pop ebx ; \*\*\*\*

1900 00005082 59 <1> pop ecx ; \*\*\*

1901 <1> swpout\_2:

1902 00005083 89DA <1> mov edx, ebx ; Page table entry address

1903 00005085 89C3 <1> mov ebx, eax ; Buffer (Page) Address

1904 <1> ;

1905 00005087 E8A6010000 <1> call link\_swap\_block

1906 0000508C 7304 <1> jnc short swpout\_3 ; It may not be needed here

1907 <1> ; because [swpd\_free] value

1908 <1> ; was checked at the beginging.

1909 0000508E 5A <1> pop edx ; \*\*

1910 0000508F 5B <1> pop ebx ; \*

1911 00005090 EB33 <1> jmp short swpout\_nfspc\_err

1912 <1> swpout\_3:

1913 00005092 A900000080 <1> test eax, 80000000h ; test bit 31 (this may not be needed!)

1914 00005097 752C <1> jnz short swpout\_nfspc\_err ; 10/06/2016 (bit 31 = 1 !)

1915 <1> ;

1916 00005099 56 <1> push esi ; \*\*

1917 0000509A 51 <1> push ecx ; \*\*\*

1918 0000509B 50 <1> push eax ; sector address ; (31 bit !, bit 31 = 0)

1919 0000509C 8B35[62050300] <1> mov esi, [swp\_drv]

1920 000050A2 B908000000 <1> mov ecx, PAGE\_SIZE / LOGIC\_SECT\_SIZE ; 8 !

1921 <1> ; Note: Even if corresponding physical disk's sector

1922 <1> ; size different than 512 bytes, logical disk sector

1923 <1> ; size is 512 bytes and disk writing procedure

1924 <1> ; will be performed for writing 4096 bytes

1925 <1> ; (2\*2048, 8\*512).

1926 <1> ; ESI = Logical disk description table address

1927 <1> ; EBX = Buffer (Page) address

1928 <1> ; EAX = Sector adress (offset address, logical sector number)

1929 <1> ; ECX = Sector count ; 8 sectors

1930 <1> ; edx = PTE address

1931 000050A7 E8E2010000 <1> call logical\_disk\_write

1932 <1> ; edx = PTE address

1933 000050AC 59 <1> pop ecx ; sector address

1934 000050AD 730C <1> jnc short swpout\_write\_ok

1935 <1> ;

1936 <1> ;; call unlink\_swap\_block ; this block must be left as 'in use'

1937 <1> swpout\_dw\_err:

1938 000050AF B82C000000 <1> mov eax, SWP\_DISK\_WRITE\_ERR ; drive not ready or write error

1939 000050B4 A3[C8030300] <1> mov [u.error], eax

1940 000050B9 EB06 <1> jmp short swpout\_retn

1941 <1> ;

1942 <1> swpout\_write\_ok:

1943 <1> ; EBX = Buffer (page) address

1944 <1> ; EDX = Page Table Entry address

1945 <1> ; ECX = Swap disk sector (file block) address (31 bit)

1946 000050BB D1E1 <1> shl ecx, 1 ; 31 bit sector address from bit 1 to bit 31

1947 000050BD 890A <1> mov [edx], ecx

1948 <1> ; bit 0 = 0 (swapped page)

1949 000050BF 89D8 <1> mov eax, ebx

1950 <1> swpout\_retn:

1951 000050C1 59 <1> pop ecx ; \*\*\*

1952 000050C2 5E <1> pop esi ; \*\*

1953 000050C3 5B <1> pop ebx ; \*

1954 000050C4 C3 <1> retn

1955 <1>

1956 <1> ;swpout\_dnp\_err:

1957 <1> ; mov eax, SWP\_DISK\_NOT\_PRESENT\_ERR ; disk not present

1958 <1> ; jmp short swpout\_err\_retn

1959 <1> swpout\_nfspc\_err:

1960 000050C5 B82B000000 <1> mov eax, SWP\_NO\_FREE\_SPACE\_ERR ; no free space

1961 <1> swpout\_err\_retn:

1962 000050CA A3[C8030300] <1> mov [u.error], eax

1963 <1> ;stc

1964 000050CF C3 <1> retn

1965 <1> swpout\_npts\_err:

1966 000050D0 B82D000000 <1> mov eax, SWP\_NO\_PAGE\_TO\_SWAP\_ERR

1967 000050D5 5B <1> pop ebx

1968 000050D6 EBF2 <1> jmp short swpout\_err\_retn

1969 <1> swpout\_im\_err:

1970 000050D8 B804000000 <1> mov eax, ERR\_MINOR\_IM ; insufficient (out of) memory

1971 000050DD EBEB <1> jmp short swpout\_err\_retn

1972 <1>

1973 <1> swap\_queue\_shift:

1974 <1> ; 26/03/2017

1975 <1> ; 10/06/2016

1976 <1> ; 09/06/2016 - TRDOS 386 (TRDOS v2.0)

1977 <1> ; 23/10/2014 - 20/07/2015 (Retro UNIX 386 v1)

1978 <1> ;

1979 <1> ; INPUT ->

1980 <1> ; EBX = Virtual (linear) address (bit 12 to 31)

1981 <1> ; and process number combination (bit 0 to 11)

1982 <1> ; EBX = 0 -> shift/drop from the head (offset 0)

1983 <1> ;

1984 <1> ; OUTPUT ->

1985 <1> ; If EBX input > 0

1986 <1> ; the queue will be shifted 4 bytes (dword),

1987 <1> ; from the tail to the head, up to entry offset

1988 <1> ; which points to EBX input value or nothing

1989 <1> ; to do if EBX value is not found on the queue.

1990 <1> ; (The entry -with EBX value- will be removed

1991 <1> ; from the queue if it is found.)

1992 <1> ;

1993 <1> ; EAX = 0

1994 <1> ;

1995 <1> ; If EBX input = 0

1996 <1> ; the queue will be shifted 4 bytes (dword),

1997 <1> ; from the tail to the head, if the PTE address

1998 <1> ; which is pointed in head of the queue is marked

1999 <1> ; as "accessed" or it is marked as "non present".

2000 <1> ; (If "accessed" flag of the PTE -which is pointed

2001 <1> ; in the head- is set -to 1-, it will be reset

2002 <1> ; -to 0- and then, the queue will be rotated

2003 <1> ; -without dropping pointer of the PTE from

2004 <1> ; the queue- for 4 bytes on head to tail direction.

2005 <1> ; Pointer in the head will be moved into the tail,

2006 <1> ; other PTEs will be shifted on head direction.)

2007 <1> ;

2008 <1> ; Swap queue will be shifted up to the first

2009 <1> ; 'present' or 'non accessed' page will be found

2010 <1> ; (as pointed) on the queue head (then it will be

2011 <1> ; removed/dropped from the queue).

2012 <1> ;

2013 <1> ; EAX (> 0) = PTE value of the page which is

2014 <1> ; (it's pointer -virtual address-) dropped

2015 <1> ; (removed) from swap queue.

2016 <1> ; EBX = PTE address of the page (if EAX > 0)

2017 <1> ; which is (it's pointer -virtual address-)

2018 <1> ; dropped (removed) from swap queue.

2019 <1> ;

2020 <1> ; EAX = 0 -> empty swap queue !

2021 <1> ;

2022 <1> ; Modified Registers -> EAX, EBX

2023 <1> ;

2024 000050DF 0FB705[60050300] <1> movzx eax, word [swpq\_count] ; Max. 1024

2025 000050E6 6621C0 <1> and ax, ax

2026 000050E9 7431 <1> jz short swpqs\_retn

2027 000050EB 57 <1> push edi

2028 000050EC 56 <1> push esi

2029 000050ED 51 <1> push ecx

2030 000050EE BE00E00800 <1> mov esi, swap\_queue

2031 000050F3 89C1 <1> mov ecx, eax

2032 000050F5 09DB <1> or ebx, ebx

2033 000050F7 7424 <1> jz short swpqs\_7

2034 <1> swpqs\_1:

2035 000050F9 AD <1> lodsd

2036 000050FA 39D8 <1> cmp eax, ebx

2037 000050FC 7406 <1> je short swpqs\_2

2038 000050FE E2F9 <1> loop swpqs\_1

2039 <1> ; 10/06/2016

2040 00005100 29C0 <1> sub eax, eax

2041 00005102 EB15 <1> jmp short swpqs\_6

2042 <1> swpqs\_2:

2043 00005104 89F7 <1> mov edi, esi

2044 00005106 83EF04 <1> sub edi, 4

2045 <1> swpqs\_3:

2046 00005109 66FF0D[60050300] <1> dec word [swpq\_count]

2047 00005110 7403 <1> jz short swpqs\_5

2048 <1> swpqs\_4:

2049 00005112 49 <1> dec ecx

2050 00005113 F3A5 <1> rep movsd ; shift up (to the head)

2051 <1> swpqs\_5:

2052 00005115 31C0 <1> xor eax, eax

2053 00005117 8907 <1> mov [edi], eax

2054 <1> swpqs\_6:

2055 00005119 59 <1> pop ecx

2056 0000511A 5E <1> pop esi

2057 0000511B 5F <1> pop edi

2058 <1> swpqs\_retn:

2059 0000511C C3 <1> retn

2060 <1> swpqs\_7:

2061 0000511D 89F7 <1> mov edi, esi ; head

2062 0000511F AD <1> lodsd

2063 <1> ; 20/07/2015

2064 00005120 89C3 <1> mov ebx, eax

2065 00005122 81E300F0FFFF <1> and ebx, ~PAGE\_OFF ; ~0FFFh

2066 <1> ; ebx = virtual address (at page boundary)

2067 00005128 25FF0F0000 <1> and eax, PAGE\_OFF ; 0FFFh

2068 <1> ; ax = process number (1 to 4095)

2069 0000512D 3A05[B3030300] <1> cmp al, [u.uno]

2070 <1> ; Max. 16 (nproc) processes for Retro UNIX 386 v1

2071 00005133 7507 <1> jne short swpqs\_8

2072 00005135 A1[B8030300] <1> mov eax, [u.pgdir]

2073 0000513A EB28 <1> jmp short swpqs\_9

2074 <1> swpqs\_8:

2075 <1> ; 09/06/2016

2076 0000513C 80B8[AF000300]00 <1> cmp byte [eax+p.stat-1], 0

2077 00005143 76C4 <1> jna short swpqs\_3 ; free (or terminated) process

2078 00005145 80B8[AF000300]02 <1> cmp byte [eax+p.stat-1], 2 ; waiting

2079 0000514C 77BB <1> ja short swpqs\_3 ; zombie (3) or undefined ?

2080 <1>

2081 <1> ;shl ax, 2

2082 0000514E C0E002 <1> shl al, 2

2083 00005151 8B80[BC000300] <1> mov eax, [eax+p.upage-4]

2084 00005157 09C0 <1> or eax, eax

2085 00005159 74AE <1> jz short swpqs\_3 ; invalid upage

2086 0000515B 83C05C <1> add eax, u.pgdir - user

2087 <1> ; u.pgdir value for the process

2088 <1> ; is in [eax]

2089 0000515E 8B00 <1> mov eax, [eax]

2090 00005160 21C0 <1> and eax, eax

2091 00005162 74A5 <1> jz short swpqs\_3 ; invalid page directory

2092 <1> swpqs\_9:

2093 00005164 52 <1> push edx

2094 <1> ; eax = page directory

2095 <1> ; ebx = virtual address

2096 00005165 E82BFBFFFF <1> call get\_pte

2097 0000516A 89D3 <1> mov ebx, edx ; PTE address

2098 0000516C 5A <1> pop edx

2099 <1> ; 10/06/2016

2100 0000516D 723A <1> jc short swpqs\_13 ; empty PDE

2101 <1> ; EAX = PTE value

2102 0000516F A801 <1> test al, PTE\_A\_PRESENT ; bit 0 = 1

2103 00005171 7436 <1> jz short swpqs\_13 ; Drop non-present page

2104 <1> ; from the queue (head)

2105 00005173 A802 <1> test al, PTE\_A\_WRITE ; bit 1 = 0 (read only)

2106 00005175 7432 <1> jz short swpqs\_13 ; Drop read only page

2107 <1> ; from the queue (head)

2108 <1> ;test al, PTE\_A\_ACCESS ; bit 5 = 1 (Accessed)

2109 <1> ;jnz short swpqs\_11 ; present

2110 <1> ; accessed page

2111 00005177 0FBAF005 <1> btr eax, PTE\_A\_ACCESS\_BIT ; reset 'accessed' bit

2112 0000517B 7210 <1> jc short swpqs\_11 ; accessed page

2113 <1>

2114 0000517D 49 <1> dec ecx

2115 0000517E 66890D[60050300] <1> mov [swpq\_count], cx

2116 00005185 7402 <1> jz short swpqs\_10

2117 <1> ; esi = head + 4

2118 <1> ; edi = head

2119 00005187 F3A5 <1> rep movsd ; n = 1 to k-1, [n - 1] = [n]

2120 <1> swpqs\_10:

2121 00005189 890F <1> mov [edi], ecx ; 0

2122 0000518B EB8C <1> jmp short swpqs\_6 ; 26/03/2017

2123 <1>

2124 <1> swpqs\_11:

2125 0000518D 8903 <1> mov [ebx], eax ; save changed attribute

2126 <1> ; Rotation (head -> tail)

2127 0000518F 49 <1> dec ecx ; entry count -> last entry number

2128 00005190 74F7 <1> jz short swpqs\_10

2129 <1> ; esi = head + 4

2130 <1> ; edi = head

2131 00005192 8B07 <1> mov eax, [edi] ; 20/07/2015

2132 00005194 F3A5 <1> rep movsd ; n = 1 to k-1, [n - 1] = [n]

2133 00005196 8907 <1> mov [edi], eax ; head -> tail ; [k] = [1]

2134 <1>

2135 00005198 668B0D[60050300] <1> mov cx, [swpq\_count]

2136 <1>

2137 <1> swpqs\_12:

2138 0000519F BE00E00800 <1> mov esi, swap\_queue ; head

2139 000051A4 E974FFFFFF <1> jmp swpqs\_7

2140 <1>

2141 <1> swpqs\_13:

2142 000051A9 49 <1> dec ecx

2143 000051AA 66890D[60050300] <1> mov [swpq\_count], cx

2144 000051B1 0F845EFFFFFF <1> jz swpqs\_5

2145 000051B7 EBE6 <1> jmp short swpqs\_12

2146 <1>

2147 <1> add\_to\_swap\_queue:

2148 <1> ; temporary - 16/09/2015

2149 000051B9 C3 <1> retn

2150 <1> ; 20/02/2017

2151 <1> ; 20/07/2015

2152 <1> ; 24/10/2014 (Retro UNIX 386 v1 - beginning)

2153 <1> ;

2154 <1> ; Adds new page to swap queue

2155 <1> ; (page directories and page tables must not be added

2156 <1> ; to swap queue)

2157 <1> ;

2158 <1> ; INPUT ->

2159 <1> ; EBX = Linear (Virtual) addr for current process

2160 <1> ; [u.uno]

2161 <1> ; 20/02/2017

2162 <1> ; (Linear address = CORE + user's virtual address)

2163 <1> ;

2164 <1> ; OUTPUT ->

2165 <1> ; EAX = [swpq\_count]

2166 <1> ; (after the PTE has been added)

2167 <1> ; EAX = 0 -> Swap queue is full, (1024 entries)

2168 <1> ; the PTE could not be added.

2169 <1> ;

2170 <1> ; Modified Registers -> EAX

2171 <1> ;

2172 000051BA 53 <1> push ebx

2173 000051BB 6681E300F0 <1> and bx, ~PAGE\_OFF ; ~0FFFh ; reset bits, 0 to 11

2174 000051C0 8A1D[B3030300] <1> mov bl, [u.uno] ; current process number

2175 000051C6 E814FFFFFF <1> call swap\_queue\_shift ; drop from the queue if

2176 <1> ; it is already on the queue

2177 <1> ; then add it to the tail of the queue

2178 000051CB 0FB705[60050300] <1> movzx eax, word [swpq\_count]

2179 000051D2 663D0004 <1> cmp ax, 1024

2180 000051D6 7205 <1> jb short atsq\_1

2181 000051D8 6629C0 <1> sub ax, ax

2182 000051DB 5B <1> pop ebx

2183 000051DC C3 <1> retn

2184 <1> atsq\_1:

2185 000051DD 56 <1> push esi

2186 000051DE BE00E00800 <1> mov esi, swap\_queue

2187 000051E3 6621C0 <1> and ax, ax

2188 000051E6 740A <1> jz short atsq\_2

2189 000051E8 66C1E002 <1> shl ax, 2 ; convert to offset

2190 000051EC 01C6 <1> add esi, eax

2191 000051EE 66C1E802 <1> shr ax, 2

2192 <1> atsq\_2:

2193 000051F2 6640 <1> inc ax

2194 000051F4 891E <1> mov [esi], ebx ; Virtual address + [u.uno] combination

2195 000051F6 66A3[60050300] <1> mov [swpq\_count], ax

2196 000051FC 5E <1> pop esi

2197 000051FD 5B <1> pop ebx

2198 000051FE C3 <1> retn

2199 <1>

2200 <1> unlink\_swap\_block:

2201 <1> ; 15/09/2015

2202 <1> ; 30/04/2015

2203 <1> ; 18/04/2015

2204 <1> ; 24/10/2014 (Retro UNIX 386 v1 - beginning)

2205 <1> ;

2206 <1> ; INPUT ->

2207 <1> ; EAX = swap disk/file offset address

2208 <1> ; (bit 1 to bit 31)

2209 <1> ; OUTPUT ->

2210 <1> ; [swpd\_free] is increased

2211 <1> ; (corresponding SWAP DISK ALLOC. TABLE bit is SET)

2212 <1> ;

2213 <1> ; Modified Registers -> EAX

2214 <1> ;

2215 000051FF 53 <1> push ebx

2216 00005200 52 <1> push edx

2217 <1> ;

2218 00005201 C1E804 <1> shr eax, SECTOR\_SHIFT+1 ;3+1 ; shift sector address to

2219 <1> ; 3 bits right

2220 <1> ; to get swap block/page number

2221 00005204 89C2 <1> mov edx, eax

2222 <1> ; 15/09/2015

2223 00005206 C1EA03 <1> shr edx, 3 ; to get offset to S.A.T.

2224 <1> ; (1 allocation bit = 1 page)

2225 <1> ; (1 allocation bytes = 8 pages)

2226 00005209 80E2FC <1> and dl, 0FCh ; clear lower 2 bits

2227 <1> ; (to get 32 bit position)

2228 <1> ;

2229 0000520C BB00000D00 <1> mov ebx, swap\_alloc\_table ; Swap Allocation Table address

2230 00005211 01D3 <1> add ebx, edx

2231 00005213 83E01F <1> and eax, 1Fh ; lower 5 bits only

2232 <1> ; (allocation bit position)

2233 00005216 3B05[6E050300] <1> cmp eax, [swpd\_next] ; is the new free block addr. lower

2234 <1> ; than the address in 'swpd\_next' ?

2235 <1> ; (next/first free block value)

2236 0000521C 7305 <1> jnb short uswpbl\_1 ; no

2237 0000521E A3[6E050300] <1> mov [swpd\_next], eax ; yes

2238 <1> uswpbl\_1:

2239 00005223 0FAB03 <1> bts [ebx], eax ; unlink/release/deallocate block

2240 <1> ; set relevant bit to 1.

2241 <1> ; set CF to the previous bit value

2242 00005226 F5 <1> cmc ; complement carry flag

2243 00005227 7206 <1> jc short uswpbl\_2 ; do not increase swfd\_free count

2244 <1> ; if the block is already deallocated

2245 <1> ; before.

2246 00005229 FF05[6A050300] <1> inc dword [swpd\_free]

2247 <1> uswpbl\_2:

2248 0000522F 5A <1> pop edx

2249 00005230 5B <1> pop ebx

2250 00005231 C3 <1> retn

2251 <1>

2252 <1> link\_swap\_block:

2253 <1> ; 01/07/2015

2254 <1> ; 18/04/2015

2255 <1> ; 24/10/2014 (Retro UNIX 386 v1 - beginning)

2256 <1> ;

2257 <1> ; INPUT -> none

2258 <1> ;

2259 <1> ; OUTPUT ->

2260 <1> ; EAX = OFFSET ADDRESS OF THE ALLOCATED BLOCK (4096 bytes)

2261 <1> ; in sectors (corresponding

2262 <1> ; SWAP DISK ALLOCATION TABLE bit is RESET)

2263 <1> ;

2264 <1> ; CF = 1 and EAX = 0

2265 <1> ; if there is not a free block to be allocated

2266 <1> ;

2267 <1> ; Modified Registers -> none (except EAX)

2268 <1> ;

2269 <1>

2270 <1> ;mov eax, [swpd\_free]

2271 <1> ;and eax, eax

2272 <1> ;jz short out\_of\_swpspc

2273 <1> ;

2274 00005232 53 <1> push ebx

2275 00005233 51 <1> push ecx

2276 <1> ;

2277 00005234 BB00000D00 <1> mov ebx, swap\_alloc\_table ; Swap Allocation Table offset

2278 00005239 89D9 <1> mov ecx, ebx

2279 0000523B 031D[6E050300] <1> add ebx, [swpd\_next] ; Free block searching starts from here

2280 <1> ; next\_free\_swap\_block >> 5

2281 00005241 030D[72050300] <1> add ecx, [swpd\_last] ; Free block searching ends here

2282 <1> ; (total\_swap\_blocks - 1) >> 5

2283 <1> lswbl\_scan:

2284 00005247 39CB <1> cmp ebx, ecx

2285 00005249 770A <1> ja short lswbl\_notfound

2286 <1> ;

2287 0000524B 0FBC03 <1> bsf eax, [ebx] ; Scans source operand for first bit set (1).

2288 <1> ; Clears ZF if a bit is found set (1) and

2289 <1> ; loads the destination with an index to

2290 <1> ; first set bit. (0 -> 31)

2291 <1> ; Sets ZF to 1 if no bits are found set.

2292 <1> ; 01/07/2015

2293 0000524E 751C <1> jnz short lswbl\_found ; ZF = 0 -> a free block has been found

2294 <1> ;

2295 <1> ; NOTE: a Swap Disk Allocation Table bit

2296 <1> ; with value of 1 means

2297 <1> ; the corresponding page is free

2298 <1> ; (Retro UNIX 386 v1 feaure only!)

2299 00005250 83C304 <1> add ebx, 4

2300 <1> ; We return back for searching next page block

2301 <1> ; NOTE: [swpd\_free] is not ZERO; so,

2302 <1> ; we always will find at least 1 free block here.

2303 00005253 EBF2 <1> jmp short lswbl\_scan

2304 <1> ;

2305 <1> lswbl\_notfound:

2306 00005255 81E900000D00 <1> sub ecx, swap\_alloc\_table

2307 0000525B 890D[6E050300] <1> mov [swpd\_next], ecx ; next/first free page = last page

2308 <1> ; (unlink\_swap\_block procedure will change it)

2309 00005261 31C0 <1> xor eax, eax

2310 00005263 A3[6A050300] <1> mov [swpd\_free], eax

2311 00005268 F9 <1> stc

2312 <1> lswbl\_ok:

2313 00005269 59 <1> pop ecx

2314 0000526A 5B <1> pop ebx

2315 0000526B C3 <1> retn

2316 <1> ;

2317 <1> ;out\_of\_swpspc:

2318 <1> ; stc

2319 <1> ; retn

2320 <1>

2321 <1> lswbl\_found:

2322 0000526C 89D9 <1> mov ecx, ebx

2323 0000526E 81E900000D00 <1> sub ecx, swap\_alloc\_table

2324 00005274 890D[6E050300] <1> mov [swpd\_next], ecx ; Set first free block searching start

2325 <1> ; address/offset (to the next)

2326 0000527A FF0D[6A050300] <1> dec dword [swpd\_free] ; 1 block has been allocated (X = X-1)

2327 <1> ;

2328 00005280 0FB303 <1> btr [ebx], eax ; The destination bit indexed by the source value

2329 <1> ; is copied into the Carry Flag and then cleared

2330 <1> ; in the destination.

2331 <1> ;

2332 <1> ; Reset the bit which is corresponding to the

2333 <1> ; (just) allocated block.

2334 00005283 C1E105 <1> shl ecx, 5 ; (block offset \* 32) + block index

2335 00005286 01C8 <1> add eax, ecx ; = block number

2336 00005288 C1E003 <1> shl eax, SECTOR\_SHIFT ; 3, sector (offset) address of the block

2337 <1> ; 1 block = 8 sectors

2338 <1> ;

2339 <1> ; EAX = offset address of swap disk/file sector (beginning of the block)

2340 <1> ;

2341 <1> ; NOTE: The relevant page table entry will be updated

2342 <1> ; according to this EAX value...

2343 <1> ;

2344 0000528B EBDC <1> jmp short lswbl\_ok

2345 <1>

2346 <1> logical\_disk\_read:

2347 <1> ; 20/07/2015

2348 <1> ; 09/03/2015 (temporary code here)

2349 <1> ;

2350 <1> ; INPUT ->

2351 <1> ; ESI = Logical disk description table address

2352 <1> ; EBX = Memory page (buffer) address (physical!)

2353 <1> ; EAX = Sector adress (offset address, logical sector number)

2354 <1> ; ECX = Sector count

2355 <1> ;

2356 <1> ;

2357 0000528D C3 <1> retn

2358 <1>

2359 <1> logical\_disk\_write:

2360 <1> ; 20/07/2015

2361 <1> ; 09/03/2015 (temporary code here)

2362 <1> ;

2363 <1> ; INPUT ->

2364 <1> ; ESI = Logical disk description table address

2365 <1> ; EBX = Memory page (buffer) address (physical!)

2366 <1> ; EAX = Sector adress (offset address, logical sector number)

2367 <1> ; ECX = Sector count

2368 <1> ;

2369 0000528E C3 <1> retn

2370 <1>

2371 <1> get\_physical\_addr:

2372 <1> ; 26/03/2017

2373 <1> ; 20/02/2017

2374 <1> ; 27/05/2016 - TRDOS 386 (TRDOS v2.0)

2375 <1> ; 18/10/2015

2376 <1> ; 29/07/2015

2377 <1> ; 20/07/2015

2378 <1> ; 04/06/2015

2379 <1> ; 20/05/2015

2380 <1> ; 28/04/2015

2381 <1> ; 18/04/2015

2382 <1> ; Get physical address

2383 <1> ; (allocates a new page for user if it is not present)

2384 <1> ;

2385 <1> ; (This subroutine is needed for mapping user's virtual

2386 <1> ; (buffer) address to physical address (of the buffer).)

2387 <1> ; ('sys write', 'sys read' system calls...)

2388 <1> ;

2389 <1> ; INPUT ->

2390 <1> ; EBX = virtual address

2391 <1> ; u.pgdir = page directory (physical) address

2392 <1> ;

2393 <1> ; OUTPUT ->

2394 <1> ; EAX = physical address

2395 <1> ; EBX = linear address

2396 <1> ; EDX = physical address of the page frame

2397 <1> ; (with attribute bits)

2398 <1> ; ECX = byte count within the page frame

2399 <1> ;

2400 <1> ; Modified Registers -> EAX, EBX, ECX, EDX

2401 <1> ;

2402 0000528F 81C300004000 <1> add ebx, CORE ; 18/10/2015

2403 <1> get\_physical\_addr\_x: ; 27/05/2016

2404 00005295 A1[B8030300] <1> mov eax, [u.pgdir]

2405 0000529A E8F6F9FFFF <1> call get\_pte

2406 <1> ; EDX = Page table entry address (if CF=0)

2407 <1> ; Page directory entry address (if CF=1)

2408 <1> ; (Bit 0 value is 0 if PT is not present)

2409 <1> ; EAX = Page table entry value (page address)

2410 <1> ; CF = 1 -> PDE not present or invalid ?

2411 0000529F 731C <1> jnc short gpa\_1

2412 <1> ;

2413 000052A1 E8D4F8FFFF <1> call allocate\_page

2414 000052A6 7248 <1> jc short gpa\_im\_err ; 'insufficient memory' error

2415 <1> gpa\_0:

2416 000052A8 E847F9FFFF <1> call clear\_page

2417 <1> ; EAX = Physical (base) address of the allocated (new) page

2418 000052AD 0C07 <1> or al, PDE\_A\_PRESENT + PDE\_A\_WRITE + PDE\_A\_USER ; 4+2+1 = 7

2419 <1> ; lower 3 bits are used as U/S, R/W, P flags

2420 <1> ; (user, writable, present page)

2421 000052AF 8902 <1> mov [edx], eax ; Let's put the new page directory entry here !

2422 000052B1 A1[B8030300] <1> mov eax, [u.pgdir]

2423 000052B6 E8DAF9FFFF <1> call get\_pte

2424 000052BB 7233 <1> jc short gpa\_im\_err ; 'insufficient memory' error

2425 <1> gpa\_1:

2426 <1> ; EAX = PTE value, EDX = PTE address

2427 000052BD A801 <1> test al, PTE\_A\_PRESENT

2428 000052BF 751F <1> jnz short gpa\_3 ; 26/03/2017

2429 000052C1 09C0 <1> or eax, eax

2430 000052C3 7456 <1> jz short gpa\_7 ; Allocate a new page

2431 <1> ; 20/07/2015

2432 000052C5 55 <1> push ebp

2433 000052C6 89DD <1> mov ebp, ebx ; virtual (linear) address

2434 <1> ; reload swapped page

2435 000052C8 E878000000 <1> call reload\_page ; 28/04/2015

2436 000052CD 5D <1> pop ebp

2437 000052CE 724A <1> jc short gpa\_retn

2438 <1> gpa\_2:

2439 <1> ; 26/03/2017

2440 <1> ; 20/02/2017

2441 <1> ; If a page will contain a Signal Response Byte

2442 <1> ; it must not be swapped out, because

2443 <1> ; timer service or irq callback service

2444 <1> ; will write a signal return/response byte

2445 <1> ; directly by using physical address of Signal

2446 <1> ; Response Byte.(Even if process is not running,

2447 <1> ; or it is running with swapped out pages.)

2448 <1> ;

2449 <1> ; 'no\_page\_swap' will be set by 'systimer' or

2450 <1> ; 'syscalbac' sistem functions/calls. (\*)

2451 <1> ;

2452 000052D0 803D[0E6B0100]00 <1> cmp byte [no\_page\_swap], 0

2453 000052D7 761D <1> jna short gpa\_4 ; this page can be swapped out

2454 <1> ; this page must not be swapped out

2455 <1> ; but 'no\_page\_swap' must be reset here

2456 <1> ; imediately for other callers (\*)

2457 <1> ; (otherwise, swap queue would not be long enough)

2458 000052D9 E84B000000 <1> call gpa\_8 ; 26/03/2017

2459 000052DE EB1D <1> jmp short gpa\_5

2460 <1> gpa\_3:

2461 <1> ; 26/03/2017

2462 000052E0 803D[0E6B0100]00 <1> cmp byte [no\_page\_swap], 0

2463 000052E7 7618 <1> jna short gpa\_6 ; this page can be swapped out

2464 000052E9 E83B000000 <1> call gpa\_8

2465 000052EE EB11 <1> jmp short gpa\_6

2466 <1>

2467 <1> gpa\_im\_err:

2468 000052F0 B804000000 <1> mov eax, ERR\_MINOR\_IM ; Insufficient memory (minor) error!

2469 <1> ; Major error = 0 (No protection fault)

2470 000052F5 C3 <1> retn

2471 <1> gpa\_4:

2472 <1> ; 20/07/2015

2473 <1> ; 20/05/2015

2474 <1> ; add this page to swap queue

2475 000052F6 50 <1> push eax

2476 <1> ; EBX = Linear (CORE+virtual) address ; 20/02/2017

2477 000052F7 E8BDFEFFFF <1> call add\_to\_swap\_queue

2478 000052FC 58 <1> pop eax

2479 <1> gpa\_5:

2480 <1> ; PTE address in EDX

2481 <1> ; virtual address in EBX

2482 <1> ; EAX = memory page address

2483 000052FD 0C07 <1> or al, PTE\_A\_PRESENT + PTE\_A\_USER + PTE\_A\_WRITE

2484 <1> ; present flag, bit 0 = 1

2485 <1> ; user flag, bit 2 = 1

2486 <1> ; writable flag, bit 1 = 1

2487 000052FF 8902 <1> mov [edx], eax ; Update PTE value

2488 <1> gpa\_6:

2489 <1> ; 18/10/2015

2490 00005301 89D9 <1> mov ecx, ebx

2491 00005303 81E1FF0F0000 <1> and ecx, PAGE\_OFF

2492 00005309 89C2 <1> mov edx, eax

2493 0000530B 662500F0 <1> and ax, PTE\_A\_CLEAR

2494 0000530F 01C8 <1> add eax, ecx

2495 00005311 F7D9 <1> neg ecx ; 1 -> -1 (0FFFFFFFFh), 4095 (0FFFh) -> -4095

2496 00005313 81C100100000 <1> add ecx, PAGE\_SIZE

2497 00005319 F8 <1> clc

2498 <1> gpa\_retn:

2499 0000531A C3 <1> retn

2500 <1> gpa\_7:

2501 0000531B E85AF8FFFF <1> call allocate\_page

2502 00005320 72CE <1> jc short gpa\_im\_err ; 'insufficient memory' error

2503 00005322 E8CDF8FFFF <1> call clear\_page

2504 00005327 EBA7 <1> jmp short gpa\_2

2505 <1>

2506 <1> gpa\_8: ; 26/03/2017

2507 00005329 C605[0E6B0100]00 <1> mov byte [no\_page\_swap], 0

2508 00005330 53 <1> push ebx

2509 00005331 50 <1> push eax ; 26/03/2017

2510 00005332 6681E300F0 <1> and bx, ~PAGE\_OFF ; ~0FFFh ; reset bits, 0 to 11

2511 00005337 8A1D[B3030300] <1> mov bl, [u.uno] ; current process number

2512 0000533D E89DFDFFFF <1> call swap\_queue\_shift ; drop from the queue if

2513 <1> ; it is already on the queue

2514 00005342 58 <1> pop eax ; 26/03/2017

2515 00005343 5B <1> pop ebx

2516 00005344 C3 <1> retn

2517 <1>

2518 <1> reload\_page:

2519 <1> ; 20/07/2015

2520 <1> ; 28/04/2015 (Retro UNIX 386 v1 - beginning)

2521 <1> ;

2522 <1> ; Reload (Restore) swapped page at memory

2523 <1> ;

2524 <1> ; INPUT ->

2525 <1> ; EBP = Virtual (linear) memory address

2526 <1> ; EAX = PTE value (swap disk sector address)

2527 <1> ; (Swap disk sector address = bit 1 to bit 31 of EAX)

2528 <1> ; OUTPUT ->

2529 <1> ; EAX = PHYSICAL (real/flat) ADDRESS OF RELOADED PAGE

2530 <1> ;

2531 <1> ; CF = 1 and EAX = error code

2532 <1> ;

2533 <1> ; Modified Registers -> none (except EAX)

2534 <1> ;

2535 00005345 D1E8 <1> shr eax, 1 ; Convert PTE value to swap disk address

2536 00005347 53 <1> push ebx ;

2537 00005348 89C3 <1> mov ebx, eax ; Swap disk (offset) address

2538 0000534A E82BF8FFFF <1> call allocate\_page

2539 0000534F 720C <1> jc short rlp\_im\_err

2540 00005351 93 <1> xchg eax, ebx

2541 <1> ; EBX = Physical memory (page) address

2542 <1> ; EAX = Swap disk (offset) address

2543 <1> ; EBP = Virtual (linear) memory address

2544 00005352 E862FCFFFF <1> call swap\_in

2545 00005357 720B <1> jc short rlp\_swp\_err ; (swap disk/file read error)

2546 00005359 89D8 <1> mov eax, ebx

2547 <1> rlp\_retn:

2548 0000535B 5B <1> pop ebx

2549 0000535C C3 <1> retn

2550 <1>

2551 <1> rlp\_im\_err:

2552 0000535D B804000000 <1> mov eax, ERR\_MINOR\_IM ; Insufficient memory (minor) error!

2553 <1> ; Major error = 0 (No protection fault)

2554 00005362 EBF7 <1> jmp short rlp\_retn

2555 <1>

2556 <1> rlp\_swp\_err:

2557 00005364 B828000000 <1> mov eax, SWP\_DISK\_READ\_ERR ; Swap disk read error !

2558 00005369 EBF0 <1> jmp short rlp\_retn

2559 <1>

2560 <1>

2561 <1> copy\_page\_dir:

2562 <1> ; 19/09/2015

2563 <1> ; temporary - 07/09/2015

2564 <1> ; 07/09/2015 (Retro UNIX 386 v1 - beginning)

2565 <1> ;

2566 <1> ; INPUT ->

2567 <1> ; [u.pgdir] = PHYSICAL (real/flat) ADDRESS of the parent's

2568 <1> ; page directory.

2569 <1> ; OUTPUT ->

2570 <1> ; EAX = PHYSICAL (real/flat) ADDRESS of the child's

2571 <1> ; page directory.

2572 <1> ; (New page directory with new page table entries.)

2573 <1> ; (New page tables with read only copies of the parent's

2574 <1> ; pages.)

2575 <1> ; EAX = 0 -> Error (CF = 1)

2576 <1> ;

2577 <1> ; Modified Registers -> none (except EAX)

2578 <1> ;

2579 0000536B E80AF8FFFF <1> call allocate\_page

2580 00005370 723E <1> jc short cpd\_err

2581 <1> ;

2582 00005372 55 <1> push ebp ; 20/07/2015

2583 00005373 56 <1> push esi

2584 00005374 57 <1> push edi

2585 00005375 53 <1> push ebx

2586 00005376 51 <1> push ecx

2587 00005377 8B35[B8030300] <1> mov esi, [u.pgdir]

2588 0000537D 89C7 <1> mov edi, eax

2589 0000537F 50 <1> push eax ; save child's page directory address

2590 <1> ; copy PDE 0 from the parent's page dir to the child's page dir

2591 <1> ; (use same system space for all user page tables)

2592 00005380 A5 <1> movsd

2593 00005381 BD00004000 <1> mov ebp, 1024\*4096 ; pass the 1st 4MB (system space)

2594 00005386 B9FF030000 <1> mov ecx, (PAGE\_SIZE / 4) - 1 ; 1023

2595 <1> cpd\_0:

2596 0000538B AD <1> lodsd

2597 <1> ;or eax, eax

2598 <1> ;jnz short cpd\_1

2599 0000538C A801 <1> test al, PDE\_A\_PRESENT ; bit 0 = 1

2600 0000538E 7508 <1> jnz short cpd\_1

2601 <1> ; (virtual address at the end of the page table)

2602 00005390 81C500004000 <1> add ebp, 1024\*4096 ; page size \* PTE count

2603 00005396 EB0F <1> jmp short cpd\_2

2604 <1> cpd\_1:

2605 00005398 662500F0 <1> and ax, PDE\_A\_CLEAR ; 0F000h ; clear attribute bits

2606 0000539C 89C3 <1> mov ebx, eax

2607 <1> ; EBX = Parent's page table address

2608 0000539E E81F000000 <1> call copy\_page\_table

2609 000053A3 720C <1> jc short cpd\_p\_err

2610 <1> ; EAX = Child's page table address

2611 000053A5 0C07 <1> or al, PDE\_A\_PRESENT + PDE\_A\_WRITE + PDE\_A\_USER

2612 <1> ; set bit 0, bit 1 and bit 2 to 1

2613 <1> ; (present, writable, user)

2614 <1> cpd\_2:

2615 000053A7 AB <1> stosd

2616 000053A8 E2E1 <1> loop cpd\_0

2617 <1> ;

2618 000053AA 58 <1> pop eax ; restore child's page directory address

2619 <1> cpd\_3:

2620 000053AB 59 <1> pop ecx

2621 000053AC 5B <1> pop ebx

2622 000053AD 5F <1> pop edi

2623 000053AE 5E <1> pop esi

2624 000053AF 5D <1> pop ebp

2625 <1> cpd\_err:

2626 000053B0 C3 <1> retn

2627 <1> cpd\_p\_err:

2628 <1> ; release the allocated pages missing (recover free space)

2629 000053B1 58 <1> pop eax ; the new page directory address (physical)

2630 000053B2 8B1D[B8030300] <1> mov ebx, [u.pgdir] ; parent's page directory address

2631 000053B8 E8F6F8FFFF <1> call deallocate\_page\_dir

2632 000053BD 29C0 <1> sub eax, eax ; 0

2633 000053BF F9 <1> stc

2634 000053C0 EBE9 <1> jmp short cpd\_3

2635 <1>

2636 <1> copy\_page\_table:

2637 <1> ; 19/09/2015

2638 <1> ; temporary - 07/09/2015

2639 <1> ; 07/09/2015 (Retro UNIX 386 v1 - beginning)

2640 <1> ;

2641 <1> ; INPUT ->

2642 <1> ; EBX = PHYSICAL (real/flat) ADDRESS of the parent's page table.

2643 <1> ; EBP = page table entry index (from 'copy\_page\_dir')

2644 <1> ; OUTPUT ->

2645 <1> ; EAX = PHYSICAL (real/flat) ADDRESS of the child's page table.

2646 <1> ; EBP = (recent) page table index (for 'add\_to\_swap\_queue')

2647 <1> ; CF = 1 -> error

2648 <1> ;

2649 <1> ; Modified Registers -> EBP (except EAX)

2650 <1> ;

2651 000053C2 E8B3F7FFFF <1> call allocate\_page

2652 000053C7 725A <1> jc short cpt\_err

2653 <1> ;

2654 000053C9 50 <1> push eax ; \*

2655 <1> ;push ebx

2656 000053CA 56 <1> push esi

2657 000053CB 57 <1> push edi

2658 000053CC 52 <1> push edx

2659 000053CD 51 <1> push ecx

2660 <1> ;

2661 000053CE 89DE <1> mov esi, ebx

2662 000053D0 89C7 <1> mov edi, eax

2663 000053D2 89C2 <1> mov edx, eax

2664 000053D4 81C200100000 <1> add edx, PAGE\_SIZE

2665 <1> cpt\_0:

2666 000053DA AD <1> lodsd

2667 000053DB A801 <1> test al, PTE\_A\_PRESENT ; bit 0 = 1

2668 000053DD 750B <1> jnz short cpt\_1

2669 000053DF 21C0 <1> and eax, eax

2670 000053E1 7430 <1> jz short cpt\_2

2671 <1> ; ebp = virtual (linear) address of the memory page

2672 000053E3 E85DFFFFFF <1> call reload\_page ; 28/04/2015

2673 000053E8 7234 <1> jc short cpt\_p\_err

2674 <1> cpt\_1:

2675 000053EA 662500F0 <1> and ax, PTE\_A\_CLEAR ; 0F000h ; clear attribute bits

2676 000053EE 89C1 <1> mov ecx, eax

2677 <1> ; Allocate a new page for the child process

2678 000053F0 E885F7FFFF <1> call allocate\_page

2679 000053F5 7227 <1> jc short cpt\_p\_err

2680 000053F7 57 <1> push edi

2681 000053F8 56 <1> push esi

2682 000053F9 89CE <1> mov esi, ecx

2683 000053FB 89C7 <1> mov edi, eax

2684 000053FD B900040000 <1> mov ecx, PAGE\_SIZE/4

2685 00005402 F3A5 <1> rep movsd ; copy page (4096 bytes)

2686 00005404 5E <1> pop esi

2687 00005405 5F <1> pop edi

2688 <1> ;

2689 00005406 53 <1> push ebx

2690 00005407 50 <1> push eax

2691 00005408 89EB <1> mov ebx, ebp

2692 <1> ; ebx = virtual address of the memory page

2693 0000540A E8AAFDFFFF <1> call add\_to\_swap\_queue

2694 0000540F 58 <1> pop eax

2695 00005410 5B <1> pop ebx

2696 <1> ;

2697 <1> ;or ax, PTE\_A\_USER+PTE\_A\_PRESENT

2698 00005411 0C07 <1> or al, PTE\_A\_USER+PTE\_A\_WRITE+PTE\_A\_PRESENT

2699 <1> cpt\_2:

2700 00005413 AB <1> stosd ; EDI points to child's PTE

2701 <1> ;

2702 00005414 81C500100000 <1> add ebp, 4096 ; 20/07/2015 (next page)

2703 <1> ;

2704 0000541A 39D7 <1> cmp edi, edx

2705 0000541C 72BC <1> jb short cpt\_0

2706 <1> cpt\_p\_err:

2707 0000541E 59 <1> pop ecx

2708 0000541F 5A <1> pop edx

2709 00005420 5F <1> pop edi

2710 00005421 5E <1> pop esi

2711 <1> ;pop ebx

2712 00005422 58 <1> pop eax ; \*

2713 <1> cpt\_err:

2714 00005423 C3 <1> retn

2715 <1>

2716 <1> allocate\_memory\_block:

2717 <1> ; 01/05/2017

2718 <1> ; 28/04/2017

2719 <1> ; 25/04/2017

2720 <1> ; 01/04/2016, 02/04/2016, 03/04/2016

2721 <1> ; 13/03/2016, 14/03/2016

2722 <1> ; 12/03/2016 (TRDOS 386 = TRDOS v2.0)

2723 <1> ; Allocating contiguous memory pages (in the kernel's memory space)

2724 <1> ;

2725 <1> ; INPUT ->

2726 <1> ; EAX = Beginning address (physical)

2727 <1> ; EAX = 0 -> Allocate memory block from the first proper aperture

2728 <1> ; ECX = Number of bytes to be allocated

2729 <1> ;

2730 <1> ; OUTPUT ->

2731 <1> ; 1) cf = 0 -> successful

2732 <1> ; EAX = Beginning (physical) address of the allocated memory block

2733 <1> ; ECX = Number of allocated bytes (rounded up to page borders)

2734 <1> ; 2) cf = 1 -> unsuccessful

2735 <1> ; 2.1) If EAX > 0 ->

2736 <1> ; (Number of requested pages is more than # of free pages

2737 <1> ; but contiguous free pages -the aperture- is not enough!)

2738 <1> ; EAX = Beginning address of available aperture

2739 <1> ; (one of all aperture with max. aperture size/length)

2740 <1> ; ECX = Size of available aperture (memory block) in bytes

2741 <1> ; 2.2) If EAX = 0 -> Out of memory error

2742 <1> ; (number of free pages is less than requested number)

2743 <1> ; ECX = Total number of free bytes (free pages \* 4096)

2744 <1> ; (It is not number of contiguous free bytes)

2745 <1> ;

2746 <1> ; (Modified Registers -> EAX, ECX)

2747 <1> ;

2748 <1> ; PURPOSE: Loading a file at memory for copying or running etc.

2749 <1> ; If this procedure returns with cf is set, ECX contains maximum

2750 <1> ; available space and EAX contains the beginning address of it.

2751 <1> ; If EAX has zero, ECX contains total number of free bytes.

2752 <1> ; If requested block has been successfully allocated (by rounding up to

2753 <1> ; the last page border), it must be deallocated later by using

2754 <1> ; 'deallocate\_memory\_block' procedure.

2755 <1>

2756 00005424 52 <1> push edx ; \*

2757 00005425 BAFF0F0000 <1> mov edx, PAGE\_SIZE - 1 ; 4095

2758 0000542A 01D0 <1> add eax, edx

2759 0000542C 01D1 <1> add ecx, edx

2760 0000542E C1E90C <1> shr ecx, PAGE\_SHIFT ; 12

2761 <1>

2762 <1> ; ECX = number of contiguous pages to be allocated

2763 00005431 8B15[40580100] <1> mov edx, [free\_pages]

2764 <1> ; 01/05/2017

2765 <1> ;or ecx, ecx

2766 <1> ;jz short amb3

2767 <1> ; If ECX=0, set cf to 1 and return with max. available mem block size

2768 <1>

2769 00005437 39D1 <1> cmp ecx, edx

2770 00005439 7760 <1> ja short amb\_3

2771 <1>

2772 0000543B C1E80C <1> shr eax, PAGE\_SHIFT ; 12

2773 <1>

2774 0000543E 89C2 <1> mov edx, eax ; page number

2775 00005440 C1EA03 <1> shr edx, 3 ; to get offset to M.A.T.

2776 <1> ; (1 allocation bit = 1 page)

2777 <1> ; (1 allocation bytes = 8 pages)

2778 00005443 80E2FC <1> and dl, 0FCh ; clear lower 2 bits

2779 <1> ; (to get 32 bit position)

2780 00005446 53 <1> push ebx ; \*\*

2781 <1> amb\_0:

2782 00005447 890D[F8640100] <1> mov [mem\_ipg\_count], ecx ; initial (reset) value of page count

2783 0000544D 890D[FC640100] <1> mov [mem\_pg\_count], ecx

2784 00005453 31C9 <1> xor ecx, ecx ; 0

2785 00005455 890D[00650100] <1> mov [mem\_aperture], ecx ; 0

2786 0000545B 890D[04650100] <1> mov [mem\_max\_aperture], ecx ; 0

2787 <1>

2788 00005461 BB00001000 <1> mov ebx, MEM\_ALLOC\_TBL ; Memory Allocation Table address.

2789 00005466 3B15[44580100] <1> cmp edx, [next\_page] ; Is the beginning page address lower

2790 <1> ; than the address in 'next\_page' ?

2791 <1> ; (the first/next free page of user space)

2792 0000546C 7208 <1> jb short amb\_1

2793 0000546E 3B15[48580100] <1> cmp edx, [last\_page] ; is the beginning page address higher

2794 <1> ; than the address in 'last\_page' ?

2795 <1> ; (end of the memory)

2796 00005474 7606 <1> jna short amb\_2 ; no

2797 <1> amb\_1:

2798 00005476 8B15[44580100] <1> mov edx, [next\_page] ; M.A.T. offset (1 M.A.T. byte = 8 pages)

2799 <1> amb\_2:

2800 0000547C 01D3 <1> add ebx, edx

2801 <1>

2802 <1> ; 28/04/2017

2803 <1> ;xor ecx, ecx

2804 0000547E 0FBC0B <1> bsf ecx, [ebx] ; 0 to 31

2805 00005481 89D0 <1> mov eax, edx

2806 00005483 C1E003 <1> shl eax, 3 ; \*8

2807 00005486 01C8 <1> add eax, ecx ; beginning page number

2808 <1>

2809 00005488 A3[08650100] <1> mov [mem\_pg\_pos], eax ; beginning page no (for curr. mem. aperture)

2810 0000548D A3[0C650100] <1> mov [mem\_max\_pg\_pos], eax ; beginning page no for max. mem. aperture

2811 <1>

2812 00005492 83E01F <1> and eax, 1Fh ; lower 5 bits only (0 to 31)

2813 <1> ; (allocation bit position)

2814 00005495 750E <1> jnz short amb\_4 ; 0

2815 00005497 B120 <1> mov cl, 32

2816 00005499 EB4B <1> jmp short amb\_10

2817 <1>

2818 <1> amb\_3: ; out\_of\_memory

2819 0000549B 31C0 <1> xor eax, eax ; 0

2820 0000549D 89D1 <1> mov ecx, edx ; free pages

2821 0000549F C1E10C <1> shl ecx, PAGE\_SHIFT

2822 000054A2 5A <1> pop edx ; \*

2823 000054A3 F9 <1> stc

2824 000054A4 C3 <1> retn

2825 <1> amb\_4:

2826 000054A5 8B13 <1> mov edx, [ebx]

2827 000054A7 88C1 <1> mov cl, al ; 1 to 31

2828 000054A9 D3EA <1> shr edx, cl

2829 000054AB 89D0 <1> mov eax, edx

2830 <1> amb\_5:

2831 000054AD D1E8 <1> shr eax, 1 ; (\*\*\*)

2832 000054AF 7317 <1> jnc short amb\_7

2833 000054B1 FF05[00650100] <1> inc dword [mem\_aperture]

2834 000054B7 FF0D[FC640100] <1> dec dword [mem\_pg\_count]

2835 000054BD 7470 <1> jz short amb\_15

2836 <1> amb\_6:

2837 <1> ; 28/04/2017

2838 000054BF FEC1 <1> inc cl

2839 000054C1 80F920 <1> cmp cl, 32

2840 000054C4 730D <1> jnb short amb\_9

2841 000054C6 EBE5 <1> jmp short amb\_5

2842 <1> amb\_7:

2843 000054C8 50 <1> push eax ; (\*\*\*) allocation bits (in shifted status)

2844 000054C9 E81B010000 <1> call amb\_26 ; set maximum memory aperture (free memory block size)

2845 000054CE 58 <1> pop eax ; (\*\*\*)

2846 000054CF EBEE <1> jmp short amb\_6

2847 <1> amb\_8:

2848 <1> ; 28/04/2017

2849 000054D1 B120 <1> mov cl, 32

2850 <1> amb\_9:

2851 000054D3 89DA <1> mov edx, ebx

2852 000054D5 81EA00001000 <1> sub edx, MEM\_ALLOC\_TBL

2853 000054DB 3B15[48580100] <1> cmp edx, [last\_page]

2854 000054E1 7336 <1> jnb short amb\_14 ; contiguous pages not enough

2855 000054E3 83C304 <1> add ebx, 4

2856 <1> amb\_10:

2857 000054E6 8B03 <1> mov eax, [ebx]

2858 000054E8 21C0 <1> and eax, eax

2859 000054EA 7408 <1> jz short amb\_11 ; there is not a free page bit in this alloc dword

2860 000054EC 40 <1> inc eax ; 0FFFFFFFFh -> 0

2861 000054ED 740C <1> jz short amb\_12 ; all of bits are set (32 free pages)

2862 000054EF 48 <1> dec eax

2863 000054F0 28C9 <1> sub cl, cl ; 0

2864 000054F2 EBB9 <1> jmp short amb\_5

2865 <1> amb\_11:

2866 000054F4 E8F0000000 <1> call amb\_26 ; set maximum memory aperture (free memory block size)

2867 000054F9 EBD8 <1> jmp short amb\_9

2868 <1> amb\_12:

2869 000054FB 390D[FC640100] <1> cmp [mem\_pg\_count], ecx ; 32

2870 00005501 7306 <1> jnb short amb\_13

2871 00005503 8B0D[FC640100] <1> mov ecx, [mem\_pg\_count]

2872 <1> amb\_13:

2873 00005509 010D[00650100] <1> add [mem\_aperture], ecx

2874 0000550F 290D[FC640100] <1> sub [mem\_pg\_count], ecx

2875 00005515 7618 <1> jna short amb\_15

2876 00005517 EBBA <1> jmp short amb\_9 ; 01/05/2017

2877 <1> amb\_14:

2878 00005519 E8CB000000 <1> call amb\_26 ; 28/04/2017

2879 0000551E A1[0C650100] <1> mov eax, [mem\_max\_pg\_pos] ; begin address of max. mem aperture

2880 00005523 8B0D[04650100] <1> mov ecx, [mem\_max\_aperture] ; max. (largest) memory aperture

2881 00005529 F9 <1> stc

2882 0000552A E9AF000000 <1> jmp amb\_25

2883 <1>

2884 <1> amb\_15: ; OK !

2885 0000552F A1[08650100] <1> mov eax, [mem\_pg\_pos] ; Beginning address as page number

2886 00005534 8B0D[00650100] <1> mov ecx, [mem\_aperture] ; Free contiguous page count (>=1)

2887 <1> amb\_16:

2888 <1> ; allocate contiguous memory pages (via memory allocation table bits)

2889 0000553A 89C2 <1> mov edx, eax

2890 <1> ; 25/04/2017

2891 0000553C C1EA03 <1> shr edx, 3 ; 8 pages in one allocation byte

2892 0000553F 80E2FC <1> and dl, 0FCh ; clear lower 2 bits

2893 <1> ; (for dword/32bit positioning)

2894 <1>

2895 00005542 BB00001000 <1> mov ebx, MEM\_ALLOC\_TBL

2896 00005547 01D3 <1> add ebx, edx

2897 00005549 83E01F <1> and eax, 1Fh ; 31

2898 <1> ; 03/04/2016

2899 0000554C BA20000000 <1> mov edx, 32

2900 00005551 28C2 <1> sub dl, al

2901 00005553 39CA <1> cmp edx, ecx ; ecx >= 1

2902 00005555 7602 <1> jna short amb\_17

2903 00005557 89CA <1> mov edx, ecx

2904 <1> amb\_17:

2905 00005559 29D1 <1> sub ecx, edx

2906 0000555B 51 <1> push ecx ; \*\*\*

2907 0000555C 89D1 <1> mov ecx, edx

2908 <1> amb\_18:

2909 0000555E 0FB303 <1> btr [ebx], eax ; The destination bit indexed by the source value

2910 <1> ; is copied into the Carry Flag and then cleared

2911 <1> ; in the destination.

2912 00005561 FF0D[40580100] <1> dec dword [free\_pages] ; 1 page has been allocated (X = X-1)

2913 00005567 49 <1> dec ecx

2914 00005568 7404 <1> jz short amb\_19

2915 0000556A FEC0 <1> inc al

2916 0000556C EBF0 <1> jmp short amb\_18

2917 <1> amb\_19:

2918 0000556E 59 <1> pop ecx ; \*\*\*

2919 0000556F 21C9 <1> and ecx, ecx ; 0 ?

2920 00005571 741E <1> jz short amb\_22

2921 <1> ; 01/04/2016

2922 00005573 B020 <1> mov al, 32

2923 <1> amb\_20:

2924 00005575 83C304 <1> add ebx, 4

2925 00005578 39C1 <1> cmp ecx, eax ; 32

2926 0000557A 7305 <1> jnb short amb\_21

2927 <1> ; ECX < 32

2928 0000557C 28C0 <1> sub al, al ; 0

2929 0000557E 50 <1> push eax ; 0 \*\*\*

2930 0000557F EBDD <1> jmp short amb\_18

2931 <1> amb\_21:

2932 00005581 2905[40580100] <1> sub [free\_pages], eax ; [free\_pages] = [free\_pages] - 32

2933 00005587 C70300000000 <1> mov dword [ebx], 0 ; reset 32 bits

2934 0000558D 29C1 <1> sub ecx, eax ; 32

2935 0000558F 75E4 <1> jnz short amb\_20

2936 <1> amb\_22:

2937 00005591 A1[08650100] <1> mov eax, [mem\_pg\_pos] ; Beginning address as page number

2938 00005596 8B0D[00650100] <1> mov ecx, [mem\_aperture] ; Free contiguous page count

2939 <1> ; [next\_page] update

2940 0000559C 89C2 <1> mov edx, eax

2941 <1> ; 03/04/2016

2942 0000559E C1EA03 <1> shr edx, 3 ; to get offset to M.A.T.

2943 <1> ; (1 allocation bit = 1 page)

2944 <1> ; (1 allocation bytes = 8 pages)

2945 000055A1 80E2FC <1> and dl, 0FCh ; clear lower 2 bits

2946 <1> ; (to get 32 bit position)

2947 000055A4 3B15[44580100] <1> cmp edx, [next\_page] ; first free page pointer offset

2948 000055AA 7732 <1> ja short amb\_25

2949 000055AC BB00001000 <1> mov ebx, MEM\_ALLOC\_TBL

2950 000055B1 833C1300 <1> cmp dword [ebx+edx], 0

2951 000055B5 7721 <1> ja short amb\_24

2952 000055B7 89C2 <1> mov edx, eax

2953 000055B9 01CA <1> add edx, ecx

2954 000055BB C1EA03 <1> shr edx, 3

2955 000055BE 80E2FC <1> and dl, 0FCh

2956 <1> amb\_23:

2957 000055C1 833C1300 <1> cmp dword [ebx+edx], 0

2958 000055C5 7711 <1> ja short amb\_24

2959 000055C7 83C204 <1> add edx, 4

2960 000055CA 3B15[48580100] <1> cmp edx, [last\_page] ; last page pointer offset

2961 000055D0 76EF <1> jna short amb\_23

2962 000055D2 8B15[4C580100] <1> mov edx, [first\_page] ; (for) beginning of user's space

2963 <1> amb\_24:

2964 000055D8 8915[44580100] <1> mov [next\_page], edx

2965 <1> amb\_25:

2966 000055DE 9C <1> pushf

2967 000055DF C1E00C <1> shl eax, PAGE\_SHIFT ; convert to phy. address in bytes

2968 000055E2 C1E10C <1> shl ecx, PAGE\_SHIFT ; convert to byte counts

2969 000055E5 9D <1> popf

2970 000055E6 5B <1> pop ebx ; \*\*

2971 000055E7 5A <1> pop edx ; \*

2972 000055E8 C3 <1> retn

2973 <1>

2974 <1> amb\_26: ; set maximum free memory aperture (free memory block size)

2975 000055E9 89DA <1> mov edx, ebx ; current address

2976 000055EB 81EA00001000 <1> sub edx, MEM\_ALLOC\_TBL ; MAT beginning address

2977 <1> ; 02/04/2016

2978 000055F1 C1E203 <1> shl edx, 3 ; MAT byte offset \* 8 = page number base

2979 000055F4 01CA <1> add edx, ecx ; current page number (ecx = 0 to 32)

2980 <1> ;

2981 000055F6 A1[00650100] <1> mov eax, [mem\_aperture]

2982 000055FB 21C0 <1> and eax, eax

2983 000055FD 7421 <1> jz short amb\_27

2984 000055FF C705[00650100]0000- <1> mov dword [mem\_aperture], 0

2984 00005607 0000 <1>

2985 00005609 3B05[04650100] <1> cmp eax, [mem\_max\_aperture]

2986 0000560F 760F <1> jna short amb\_27

2987 00005611 A3[04650100] <1> mov [mem\_max\_aperture], eax

2988 <1> ; 25/04/2017

2989 00005616 A1[08650100] <1> mov eax, [mem\_pg\_pos]

2990 <1> ; EAX = Beginning page number of the max. aperture

2991 0000561B A3[0C650100] <1> mov [mem\_max\_pg\_pos], eax

2992 <1> amb\_27:

2993 00005620 8915[08650100] <1> mov [mem\_pg\_pos], edx ; current page

2994 <1>

2995 00005626 A1[F8640100] <1> mov eax, [mem\_ipg\_count] ; initial (reset) value of page count

2996 0000562B A3[FC640100] <1> mov [mem\_pg\_count], eax

2997 <1>

2998 00005630 C3 <1> retn

2999 <1>

3000 <1> deallocate\_memory\_block:

3001 <1> ; 03/04/2016

3002 <1> ; 14/03/2016 (TRDOS 386 = TRDOS v2.0)

3003 <1> ; Deallocating contiguous memory pages (in the kernel's memory space)

3004 <1> ;

3005 <1> ; INPUT ->

3006 <1> ; EAX = Beginning address (physical)

3007 <1> ; ECX = Number of bytes to be deallocated

3008 <1> ;

3009 <1> ; OUTPUT ->

3010 <1> ; Memory Allocation Table bits will be updated

3011 <1> ; [free\_pages] will be changed (increased)

3012 <1> ;

3013 <1> ; (Modified Registers -> EAX, ECX)

3014 <1> ;

3015 <1> ; PURPOSE: Unloading/Freeing a file -or an allocated memory block-

3016 <1> ; at memory after copying, running, saving, reading, writing etc.

3017 <1> ;

3018 <1>

3019 00005631 52 <1> push edx ; \*

3020 00005632 53 <1> push ebx ; \*\*

3021 <1>

3022 00005633 C1E80C <1> shr eax, PAGE\_SHIFT ; 12

3023 00005636 C1E90C <1> shr ecx, PAGE\_SHIFT ; 12

3024 <1>

3025 <1> ; EAX = Beginning page number

3026 <1> ; ECX = Number of contiguous pages to be deallocated

3027 <1> damb\_0:

3028 <1> ; deallocate contiguous memory pages (via memory allocation table bits)

3029 00005639 89C2 <1> mov edx, eax

3030 0000563B C1EA03 <1> shr edx, 3 ; to get offset to M.A.T.

3031 <1> ; (1 allocation bit = 1 page)

3032 <1> ; (1 allocation bytes = 8 pages)

3033 0000563E 80E2FC <1> and dl, 0FCh ; clear lower 2 bits

3034 <1> ; (to get 32 bit position)

3035 00005641 3B15[44580100] <1> cmp edx, [next\_page] ; next free page

3036 00005647 7306 <1> jnb short damb\_1

3037 00005649 8915[44580100] <1> mov [next\_page], edx

3038 <1> damb\_1:

3039 0000564F BB00001000 <1> mov ebx, MEM\_ALLOC\_TBL

3040 00005654 01D3 <1> add ebx, edx

3041 00005656 83E01F <1> and eax, 1Fh ; 31

3042 <1>

3043 <1> ; 03/04/2016

3044 00005659 BA20000000 <1> mov edx, 32

3045 0000565E 28C2 <1> sub dl, al

3046 00005660 39CA <1> cmp edx, ecx

3047 00005662 7602 <1> jna short damb\_2

3048 00005664 89CA <1> mov edx, ecx

3049 <1> damb\_2:

3050 00005666 29D1 <1> sub ecx, edx

3051 00005668 51 <1> push ecx ; \*\*\*

3052 00005669 89D1 <1> mov ecx, edx

3053 <1> damb\_3:

3054 0000566B 0FAB03 <1> bts [ebx], eax ; unlink/release/deallocate page

3055 <1> ; set relevant bit to 1.

3056 <1> ; set CF to the previous bit value

3057 0000566E FF05[40580100] <1> inc dword [free\_pages] ; 1 page has been deallocated (X = X+1)

3058 00005674 49 <1> dec ecx

3059 00005675 7404 <1> jz short damb\_4

3060 00005677 FEC0 <1> inc al

3061 00005679 EBF0 <1> jmp short damb\_3

3062 <1> damb\_4:

3063 0000567B 59 <1> pop ecx ; \*\*\*

3064 0000567C 21C9 <1> and ecx, ecx ; 0 ?

3065 0000567E 741E <1> jz short damb\_7

3066 <1> ; 03/04/2016

3067 00005680 B020 <1> mov al, 32

3068 <1> damb\_5:

3069 00005682 83C304 <1> add ebx, 4

3070 00005685 39C1 <1> cmp ecx, eax ; 32

3071 00005687 7305 <1> jnb short damb\_6

3072 <1> ; ECX < 32

3073 00005689 28C0 <1> sub al, al ; 0

3074 0000568B 50 <1> push eax ; 0 \*\*\*

3075 0000568C EBDD <1> jmp short damb\_3

3076 <1> damb\_6:

3077 0000568E 0105[40580100] <1> add [free\_pages], eax ; [free\_pages] = [free\_pages] + 32

3078 00005694 C703FFFFFFFF <1> mov dword [ebx], 0FFFFFFFFh ; set 32 bits

3079 0000569A 29C1 <1> sub ecx, eax ; 32

3080 0000569C 75E4 <1> jnz short damb\_5

3081 <1> damb\_7:

3082 0000569E 5B <1> pop ebx ; \*\*

3083 0000569F 5A <1> pop edx ; \*

3084 000056A0 C3 <1> retn

3085 <1>

3086 <1> direct\_memory\_access:

3087 <1> ; 22/07/2017

3088 <1> ; 12/05/2017

3089 <1> ; 16/07/2016

3090 <1> ; 12/07/2016 (TRDOS 386 = TRDOS v2.0)

3091 <1> ; This processure will be called to map

3092 <1> ; user's (ring 3) page tables to access phsical

3093 <1> ; (flat/linear) memory addresses, directly (without

3094 <1> ; kernel's data transfer functions).

3095 <1> ;

3096 <1> ; Purpose: Video memory access and shared memory access.

3097 <1> ;

3098 <1> ; INPUT ->

3099 <1> ; EAX = Beginning address (physical).

3100 <1> ; EBX = User's buffer address ; 12/05/2017

3101 <1> ; ECX = Number of contiguous pages to be mapped.

3102 <1> ; OUTPUT ->

3103 <1> ; User's page directory and pages tables

3104 <1> ; will be updated.

3105 <1> ;

3106 <1> ; If an old page table entry has valid page address,

3107 <1> ; that page will be deallocated just before PTE will

3108 <1> ; be changed for direct (1 to 1) memory page access.

3109 <1> ;

3110 <1> ; If old PTE value points to a swapped page,

3111 <1> ; that page (block) will be unlinked on swap disk.

3112 <1> ;

3113 <1> ; Newly allocated pages (except page tables) will not

3114 <1> ; be applied to Memory Allocation Table.

3115 <1> ; AVL bit 1 (PTE bit 10) of page table entry will be

3116 <1> ; used to indicate shared (direct) memory page; then,

3117 <1> ; this page will not be deallocated later during

3118 <1> ; process termination. (Memory Allocation Table and

3119 <1> ; free memory count will not be affected.

3120 <1> ; (Except deallocating page table's itself.)

3121 <1> ;

3122 <1> ; CF = 1 -> error (EAX = error code)

3123 <1> ; CF = 0 -> success (EAX = beginning address)

3124 <1> ;

3125 <1> ;; (Modified Registers -> none)

3126 <1> ; Modified registers: ebp, edx, ecx, ebx, esi, edi

3127 <1> ;

3128 <1>

3129 <1> ;push ebp

3130 <1> ;push ebx

3131 <1> ;push ecx

3132 <1> ;push edx

3133 000056A1 662500F0 <1> and ax, PTE\_A\_CLEAR ; clear page offset

3134 000056A5 50 <1> push eax

3135 <1> ;and ecx, ecx ; page count

3136 <1> ;jz dmem\_acc\_7 ; 'insufficient memory' error

3137 000056A6 89C5 <1> mov ebp, eax

3138 000056A8 81C300004000 <1> add ebx, CORE ; 12/05/2017

3139 <1> dmem\_acc\_0:

3140 000056AE 891D[F86F0100] <1> mov [base\_addr], ebx ; 12/05/2017

3141 000056B4 A1[B8030300] <1> mov eax, [u.pgdir] ; page dir address (physical)

3142 000056B9 E8D7F5FFFF <1> call get\_pte

3143 <1> ; EDX = Page table entry address (if CF=0)

3144 <1> ; Page directory entry address (if CF=1)

3145 <1> ; (Bit 0 value is 0 if PT is not present)

3146 <1> ; EAX = Page table entry value (page address)

3147 <1> ; CF = 1 -> PDE not present or invalid ?

3148 000056BE 7324 <1> jnc short dmem\_acc\_1

3149 <1> ;

3150 000056C0 E8B5F4FFFF <1> call allocate\_page

3151 000056C5 0F82AB000000 <1> jc dmem\_acc\_7 ; 'insufficient memory' error

3152 <1> ;

3153 000056CB E824F5FFFF <1> call clear\_page

3154 <1> ; EAX = Physical (base) address of the allocated (new) page

3155 000056D0 0C07 <1> or al, PDE\_A\_PRESENT + PDE\_A\_WRITE + PDE\_A\_USER ; 4+2+1 = 7

3156 <1> ; lower 3 bits are used as U/S, R/W, P flags

3157 <1> ; (user, writable, present page)

3158 000056D2 8902 <1> mov [edx], eax ; Let's put the new page directory entry here !

3159 000056D4 A1[B8030300] <1> mov eax, [u.pgdir]

3160 000056D9 E8B7F5FFFF <1> call get\_pte

3161 000056DE 0F8292000000 <1> jc dmem\_acc\_7 ; 'insufficient memory' error

3162 <1> dmem\_acc\_1:

3163 <1> ; EAX = PTE value, EDX = PTE address

3164 000056E4 A801 <1> test al, PTE\_A\_PRESENT

3165 000056E6 750D <1> jnz short dmem\_acc\_2

3166 000056E8 09C0 <1> or eax, eax

3167 000056EA 7468 <1> jz short dmem\_acc\_6 ; Change PTE

3168 000056EC D1E8 <1> shr eax, 1 ; swap disk block (8 sectors) address

3169 <1> ; unlink swap disk block

3170 000056EE E80CFBFFFF <1> call unlink\_swap\_block

3171 000056F3 EB5F <1> jmp short dmem\_acc\_6

3172 <1>

3173 <1> dmem\_acc\_2:

3174 000056F5 A802 <1> test al, PTE\_A\_WRITE ; bit 1, writable (r/w) flag

3175 <1> ; (must be 1)

3176 000056F7 7550 <1> jnz short dmem\_acc\_4

3177 <1> ; Read only -duplicated- page (belongs to a parent or a child)

3178 000056F9 66A90002 <1> test ax, PTE\_DUPLICATED ; Was this page duplicated

3179 <1> ; as child's page ?

3180 000056FD 7455 <1> jz short dmem\_acc\_5 ; Change PTE but don't deallocate the page!

3181 <1>

3182 <1> ;push edi

3183 <1> ;push esi

3184 <1>

3185 000056FF 51 <1> push ecx

3186 <1> ;push ebx

3187 00005700 8B1D[BC030300] <1> mov ebx, [u.ppgdir] ; parent's page dir address (physical)

3188 <1>

3189 <1> ; check the parent's PTE value is read only & same page or not..

3190 00005706 89EF <1> mov edi, ebp

3191 00005708 C1EF16 <1> shr edi, PAGE\_D\_SHIFT ; 22

3192 <1> ; EDI = page directory entry index (0-1023)

3193 0000570B 89EE <1> mov esi, ebp

3194 0000570D C1EE0C <1> shr esi, PAGE\_SHIFT ; 12

3195 00005710 81E6FF030000 <1> and esi, PTE\_MASK

3196 <1> ; ESI = page table entry index (0-1023)

3197 <1>

3198 00005716 66C1E702 <1> shl di, 2 ; \* 4

3199 0000571A 01FB <1> add ebx, edi ; PDE offset (for the parent)

3200 0000571C 8B0F <1> mov ecx, [edi]

3201 0000571E F6C101 <1> test cl, PDE\_A\_PRESENT ; present (valid) or not ?

3202 00005721 7425 <1> jz short dmem\_acc\_3 ; parent process does not use this page

3203 00005723 6681E100F0 <1> and cx, PDE\_A\_CLEAR ; 0F000h ; Clear attribute bits

3204 00005728 66C1E602 <1> shl si, 2 ; \*4

3205 0000572C 01CE <1> add esi, ecx ; PTE offset (for the parent)

3206 0000572E 8B1E <1> mov ebx, [esi]

3207 00005730 F6C301 <1> test bl, PTE\_A\_PRESENT ; present or not ?

3208 00005733 7413 <1> jz short dmem\_acc\_3 ; parent process does not use this page

3209 00005735 662500F0 <1> and ax, PTE\_A\_CLEAR ; 0F000h ; Clear attribute bits

3210 00005739 6681E300F0 <1> and bx, PTE\_A\_CLEAR ; 0F000h ; Clear attribute bits

3211 0000573E 39D8 <1> cmp eax, ebx ; parent's and child's pages are same ?

3212 00005740 7506 <1> jne short dmem\_acc\_3 ; not same page

3213 <1> ; deallocate the child's page

3214 00005742 800E02 <1> or byte [esi], PTE\_A\_WRITE ; convert to writable page (parent)

3215 <1> ;pop ebx

3216 00005745 59 <1> pop ecx

3217 00005746 EB0C <1> jmp short dmem\_acc\_5

3218 <1> dmem\_acc\_3:

3219 <1> ;pop ebx

3220 00005748 59 <1> pop ecx

3221 <1> dmem\_acc\_4:

3222 00005749 66A90004 <1> test ax, PTE\_SHARED ; shared or direct memory access indicator

3223 0000574D 7505 <1> jnz short dmem\_acc\_5 ; AVL bit 1 = 1, do not deallocate this page!

3224 <1> ;

3225 <1> ;and ax, PTE\_A\_CLEAR ; 0F000h ; clear lower 12 (attribute) bits

3226 0000574F E804F6FFFF <1> call deallocate\_page

3227 <1> dmem\_acc\_5:

3228 <1> ;pop esi

3229 <1> ;pop edi

3230 <1> dmem\_acc\_6:

3231 00005754 89E8 <1> mov eax, ebp ; physical page (offset=0) address

3232 <1> ; EAX = memory page address

3233 <1> ; EDX = PTE entry address (physical)

3234 00005756 660D0704 <1> or ax, PTE\_A\_PRESENT+PTE\_A\_USER+PTE\_A\_WRITE+PTE\_SHARED

3235 <1> ; present flag, bit 0 = 1

3236 <1> ; user flag, bit 2 = 1

3237 <1> ; writable flag, bit 1 = 1

3238 <1> ; direct memory access flag, bit 10 = 1

3239 <1> ; (This page must not be deallocated!)

3240 0000575A 8902 <1> mov [edx], eax ; Update PTE value

3241 0000575C 49 <1> dec ecx ; remain count of contiguous pages

3242 0000575D 741E <1> jz short dmem\_acc\_8

3243 0000575F 81C500100000 <1> add ebp, PAGE\_SIZE ; next physical page address

3244 <1> ; 22/07/2017

3245 <1> ;mov eax, ebp

3246 <1> ; 12/05/2017

3247 00005765 8B1D[F86F0100] <1> mov ebx, [base\_addr] ; linear address (virtual+CORE)

3248 0000576B 81C300100000 <1> add ebx, PAGE\_SIZE ; next linear address

3249 00005771 E938FFFFFF <1> jmp dmem\_acc\_0

3250 <1> dmem\_acc\_7: ; ERROR !

3251 00005776 C7042404000000 <1> mov dword [esp], ERR\_MINOR\_IM

3252 <1> ; Insufficient memory (minor) error!

3253 <1> ; Major error = 0 (No protection fault)

3254 <1> ; cf = 1

3255 <1> dmem\_acc\_8:

3256 0000577D 58 <1> pop eax

3257 <1> ;pop edx

3258 <1> ;pop ecx

3259 <1> ;pop ebx

3260 <1> ;pop ebp

3261 0000577E C3 <1> retn

3262 <1>

3263 <1> deallocate\_user\_pages:

3264 <1> ; 20/05/2017

3265 <1> ; 15/05/2017

3266 <1> ; 20/02/2017

3267 <1> ; 19/02/2017 (TRDOS 386 = TRDOS v2.0)

3268 <1> ;

3269 <1> ; Deallocate virtually contiguous user pages (memory block)

3270 <1> ; (caller: 'sysdalloc' system call)

3271 <1> ;

3272 <1> ; INPUT ->

3273 <1> ; EBX = VIRTUAL ADDRESS (beginning address)

3274 <1> ; ECX = byte count

3275 <1> ; [u.pgdir] = user's page directory

3276 <1> ; [u.ppdir] = parent's page directory

3277 <1> ;

3278 <1> ; OUTPUT ->

3279 <1> ; If CF = 0

3280 <1> ; EAX = Deallocated memory bytes

3281 <1> ; (Even if shared or read only pages will not be

3282 <1> ; deallocated on M.A.T., this byte count will be

3283 <1> ; returned as virtually deallocated bytes; in fact

3284 <1> ; virtually deallocated user pages \* 4096.)

3285 <1> ; EBX = Virtual address (as rounded up)

3286 <1> ; If CF = 1

3287 <1> ; EAX = 0 (there is not any deallocated pages)

3288 <1> ;

3289 <1> ; Note: Empty page tables will not be deallocated!!!

3290 <1> ; (they will be deallocated at process termination stage)

3291 <1> ;

3292 <1> ; Modified Registers -> EAX, EDX, ESI, EDI, EBX, ECX, EBP

3293 <1> ;

3294 0000577F 89DE <1> mov esi, ebx

3295 00005781 89F7 <1> mov edi, esi

3296 00005783 01CF <1> add edi, ecx

3297 00005785 81C6FF0F0000 <1> add esi, PAGE\_SIZE - 1 ; 4095 (round up)

3298 0000578B C1EE0C <1> shr esi, PAGE\_SHIFT

3299 0000578E C1EF0C <1> shr edi, PAGE\_SHIFT

3300 00005791 89F8 <1> mov eax, edi ; end page

3301 00005793 29F0 <1> sub eax, esi ; end page - start page

3302 00005795 0F86D5000000 <1> jna da\_u\_pd\_err ; < 1

3303 0000579B 89F3 <1> mov ebx, esi

3304 0000579D C1E30C <1> shl ebx, PAGE\_SHIFT ; virtual address (as rounded up)

3305 000057A0 53 <1> push ebx ; \*

3306 000057A1 89C1 <1> mov ecx, eax ; page count

3307 000057A3 C1E00C <1> shl eax, PAGE\_SHIFT ; byte count as adjusted

3308 000057A6 50 <1> push eax ; \*\*

3309 000057A7 8B1D[B8030300] <1> mov ebx, [u.pgdir] ; physical addr of user's page dir

3310 000057AD 81C600040000 <1> add esi, CORE/PAGE\_SIZE

3311 000057B3 89F7 <1> mov edi, esi

3312 000057B5 81E7FF030000 <1> and edi, PTE\_MASK ; PTE entry in the page table

3313 000057BB 57 <1> push edi ; \*\*\* ; PTE index (of page directory)

3314 000057BC C1EE0A <1> shr esi, PAGE\_D\_SHIFT - PAGE\_SHIFT ; 22-12=10

3315 000057BF 89F2 <1> mov edx, esi

3316 <1> ; EDX = PDE index

3317 000057C1 C1E602 <1> shl esi, 2 ; convert PDE index to dword offset

3318 000057C4 01DE <1> add esi, ebx ; add page directory address

3319 <1> da\_u\_pd\_1:

3320 000057C6 AD <1> lodsd

3321 <1> ;

3322 000057C7 89F5 <1> mov ebp, esi ; 20/02/2017

3323 <1> ; EBP = next PDE address

3324 <1> ;

3325 000057C9 A801 <1> test al, PDE\_A\_PRESENT ; bit 0, present flag (must be 1)

3326 000057CB 0F8494000000 <1> jz da\_u\_pd\_3 ; 20/05/2017

3327 000057D1 662500F0 <1> and ax, PDE\_A\_CLEAR ; 0F000h ; clear lower 12 (attribute) bits

3328 <1> ; EAX = PHYSICAL (flat) ADDRESS OF THE PAGE TABLE

3329 000057D5 8B3C24 <1> mov edi, [esp] ; \*\*\*

3330 <1> ; EDI = PTE index (of complete page directory)

3331 <1> ;and edi, PTE\_MASK ; PTE entry in the page table

3332 000057D8 C1E702 <1> shl edi, 2 ; convert PTE index to dword offset

3333 000057DB 89FE <1> mov esi, edi ; PTE offset in page table (0-4092)

3334 000057DD 01C6 <1> add esi, eax ; now, esi points to requested PTE

3335 <1> da\_u\_pt\_0:

3336 000057DF AD <1> lodsd

3337 000057E0 A801 <1> test al, PTE\_A\_PRESENT ; bit 0, present flag (must be 1)

3338 000057E2 743F <1> jz short da\_u\_pt\_1

3339 <1> ;

3340 000057E4 A802 <1> test al, PTE\_A\_WRITE ; bit 1, writable (r/w) flag

3341 <1> ; (must be 1)

3342 000057E6 7549 <1> jnz short da\_u\_pt\_3

3343 <1> ; Read only -duplicated- page (belongs to a parent or a child)

3344 000057E8 66A90002 <1> test ax, PTE\_DUPLICATED ; Was this page duplicated

3345 <1> ; as child's page ?

3346 000057EC 744E <1> jz short da\_u\_pt\_4 ; Clear PTE but don't deallocate the page!

3347 <1> ;

3348 <1> ; check the parent's PTE value is read only & same page or not..

3349 <1> ; EDX = page directory entry index (0-1023)

3350 000057EE 52 <1> push edx ; \*\*\*\*

3351 <1> ; EDI = page table entry offset (0-4092)

3352 000057EF 8B1D[BC030300] <1> mov ebx, [u.ppgdir] ; page directory of the parent process

3353 000057F5 66C1E202 <1> shl dx, 2 ; \*4

3354 000057F9 01D3 <1> add ebx, edx ; PDE address (for the parent)

3355 000057FB 8B13 <1> mov edx, [ebx] ; page table address

3356 000057FD F6C201 <1> test dl, PDE\_A\_PRESENT ; present (valid) or not ?

3357 00005800 742E <1> jz short da\_u\_pt\_2 ; parent process does not use this page

3358 00005802 6681E200F0 <1> and dx, PDE\_A\_CLEAR ; 0F000h ; Clear attribute bits

3359 <1> ; EDI = page table entry offset (0-4092)

3360 00005807 01D7 <1> add edi, edx ; PTE address (for the parent)

3361 00005809 8B1F <1> mov ebx, [edi]

3362 0000580B F6C301 <1> test bl, PTE\_A\_PRESENT ; present or not ?

3363 0000580E 7420 <1> jz short da\_u\_pt\_2 ; parent process does not use this page

3364 00005810 662500F0 <1> and ax, PTE\_A\_CLEAR ; 0F000h ; Clear attribute bits

3365 00005814 6681E300F0 <1> and bx, PTE\_A\_CLEAR ; 0F000h ; Clear attribute bits

3366 00005819 39D8 <1> cmp eax, ebx ; parent's and child's pages are same ?

3367 0000581B 7513 <1> jne short da\_u\_pt\_2 ; not same page

3368 <1> ; deallocate the child's page

3369 0000581D 800F02 <1> or byte [edi], PTE\_A\_WRITE ; convert to writable page (parent)

3370 00005820 5A <1> pop edx ; \*\*\*\*

3371 00005821 EB19 <1> jmp short da\_u\_pt\_4

3372 <1> da\_u\_pt\_1:

3373 00005823 09C0 <1> or eax, eax ; swapped page ?

3374 00005825 741C <1> jz short da\_u\_pt\_5 ; no

3375 <1> ; yes

3376 00005827 D1E8 <1> shr eax, 1

3377 00005829 E8D1F9FFFF <1> call unlink\_swap\_block ; Deallocate swapped page block

3378 <1> ; on the swap disk (or in file)

3379 0000582E EB13 <1> jmp short da\_u\_pt\_5

3380 <1> da\_u\_pt\_2:

3381 00005830 5A <1> pop edx ; \*\*\*\*

3382 <1> da\_u\_pt\_3:

3383 00005831 66A90004 <1> test ax, PTE\_SHARED ; shared or direct memory access indicator

3384 00005835 7505 <1> jnz short da\_u\_pt\_4 ; AVL bit 1 = 1, do not deallocate this page!

3385 <1> ;

3386 <1> ;and ax, PTE\_A\_CLEAR ; 0F000h ; clear lower 12 (attribute) bits

3387 00005837 E81CF5FFFF <1> call deallocate\_page ; set the mem allocation bit of this page

3388 <1> da\_u\_pt\_4:

3389 0000583C C746FC00000000 <1> mov dword [esi-4], 0 ; clear/reset PTE (child, dupl. as parent)

3390 <1> da\_u\_pt\_5:

3391 <1> ; 20/05/2017

3392 00005843 58 <1> pop eax ; \*\*\* PTE index (of page directory)

3393 00005844 49 <1> dec ecx ; remain page count

3394 00005845 7426 <1> jz short da\_u\_pd\_4

3395 00005847 40 <1> inc eax ; next PTE

3396 00005848 6625FF03 <1> and ax, PTE\_MASK ; PTE entry index in the page table

3397 0000584C 50 <1> push eax ; \*\*\* (save again)

3398 <1> ;mov edi, eax

3399 <1> ;and di, PTE\_MASK

3400 <1> ;cmp edi, PAGE\_SIZE / 4 ; 1024

3401 <1> ;jnb short da\_u\_pd\_2

3402 0000584D 89C7 <1> mov edi, eax

3403 0000584F C1E702 <1> shl edi, 2 ; convert index to dword offset

3404 <1> ;test ax, PTE\_MASK ; 3FFh

3405 00005852 09C0 <1> or eax, eax

3406 00005854 7589 <1> jnz short da\_u\_pt\_0 ; 1-1023

3407 <1> da\_u\_pd\_2:

3408 00005856 42 <1> inc edx

3409 <1> ; 20/05/2017

3410 00005857 6681E2FF03 <1> and dx, PTE\_MASK ; 3FFh

3411 0000585C 740F <1> jz short da\_u\_pd\_4 ; 0 (1024)

3412 <1> ;cmp edx, 1024

3413 <1> ;jnb short da\_u\_pd\_4

3414 0000585E 89EE <1> mov esi, ebp ; 20/02/2017

3415 00005860 E961FFFFFF <1> jmp da\_u\_pd\_1

3416 <1> da\_u\_pd\_3:

3417 <1> ; 15/05/2017 (empty page directory entry)

3418 00005865 81E900040000 <1> sub ecx, 1024

3419 0000586B 77E9 <1> ja short da\_u\_pd\_2 ; 20/05/2017

3420 <1> da\_u\_pd\_4:

3421 0000586D 58 <1> pop eax ; \*\*

3422 0000586E 5B <1> pop ebx ; \*

3423 0000586F C3 <1> retn

3424 <1>

3425 <1> da\_u\_pd\_err:

3426 00005870 31C0 <1> xor eax, eax

3427 00005872 F9 <1> stc

3428 00005873 C3 <1> retn

3429 <1>

3430 <1> allocate\_user\_pages:

3431 <1> ; 20/05/2017

3432 <1> ; 01/05/2017, 02/05/2017, 15/05/2017

3433 <1> ; 04/03/2017

3434 <1> ; 20/02/2017 (TRDOS 386 = TRDOS v2.0)

3435 <1> ;

3436 <1> ; Allocate physically contiguous user pages (memory block)

3437 <1> ; (caller: 'sysalloc' system call)

3438 <1> ;

3439 <1> ; Note: This procedure does not alloc a page's itself

3440 <1> ; (page bit) on Memory Allocation Table.

3441 <1> ; (allocate\_memory\_block is needed before this proc)

3442 <1> ;

3443 <1> ; INPUT ->

3444 <1> ; EAX = PHYSICAL ADDRESS (beginning address)

3445 <1> ; EBX = VIRTUAL ADDRESS (beginning address)

3446 <1> ; ECX = byte count (>=4096)

3447 <1> ; [u.pgdir] = user's page directory

3448 <1> ;

3449 <1> ; Note: All addresses are (must be) already adjusted

3450 <1> ; to page borders, otherwise, lower 12bits of addresses

3451 <1> ; and byte count would be truncated.

3452 <1> ;

3453 <1> ; OUTPUT ->

3454 <1> ; none

3455 <1> ;

3456 <1> ; CF = 1 -> insufficient memory error

3457 <1> ;

3458 <1> ; Note: All pages will be allocated in physical page order

3459 <1> ; from the beginning page address.

3460 <1> ; \* A new page table will be added to the page dir

3461 <1> ; when the requested PDE is invalid.

3462 <1> ; \* Those pages will not be added to swap queue

3463 <1> ; because main purpose of this allocation is to

3464 <1> ; set a direct memory access (DMA controller) buffer.

3465 <1> ; (Swapping out a page in a DMA buffer would be wrong!)

3466 <1> ; \* Previous content of page tables (PTEs) would be

3467 <1> ; (should be) deallocated before entering this

3468 <1> ; procedure. So, new page table entries (PTEs)

3469 <1> ; directly will be written without checking

3470 <1> ; their previous content.

3471 <1> ; \* Only solution to increase free memory by removing

3472 <1> ; that non-swappable memory block is to terminate

3473 <1> ; the process or to wait until the process will

3474 <1> ; deallocate that memory block as itself. ('sysdalloc')

3475 <1> ; (No problem, if the process does not grab all of

3476 <1> ; -very big amount of- free memory by using

3477 <1> ; 'sysalloc' system call!?)

3478 <1> ; (Even if the process has grabbed all of free memory,

3479 <1> ; no problem if the process is not running in

3480 <1> ; multitasking mode. No problem in multitasking

3481 <1> ; mode if there is not another process which is running

3482 <1> ; or waiting or sleeping for an event as it's pages

3483 <1> ; are swapped-out. But a new process can not start to

3484 <1> ; run if all of free memory has beeen allocated

3485 <1> ; by running processes. Deallocation -'sysdalloc'-

3486 <1> ; or terminate a running process is needed

3487 <1> ; in order to run a new process.)

3488 <1> ;

3489 <1> ; Modified Registers -> EAX, EDX, ESI, EDI, EBX, ECX, EBP

3490 <1> ;

3491 <1>

3492 <1> ; 01/05/2017

3493 00005874 662500F0 <1> and ax, ~PAGE\_OFF

3494 00005878 6681E300F0 <1> and bx, ~PAGE\_OFF

3495 <1> ; 02/05/2017

3496 0000587D BD00F0FFFF <1> mov ebp, 0FFFFF000h ; 4 Giga Bytes - 4096 Bytes (for Stack)

3497 00005882 C1E90C <1> shr ecx, PAGE\_SHIFT ; page count

3498 00005885 83F901 <1> cmp ecx, 1

3499 00005888 7251 <1> jb short a\_u\_im\_retn

3500 0000588A 89C2 <1> mov edx, eax

3501 0000588C 01CA <1> add edx, ecx

3502 0000588E 724B <1> jc short a\_u\_im\_retn

3503 00005890 39D5 <1> cmp ebp, edx

3504 00005892 7247 <1> jb short a\_u\_im\_retn

3505 00005894 89DA <1> mov edx, ebx

3506 00005896 81C200004000 <1> add edx, CORE

3507 0000589C 723D <1> jc short a\_u\_im\_retn

3508 0000589E 01CA <1> add edx, ecx

3509 000058A0 7239 <1> jc short a\_u\_im\_retn

3510 000058A2 39D5 <1> cmp ebp, edx

3511 000058A4 7235 <1> jb short a\_u\_im\_retn

3512 <1> ;

3513 000058A6 89C5 <1> mov ebp, eax ; physical address

3514 000058A8 89DE <1> mov esi, ebx

3515 000058AA 81C600004000 <1> add esi, CORE ; start of user's memory (4M)

3516 000058B0 C1EE0C <1> shr esi, PAGE\_SHIFT ; higher 20 bits of the linear address

3517 <1> ;shr ecx, PAGE\_SHIFT ; page count

3518 000058B3 8B1D[B8030300] <1> mov ebx, [u.pgdir] ; physical addr of user's page dir

3519 000058B9 89F7 <1> mov edi, esi

3520 000058BB 81E7FF030000 <1> and edi, PTE\_MASK ; PTE entry index in the page table

3521 000058C1 57 <1> push edi ; \* ; PTE index (in page directory)

3522 000058C2 C1EE0A <1> shr esi, PAGE\_D\_SHIFT - PAGE\_SHIFT ; 22-12=10

3523 000058C5 89F2 <1> mov edx, esi

3524 <1> ; EDX = PDE index

3525 000058C7 C1E602 <1> shl esi, 2 ; convert PDE index to dword offset

3526 000058CA 01DE <1> add esi, ebx ; add page directory address

3527 <1> a\_u\_pd\_0:

3528 000058CC AD <1> lodsd

3529 <1> ;

3530 000058CD 89F3 <1> mov ebx, esi ; next PDE address

3531 <1> ;

3532 000058CF A801 <1> test al, PDE\_A\_PRESENT ; bit 0, present flag (must be 1)

3533 000058D1 7513 <1> jnz short a\_u\_pd\_2

3534 <1> ;

3535 <1> ; empty PDE (it does not point to valid page table address)

3536 000058D3 E8A2F2FFFF <1> call allocate\_page ; (allocate a new page table)

3537 000058D8 7302 <1> jnc short a\_u\_pd\_1 ; OK... now, we have a new page table.

3538 <1> ; cf = 1

3539 <1> ; There is not a free memory page to allocate a new page table !!!

3540 000058DA 5E <1> pop esi ; \*

3541 <1> a\_u\_im\_retn:

3542 000058DB C3 <1> retn ; return to 'sysalloc' with 'insufficient memory' error

3543 <1> ;

3544 <1> a\_u\_pd\_1: ; clear the new page table content

3545 <1> ; EAX = Physical (base) address of the new page table

3546 000058DC E813F3FFFF <1> call clear\_page ; Clear page content

3547 <1> ;

3548 000058E1 0C07 <1> or al, PDE\_A\_PRESENT + PDE\_A\_WRITE + PDE\_A\_USER

3549 <1> ; set bit 0, bit 1 and bit 2 to 1

3550 <1> ; (present, writable, user)

3551 000058E3 8946FC <1> mov [esi-4], eax

3552 <1> a\_u\_pd\_2:

3553 000058E6 662500F0 <1> and ax, PDE\_A\_CLEAR ; 0F000h ; clear lower 12 (attribute) bits

3554 <1> ; EAX = PHYSICAL (flat) ADDRESS OF THE PAGE TABLE

3555 000058EA 8B3C24 <1> mov edi, [esp] ; \*

3556 <1> ; EDI = PTE index (of page directory)

3557 <1> ;and edi, PTE\_MASK ; PTE entry index in the page table

3558 <1> ; EBX = next PDE address

3559 000058ED 89FE <1> mov esi, edi ; PTE index in page table (0-1023)

3560 000058EF C1E702 <1> shl edi, 2 ; convert PTE index to dword offset

3561 000058F2 01C7 <1> add edi, eax ; now, edi points to requested PTE

3562 <1> a\_u\_pt\_0:

3563 <1> ; 02/05/2017

3564 000058F4 8B07 <1> mov eax, [edi]

3565 <1> ;

3566 000058F6 A801 <1> test al, PTE\_A\_PRESENT ; bit 0, present flag (must be 1)

3567 000058F8 7445 <1> jz short a\_u\_pt\_1

3568 <1> ;

3569 000058FA A802 <1> test al, PTE\_A\_WRITE ; bit 1, writable (r/w) flag

3570 <1> ; (must be 1)

3571 000058FC 7550 <1> jnz short a\_u\_pt\_3

3572 <1> ; Read only -duplicated- page (belongs to a parent or a child)

3573 000058FE 66A90002 <1> test ax, PTE\_DUPLICATED ; Was this page duplicated

3574 <1> ; as child's page ?

3575 00005902 7455 <1> jz short a\_u\_pt\_4 ; Clear PTE but don't deallocate the page!

3576 <1> ;

3577 <1> ; check the parent's PTE value is read only & same page or not..

3578 <1> ; EDX = page directory entry index (0-1023)

3579 00005904 52 <1> push edx ; \*\*

3580 00005905 53 <1> push ebx ; \*\*\*

3581 <1> ; ESI = page table entry index (0-1023)

3582 <1> ;push esi ; \*\*\*\* ; 20/05/2017

3583 00005906 8B1D[BC030300] <1> mov ebx, [u.ppgdir] ; page directory of the parent process

3584 0000590C 66C1E202 <1> shl dx, 2 ; \*4

3585 00005910 01D3 <1> add ebx, edx ; PTE address,0 (for the parent)

3586 00005912 8B13 <1> mov edx, [ebx] ; page table address

3587 00005914 F6C201 <1> test dl, PDE\_A\_PRESENT ; present (valid) or not ?

3588 00005917 7433 <1> jz short a\_u\_pt\_2 ; parent process does not use this page

3589 00005919 6681E200F0 <1> and dx, PDE\_A\_CLEAR ; 0F000h ; Clear attribute bits

3590 0000591E 66C1E602 <1> shl si, 2 ; \*4

3591 <1> ; ESI = page table entry offset (0-4092)

3592 00005922 01D6 <1> add esi, edx ; PTE address (for the parent)

3593 00005924 8B1E <1> mov ebx, [esi]

3594 00005926 F6C301 <1> test bl, PTE\_A\_PRESENT ; present or not ?

3595 00005929 7421 <1> jz short a\_u\_pt\_2 ; parent process does not use this page

3596 0000592B 662500F0 <1> and ax, PTE\_A\_CLEAR ; 0F000h ; Clear attribute bits

3597 0000592F 6681E300F0 <1> and bx, PTE\_A\_CLEAR ; 0F000h ; Clear attribute bits

3598 00005934 39D8 <1> cmp eax, ebx ; parent's and child's pages are same ?

3599 00005936 7514 <1> jne short a\_u\_pt\_2 ; not same page

3600 <1> ; deallocate the child's page

3601 00005938 800E02 <1> or byte [esi], PTE\_A\_WRITE ; convert to writable page (parent)

3602 <1> ;pop esi ; \*\*\*\* ; 20/05/2017

3603 0000593B 5B <1> pop ebx ; \*\*\*

3604 0000593C 5A <1> pop edx ; \*\*

3605 0000593D EB1A <1> jmp short a\_u\_pt\_4

3606 <1> a\_u\_pt\_1:

3607 0000593F 09C0 <1> or eax, eax ; swapped page ?

3608 00005941 7416 <1> jz short a\_u\_pt\_4 ; no

3609 <1> ; yes

3610 00005943 D1E8 <1> shr eax, 1

3611 00005945 E8B5F8FFFF <1> call unlink\_swap\_block ; Deallocate swapped page block

3612 <1> ; on the swap disk (or in file)

3613 0000594A EB0D <1> jmp short a\_u\_pt\_4

3614 <1> a\_u\_pt\_2:

3615 <1> ;pop esi ; \*\*\*\* ; 20/05/2017

3616 0000594C 5B <1> pop ebx ; \*\*\*

3617 0000594D 5A <1> pop edx ; \*\*

3618 <1> a\_u\_pt\_3:

3619 0000594E 66A90004 <1> test ax, PTE\_SHARED ; shared or direct memory access indicator

3620 00005952 7505 <1> jnz short a\_u\_pt\_4 ; AVL bit 1 = 1, do not deallocate this page!

3621 <1> ;

3622 <1> ;and ax, PTE\_A\_CLEAR ; 0F000h ; clear lower 12 (attribute) bits

3623 00005954 E8FFF3FFFF <1> call deallocate\_page ; set the mem allocation bit of this page

3624 <1> ;

3625 <1> a\_u\_pt\_4:

3626 00005959 89E8 <1> mov eax, ebp ; physical address

3627 0000595B 0C07 <1> or al, PTE\_A\_PRESENT + PTE\_A\_WRITE + PTE\_A\_USER ; 04/03/2017

3628 0000595D AB <1> stosd

3629 0000595E 5E <1> pop esi ; \* ; 20/05/2017

3630 0000595F 49 <1> dec ecx ; remain page count

3631 00005960 7417 <1> jz short a\_u\_pd\_5

3632 00005962 81C500100000 <1> add ebp, PAGE\_SIZE

3633 00005968 46 <1> inc esi ; next PTE (index)

3634 <1> ; 20/05/2017

3635 <1> ;cmp esi, PAGE\_SIZE/4 ; 1024

3636 <1> ;jb short a\_u\_pt\_0

3637 00005969 6681E6FF03 <1> and si, PTE\_MASK ; 3FFh (0 to 1023)

3638 0000596E 56 <1> push esi ; \*

3639 0000596F 7583 <1> jnz short a\_u\_pt\_0 ; > 0 (<1024)

3640 <1> a\_u\_pd\_3:

3641 00005971 42 <1> inc edx

3642 <1> ; cmp edx, 1024

3643 <1> ; jnb short a\_u\_pd\_4 ; 02/05/2017 (error!, ecx > 0)

3644 00005972 89DE <1> mov esi, ebx ; the next PDE address

3645 00005974 E953FFFFFF <1> jmp a\_u\_pd\_0

3646 <1> a\_u\_pd\_4:

3647 <1> ; 02/05/2017

3648 <1> ; stc

3649 <1> a\_u\_pd\_5:

3650 <1> ; 20/05/2017

3651 <1> ;pop edi ; \*

3652 00005979 C3 <1> retn

3653 <1>

3654 <1>

3655 <1> ; /// End Of MEMORY MANAGEMENT FUNCTIONS ///

3656 <1>

3657 <1> ;; Data:

3658 <1>

3659 <1> ; 09/03/2015

3660 <1> ;swpq\_count: dw 0 ; count of pages on the swap que

3661 <1> ;swp\_drv: dd 0 ; logical drive description table address of the swap drive/disk

3662 <1> ;swpd\_size: dd 0 ; size of swap drive/disk (volume) in sectors (512 bytes).

3663 <1> ;swpd\_free: dd 0 ; free page blocks (4096 bytes) on swap disk/drive (logical)

3664 <1> ;swpd\_next: dd 0 ; next free page block

3665 <1> ;swpd\_last: dd 0 ; last swap page block

2160 %include 'timer.s' ; 17/01/2015

1 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2 <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0 - timer.s

3 <1> ; ----------------------------------------------------------------------------

4 <1> ; Last Update: 15/01/2017

5 <1> ; ----------------------------------------------------------------------------

6 <1> ; Beginning: 17/01/2016

7 <1> ; ----------------------------------------------------------------------------

8 <1> ; Assembler: NASM version 2.11 (trdos386.s)

9 <1> ; ----------------------------------------------------------------------------

10 <1> ; Turkish Rational DOS

11 <1> ; Operating System Project v2.0 by ERDOGAN TAN (Beginning: 04/01/2016)

12 <1> ;

13 <1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan

14 <1> ;

15 <1> ; Derived from 'IBM PC-AT' BIOS source code (1985)

16 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

17 <1>

18 <1> ; TRDOS 386 (TRDOS v2.0) Kernel - TIMER & REAL TIME CLOCK (BIOS) FUNCTIONS

19 <1>

20 <1> ; IBM PC-AT BIOS Source Code ('BIOS2.ASM')

21 <1> ; TITLE BIOS2 ---- 06/10/85 BIOS INTERRUPT ROUTINES

22 <1>

23 <1> ;

24 <1> ; ///////// TIMER (& REAL TIME CLOCK) FUNCTIONS ///////////////

25 <1>

26 <1> int1Ah:

27 <1> ; 29/01/2016

28 <1> ; 17/01/2016 (TRDOS 386 = TRDOS v2.0)

29 0000597A 9C <1> pushfd

30 0000597B 0E <1> push cs

31 0000597C E801000000 <1> call TIME\_OF\_DAY\_1

32 00005981 C3 <1> retn

33 <1>

34 <1> ;--- INT 1A H -- (TIME OF DAY) -------------------------------------------------

35 <1> ; THIS BIOS ROUTINE ALLOWS THE CLOCKS TO BE SET OR READ :

36 <1> ; :

37 <1> ; PARAMETERS: :

38 <1> ; (AH) = 00H READ THE CURRENT SETTING AND RETURN WITH, :

39 <1> ; (CX) = HIGH PORTION OF COUNT :

40 <1> ; (DX) = LOW PORTION OF COUNT :

41 <1> ; (AL) = 0 TIMER HAS NOT PASSED 24 HOURS SINCE LAST READ :

42 <1> ; 1 IF ON ANOTHER DAY. (RESET TO ZERO AFTER READ) :

43 <1> ; :

44 <1> ; (AH) = 01H SET THE CURRENT CLOCK USING, :

45 <1> ; (CX) = HIGH PORTION OF COUNT :

46 <1> ; (DX) = LOW PORTION OF COUNT. :

47 <1> ; :

48 <1> ; NOTE: COUNTS OCCUR AT THE RATE OF 1193180/65536 COUNTS/SECOND :

49 <1> ; (OR ABOUT 18.2 PER SECOND -- SEE EQUATES) :

50 <1> ; :

51 <1> ; (AH) = 02H READ THE REAL TIME CLOCK AND RETURN WITH, :

52 <1> ; (CH) = HOURS IN BCD (00-23) :

53 <1> ; (CL) = MINUTES IN BCD (00-59) :

54 <1> ; (DH) = SECONDS IN BCD (00-59) :

55 <1> ; (DL) = DAYLIGHT SAVINGS ENABLE (00-01) :

56 <1> ; :

57 <1> ; (AH) = 03H SET THE REAL TIME CLOCK USING, :

58 <1> ; (CH) = HOURS IN BCD (00-23) :

59 <1> ; (CL) = MINUTES IN BCD (00-59) :

60 <1> ; (DH) = SECONDS IN BCD (00-59) :

61 <1> ; (DL) = 01 IF DAYLIGHT SAVINGS ENABLE OPTION, ELSE 00. :

62 <1> ; :

63 <1> ; NOTE: (DL) = 00 IF DAYLIGHT SAVINGS TIME ENABLE IS NOT ENABLED. :

64 <1> ; (DL) = 01 ENABLES TWO SPECIAL UPDATES THE LAST SUNDAY IN :

65 <1> ; APRIL (1:59:59 --> 3:00:00 AM) AND THE LAST SUNDAY IN :

66 <1> ; OCTOBER (1:59:59 --> 1:00:00 AM) THE FIRST TIME. :

67 <1> ; :

68 <1> ; (AH) = 04H READ THE DATE FROM THE REAL TIME CLOCK AND RETURN WITH, :

69 <1> ; (CH) = CENTURY IN BCD (19 OR 20) :

70 <1> ; (CL) = YEAR IN BCD (00-99) :

71 <1> ; (DH) = MONTH IN BCD (01-12) :

72 <1> ; (DL) = DAY IN BCD (01-31). :

73 <1> ; :

74 <1> ; (AH) = 05H SET THE DATE INTO THE REAL TIME CLOCK USING, :

75 <1> ; (CH) = CENTURY IN BCD (19 OR 20) :

76 <1> ; (CL) = YEAR IN BCD (00-99) :

77 <1> ; (DH) = MONTH IN BCD (01-12) :

78 <1> ; (DL) = DAY IN BCD (01-31). :

79 <1> ; :

80 <1> ; (AH) = 06H SET THE ALARM TO INTERRUPT AT SPECIFIED TIME, :

81 <1> ; (CH) = HOURS IN BCD (00-23 (OR FFH)) :

82 <1> ; (CL) = MINUTES IN BCD (00-59 (OR FFH)) :

83 <1> ; (DH) = SECONDS IN BCD (00-59 (OR FFH)) :

84 <1> ; :

85 <1> ; (AH) = 07H RESET THE ALARM INTERRUPT FUNCTION. :

86 <1> ; :

87 <1> ; NOTES: FOR ALL RETURNS CY= 0 FOR SUCCESSFUL OPERATION. :

88 <1> ; FOR (AH)= 2, 4, 6 - CARRY FLAG SET IF REAL TIME CLOCK NOT OPERATING. :

89 <1> ; FOR (AH)= 6 - CARRY FLAG SET IF ALARM ALREADY ENABLED. :

90 <1> ; FOR THE ALARM FUNCTION (AH = 6) THE USER MUST SUPPLY A ROUTINE AND :

91 <1> ; INTERCEPT THE CORRECT ADDRESS IN THE VECTOR TABLE FOR INTERRUPT 4AH. :

92 <1> ; USE 0FFH FOR ANY "DO NOT CARE" POSITION FOR INTERVAL INTERRUPTS. :

93 <1> ; INTERRUPTS ARE DISABLED DURING DATA MODIFICATION. :

94 <1> ; AH & AL ARE RETURNED MODIFIED AND NOT DEFINED EXCEPT WHERE INDICATED. :

95 <1> ;--------------------------------------------------------------------------------

96 <1>

97 <1> ; 15/01/2017

98 <1> ; 14/01/2017

99 <1> ; 07/01/2017

100 <1> ; 02/01/2017

101 <1> ; 29/05/2016

102 <1> ; 29/01/2016

103 <1> ; 17/01/2016 (TRDOS 386 = TRDOS v2.0)

104 <1>

105 <1> ; 29/05/2016

106 <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)

107 <1> int35h: ; Date/Time functions

108 <1>

109 <1> TIME\_OF\_DAY\_1:

110 <1> ;sti ; INTERRUPTS BACK ON

111 <1> ; 29/05/2016

112 00005982 80642408FE <1> and byte [esp+8], 11111110b ; clear carry bit of eflags register

113 <1> ;

114 00005987 80FC08 <1> cmp ah, (RTC\_TBE-RTC\_TB)/4 ; CHECK IF COMMAND IN VALID RANGE (0-7)

115 0000598A F5 <1> cmc ; COMPLEMENT CARRY FOR ERROR EXIT

116 <1> ; (\*) jc short TIME\_9 ; EXIT WITH CARRY = 1 IF NOT VALID

117 0000598B 721A <1> jc short \_TIME\_9 ; 29/05/2016

118 <1>

119 0000598D 1E <1> push ds

120 0000598E 56 <1> push esi

121 0000598F 66BE1000 <1> mov si, KDATA ; kernel data segment

122 00005993 8EDE <1> mov ds, si

123 <1>

124 <1> ;;15/01/2017

125 <1> ; 14/01/2017

126 <1> ; 02/01/2017

127 <1> ;;mov byte [intflg], 35h ; date & time interrupt

128 <1> ;sti

129 <1> ;

130 00005995 C0E402 <1> shl ah, 2 ; convert function to dword offset

131 00005998 0FB6F4 <1> movzx esi, ah ; PLACE INTO ADDRESSING REGISTER

132 <1> ;cli ; NO INTERRUPTS DURING TIME FUNCTIONS

133 0000599B FF96[AD590000] <1> call [esi+RTC\_TB] ; VECTOR TO FUNCTION REQUESTED WITH CY=0

134 <1> ; RETURN WITH CARRY FLAG SET FOR RESULT

135 <1> ;sti ; INTERRUPTS BACK ON

136 000059A1 B400 <1> mov ah, 0 ; CLEAR (AH) TO ZERO

137 000059A3 5E <1> pop esi ; RECOVER USERS REGISTER

138 000059A4 1F <1> pop ds ; RECOVER USERS SEGMENT SELECTOR

139 <1>

140 <1> ;;15/01/2017

141 <1> ; 02/01/2017

142 <1> ;;mov byte [ss:intflg], 0 ; 07/01/2017

143 <1>

144 <1> ;TIME\_9:

145 <1> ; RETURN WITH CY= 0 IF NO ERROR

146 <1> ; (\*) 29/05/2016

147 <1> ; (\*) retf 4 ; skip eflags on stack

148 000059A5 7305 <1> jnc short \_TIME\_10

149 <1> \_TIME\_9:

150 <1> ; 29/05/2016 -set carry flag on stack-

151 <1> ; [esp] = EIP

152 <1> ; [esp+4] = CS

153 <1> ; [esp+8] = E-FLAGS

154 000059A7 804C240801 <1> or byte [esp+8], 1 ; set carry bit of eflags register

155 <1> ; [esp+12] = ESP (user)

156 <1> ; [esp+16] = SS (User)

157 <1> \_TIME\_10:

158 000059AC CF <1> iretd

159 <1>

160 <1> ; (\*) 29/05/2016 - 'ref 4' intruction causes to stack fault

161 <1> ; (OUTER-PRIVILEGE-LEVEL)

162 <1> ; INTEL 80386 PROGRAMMER'S REFERENCE MANUAL 1986

163 <1> ; // RETF instruction:

164 <1> ;

165 <1> ; IF OperandMode=32 THEN

166 <1> ; Load CS:EIP from stack;

167 <1> ; Set CS RPL to CPL;

168 <1> ; Increment eSP by 8 plus the immediate offset if it exists;

169 <1> ; Load SS:eSP from stack;

170 <1> ; ELSE (\* OperandMode=16 \*)

171 <1> ; Load CS:IP from stack;

172 <1> ; Set CS RPL to CPL;

173 <1> ; Increment eSP by 4 plus the immediate offset if it exists;

174 <1> ; Load SS:eSP from stack;

175 <1> ; FI;

176 <1> ;

177 <1> ; //

178 <1> ; ROUTINE VECTOR TABLE (AH)=

179 <1> RTC\_TB:

180 000059AD [CD590000] <1> dd RTC\_00 ; 0 = READ CURRENT CLOCK COUNT

181 000059B1 [E0590000] <1> dd RTC\_10 ; 1 = SET CLOCK COUNT

182 000059B5 [EE590000] <1> dd RTC\_20 ; 2 = READ THE REAL TIME CLOCK TIME

183 000059B9 [1D5A0000] <1> dd RTC\_30 ; 3 = SET REAL TIME CLOCK TIME

184 000059BD [5F5A0000] <1> dd RTC\_40 ; 4 = READ THE REAL TIME CLOCK DATE

185 000059C1 [8C5A0000] <1> dd RTC\_50 ; 5 = SET REAL TIME CLOCK DATE

186 000059C5 [D95A0000] <1> dd RTC\_60 ; 6 = SET THE REAL TIME CLOCK ALARM

187 000059C9 [2C5B0000] <1> dd RTC\_70 ; 7 = RESET ALARM

188 <1>

189 <1> RTC\_TBE equ $

190 <1>

191 <1> RTC\_00: ; READ TIME COUNT

192 000059CD A0[BC580100] <1> mov al, [TIMER\_OFL] ; GET THE OVERFLOW FLAG

193 000059D2 C605[BC580100]00 <1> mov byte [TIMER\_OFL], 0 ; AND THEN RESET THE OVERFLOW FLAG

194 000059D9 8B0D[B8580100] <1> mov ecx, [TIMER\_LH] ; GET COUNT OF TIME

195 000059DF C3 <1> retn

196 <1>

197 <1> RTC\_10: ; SET TIME COUNT

198 000059E0 890D[B8580100] <1> mov [TIMER\_LH], ecx ; SET TIME COUNT

199 000059E6 C605[BC580100]00 <1> mov byte [TIMER\_OFL], 0 ; RESET OVERFLOW FLAG

200 000059ED C3 <1> retn ; RETURN WITH NO CARRY

201 <1>

202 <1> RTC\_20: ; GET RTC TIME

203 000059EE E8EB010000 <1> call UPD\_IPR ; CHECK FOR UPDATE IN PROCESS

204 000059F3 7227 <1> jc short RTC\_29 ; EXIT IF ERROR (CY= 1)

205 <1>

206 000059F5 B000 <1> mov al, CMOS\_SECONDS ; SET ADDRESS OF SECONDS

207 000059F7 E8FD010000 <1> call CMOS\_READ ; GET SECONDS

208 000059FC 88C6 <1> mov dh, al ; SAVE

209 000059FE B00B <1> mov al, CMOS\_REG\_B ; ADDRESS ALARM REGISTER

210 00005A00 E8F4010000 <1> call CMOS\_READ ; READ CURRENT VALUE OF DSE BIT

211 00005A05 2401 <1> and al, 00000001b ; MASK FOR VALID DSE BIT

212 00005A07 88C2 <1> mov dl, al ; SET [DL] TO ZERO FOR NO DSE BIT

213 00005A09 B002 <1> mov al, CMOS\_MINUTES ; SET ADDRESS OF MINUTES

214 00005A0B E8E9010000 <1> call CMOS\_READ ; GET MINUTES

215 00005A10 88C1 <1> mov cl, al ; SAVE

216 00005A12 B004 <1> mov al, CMOS\_HOURS ; SET ADDRESS OF HOURS

217 00005A14 E8E0010000 <1> call CMOS\_READ ; GET HOURS

218 00005A19 88C5 <1> mov ch, al ; SAVE

219 00005A1B F8 <1> clc ; SET CY= 0

220 <1> RTC\_29:

221 00005A1C C3 <1> retn ; RETURN WITH RESULT IN CARRY FLAG

222 <1>

223 <1> RTC\_30: ; SET RTC TIME

224 00005A1D E8BC010000 <1> call UPD\_IPR ; CHECK FOR UPDATE IN PROCESS

225 00005A22 7305 <1> jnc short RTC\_35 ; GO AROUND IF CLOCK OPERATING

226 00005A24 E817010000 <1> call RTC\_STA ; ELSE TRY INITIALIZING CLOCK

227 <1> RTC\_35:

228 00005A29 88F4 <1> mov ah, dh ; GET TIME BYTE - SECONDS

229 00005A2B B000 <1> mov al, CMOS\_SECONDS ; ADDRESS SECONDS

230 00005A2D E8E0010000 <1> call CMOS\_WRITE ; UPDATE SECONDS

231 00005A32 88CC <1> mov ah, cl ; GET TIME BYTE - MINUTES

232 00005A34 B002 <1> mov al, CMOS\_MINUTES ; ADDRESS MINUTES

233 00005A36 E8D7010000 <1> call CMOS\_WRITE ; UPDATE MINUTES

234 00005A3B 88EC <1> mov ah, ch ; GET TIME BYTE - HOURS

235 00005A3D B004 <1> mov al, CMOS\_HOURS ; ADDRESS HOURS

236 00005A3F E8CE010000 <1> call CMOS\_WRITE ; UPDATE ADDRESS

237 <1> ;mov al, CMOS\_REG\_B ; ADDRESS ALARM REGISTER

238 <1> ;mov ah, al

239 00005A44 66B80B0B <1> mov ax, CMOS\_REG\_B \* 257

240 00005A48 E8AC010000 <1> call CMOS\_READ ; READ CURRENT TIME

241 00005A4D 2462 <1> and al, 01100010b ; MASK FOR VALID BIT POSITIONS

242 00005A4F 0C02 <1> or al, 00000010b ; TURN ON 24 HOUR MODE

243 00005A51 80E201 <1> and dl, 00000001b ; USE ONLY THE DSE BIT

244 00005A54 08D0 <1> or al, dl ; GET DAY LIGHT SAVINGS TIME BIT (OSE)

245 00005A56 86E0 <1> xchg ah, al ; PLACE IN WORK REGISTER AND GET ADDRESS

246 00005A58 E8B5010000 <1> call CMOS\_WRITE ; SET NEW ALARM SITS

247 00005A5D F8 <1> clc ; SET CY= 0

248 00005A5E C3 <1> retn ; RETURN WITH CY= 0

249 <1>

250 <1> RTC\_40: ; GET RTC DATE

251 00005A5F E87A010000 <1> call UPD\_IPR ; CHECK FOR UPDATE IN PROCESS

252 00005A64 7225 <1> jc short RTC\_49 ; EXIT IF ERROR (CY= 1)

253 <1>

254 00005A66 B007 <1> mov al, CMOS\_DAY\_MONTH ; ADDRESS DAY OF MONTH

255 00005A68 E88C010000 <1> call CMOS\_READ ; READ DAY OF MONTH

256 00005A6D 88C2 <1> mov dl, al ; SAVE

257 00005A6F B008 <1> mov al, CMOS\_MONTH ; ADDRESS MONTH

258 00005A71 E883010000 <1> call CMOS\_READ ; READ MONTH

259 00005A76 88C6 <1> mov dh, al ; SAVE

260 00005A78 B009 <1> mov al, CMOS\_YEAR ; ADDRESS YEAR

261 00005A7A E87A010000 <1> call CMOS\_READ ; READ YEAR

262 00005A7F 88C1 <1> mov cl, al ; SAVE

263 00005A81 B032 <1> mov al, CMOS\_CENTURY ; ADDRESS CENTURY LOCATION

264 00005A83 E871010000 <1> call CMOS\_READ ; GET CENTURY BYTE

265 00005A88 88C5 <1> mov ch, al ; SAVE

266 00005A8A F8 <1> clc ; SET CY=0

267 <1> RTC\_49:

268 00005A8B C3 <1> retn ; RETURN WITH RESULTS IN CARRY FLAG

269 <1>

270 <1> RTC\_50: ; SET RTC DATE

271 00005A8C E84D010000 <1> call UPD\_IPR ; CHECK FOR UPDATE IN PROCESS

272 00005A91 7305 <1> jnc short RTC\_55 ; GO AROUND IF NO ERROR

273 00005A93 E8A8000000 <1> call RTC\_STA ; ELSE INITIALIZE CLOCK

274 <1> RTC\_55:

275 00005A98 66B80600 <1> mov ax, CMOS\_DAY\_WEEK ; ADDRESS OF DAY OF WEEK BYTE

276 00005A9C E871010000 <1> call CMOS\_WRITE ; LOAD ZEROS TO DAY OF WEEK

277 00005AA1 88D4 <1> mov ah, dl ; GET DAY OF MONTH BYTE

278 00005AA3 B007 <1> mov al, CMOS\_DAY\_MONTH ; ADDRESS DAY OF MONTH BYTE

279 00005AA5 E868010000 <1> call CMOS\_WRITE ; WRITE OF DAY OF MONTH REGISTER

280 00005AAA 88F4 <1> mov ah, dh ; GET MONTH

281 00005AAC B008 <1> mov al, CMOS\_MONTH ; ADDRESS MONTH BYTE

282 00005AAE E85F010000 <1> call CMOS\_WRITE ; WRITE MONTH REGISTER

283 00005AB3 88CC <1> mov ah, cl ; GET YEAR BYTE

284 00005AB5 B009 <1> mov al, CMOS\_YEAR ; ADDRESS YEAR REGISTER

285 00005AB7 E856010000 <1> call CMOS\_WRITE ; WRITE YEAR REGISTER

286 00005ABC 88EC <1> mov ah, ch ; GET CENTURY BYTE

287 00005ABE B032 <1> mov al, CMOS\_CENTURY ; ADDRESS CENTURY BYTE

288 00005AC0 E84D010000 <1> call CMOS\_WRITE ; WRITE CENTURY LOCATION

289 <1> ;mov al, CMOS\_REG\_B ; ADDRESS ALARM REGISTER

290 <1> ;mov ah, al

291 00005AC5 66B80B0B <1> mov ax, CMOS\_REG\_B \* 257

292 00005AC9 E82B010000 <1> call CMOS\_READ ; READ WIRRENT SETTINGS

293 00005ACE 247F <1> and al, 07Fh ; CLEAR 'SET BIT'

294 00005AD0 86E0 <1> xchg ah, al ; MOVE TO WORK REGISTER

295 00005AD2 E83B010000 <1> call CMOS\_WRITE ; AND START CLOCK UPDATING

296 00005AD7 F8 <1> clc ; SET CY= 0

297 00005AD8 C3 <1> retn ; RETURN CY=0

298 <1>

299 <1> RTC\_60: ; SET RTC ALARM

300 00005AD9 B00B <1> mov al, CMOS\_REG\_B ; ADDRESS ALARM

301 00005ADB E819010000 <1> call CMOS\_READ ; READ ALARM REGISTER

302 00005AE0 A820 <1> test al, 20h ; CHECK FOR ALARM ALREADY ENABLED

303 00005AE2 F9 <1> stc ; SET CARRY IN CASE OF ERROR

304 00005AE3 7542 <1> jnz short RTC\_69 ; ERROR EXIT IF ALARM SET

305 00005AE5 E8F4000000 <1> call UPD\_IPR ; CHECK FOR UPDATE IN PROCESS

306 00005AEA 7305 <1> jnc short RTC\_65 ; SKIP INITIALIZATION IF NO ERROR

307 00005AEC E84F000000 <1> call RTC\_STA ; ELSE INITIALIZE CLOCK

308 <1> RTC\_65:

309 00005AF1 88F4 <1> mov ah, dh ; GET SECONDS BYTE

310 00005AF3 B001 <1> mov al, CMOS\_SEC\_ALARM ; ADDRESS THE SECONDS ALARM REGISTER

311 00005AF5 E818010000 <1> call CMOS\_WRITE ; INSERT SECONDS

312 00005AFA 88CC <1> mov ah, cl ; GET MINUTES PARAMETER

313 00005AFC B003 <1> mov al, CMOS\_MIN\_ALARM ; ADDRESS MINUTES ALARM REGISTER

314 00005AFE E80F010000 <1> call CMOS\_WRITE ; INSERT MINUTES

315 00005B03 88EC <1> mov ah, ch ; GET HOURS PARAMETER

316 00005B05 B005 <1> mov al, CMOS\_HR\_ALARM ; ADDRESS HOUR ALARM REGISTER

317 00005B07 E806010000 <1> call CMOS\_WRITE ; INSERT HOURS

318 00005B0C E4A1 <1> in al, INTB01 ; READ SECOND INTERRUPT MASK REGISTER

319 00005B0E 24FE <1> and al, 0FEh ; ENABLE ALARM TIMER BIT (CY= 0)

320 00005B10 E6A1 <1> out INTB01, al ; WRITE UPDATED MASK

321 <1> ;mov al, CMOS\_REG\_B ; ADDRESS ALARM REGISTER

322 <1> ;mov ah, al

323 00005B12 66B80B0B <1> mov ax, CMOS\_REG\_B \* 257

324 00005B16 E8DE000000 <1> call CMOS\_READ ; READ CURRENT ALARM REGISTER

325 00005B1B 247F <1> and al, 07Fh ; ENSURE SET BIT TURNED OFF

326 00005B1D 0C20 <1> or al, 20h ; TURN ON ALARM ENABLE

327 00005B1F 86E0 <1> xchg ah, al ; MOVE MASK TO OUTPUT REGISTER

328 00005B21 E8EC000000 <1> call CMOS\_WRITE ; WRITE NEW ALARM MASK

329 00005B26 F8 <1> clc ; SET CY= 0

330 <1> RTC\_69:

331 00005B27 66B80000 <1> mov ax, 0 ; CLEAR AX REGISTER

332 00005B2B C3 <1> retn ; RETURN WITH RESULTS IN CARRY FLAC

333 <1>

334 <1> RTC\_70: ; RESET ALARM

335 <1> ;mov al, CMOS\_REG\_B ; ADDRESS ALARM REGISTER

336 <1> ;mov ah, al

337 00005B2C 66B80B0B <1> mov ax, CMOS\_REG\_B \* 257 ; ADDRESS ALARM REGISTER (TO BOTH AH,AL)

338 00005B30 E8C4000000 <1> call CMOS\_READ ; READ ALARM REGISTER

339 00005B35 2457 <1> and al, 57h ; TURN OFF ALARM ENABLE

340 00005B37 86E0 <1> xchg ah, al ; SAVE DATA AND RECOVER ADDRESS

341 00005B39 E8D4000000 <1> call CMOS\_WRITE ; RESTORE NEW VALUE

342 00005B3E F8 <1> clc ; SET CY= 0

343 00005B3F C3 <1> retn ; RETURN WITH NO CARRY

344 <1>

345 <1> RTC\_STA: ; INITIALIZE REAL TIME CLOCK

346 <1> ;mov al, CMOS\_REG\_A ; ADDRESS REGISTER A AND LOAD DATA MASK

347 <1> ;mov ah, 26h

348 00005B40 66B80A26 <1> mov ax, (26h\*100h)+CMOS\_REG\_A

349 00005B44 E8C9000000 <1> call CMOS\_WRITE ; INITIALIZE STATUS REGISTER A

350 <1> ;mov al, CMOS\_REG\_B ; SET "SET BIT" FOR CLOCK INITIALIZATION

351 <1> ;mov ah, 82h

352 00005B49 66B80B82 <1> mov ax, (82h\*100h)+CMOS\_REG\_B

353 00005B4D E8C0000000 <1> call CMOS\_WRITE ; AND 24 HOUR MODE TO REGISTER B

354 00005B52 B00C <1> mov al, CMOS\_REG\_C ; ADDRESS REGISTER C

355 00005B54 E8A0000000 <1> call CMOS\_READ ; READ REGISTER C TO INITIALIZE

356 00005B59 B00D <1> mov al, CMOS\_REG\_D ; ADDRESS REGISTER D

357 00005B5B E899000000 <1> call CMOS\_READ ; READ REGISTER D TO INITIALIZE

358 00005B60 C3 <1> retn

359 <1>

360 <1> ; 17/01/2016 (TRDOS 386 = TRDOS v2.0)

361 <1>

362 <1> ;--- HARDWARE INT 70 H -- ( IRQ LEVEL 8) --------------------------------------

363 <1> ; ALARM INTERRUPT HANDLER (RTC) :

364 <1> ; THIS ROUTINE HANDLES THE PERIODIC AND ALARM INTERRUPTS FROM THE CMOS :

365 <1> ; TIMER. INPUT FREQUENCY IS 1.024 KHZ OR APPROXIMATELY 1024 INTERRUPTS :

366 <1> ; EVERY SECOND FOR THE PERIODIC INTERRUPT. FOR THE ALARM FUNCTION, :

367 <1> ; THE INTERRUPT WILL OCCUR AT THE DESIGNATED TIME. :

368 <1> ; :

369 <1> ; INTERRUPTS ARE ENABLED WHEN THE EVENT OR ALARM FUNCTION IS ACTIVATED. :

370 <1> ; FOR THE EVENT INTERRUPT, THE HANDLER WILL DECREMENT THE WAIT COUNTER :

371 <1> ; AND WHEN IT EXPIRES WILL SET THE DESIGNATED LOCATION TO 80H. FOR :

372 <1> ; THE ALARM INTERRUPT. THE USER MUST PROVIDE A ROUTINE TO INTERCEPT :

373 <1> ; THE CORRECT ADDRESS FROM THE VECTOR TABLE INVOKED BY INTERRUPT 4AH :

374 <1> ; PRIOR TO SETTING THE REAL TIME CLOCK ALARM (INT 1AH, AH= 06H). :

375 <1> ;--------------------------------------------------------------------------------

376 <1>

377 <1> RTC\_A\_INT: ; 07/01/2017

378 <1> ;RTC\_INT: ; ALARM INTERRUPT

379 00005B61 1E <1> push ds ; LEAVE INTERRUPTS DISABLED

380 00005B62 50 <1> push eax ; SAVE REGISTERS

381 00005B63 57 <1> push edi

382 <1> RTC\_I\_1: ; CHECK FOR SECOND INTERRUPT

383 00005B64 66B88C8B <1> mov ax, 256\*(CMOS\_REG\_B+NMI)+CMOS\_REG\_C+NMI ; ALARM AND STATUS

384 00005B68 E670 <1> out CMOS\_PORT, al ; WRITE ALARM FLAG MASK ADDRESS

385 00005B6A 90 <1> nop ; I/O DELAY

386 00005B6B EB00 <1> jmp short $+2

387 00005B6D E471 <1> in al, CMOS\_DATA ; READ AND RESET INTERRUPT REQUEST FLAGS

388 00005B6F A860 <1> test al, 01100000b ; CHECK FOR EITHER INTERRUPT PENDING

389 00005B71 745D <1> jz short RTC\_I\_9 ; EXIT IF NOT A VALID RTC INTERRUPT

390 <1>

391 00005B73 86E0 <1> xchg ah, al ; SAVE FLAGS AND GET ENABLE ADDRESS

392 00005B75 E670 <1> out CMOS\_PORT, al ; WRITE ALARM ENABLE MASK ADDRESS

393 00005B77 90 <1> nop ; I/O DELAY

394 00005B78 EB00 <1> jmp short $+2

395 00005B7A E471 <1> in al, CMOS\_DATA ; READ CURRENT ALARM ENABLE MASK

396 00005B7C 20E0 <1> and al, ah ; ALLOW ONLY SOURCES THAT ARE ENABLED

397 00005B7E A840 <1> test al, 01000000b ; CHECK FOR PERIODIC INTERRUPT

398 00005B80 743B <1> jz short RTC\_I\_5 ; SKIP IF NOT A PERIODIC INTERRUPT

399 <1>

400 <1> ;----- DECREMENT WAIT COUNT BY INTERRUPT INTERVAL

401 <1>

402 00005B82 66BF1000 <1> mov di, KDATA ; kernel data segment

403 00005B86 8EDF <1> mov ds, di

404 <1>

405 00005B88 812D[B0580100]D003- <1> sub dword [RTC\_LH], 976 ; DECREMENT COUNT BY 1/1024

405 00005B90 0000 <1>

406 00005B92 7329 <1> jnc short RTC\_I\_5 ; SKIP TILL 32 BIT WORD LESS THAN ZERO

407 <1>

408 <1> ;----- TURN OFF PERIODIC INTERRUPT ENABLE

409 <1>

410 00005B94 6650 <1> push ax ; SAVE INTERRUPT FLAG MASK

411 00005B96 66B88B8B <1> mov ax, 257\*(CMOS\_REG\_B+NMI) ; INTERRUPT ENABLE REGISTER

412 00005B9A E670 <1> out CMOS\_PORT, al ; WRITE ADDRESS TO CMOS CLOCK

413 00005B9C 90 <1> nop ; I/O DELAY

414 00005B9D EB00 <1> jmp short $+2

415 00005B9F E471 <1> in al, CMOS\_DATA ; READ CURRENT ENABLES

416 00005BA1 24BF <1> and al, 0BFh ; TURN OFF PIE

417 00005BA3 86C4 <1> xchg al, ah ; GET CMOS ADDRESS AND SAVE VALUE

418 00005BA5 E670 <1> out CMOS\_PORT, al ; ADDRESS REGISTER B

419 00005BA7 86C4 <1> xchg al, ah ; GET NEW INTERRUPT ENABLE MASK

420 00005BA9 E671 <1> out CMOS\_DATA, al ; SET MASK IN INTERRUPT ENABLE REGISTER

421 00005BAB C605[B4580100]00 <1> mov byte [RTC\_WAIT\_FLAG], 0 ; SET FUNCTION ACTIVE FLAG OFF

422 00005BB2 8B3D[B5580100] <1> mov edi, [USER\_FLAG] ; SET UP (DS:DI) TO POINT TO USER FLAG

423 00005BB8 C60780 <1> mov byte [edi], 80h ; TURN ON USERS FLAG

424 00005BBB 6658 <1> pop ax ; GET INTERRUPT SOURCE BACK

425 <1> RTC\_I\_5:

426 00005BBD A820 <1> test al, 00100000b ; TEST FOR ALARM INTERRUPT

427 00005BBF 740D <1> jz short RTC\_I\_7 ; SKIP USER INTERRUPT CALL IF NOT ALARM

428 <1>

429 00005BC1 B00D <1> mov al, CMOS\_REG\_D ; POINT TO DEFAULT READ ONLY REGISTER

430 00005BC3 E670 <1> out CMOS\_PORT, al ; ENABLE NMI AND CMOS ADDRESS TO DEFAULT

431 00005BC5 FB <1> sti ; INTERRUPTS BACK ON NOW

432 00005BC6 52 <1> push edx

433 00005BC7 E8099E0000 <1> call INT4Ah ; TRANSFER TO USER ROUTINE

434 00005BCC 5A <1> pop edx

435 00005BCD FA <1> cli ; BLOCK INTERRUPT FOR RETRY

436 <1> RTC\_I\_7: ; RESTART ROUTINE TO HANDLE DELAYED

437 00005BCE EB94 <1> jmp short RTC\_I\_1 ; ENTRY AND SECOND EVENT BEFORE DONE

438 <1>

439 <1> RTC\_I\_9: ; EXIT - NO PENDING INTERRUPTS

440 00005BD0 B00D <1> mov al, CMOS\_REG\_D ; POINT TO DEFAULT READ ONLY REGISTER

441 00005BD2 E670 <1> out CMOS\_PORT, al ; ENABLE NMI AND CMOS ADDRESS TO DEFAULT

442 00005BD4 B020 <1> mov al, EOI ; END OF INTERRUPT MASK TO 8259 - 2

443 00005BD6 E6A0 <1> out INTB00, al ; TO 8259 - 2

444 00005BD8 E620 <1> out INTA00, al ; TO 8259 - 1

445 00005BDA 5F <1> pop edi ; RESTORE REGISTERS

446 00005BDB 58 <1> pop eax

447 00005BDC 1F <1> pop ds

448 00005BDD CF <1> iretd ; END OF INTERRUPT

449 <1>

450 <1>

451 <1> ; 29/05/2016 - TRDOS 386 (TRDOS v2.0)

452 <1> ; 22/08/2014 (Retro UNIX 386 v1)

453 <1> ; IBM PC/AT BIOS source code ----- 10/06/85 (bios2.asm)

454 <1> UPD\_IPR: ; WAIT TILL UPDATE NOT IN PROGRESS

455 00005BDE 51 <1> push ecx

456 <1>

457 <1> ; 29/05/2016

458 00005BDF B968110000 <1> mov ecx, ((1984+244)\*4)/2 ; AWARD BIOS 1999, ATIME.ASM

459 <1> ; 'WAITCPU\_CK\_UD\_STAT'

460 <1> ; (244Us + 1984Us)

461 <1> ; (assume each read takes

462 <1> ; 2 microseconds).

463 <1> ;mov ecx, 65535

464 <1> ;mov cx, 800 ; SET TIMEOUT LOOP COUNT (= 800)

465 <1> UPD\_10:

466 00005BE4 B00A <1> mov al, CMOS\_REG\_A ; ADDRESS STATUS REGISTER A

467 00005BE6 FA <1> cli ; NO TIMER INTERRUPTS DURING UPDATES

468 00005BE7 E80D000000 <1> call CMOS\_READ ; READ UPDATE IN PROCESS FLAG

469 00005BEC A880 <1> test al, 80h ; IF UIP BIT IS ON ( CANNOT READ TIME )

470 00005BEE 7406 <1> jz short UPD\_90 ; EXIT WITH CY= 0 IF CAN READ CLOCK NOW

471 00005BF0 FB <1> sti ; ALLOW INTERRUPTS WHILE WAITING

472 00005BF1 E2F1 <1> loop UPD\_10 ; LOOP TILL READY OR TIMEOUT

473 00005BF3 31C0 <1> xor eax, eax ; CLEAR RESULTS IF ERROR

474 <1> ; xor ax, ax

475 00005BF5 F9 <1> stc ; SET CARRY FOR ERROR

476 <1> UPD\_90:

477 00005BF6 59 <1> pop ecx ; RESTORE CALLERS REGISTER

478 00005BF7 FA <1> cli ; INTERRUPTS OFF DURING SET

479 00005BF8 C3 <1> retn ; RETURN WITH CY FLAG SET

480 <1>

481 <1>

482 <1> ; 29/05/2016 - TRDOS 386 (TRDOS v2.0)

483 <1> ; 22/08/2014 (Retro UNIX 386 v1)

484 <1> ; IBM PC/AT BIOS source code ----- 10/06/85 (test4.asm)

485 <1>

486 <1> ;--- CMOS\_READ -----------------------------------------------------------------

487 <1> ; READ BYTE FROM CMOS\_SYSTEM CLOCK CONFIGURATION TABLE :

488 <1> ; :

489 <1> ; INPUT: (AL)= CMOS\_TABLE ADDRESS TO BE READ :

490 <1> ; BIT 7 = 0 FOR NMI ENABLED AND 1 FOR NMI DISABLED ON EXIT :

491 <1> ; BITS 6-0 = ADDRESS OF TABLE LOCATION TO READ :

492 <1> ; :

493 <1> ; OUTPUT: (AL) VALUE AT LOCATION (AL) MOVED INTO (AL). IF BIT 7 OF (AL) WAS :

494 <1> ; ON THEN NMI LEFT DISABLED, DURING THE CMOS READ BOTH NMI AND :

495 <1> ; NORMAL INTERRUPTS ARE DISABLED TO PROTECT CMOS DATA INTEGRITY. :

496 <1> ; THE CMOS ADDRESS REGISTER IS POINTED TO A DEFAULT VALUE AND :

497 <1> ; THE INTERRUPT FLAG RESTORED TO THE ENTRY STATE ON RETURN. :

498 <1> ; ONLY THE (AL) REGISTER AND THE NMI STATE IS CHANGED. :

499 <1> ;-------------------------------------------------------------------------------

500 <1>

501 <1> CMOS\_READ:

502 00005BF9 9C <1> pushf ; SAVE INTERRUPT ENABLE STATUS AND FLAGS

503 00005BFA D0C0 <1> rol al, 1 ; MOVE NMI BIT TO LOW POSITION

504 00005BFC F9 <1> stc ; FORCE NMI BIT ON IN CARRY FLAG

505 00005BFD D0D8 <1> rcr al, 1 ; HIGH BIT ON TO DISABLE NMI - OLD IN CY

506 00005BFF FA <1> cli ; DISABLE INTERRUPTS

507 00005C00 E670 <1> out CMOS\_PORT, al ; ADDRESS LOCATION AND DISABLE NMI

508 <1> ; 29/05/2016

509 <1> ;nop ; I/O DELAY

510 00005C02 E6EB <1> out 0ebh,al ; NEWIODELAY ; AWARD BIOS 1999, ATIME.ASM

511 <1> ;

512 00005C04 E471 <1> in al, CMOS\_DATA ; READ THE REQUESTED CMOS LOCATION

513 00005C06 6650 <1> push ax ; SAVE (AH) REGISTER VALUE AND CMOS BYTE

514 <1> ; 15/03/2015 ; IBM PC/XT Model 286 BIOS source code

515 <1> ; ----- 10/06/85 (test4.asm)

516 00005C08 B01E <1> mov al, CMOS\_SHUT\_DOWN\*2 ; GET ADDRESS OF DEFAULT LOCATION

517 <1> ;mov al, CMOS\_REG\_D\*2 ; GET ADDRESS OF DEFAULT LOCATION

518 00005C0A D0D8 <1> rcr al, 1 ; PUT ORIGINAL NMI MASK BIT INTO ADDRESS

519 00005C0C E670 <1> out CMOS\_PORT, al ; SET DEFAULT TO READ ONLY REGISTER

520 00005C0E 6658 <1> pop ax ; RESTORE (AH) AND (AL), CMOS BYTE

521 00005C10 9D <1> popf

522 00005C11 C3 <1> retn ; RETURN WITH FLAGS RESTORED

523 <1>

524 <1> ; 17/01/2016 (TRDOS 386 = TRDOS v2.0)

525 <1>

526 <1> ;--- CMOS\_WRITE ----------------------------------------------------------------

527 <1> ; WRITE BYTE TO CMOS SYSTEM CLOCK CONFIGURATION TABLE :

528 <1> ; :

529 <1> ; INPUT: (AL)= CMOS TABLE ADDRESS TO BE WRITTEN TO :

530 <1> ; BIT 7 = 0 FOR NMI ENABLED AND 1 FOR NMI DISABLED ON EXIT :

531 <1> ; BITS 6-0 = ADDRESS OF TABLE LOCATION TO WRITE :

532 <1> ; (AH)= NEW VALUE TO BE PLACED IN THE ADDRESSED TABLE LOCATION :

533 <1> ; :

534 <1> ; OUTPUT: VALUE IN (AH) PLACED IN LOCATION (AL) WITH NMI LEFT DISABLED :

535 <1> ; IF BIT 7 OF (AL) IS ON, DURING THE CMOS UPDATE BOTH NMI AND :

536 <1> ; NORMAL INTERRUPTS ARE DISABLED TO PROTECT CMOS DATA INTEGRITY. :

537 <1> ; THE CMOS ADDRESS REGISTER IS POINTED TO A DEFAULT VALUE AND :

538 <1> ; THE INTERRUPT FLAG RESTORED TO THE ENTRY STATE ON RETURN. :

539 <1> ; ONLY THE CMOS LOCATION AND THE NMI STATE IS CHANGED. :

540 <1> ;-------------------------------------------------------------------------------

541 <1>

542 <1> CMOS\_WRITE: ; WRITE (AH) TO LOCATION (AL)

543 00005C12 9C <1> pushf ; SAVE INTERRUPT ENABLE STATUS AND FLAGS

544 00005C13 6650 <1> push ax ; SAVE WORK REGISTER VALUES

545 00005C15 D0C0 <1> rol al, 1 ; MOVE NMI BIT TO LOW POSITION

546 00005C17 F9 <1> stc ; FORCE NMI BIT ON IN CARRY FLAG

547 00005C18 D0D8 <1> rcr al, 1 ; HIGH BIT ON TO DISABLE NMI - OLD IN CY

548 00005C1A FA <1> cli ; DISABLE INTERRUPTS

549 00005C1B E670 <1> out CMOS\_PORT, al ; ADDRESS LOCATION AND DISABLE NMI

550 00005C1D 88E0 <1> mov al, ah ; GET THE DATA BYTE TO WRITE

551 00005C1F E671 <1> out CMOS\_DATA, al ; PLACE IN REQUESTED CMOS LOCATION

552 00005C21 B01E <1> mov al, CMOS\_SHUT\_DOWN\*2 ; GET ADDRESS OF DEFAULT LOCATION

553 <1> ;mov al, CMOS\_REG\_D\*2 ; GET ADDRESS OF DEFAULT LOCATION

554 00005C23 D0D8 <1> rcr al, 1 ; PUT ORIGINAL NMI MASK BIT INTO ADDRESS

555 00005C25 E670 <1> out CMOS\_PORT, al ; SET DEFAULT TO READ ONLY REGISTER

556 00005C27 90 <1> nop ; I/O DELAY

557 00005C28 E471 <1> in al, CMOS\_DATA ; OPEN STANDBY LATCH

558 00005C2A 6658 <1> pop ax ; RESTORE WORK REGISTERS

559 00005C2C 9D <1> popf

560 00005C2D C3 <1> retn

561 <1>

562 <1> ; /// End Of TIMER FUNCTIONS ///

2161

2162 00005C2E 90<rept> Align 16

2163

2164 gdt: ; Global Descriptor Table

2165 ; (30/07/2015, conforming cs)

2166 ; (26/03/2015)

2167 ; (24/03/2015, tss)

2168 ; (19/03/2015)

2169 ; (29/12/2013)

2170 ;

2171 00005C30 0000000000000000 dw 0, 0, 0, 0 ; NULL descriptor

2172 ; 18/08/2014

2173 ; 8h kernel code segment, base = 00000000h

2174 ;dw 0FFFFh, 0, 9E00h, 00CFh ; KCODE ; 30/12/2016

2175 00005C38 FFFF0000009ACF00 dw 0FFFFh, 0, 9A00h, 00CFh ; KCODE

2176 ; 10h kernel data segment, base = 00000000h

2177 00005C40 FFFF00000092CF00 dw 0FFFFh, 0, 9200h, 00CFh ; KDATA

2178 ; 1Bh user code segment, base address = 400000h ; CORE

2179 ;dw 0FBFFh, 0, 0FE40h, 00CFh ; UCODE ; 30/12/2016

2180 00005C48 FFFB000040FACF00 dw 0FBFFh, 0, 0FA40h, 00CFh ; UCODE

2181 ; 23h user data segment, base address = 400000h ; CORE

2182 00005C50 FFFB000040F2CF00 dw 0FBFFh, 0, 0F240h, 00CFh ; UDATA

2183 ; Task State Segment

2184 00005C58 6700 dw 0067h ; Limit = 103 ; (104-1, tss size = 104 byte,

2185 ; no IO permission in ring 3)

2186 gdt\_tss0:

2187 00005C5A 0000 dw 0 ; TSS base address, bits 0-15

2188 gdt\_tss1:

2189 00005C5C 00 db 0 ; TSS base address, bits 16-23

2190 ; 49h

2191 00005C5D E9 db 11101001b ; E9h => P=1/DPL=11/0/1/0/B/1 --> B = Task is busy (1)

2192 00005C5E 00 db 0 ; G/0/0/AVL/LIMIT=0000 ; (Limit bits 16-19 = 0000) (G=0, 1 byte)

2193 gdt\_tss2:

2194 00005C5F 00 db 0 ; TSS base address, bits 24-31

2195

2196 gdt\_end:

2197 ;; 9Eh = 1001 1110b (GDT byte 5) P=1/DPL=00/1/TYPE=1110,

2198 ;; Type= 1 (code)/C=1/R=1/A=0

2199 ; P= Present, DPL=0=ring 0, 1= user (0= system)

2200 ; 1= Code C= Conforming, R= Readable, A = Accessed

2201

2202 ;; 9Ah = 1001 1010b (GDT byte 5) P=1/DPL=00/1/TYPE=1010,

2203 ;; Type= 1 (code)/C=0/R=1/A=0

2204 ; P= Present, DPL=0=ring 0, 1= user (0= system)

2205 ; 1= Code C= non-Conforming, R= Readable, A = Accessed

2206

2207 ;; 92h = 1001 0010b (GDT byte 5) P=1/DPL=00/1/TYPE=1010,

2208 ;; Type= 0 (data)/E=0/W=1/A=0

2209 ; P= Present, DPL=0=ring 0, 1= user (0= system)

2210 ; 0= Data E= Expansion direction (1= down, 0= up)

2211 ; W= Writeable, A= Accessed

2212

2213 ;; FEh = 1111 1110b (GDT byte 5) P=1/DPL=11/1/TYPE=1110,

2214 ;; Type= 1 (code)/C=1/R=1/A=0

2215 ; P= Present, DPL=3=ring 3, 1= user (0= system)

2216 ; 1= Code C= Conforming, R= Readable, A = Accessed

2217

2218 ;; FAh = 1111 1010b (GDT byte 5) P=1/DPL=11/1/TYPE=1010,

2219 ;; Type= 1 (code)/C=0/R=1/A=0

2220 ; P= Present, DPL=3=ring 3, 1= user (0= system)

2221 ; 1= Code C= non-Conforming, R= Readable, A = Accessed

2222

2223 ;; F2h = 1111 0010b (GDT byte 5) P=1/DPL=11/1/TYPE=0010,

2224 ;; Type= 0 (data)/E=0/W=1/A=0

2225 ; P= Present, DPL=3=ring 3, 1= user (0= system)

2226 ; 0= Data E= Expansion direction (1= down, 0= up)

2227

2228 ;; CFh = 1100 1111b (GDT byte 6) G=1/B=1/0/AVL=0, Limit=1111b (3)

2229

2230 ;; Limit = FFFFFh (=> FFFFFh+1= 100000h) // bits 0-15, 48-51 //

2231 ; = 100000h \* 1000h (G=1) = 4GB

2232 ;; Limit = FFBFFh (=> FFBFFh+1= FFC00h) // bits 0-15, 48-51 //

2233 ; = FFC00h \* 1000h (G=1) = 4GB - 4MB

2234 ; G= Granularity (1= 4KB), B= Big (32 bit),

2235 ; AVL= Available to programmers

2236

2237 gdtd:

2238 00005C60 2F00 dw gdt\_end - gdt - 1 ; Limit (size)

2239 00005C62 [305C0000] dd gdt ; Address of the GDT

2240

2241 ; 20/08/2014

2242 idtd:

2243 00005C66 7F02 dw idt\_end - idt - 1 ; Limit (size)

2244 00005C68 [50550100] dd idt ; Address of the IDT

2245

2246 ; 20/02/2017

2247 ;;; 11/03/2015

2248 %include 'diskdata.s' ; DISK (BIOS) DATA (initialized)

1 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2 <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0 - diskdata.s

3 <1> ; ----------------------------------------------------------------------------

4 <1> ; Last Update: 24/01/2016

5 <1> ; ----------------------------------------------------------------------------

6 <1> ; Beginning: 24/01/2016

7 <1> ; ----------------------------------------------------------------------------

8 <1> ; Assembler: NASM version 2.11 (trdos386.s)

9 <1> ; ----------------------------------------------------------------------------

10 <1> ; Turkish Rational DOS

11 <1> ; Operating System Project v2.0 by ERDOGAN TAN (Beginning: 04/01/2016)

12 <1> ;

13 <1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan

14 <1> ; diskdata.inc (11/03/2015)

15 <1> ;

16 <1> ; Derived from 'IBM PC-XT-286' BIOS source code (1986)

17 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

18 <1>

19 <1> ; Retro UNIX 386 v1 Kernel - DISKDATA.INC

20 <1> ; Last Modification: 11/03/2015

21 <1> ; (Initialized Disk Parameters Data section for 'DISKIO.INC')

22 <1> ;

23 <1>

24 <1> ;----------------------------------------

25 <1> ; 80286 INTERRUPT LOCATIONS :

26 <1> ; REFERENCED BY POST & BIOS :

27 <1> ;----------------------------------------

28 <1>

29 00005C6C [CF5C0000] <1> DISK\_POINTER: dd MD\_TBL6 ; Pointer to Diskette Parameter Table

30 <1>

31 <1> ; IBM PC-XT Model 286 source code ORGS.ASM (06/10/85) - 14/12/2014

32 <1> ;----------------------------------------------------------------

33 <1> ; DISK\_BASE :

34 <1> ; THIS IS THE SET OF PARAMETERS REQUIRED FOR :

35 <1> ; DISKETTE OPERATION. THEY ARE POINTED AT BY THE :

36 <1> ; DATA VARIABLE @DISK\_POINTER. TO MODIFY THE PARAMETERS, :

37 <1> ; BUILD ANOTHER PARAMETER BLOCK AND POINT AT IT :

38 <1> ;----------------------------------------------------------------

39 <1>

40 <1> ;DISK\_BASE:

41 <1> ; DB 11011111B ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE

42 <1> ; DB 2 ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE

43 <1> ; DB MOTOR\_WAIT ; WAIT TIME AFTER OPERATION TILL MOTOR OFF

44 <1> ; DB 2 ; 512 BYTES/SECTOR

45 <1> ; ;DB 15 ; EOT (LAST SECTOR ON TRACK)

46 <1> ; db 18 ; (EOT for 1.44MB diskette)

47 <1> ; DB 01BH ; GAP LENGTH

48 <1> ; DB 0FFH ; DTL

49 <1> ; ;DB 054H ; GAP LENGTH FOR FORMAT

50 <1> ; db 06ch ; (for 1.44MB dsikette)

51 <1> ; DB 0F6H ; FILL BYTE FOR FORMAT

52 <1> ; DB 15 ; HEAD SETTLE TIME (MILLISECONDS)

53 <1> ; DB 8 ; MOTOR START TIME (1/8 SECONDS)

54 <1>

55 <1> ;----------------------------------------

56 <1> ; ROM BIOS DATA AREAS :

57 <1> ;----------------------------------------

58 <1>

59 <1> ;DATA SEGMENT AT 40H ; ADDRESS= 0040:0000

60 <1>

61 <1> ;@EQUIP\_FLAG DW ? ; INSTALLED HARDWARE FLAGS

62 <1>

63 <1> ;----------------------------------------

64 <1> ; DISKETTE DATA AREAS :

65 <1> ;----------------------------------------

66 <1>

67 <1> ;@SEEK\_STATUS DB ? ; DRIVE RECALIBRATION STATUS

68 <1> ; ; BIT 3-0 = DRIVE 3-0 RECALIBRATION

69 <1> ; ; BEFORE NEXT SEEK IF BIT IS = 0

70 <1> ;@MOTOR\_STATUS DB ? ; MOTOR STATUS

71 <1> ; ; BIT 3-0 = DRIVE 3-0 CURRENTLY RUNNING

72 <1> ; ; BIT 7 = CURRENT OPERATION IS A WRITE

73 <1> ;@MOTOR\_COUNT DB ? ; TIME OUT COUNTER FOR MOTOR(S) TURN OFF

74 <1> ;@DSKETTE\_STATUS DB ? ; RETURN CODE STATUS BYTE

75 <1> ; ; CMD\_BLOCK IN STACK FOR DISK OPERATION

76 <1> ;@NEC\_STATUS DB 7 DUP(?) ; STATUS BYTES FROM DISKETTE OPERATION

77 <1>

78 <1> ;----------------------------------------

79 <1> ; POST AND BIOS WORK DATA AREA :

80 <1> ;----------------------------------------

81 <1>

82 <1> ;@INTR\_FLAG DB ? ; FLAG INDICATING AN INTERRUPT HAPPENED

83 <1>

84 <1> ;----------------------------------------

85 <1> ; TIMER DATA AREA :

86 <1> ;----------------------------------------

87 <1>

88 <1> ; 17/12/2014 (IRQ 0 - INT 08H)

89 <1> ;TIMER\_LOW equ 46Ch ; Timer ticks (counter) @ 40h:006Ch

90 <1> ;TIMER\_HIGH equ 46Eh ; (18.2 timer ticks per second)

91 <1> ;TIMER\_OFL equ 470h ; Timer - 24 hours flag @ 40h:0070h

92 <1>

93 <1> ;----------------------------------------

94 <1> ; ADDITIONAL MEDIA DATA :

95 <1> ;----------------------------------------

96 <1>

97 <1> ;@LASTRATE DB ? ; LAST DISKETTE DATA RATE SELECTED

98 <1> ;@DSK\_STATE DB ? ; DRIVE 0 MEDIA STATE

99 <1> ; DB ? ; DRIVE 1 MEDIA STATE

100 <1> ; DB ? ; DRIVE 0 OPERATION START STATE

101 <1> ; DB ? ; DRIVE 1 OPERATION START STATE

102 <1> ;@DSK\_TRK DB ? ; DRIVE 0 PRESENT CYLINDER

103 <1> ; DB ? ; DRIVE 1 PRESENT CYLINDER

104 <1>

105 <1> ;DATA ENDS ; END OF BIOS DATA SEGMENT

106 <1>

107 <1> ;--------------------------------------------------------

108 <1> ; DRIVE TYPE TABLE :

109 <1> ;--------------------------------------------------------

110 <1> ; 16/02/2015 (unix386.s, 32 bit modifications)

111 <1> DR\_TYPE:

112 00005C70 01 <1> DB 01 ;DRIVE TYPE, MEDIA TABLE

113 <1> ;DW MD\_TBL1

114 00005C71 [8E5C0000] <1> dd MD\_TBL1

115 00005C75 82 <1> DB 02+BIT7ON

116 <1> ;DW MD\_TBL2

117 00005C76 [9B5C0000] <1> dd MD\_TBL2

118 00005C7A 02 <1> DR\_DEFAULT: DB 02

119 <1> ;DW MD\_TBL3

120 00005C7B [A85C0000] <1> dd MD\_TBL3

121 00005C7F 03 <1> DB 03

122 <1> ;DW MD\_TBL4

123 00005C80 [B55C0000] <1> dd MD\_TBL4

124 00005C84 84 <1> DB 04+BIT7ON

125 <1> ;DW MD\_TBL5

126 00005C85 [C25C0000] <1> dd MD\_TBL5

127 00005C89 04 <1> DB 04

128 <1> ;DW MD\_TBL6

129 00005C8A [CF5C0000] <1> dd MD\_TBL6

130 <1> DR\_TYPE\_E equ $ ; END OF TABLE

131 <1> ;DR\_CNT EQU (DR\_TYPE\_E-DR\_TYPE)/3

132 <1> DR\_CNT equ (DR\_TYPE\_E-DR\_TYPE)/5

133 <1> ;--------------------------------------------------------

134 <1> ; MEDIA/DRIVE PARAMETER TABLES :

135 <1> ;--------------------------------------------------------

136 <1> ;--------------------------------------------------------

137 <1> ; 360 KB MEDIA IN 360 KB DRIVE :

138 <1> ;--------------------------------------------------------

139 <1> MD\_TBL1:

140 00005C8E DF <1> DB 11011111B ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE

141 00005C8F 02 <1> DB 2 ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE

142 00005C90 25 <1> DB MOTOR\_WAIT ; WAIT TIME AFTER OPERATION TILL MOTOR OFF

143 00005C91 02 <1> DB 2 ; 512 BYTES/SECTOR

144 00005C92 09 <1> DB 09 ; EOT (LAST SECTOR ON TRACK)

145 00005C93 2A <1> DB 02AH ; GAP LENGTH

146 00005C94 FF <1> DB 0FFH ; DTL

147 00005C95 50 <1> DB 050H ; GAP LENGTH FOR FORMAT

148 00005C96 F6 <1> DB 0F6H ; FILL BYTE FOR FORMAT

149 00005C97 0F <1> DB 15 ; HEAD SETTLE TIME (MILLISECONDS)

150 00005C98 08 <1> DB 8 ; MOTOR START TIME (1/8 SECONDS)

151 00005C99 27 <1> DB 39 ; MAX. TRACK NUMBER

152 00005C9A 80 <1> DB RATE\_250 ; DATA TRANSFER RATE

153 <1> ;--------------------------------------------------------

154 <1> ; 360 KB MEDIA IN 1.2 MB DRIVE :

155 <1> ;--------------------------------------------------------

156 <1> MD\_TBL2:

157 00005C9B DF <1> DB 11011111B ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE

158 00005C9C 02 <1> DB 2 ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE

159 00005C9D 25 <1> DB MOTOR\_WAIT ; WAIT TIME AFTER OPERATION TILL MOTOR OFF

160 00005C9E 02 <1> DB 2 ; 512 BYTES/SECTOR

161 00005C9F 09 <1> DB 09 ; EOT (LAST SECTOR ON TRACK)

162 00005CA0 2A <1> DB 02AH ; GAP LENGTH

163 00005CA1 FF <1> DB 0FFH ; DTL

164 00005CA2 50 <1> DB 050H ; GAP LENGTH FOR FORMAT

165 00005CA3 F6 <1> DB 0F6H ; FILL BYTE FOR FORMAT

166 00005CA4 0F <1> DB 15 ; HEAD SETTLE TIME (MILLISECONDS)

167 00005CA5 08 <1> DB 8 ; MOTOR START TIME (1/8 SECONDS)

168 00005CA6 27 <1> DB 39 ; MAX. TRACK NUMBER

169 00005CA7 40 <1> DB RATE\_300 ; DATA TRANSFER RATE

170 <1> ;--------------------------------------------------------

171 <1> ; 1.2 MB MEDIA IN 1.2 MB DRIVE :

172 <1> ;--------------------------------------------------------

173 <1> MD\_TBL3:

174 00005CA8 DF <1> DB 11011111B ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE

175 00005CA9 02 <1> DB 2 ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE

176 00005CAA 25 <1> DB MOTOR\_WAIT ; WAIT TIME AFTER OPERATION TILL MOTOR OFF

177 00005CAB 02 <1> DB 2 ; 512 BYTES/SECTOR

178 00005CAC 0F <1> DB 15 ; EOT (LAST SECTOR ON TRACK)

179 00005CAD 1B <1> DB 01BH ; GAP LENGTH

180 00005CAE FF <1> DB 0FFH ; DTL

181 00005CAF 54 <1> DB 054H ; GAP LENGTH FOR FORMAT

182 00005CB0 F6 <1> DB 0F6H ; FILL BYTE FOR FORMAT

183 00005CB1 0F <1> DB 15 ; HEAD SETTLE TIME (MILLISECONDS)

184 00005CB2 08 <1> DB 8 ; MOTOR START TIME (1/8 SECONDS)

185 00005CB3 4F <1> DB 79 ; MAX. TRACK NUMBER

186 00005CB4 00 <1> DB RATE\_500 ; DATA TRANSFER RATE

187 <1> ;--------------------------------------------------------

188 <1> ; 720 KB MEDIA IN 720 KB DRIVE :

189 <1> ;--------------------------------------------------------

190 <1> MD\_TBL4:

191 00005CB5 DF <1> DB 11011111B ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE

192 00005CB6 02 <1> DB 2 ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE

193 00005CB7 25 <1> DB MOTOR\_WAIT ; WAIT TIME AFTER OPERATION TILL MOTOR OFF

194 00005CB8 02 <1> DB 2 ; 512 BYTES/SECTOR

195 00005CB9 09 <1> DB 09 ; EOT (LAST SECTOR ON TRACK)

196 00005CBA 2A <1> DB 02AH ; GAP LENGTH

197 00005CBB FF <1> DB 0FFH ; DTL

198 00005CBC 50 <1> DB 050H ; GAP LENGTH FOR FORMAT

199 00005CBD F6 <1> DB 0F6H ; FILL BYTE FOR FORMAT

200 00005CBE 0F <1> DB 15 ; HEAD SETTLE TIME (MILLISECONDS)

201 00005CBF 08 <1> DB 8 ; MOTOR START TIME (1/8 SECONDS)

202 00005CC0 4F <1> DB 79 ; MAX. TRACK NUMBER

203 00005CC1 80 <1> DB RATE\_250 ; DATA TRANSFER RATE

204 <1> ;--------------------------------------------------------

205 <1> ; 720 KB MEDIA IN 1.44 MB DRIVE :

206 <1> ;--------------------------------------------------------

207 <1> MD\_TBL5:

208 00005CC2 DF <1> DB 11011111B ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE

209 00005CC3 02 <1> DB 2 ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE

210 00005CC4 25 <1> DB MOTOR\_WAIT ; WAIT TIME AFTER OPERATION TILL MOTOR OFF

211 00005CC5 02 <1> DB 2 ; 512 BYTES/SECTOR

212 00005CC6 09 <1> DB 09 ; EOT (LAST SECTOR ON TRACK)

213 00005CC7 2A <1> DB 02AH ; GAP LENGTH

214 00005CC8 FF <1> DB 0FFH ; DTL

215 00005CC9 50 <1> DB 050H ; GAP LENGTH FOR FORMAT

216 00005CCA F6 <1> DB 0F6H ; FILL BYTE FOR FORMAT

217 00005CCB 0F <1> DB 15 ; HEAD SETTLE TIME (MILLISECONDS)

218 00005CCC 08 <1> DB 8 ; MOTOR START TIME (1/8 SECONDS)

219 00005CCD 4F <1> DB 79 ; MAX. TRACK NUMBER

220 00005CCE 80 <1> DB RATE\_250 ; DATA TRANSFER RATE

221 <1> ;--------------------------------------------------------

222 <1> ; 1.44 MB MEDIA IN 1.44 MB DRIVE :

223 <1> ;--------------------------------------------------------

224 <1> MD\_TBL6:

225 00005CCF AF <1> DB 10101111B ; SRT=A, HD UNLOAD=0F - 1ST SPECIFY BYTE

226 00005CD0 02 <1> DB 2 ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE

227 00005CD1 25 <1> DB MOTOR\_WAIT ; WAIT TIME AFTER OPERATION TILL MOTOR OFF

228 00005CD2 02 <1> DB 2 ; 512 BYTES/SECTOR

229 00005CD3 12 <1> DB 18 ; EOT (LAST SECTOR ON TRACK)

230 00005CD4 1B <1> DB 01BH ; GAP LENGTH

231 00005CD5 FF <1> DB 0FFH ; DTL

232 00005CD6 6C <1> DB 06CH ; GAP LENGTH FOR FORMAT

233 00005CD7 F6 <1> DB 0F6H ; FILL BYTE FOR FORMAT

234 00005CD8 0F <1> DB 15 ; HEAD SETTLE TIME (MILLISECONDS)

235 00005CD9 08 <1> DB 8 ; MOTOR START TIME (1/8 SECONDS)

236 00005CDA 4F <1> DB 79 ; MAX. TRACK NUMBER

237 00005CDB 00 <1> DB RATE\_500 ; DATA TRANSFER RATE

238 <1>

239 <1>

240 <1> ; << diskette.inc >>

241 <1> ; +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

242 <1> ;

243 <1> ;----------------------------------------

244 <1> ; ROM BIOS DATA AREAS :

245 <1> ;----------------------------------------

246 <1>

247 <1> ;DATA SEGMENT AT 40H ; ADDRESS= 0040:0000

248 <1>

249 <1> ;----------------------------------------

250 <1> ; FIXED DISK DATA AREAS :

251 <1> ;----------------------------------------

252 <1>

253 <1> ;DISK\_STATUS1: DB 0 ; FIXED DISK STATUS

254 <1> ;HF\_NUM: DB 0 ; COUNT OF FIXED DISK DRIVES

255 <1> ;CONTROL\_BYTE: DB 0 ; HEAD CONTROL BYTE

256 <1> ;@PORT\_OFF DB ? ; RESERVED (PORT OFFSET)

257 <1>

258 <1> ;----------------------------------------

259 <1> ; ADDITIONAL MEDIA DATA :

260 <1> ;----------------------------------------

261 <1>

262 <1> ;@LASTRATE DB ? ; LAST DISKETTE DATA RATE SELECTED

263 <1> ;HF\_STATUS DB 0 ; STATUS REGISTER

264 <1> ;HF\_ERROR DB 0 ; ERROR REGISTER

265 <1> ;HF\_INT\_FLAG DB 0 ; FIXED DISK INTERRUPT FLAG

266 <1> ;HF\_CNTRL DB 0 ; COMBO FIXED DISK/DISKETTE CARD BIT 0=1

267 <1> ;@DSK\_STATE DB ? ; DRIVE 0 MEDIA STATE

268 <1> ; DB ? ; DRIVE 1 MEDIA STATE

269 <1> ; DB ? ; DRIVE 0 OPERATION START STATE

270 <1> ; DB ? ; DRIVE 1 OPERATION START STATE

271 <1> ;@DSK\_TRK DB ? ; DRIVE 0 PRESENT CYLINDER

272 <1> ; DB ? ; DRIVE 1 PRESENT CYLINDER

273 <1>

274 <1> ;DATA ENDS ; END OF BIOS DATA SEGMENT

275 <1> ;

276 <1> ; +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

277 <1>

278 <1> ERR\_TBL:

279 00005CDC E0 <1> db NO\_ERR

280 00005CDD 024001BB <1> db BAD\_ADDR\_MARK,BAD\_SEEK,BAD\_CMD,UNDEF\_ERR

281 00005CE1 04BB100A <1> db RECORD\_NOT\_FND,UNDEF\_ERR,BAD\_ECC,BAD\_SECTOR

282 <1>

283 <1> ; 17/12/2014 (mov ax, [cfd])

284 <1> ; 11/12/2014

285 00005CE5 00 <1> cfd: db 0 ; current floppy drive (for GET\_PARM)

286 <1> ; 17/12/2014 ; instead of 'DISK\_POINTER'

287 00005CE6 01 <1> pfd: db 1 ; previous floppy drive (for GET\_PARM)

288 <1> ; (initial value of 'pfd

289 <1> ; must be different then 'cfd' value

290 <1> ; to force updating/initializing

291 <1> ; current drive parameters)

292 00005CE7 90 <1> align 2

293 <1>

294 00005CE8 F001 <1> HF\_PORT: dw 1F0h ; Default = 1F0h

295 <1> ; (170h)

296 00005CEA F603 <1> HF\_REG\_PORT: dw 3F6h ; HF\_PORT + 206h

297 <1>

298 <1> ; 05/01/2015

299 00005CEC 00 <1> hf\_m\_s: db 0 ; (0 = Master, 1 = Slave)

300 <1>

301 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2249

2250 00005CED 90 Align 2

2251

2252 ; 04/11/2014 (Retro UNIX 386 v1)

2253 00005CEE 0000 mem\_1m\_1k: dw 0 ; Number of contiguous KB between

2254 ; 1 and 16 MB, max. 3C00h = 15 MB.

2255 00005CF0 0000 mem\_16m\_64k: dw 0 ; Number of contiguous 64 KB blocks

2256 ; between 16 MB and 4 GB.

2257

2258 ; 12/11/2014 (Retro UNIX 386 v1)

2259 00005CF2 00 boot\_drv: db 0 ; boot drive number (physical)

2260 ; 24/11/2014

2261 00005CF3 00 drv: db 0

2262 00005CF4 00 last\_drv: db 0 ; last hdd

2263 00005CF5 00 hdc: db 0 ; number of hard disk drives

2264 ; (present/detected)

2265

2266 ; 24/11/2014 (Retro UNIX 386 v1)

2267 ; Physical drive type & flags

2268 00005CF6 00 fd0\_type: db 0 ; floppy drive type

2269 00005CF7 00 fd1\_type: db 0 ; 4 = 1.44 Mb, 80 track, 3.5" (18 spt)

2270 ; 6 = 2.88 Mb, 80 track, 3.5" (36 spt)

2271 ; 3 = 720 Kb, 80 track, 3.5" (9 spt)

2272 ; 2 = 1.2 Mb, 80 track, 5.25" (15 spt)

2273 ; 1 = 360 Kb, 40 track, 5.25" (9 spt)

2274 00005CF8 00 hd0\_type: db 0 ; EDD status for hd0 (bit 7 = present flag)

2275 00005CF9 00 hd1\_type: db 0 ; EDD status for hd1 (bit 7 = present flag)

2276 00005CFA 00 hd2\_type: db 0 ; EDD status for hd2 (bit 7 = present flag)

2277 00005CFB 00 hd3\_type: db 0 ; EDD status for hd3 (bit 7 = present flag)

2278 ; bit 0 - Fixed disk access subset supported

2279 ; bit 1 - Drive locking and ejecting

2280 ; bit 2 - Enhanced disk drive support

2281 ; bit 3 = Reserved (64 bit EDD support)

2282 ; (If bit 0 is '1' Retro UNIX 386 v1

2283 ; will interpret it as 'LBA ready'!)

2284

2285 ; 11/03/2015 - 10/07/2015

2286 00005CFC 000000000000000000- drv.cylinders: dw 0,0,0,0,0,0,0

2286 00005D05 0000000000

2287 00005D0A 000000000000000000- drv.heads: dw 0,0,0,0,0,0,0

2287 00005D13 0000000000

2288 00005D18 000000000000000000- drv.spt: dw 0,0,0,0,0,0,0

2288 00005D21 0000000000

2289 00005D26 000000000000000000- drv.size: dd 0,0,0,0,0,0,0

2289 00005D2F 000000000000000000-

2289 00005D38 000000000000000000-

2289 00005D41 00

2290 00005D42 00000000000000 drv.status: db 0,0,0,0,0,0,0

2291 00005D49 00000000000000 drv.error: db 0,0,0,0,0,0,0

2292

2293 Align 2

2294

2295 ;;; 11/03/2015

2296 %include 'kybdata.s' ; KEYBOARD (BIOS) DATA

1 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2 <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0 - kybdata.s

3 <1> ; ----------------------------------------------------------------------------

4 <1> ; Last Update: 17/01/2016

5 <1> ; ----------------------------------------------------------------------------

6 <1> ; Beginning: 17/01/2016

7 <1> ; ----------------------------------------------------------------------------

8 <1> ; Assembler: NASM version 2.11 (trdos386.s)

9 <1> ; ----------------------------------------------------------------------------

10 <1> ; Turkish Rational DOS

11 <1> ; Operating System Project v2.0 by ERDOGAN TAN (Beginning: 04/01/2016)

12 <1> ;

13 <1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan

14 <1> ; kybdata.inc (11/03/2015)

15 <1> ;

16 <1> ; Derived from 'IBM PC-XT-286' BIOS source code (1986)

17 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

18 <1>

19 <1> ; Retro UNIX 386 v1 Kernel - KYBDATA.INC

20 <1> ; Last Modification: 11/03/2015

21 <1> ; (Data Section for 'KEYBOARD.INC')

22 <1> ;

23 <1> ; ///////// KEYBOARD DATA ///////////////

24 <1>

25 <1> ; 05/12/2014

26 <1> ; 04/12/2014 (derived from pc-xt-286 bios source code -1986-)

27 <1> ; 03/06/86 KEYBOARD BIOS

28 <1>

29 <1> ;---------------------------------------------------------------------------------

30 <1> ; KEY IDENTIFICATION SCAN TABLES

31 <1> ;---------------------------------------------------------------------------------

32 <1>

33 <1> ;----- TABLES FOR ALT CASE ------------

34 <1> ;----- ALT-INPUT-TABLE

35 00005D50 524F50514B <1> K30: db 82,79,80,81,75

36 00005D55 4C4D474849 <1> db 76,77,71,72,73 ; 10 NUMBER ON KEYPAD

37 <1> ;----- SUPER-SHIFT-TABLE

38 00005D5A 101112131415 <1> db 16,17,18,19,20,21 ; A-Z TYPEWRITER CHARS

39 00005D60 161718191E1F <1> db 22,23,24,25,30,31

40 00005D66 202122232425 <1> db 32,33,34,35,36,37

41 00005D6C 262C2D2E2F30 <1> db 38,44,45,46,47,48

42 00005D72 3132 <1> db 49,50

43 <1>

44 <1> ;----- TABLE OF SHIFT KEYS AND MASK VALUES

45 <1> ;----- KEY\_TABLE

46 00005D74 52 <1> \_K6: db INS\_KEY ; INSERT KEY

47 00005D75 3A4546381D <1> db CAPS\_KEY,NUM\_KEY,SCROLL\_KEY,ALT\_KEY,CTL\_KEY

48 00005D7A 2A36 <1> db LEFT\_KEY,RIGHT\_KEY

49 <1> \_K6L equ $-\_K6

50 <1>

51 <1> ;----- MASK\_TABLE

52 00005D7C 80 <1> \_K7: db INS\_SHIFT ; INSERT MODE SHIFT

53 00005D7D 4020100804 <1> db CAPS\_SHIFT,NUM\_SHIFT,SCROLL\_SHIFT,ALT\_SHIFT,CTL\_SHIFT

54 00005D82 0201 <1> db LEFT\_SHIFT,RIGHT\_SHIFT

55 <1>

56 <1> ;----- TABLES FOR CTRL CASE ;---- CHARACTERS ------

57 00005D84 1BFF00FFFFFF <1> \_K8: db 27,-1,0,-1,-1,-1 ; Esc, 1, 2, 3, 4, 5

58 00005D8A 1EFFFFFFFF1F <1> db 30,-1,-1,-1,-1,31 ; 6, 7, 8, 9, 0, -

59 00005D90 FF7FFF111705 <1> db -1,127,-1,17,23,5 ; =, Bksp, Tab, Q, W, E

60 00005D96 12141915090F <1> db 18,20,25,21,9,15 ; R, T, Y, U, I, O

61 00005D9C 101B1D0AFF01 <1> db 16,27,29,10,-1,1 ; P, [, ], Enter, Ctrl, A

62 00005DA2 13040607080A <1> db 19,4,6,7,8,10 ; S, D, F, G, H, J

63 00005DA8 0B0CFFFFFFFF <1> db 11,12,-1,-1,-1,-1 ; K, L, :, ', `, LShift

64 00005DAE 1C1A18031602 <1> db 28,26,24,3,22,2 ; Bkslash, Z, X, C, V, B

65 00005DB4 0E0DFFFFFFFF <1> db 14,13,-1,-1,-1,-1 ; N, M, ,, ., /, RShift

66 00005DBA 96FF20FF <1> db 150,-1,' ',-1 ; \*, ALT, Spc, CL

67 <1> ; ;----- FUNCTIONS ------

68 00005DBE 5E5F60616263 <1> db 94,95,96,97,98,99 ; F1 - F6

69 00005DC4 64656667FFFF <1> db 100,101,102,103,-1,-1 ; F7 - F10, NL, SL

70 00005DCA 778D848E738F <1> db 119,141,132,142,115,143 ; Home, Up, PgUp, -, Left, Pad5

71 00005DD0 749075917692 <1> db 116,144,117,145,118,146 ; Right, +, End, Down, PgDn, Ins

72 00005DD6 93FFFFFF898A <1> db 147,-1,-1,-1,137,138 ; Del, SysReq, Undef, WT, F11, F12

73 <1>

74 <1> ;----- TABLES FOR LOWER CASE ----------

75 00005DDC 1B3132333435363738- <1> K10: db 27,'1234567890-=',8,9

75 00005DE5 39302D3D0809 <1>

76 00005DEB 71776572747975696F- <1> db 'qwertyuiop[]',13,-1,'asdfghjkl;',39

76 00005DF4 705B5D0DFF61736466- <1>

76 00005DFD 67686A6B6C3B27 <1>

77 00005E04 60FF5C7A786376626E- <1> db 96,-1,92,'zxcvbnm,./',-1,'\*',-1,' ',-1

77 00005E0D 6D2C2E2FFF2AFF20FF <1>

78 <1> ;----- LC TABLE SCAN

79 00005E16 3B3C3D3E3F <1> db 59,60,61,62,63 ; BASE STATE OF F1 - F10

80 00005E1B 4041424344 <1> db 64,65,66,67,68

81 00005E20 FFFF <1> db -1,-1 ; NL, SL

82 <1>

83 <1> ;----- KEYPAD TABLE

84 00005E22 474849FF4BFF <1> K15: db 71,72,73,-1,75,-1 ; BASE STATE OF KEYPAD KEYS

85 00005E28 4DFF4F50515253 <1> db 77,-1,79,80,81,82,83

86 00005E2F FFFF5C8586 <1> db -1,-1,92,133,134 ; SysRq, Undef, WT, F11, F12

87 <1>

88 <1> ;----- TABLES FOR UPPER CASE ----------

89 00005E34 1B21402324255E262A- <1> K11: db 27,'!@#$%',94,'&\*()\_+',8,0

89 00005E3D 28295F2B0800 <1>

90 00005E43 51574552545955494F- <1> db 'QWERTYUIOP{}',13,-1,'ASDFGHJKL:"'

90 00005E4C 507B7D0DFF41534446- <1>

90 00005E55 47484A4B4C3A22 <1>

91 00005E5C 7EFF7C5A584356424E- <1> db 126,-1,'|ZXCVBNM<>?',-1,'\*',-1,' ',-1

91 00005E65 4D3C3E3FFF2AFF20FF <1>

92 <1> ;----- UC TABLE SCAN

93 00005E6E 5455565758 <1> K12: db 84,85,86,87,88 ; SHIFTED STATE OF F1 - F10

94 00005E73 595A5B5C5D <1> db 89,90,91,92,93

95 00005E78 FFFF <1> db -1,-1 ; NL, SL

96 <1>

97 <1> ;----- NUM STATE TABLE

98 00005E7A 3738392D3435362B31- <1> K14: db '789-456+1230.' ; NUMLOCK STATE OF KEYPAD KEYS

98 00005E83 3233302E <1>

99 <1> ;

100 00005E87 FFFF7C8788 <1> db -1,-1,124,135,136 ; SysRq, Undef, WT, F11, F12

101 <1>

102 <1> ; 26/08/2014

103 <1> ; Retro UNIX 8086 v1 - UNIX.ASM (03/03/2014)

104 <1> ; Derived from IBM "pc-at"

105 <1> ; rombios source code (06/10/1985)

106 <1> ; 'dseg.inc'

107 <1>

108 <1> ;---------------------------------------;

109 <1> ; SYSTEM DATA AREA ;

110 <1> ;----------------------------------------

111 00005E8C 00 <1> BIOS\_BREAK db 0 ; BIT 7=1 IF BREAK KEY HAS BEEN PRESSED

112 <1>

113 <1> ;----------------------------------------

114 <1> ; KEYBOARD DATA AREAS ;

115 <1> ;----------------------------------------

116 <1>

117 00005E8D 00 <1> KB\_FLAG db 0 ; KEYBOARD SHIFT STATE AND STATUS FLAGS

118 00005E8E 00 <1> KB\_FLAG\_1 db 0 ; SECOND BYTE OF KEYBOARD STATUS

119 00005E8F 00 <1> KB\_FLAG\_2 db 0 ; KEYBOARD LED FLAGS

120 00005E90 00 <1> KB\_FLAG\_3 db 0 ; KEYBOARD MODE STATE AND TYPE FLAGS

121 00005E91 00 <1> ALT\_INPUT db 0 ; STORAGE FOR ALTERNATE KEY PAD ENTRY

122 00005E92 [A25E0000] <1> BUFFER\_START dd KB\_BUFFER ; OFFSET OF KEYBOARD BUFFER START

123 00005E96 [C25E0000] <1> BUFFER\_END dd KB\_BUFFER + 32 ; OFFSET OF END OF BUFFER

124 00005E9A [A25E0000] <1> BUFFER\_HEAD dd KB\_BUFFER ; POINTER TO HEAD OF KEYBOARD BUFFER

125 00005E9E [A25E0000] <1> BUFFER\_TAIL dd KB\_BUFFER ; POINTER TO TAIL OF KEYBOARD BUFFER

126 <1> ; ------ HEAD = TAIL INDICATES THAT THE BUFFER IS EMPTY

127 00005EA2 0000<rept> <1> KB\_BUFFER times 16 dw 0 ; ROOM FOR 16 SCAN CODE ENTRIES

128 <1>

129 <1> ; /// End Of KEYBOARD DATA ///

2297 %include 'vidata.s' ; VIDEO (BIOS) DATA

1 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2 <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0 - vidata.s

3 <1> ; ----------------------------------------------------------------------------

4 <1> ; Last Update: 31/07/2016

5 <1> ; ----------------------------------------------------------------------------

6 <1> ; Beginning: 16/01/2016

7 <1> ; ----------------------------------------------------------------------------

8 <1> ; Assembler: NASM version 2.11 (trdos386.s)

9 <1> ; ----------------------------------------------------------------------------

10 <1> ; Turkish Rational DOS

11 <1> ; Operating System Project v2.0 by ERDOGAN TAN (Beginning: 04/01/2016)

12 <1> ;

13 <1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan

14 <1> ; vidata.inc (11/03/2015)

15 <1> ;

16 <1> ; Derived from 'IBM PC-AT' BIOS source code (1985)

17 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

18 <1>

19 <1> ; Retro UNIX 386 v1 Kernel - VIDATA.S

20 <1> ; Last Modification: 11/03/2015

21 <1> ; (Data section for 'VIDEO.INC')

22 <1> ;

23 <1> ; ///////// VIDEO DATA ///////////////

24 <1>

25 <1> ;----------------------------------------

26 <1> ; VIDEO DISPLAY DATA AREA ;

27 <1> ;----------------------------------------

28 00005EC2 03 <1> CRT\_MODE: db 3 ; CURRENT DISPLAY MODE (TYPE)

29 00005EC3 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 00005EC4 50 <1> CRT\_COLS: db 80 ; Number of columns

52 <1>

53 <1> ; 01/07/2016

54 00005EC5 00 <1> CRT\_PALETTE: db 0 ; Current palette setting

55 <1>

56 <1> ; 03/07/2016

57 00005EC6 10 <1> CHAR\_HEIGHT: db 16 ; Default character height

58 00005EC7 60 <1> VGA\_VIDEO\_CTL: db 60h ; ROM BIOS DATA AREA Offset 87h

59 00005EC8 F9 <1> VGA\_SWITCHES: db 0F9h ; Feature Bit Switches (the basic screen)

60 00005EC9 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 00005ECA 19 <1> VGA\_ROWS: db 25

73 <1>

74 <1> ; 16/01/2016

75 <1> chr\_attrib: ; Character color/attributes for viode pages (0 to 7)

76 00005ECB 0707070707070707 <1> db 07h, 07h, 07h, 07h, 07h, 07h, 07h, 07h

77 <1> ; 30/01/2016

78 <1> vmode:

79 00005ED3 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 00005EDB 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 00005EDD 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 00005EDE 0302010007040506 <1> db 03h, 02h, 01h, 00h, 07h, 04h, 05h, 06h

96 <1> vga\_g\_modes: ; 31/07/2016

97 00005EE6 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 00005EEE [4E5F0000] <1> dd vga\_mode\_03h

104 00005EF2 [4E5F0000] <1> dd vga\_mode\_03h ; mode 02h -> mode 03h

105 00005EF6 [8E5F0000] <1> dd vga\_mode\_01h

106 00005EFA [8E5F0000] <1> dd vga\_mode\_01h ; mode 00h -> mode 01h

107 <1> ;dd vga\_mode\_07h

108 00005EFE [4E5F0000] <1> dd vga\_mode\_03h ; mode 07h -> mode 03h

109 00005F02 [CE5F0000] <1> dd vga\_mode\_04h

110 00005F06 [CE5F0000] <1> dd vga\_mode\_04h ; mode 05h -> mode 04h

111 00005F0A [0E600000] <1> dd vga\_mode\_06h

112 00005F0E [4E600000] <1> dd vga\_mode\_13h

113 00005F12 [8E600000] <1> dd vga\_mode\_F0h

114 00005F16 [CE600000] <1> dd vga\_mode\_12h

115 00005F1A [0E610000] <1> dd vga\_mode\_6Ah

116 00005F1E [4E610000] <1> dd vga\_mode\_0Dh

117 00005F22 [8E610000] <1> dd vga\_mode\_0Eh

118 00005F26 [CE610000] <1> dd vga\_mode\_10h

119 00005F2A [0E620000] <1> dd vga\_mode\_11h

120 <1>

121 <1> vga\_memmodel:

122 <1> ; 25/07/2016

123 <1> ; 07/07/2016

124 <1> CTEXT equ 0

125 <1> ;MTEXT equ 1

126 <1> MTEXT equ 0 ; mode 07h -> mode 03h

127 <1> CGA equ 2

128 <1> LINEAR8 equ 5

129 <1> PLANAR4 equ 4

130 <1> PLANAR1 equ 3

131 00005F2E 0000000000020202 <1> db CTEXT, CTEXT, CTEXT, CTEXT, MTEXT, CGA, CGA, CGA

132 <1> vga\_g\_memmodel: ; 31/07/2016

133 00005F36 0504040404040403 <1> db LINEAR8, PLANAR4, PLANAR4, PLANAR4, PLANAR4, PLANAR4, PLANAR4, PLANAR1

134 <1> ;vga\_pixbits:

135 <1> ; ; 25/07/2016

136 <1> ; ; 08/07/2016

137 <1> ; db 4, 4, 4, 4, 4, 2, 2, 1, 8, 4, 4, 4, 4, 4, 4, 1

138 <1> vga\_dac\_s:

139 00005F3E 020202020001010103- <1> db 2, 2, 2, 2, 0, 1, 1, 1, 3, 3, 2, 2, 1, 1, 2, 2

139 00005F47 03020201010202 <1>

140 <1>

141 <1> vga\_params:

142 <1> ; 25/07/2016

143 <1> ; 19/07/2016

144 <1> ; 03/07/2016

145 <1> ; derived from 'Plex86/Bochs VGABios' source code

146 <1> ; vgabios-0.7a (2011)

147 <1> ; by the LGPL VGABios Developers Team (2001-2008)

148 <1> ; 'vgatables.h'

149 <1> ; Oracle VirtualBox 5.0.24 VGABios Source Code

150 <1> ; ('vgabios.c', 'vgatables.h', 'vgafonts.h', 'vgarom.asm')

151 <1> ;

152 <1> vga\_mode\_03h: ; mode 03h, 80\*25 text, CGA colors

153 00005F4E 5018100010 <1> db 80, 24, 16, 00h, 10h ; tw, th-1, ch, slength (5)

154 00005F53 00030002 <1> db 00h, 03h, 00h, 02h ; sequ regs (4)

155 00005F57 67 <1> db 67h ; misc reg (1)

156 00005F58 5F4F50825581BF1F <1> db 5Fh, 4Fh, 50h, 82h, 55h, 81h, 0BFh, 1Fh

157 00005F60 004F <1> db 00h, 4Fh

158 <1> vga\_p\_cm\_pos equ $ - vga\_mode\_03h

159 00005F62 0D0E00000000 <1> db 0Dh, 0Eh, 00h, 00h, 00h, 00h

160 00005F68 9C8E8F281F96B9A3 <1> db 9Ch, 8Eh, 8Fh, 28h, 1Fh, 96h, 0B9h, 0A3h

161 00005F70 FF <1> db 0FFh ; crtc\_regs (25)

162 00005F71 0001020304051407 <1> db 00h, 01h, 02h, 03h, 04h, 05h, 14h, 07h

163 00005F79 38393A3B3C3D3E3F <1> db 38h, 39h, 3Ah, 3Bh, 3Ch, 3Dh, 3Eh, 3Fh

164 00005F81 0C000F08 <1> db 0Ch, 00h, 0Fh, 08h ; actl regs (20)

165 00005F85 0000000000100E0FFF <1> db 00h, 00h, 00h, 00h, 00h, 10h, 0Eh, 0Fh, 0FFh ; grdc regs (9)

166 <1> vga\_mode\_01h: ; mode 01h, 40\*25 text, CGA colors

167 00005F8E 2818100008 <1> db 40, 24, 16, 00h, 08h ; tw, th-1, ch, slength

168 00005F93 08030002 <1> db 08h, 03h, 00h, 02h ; sequ regs

169 00005F97 67 <1> db 67h ; misc reg

170 00005F98 2D2728902BA0BF1F <1> db 2Dh, 27h, 28h, 90h, 2Bh, 0A0h, 0BFh, 1Fh

171 00005FA0 004F0D0E00000000 <1> db 00h, 4Fh, 0Dh, 0Eh, 00h, 00h, 00h, 00h

172 00005FA8 9C8E8F141F96B9A3 <1> db 9Ch, 8Eh, 8Fh, 14h, 1Fh, 96h, 0B9h, 0A3h

173 00005FB0 FF <1> db 0FFh ; crtc\_regs

174 00005FB1 0001020304051407 <1> db 00h, 01h, 02h, 03h, 04h, 05h, 14h, 07h

175 00005FB9 38393A3B3C3D3E3F <1> db 38h, 39h, 3Ah, 3Bh, 3Ch, 3Dh, 3Eh, 3Fh

176 00005FC1 0C000F08 <1> db 0Ch, 00h, 0Fh, 08h ; actl regs

177 00005FC5 0000000000100E0FFF <1> db 00h, 00h, 00h, 00h, 00h, 10h, 0Eh, 0Fh, 0FFh ; grdc regs

178 <1> ;vga\_mode\_07h: ; mode 07h, 80\*25 text, mono color

179 <1> ; db 80, 24, 16, 00h, 10h ; tw, th-1, ch, slength

180 <1> ; db 00h, 03h, 00h, 02h ; sequ regs

181 <1> ; db 66h ; misc reg

182 <1> ; db 5Fh, 4Fh, 50h, 82h, 55h, 81h, 0BFh, 1Fh

183 <1> ; db 00h, 4Fh, 0Dh, 0Eh, 00h, 00h, 00h, 00h

184 <1> ; db 9Ch, 8Eh, 8Fh, 28h, 0Fh, 96h, 0B9h, 0A3h

185 <1> ; db 0FFh ; crtc regs

186 <1> ; db 00h, 08h, 08h, 08h, 08h, 08h, 08h, 08h

187 <1> ; db 10h, 18h, 18h, 18h, 18h, 18h, 18h, 18h

188 <1> ; db 0Eh, 00h, 0Fh, 08h ; actl regs

189 <1> ; db 00h, 00h, 00h, 00h, 00h, 10h, 0Ah, 0Fh, 0FFh ; grdc regs

190 <1> vga\_mode\_04h: ; 320\*200 graphics, 4 colors, CGA

191 00005FCE 2818080008 <1> db 40, 24, 8, 00h, 08h ; tw, th-1, ch, slength

192 00005FD3 09030002 <1> db 09h, 03h, 00h, 02h ; sequ regs

193 00005FD7 63 <1> db 63h ; misc reg

194 00005FD8 2D2728902B80BF1F <1> db 2Dh, 27h, 28h, 90h, 2Bh, 80h, 0BFh, 1Fh

195 00005FE0 00C1000000000000 <1> db 00h, 0C1h, 00h, 00h, 00h, 00h, 00h, 00h

196 00005FE8 9C8E8F140096B9A2 <1> db 9Ch, 8Eh, 8Fh, 14h, 00h, 96h, 0B9h, 0A2h

197 00005FF0 FF <1> db 0FFh ; crtc\_regs

198 00005FF1 0013151702040607 <1> db 00h, 13h, 15h, 17h, 02h, 04h, 06h, 07h

199 00005FF9 1011121314151617 <1> db 10h, 11h, 12h, 13h, 14h, 15h, 16h, 17h

200 00006001 01000300 <1> db 01h, 00h, 03h, 00h ; actl regs

201 00006005 0000000000300F0FFF <1> db 00h, 00h, 00h, 00h, 00h, 30h, 0Fh, 0Fh, 0FFh ; grdc regs

202 <1> vga\_mode\_06h: ; 640\*200 graphics, 2 colors, CGA

203 0000600E 5018080010 <1> db 80, 24, 8, 00h, 10h ; tw, th-1, ch, slength

204 00006013 01010006 <1> db 01h, 01h, 00h, 06h ; sequ regs

205 00006017 63 <1> db 63h ; misc reg

206 00006018 5F4F50825480BF1F <1> db 5Fh, 4Fh, 50h, 82h, 54h, 80h, 0BFh, 1Fh

207 00006020 00C1000000000000 <1> db 00h, 0C1h, 00h, 00h, 00h, 00h, 00h, 00h

208 00006028 9C8E8F280096B9C2 <1> db 9Ch, 8Eh, 8Fh, 28h, 00h, 96h, 0B9h, 0C2h

209 00006030 FF <1> db 0FFh ; crtc regs

210 00006031 0017171717171717 <1> db 00h, 17h, 17h, 17h, 17h, 17h, 17h, 17h

211 00006039 1717171717171717 <1> db 17h, 17h, 17h, 17h, 17h, 17h, 17h, 17h

212 00006041 01000100 <1> db 01h, 00h, 01, 00h ; actl regs

213 00006045 0000000000000D0FFF <1> db 00h, 00h, 00h, 00h, 00h, 00h, 0Dh, 0Fh, 0FFh ; grdc regs

214 <1> vga\_mode\_13h: ; mode 13h, 300\*200, 256 colors, linear

215 0000604E 2818080000 <1> db 40, 24, 8, 0, 0 ; tw, th-1, ch, slength (5)

216 00006053 010F000E <1> db 01h, 0Fh, 00h, 0Eh ; sequ regs (4)

217 00006057 63 <1> db 63h ; misc reg (1)

218 00006058 5F4F50825480BF1F <1> db 5Fh, 4Fh, 50h, 82h, 54h, 80h, 0BFh, 1Fh

219 00006060 0041000000000000 <1> db 00h, 41h, 00h, 00h, 00h, 00h, 00h, 00h

220 00006068 9C8E8F284096B9A3 <1> db 9Ch, 8Eh, 8Fh, 28h, 40h, 96h, 0B9h, 0A3h

221 00006070 FF <1> db 0FFh ; crtc regs (25)

222 00006071 0001020304050607 <1> db 00h, 01h, 02h, 03h, 04h, 05h, 06h, 07h

223 00006079 08090A0B0C0D0E0F <1> db 08h, 09h, 0Ah, 0Bh, 0Ch, 0Dh, 0Eh, 0Fh

224 00006081 41000F00 <1> db 41h, 00h, 0Fh, 00h ; actl regs (20)

225 00006085 000000000040050FFF <1> db 00h, 00h, 00h, 00h, 00h, 40h, 05h, 0Fh, 0FFh ; grdc regs (9)

226 <1> vga\_mode\_setl equ $ - vga\_mode\_13h ; = 64

227 <1> vga\_mode\_F0h: ; mode X ; 320\*240, 256 colors, planar

228 0000608E 2818080000 <1> db 40, 24, 8, 0, 0 ; tw, th-1, ch, slength

229 00006093 010F0006 <1> db 01h, 0Fh, 00h, 06h ; sequ regs

230 00006097 E3 <1> db 0E3h ; misc reg

231 00006098 5F4F508254800D3E <1> db 5Fh, 4Fh, 50h, 82h, 54h, 80h, 0Dh, 3Eh

232 000060A0 0041000000000000 <1> db 00h, 41h, 00h, 00h, 00h, 00h, 00h, 00h

233 000060A8 EAACDF2800E706E3 <1> db 0EAh, 0ACh, 0DFh, 28h, 00h, 0E7h, 06h, 0E3h

234 000060B0 FF <1> db 0FFh ; crtc regs (25)

235 000060B1 0001020304050607 <1> db 00h, 01h, 02h, 03h, 04h, 05h, 06h, 07h

236 000060B9 08090A0B0C0D0E0F <1> db 08h, 09h, 0Ah, 0Bh, 0Ch, 0Dh, 0Eh, 0Fh

237 000060C1 41000F00 <1> db 41h, 00h, 0Fh, 00h ; actl regs

238 000060C5 000000000040050FFF <1> db 00h, 00h, 00h, 00h, 00h, 40h, 05h, 0Fh, 0FFh ; grdc regs

239 <1> vga\_mode\_12h: ; mode 12h, 640\*480, 16 colors, planar

240 000060CE 501D100000 <1> db 80, 29, 16, 0, 0 ; tw, th-1, ch, slength

241 000060D3 010F0006 <1> db 01h, 0Fh, 00h, 06h ; sequ regs

242 000060D7 E3 <1> db 0E3h ; misc reg

243 000060D8 5F4F508254800B3E <1> db 5Fh, 4Fh, 50h, 82h, 54h, 80h, 0Bh, 3Eh

244 000060E0 0040000000000000 <1> db 00h, 40h, 00h, 00h, 00h, 00h, 00h, 00h

245 000060E8 EA8CDF2800E704E3 <1> db 0EAh, 8Ch, 0DFh, 28h, 00h, 0E7h, 04h, 0E3h

246 000060F0 FF <1> db 0FFh ; crtc regs

247 000060F1 0001020304051407 <1> db 00h, 01h, 02h, 03h, 04h, 05h, 14h, 07h

248 000060F9 38393A3B3C3D3E3F <1> db 38h, 39h, 3Ah, 3Bh, 3Ch, 3Dh, 3Eh, 3Fh

249 00006101 01000F00 <1> db 01h, 00h, 0Fh, 00h ; actl regs

250 00006105 000000000000050FFF <1> db 00h, 00h, 00h, 00h, 00h, 00h, 05h, 0Fh, 0FFh ; grdc regs

251 <1> vga\_mode\_6Ah: ; mode 6Ah, 800\*600, 16 colors, planar

252 0000610E 6424100000 <1> db 100, 36, 16, 0, 0 ; tw, th-1, ch, slength

253 00006113 010F0006 <1> db 01h, 0Fh, 00h, 06h ; sequ regs

254 00006117 E3 <1> db 0E3h ; misc reg

255 00006118 7F6363836B1B72F0 <1> db 7Fh, 63h, 63h, 83h, 6Bh, 1Bh, 72h, 0F0h

256 00006120 0060000000000000 <1> db 00h, 60h, 00h, 00h, 00h, 00h, 00h, 00h

257 00006128 598D5732005773E3 <1> db 59h, 8Dh, 57h, 32h, 00h, 57h, 73h, 0E3h

258 00006130 FF <1> db 0FFh ; crtc regs

259 00006131 0001020304051407 <1> db 00h, 01h, 02h, 03h, 04h, 05h, 14h, 07h

260 00006139 38393A3B3C3D3E3F <1> db 38h, 39h, 3Ah, 3Bh, 3Ch, 3Dh, 3Eh, 3Fh

261 00006141 01000F00 <1> db 01h, 00h, 0Fh, 00h ; actl regs

262 00006145 000000000000050FFF <1> db 00h, 00h, 00h, 00h, 00h, 00h, 05h, 0Fh, 0FFh ; grdc regs

263 <1> vga\_mode\_0Dh: ; mode 0Dh, 320\*200, 16 colors, planar

264 0000614E 2818080020 <1> db 40, 24, 8, 0, 20h ; tw, th-1, ch, slength

265 00006153 090F0006 <1> db 09h, 0Fh, 00h, 06h ; sequ regs

266 00006157 63 <1> db 63h ; misc reg

267 00006158 2D2728902B80BF1F <1> db 2Dh, 27h, 28h, 90h, 2Bh, 80h, 0BFh, 1Fh

268 00006160 00C0000000000000 <1> db 00h, 0C0h, 00h, 00h, 00h, 00h, 00h, 00h

269 00006168 9C8E8F140096B9E3 <1> db 9Ch, 8Eh, 8Fh, 14h, 00h, 96h, 0B9h, 0E3h

270 00006170 FF <1> db 0FFh ; crtc regs

271 00006171 0001020304050607 <1> db 00h, 01h, 02h, 03h, 04h, 05h, 06h, 07h

272 00006179 1011121314151617 <1> db 10h, 11h, 12h, 13h, 14h, 15h, 16h, 17h

273 00006181 01000F00 <1> db 01h, 00h, 0Fh, 00h ; actl regs

274 00006185 000000000000050FFF <1> db 00h, 00h, 00h, 00h, 00h, 00h, 05h, 0Fh, 0FFh ; grdc regs

275 <1> vga\_mode\_0Eh: ; mode 0Eh, 640\*200, 16 colors, planar

276 0000618E 5018080040 <1> db 80, 24, 8, 0, 40h ; tw, th-1, ch, slength

277 00006193 010F0006 <1> db 01h, 0Fh, 00h, 06h ; sequ regs

278 00006197 63 <1> db 63h ; misc reg

279 00006198 5F4F50825480BF1F <1> db 5Fh, 4Fh, 50h, 82h, 54h, 80h, 0BFh, 1Fh

280 000061A0 00C0000000000000 <1> db 00h, 0C0h, 00h, 00h, 00h, 00h, 00h, 00h

281 000061A8 9C8E8F280096B9E3 <1> db 9Ch, 8Eh, 8Fh, 28h, 00h, 96h, 0B9h, 0E3h

282 000061B0 FF <1> db 0FFh ; crtc regs

283 000061B1 0001020304050607 <1> db 00h, 01h, 02h, 03h, 04h, 05h, 06h, 07h

284 000061B9 1011121314151617 <1> db 10h, 11h, 12h, 13h, 14h, 15h, 16h, 17h

285 000061C1 01000F00 <1> db 01h, 00h, 0Fh, 00h ; actl regs

286 000061C5 000000000000050FFF <1> db 00h, 00h, 00h, 00h, 00h, 00h, 05h, 0Fh, 0FFh ; grdc regs

287 <1> vga\_mode\_10h: ; mode 10h, 640\*350, 16 colors, planar

288 000061CE 50180E0080 <1> db 80, 24, 14, 0, 80h ; tw, th-1, ch, slength

289 000061D3 010F0006 <1> db 01h, 0Fh, 00h, 06h ; sequ regs

290 000061D7 A3 <1> db 0A3h ; misc reg

291 000061D8 5F4F50825480BF1F <1> db 5Fh, 4Fh, 50h, 82h, 54h, 80h, 0BFh, 1Fh

292 000061E0 0040000000000000 <1> db 00h, 40h, 00h, 00h, 00h, 00h, 00h, 00h

293 000061E8 83855D280F63BAE3 <1> db 83h, 85h, 5Dh, 28h, 0Fh, 63h, 0BAh, 0E3h

294 000061F0 FF <1> db 0FFh ; crtc regs

295 000061F1 0001020304051407 <1> db 00h, 01h, 02h, 03h, 04h, 05h, 14h, 07h

296 000061F9 38393A3B3C3D3E3F <1> db 38h, 39h, 3Ah, 3Bh, 3Ch, 3Dh, 3Eh, 3Fh

297 00006201 01000F00 <1> db 01h, 00h, 0Fh, 00h ; actl regs

298 00006205 000000000000050FFF <1> db 00h, 00h, 00h, 00h, 00h, 00h, 05h, 0Fh, 0FFh ; grdc regs

299 <1> vga\_mode\_11h: ; mode 11h, 640\*480, mono color, planar

300 0000620E 501D100000 <1> db 80, 29, 16, 0, 0 ; tw, th-1, ch, slength

301 00006213 010F0006 <1> db 01h, 0Fh, 00h, 06h ; sequ regs

302 00006217 E3 <1> db 0E3h ; misc reg

303 00006218 5F4F508254800B3E <1> db 5Fh, 4Fh, 50h, 82h, 54h, 80h, 0Bh, 3Eh

304 00006220 0040000000000000 <1> db 00h, 40h, 00h, 00h, 00h, 00h, 00h, 00h

305 00006228 EA8CDF2800E704E3 <1> db 0EAh, 8Ch, 0DFh, 28h, 00h, 0E7h, 04h, 0E3h

306 00006230 FF <1> db 0FFh ; crtc regs

307 00006231 003F003F003F003F <1> db 00h, 3Fh, 00h, 3Fh, 00h, 3Fh, 00h, 3Fh

308 00006239 003F003F003F003F <1> db 00h, 3Fh, 00h, 3Fh, 00h, 3Fh, 00h, 3Fh

309 00006241 01000F00 <1> db 01h, 00h, 0Fh, 00h ; actl regs

310 00006245 000000000000050FFF <1> db 00h, 00h, 00h, 00h, 00h, 00h, 05h, 0Fh, 0FFh ; grdc regs

311 <1> end\_of\_vga\_params:

312 <1>

313 <1> ; /// End Of VIDEO DATA ///

2298 ;%include 'diskdata.s' ; DISK (BIOS) DATA (initialized)

2299 ;;;

2300

2301 Align 2

2302

2303 %include 'sysdefs.s' ; 24/01/2015

1 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2 <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.0) - SYSTEM DEFINITIONS : sysdefs.s

3 <1> ; ----------------------------------------------------------------------------

4 <1> ; Last Update: 31/12/2017

5 <1> ; ----------------------------------------------------------------------------

6 <1> ; Beginning: 24/01/2016

7 <1> ; ----------------------------------------------------------------------------

8 <1> ; Assembler: NASM version 2.11 (trdos386.s)

9 <1> ; ----------------------------------------------------------------------------

10 <1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan

11 <1> ; sysdefs.inc (14/11/2015)

12 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

13 <1>

14 <1> ; Retro UNIX 386 v1 Kernel - SYSDEFS.INC

15 <1> ; Last Modification: 14/11/2015

16 <1> ;

17 <1> ; ///////// RETRO UNIX 386 V1 SYSTEM DEFINITIONS ///////////////

18 <1> ; (Modified from

19 <1> ; Retro UNIX 8086 v1 system definitions in 'UNIX.ASM', 01/09/2014)

20 <1> ; ((UNIX.ASM (RETRO UNIX 8086 V1 Kernel), 11/03/2013 - 01/09/2014))

21 <1> ; UNIX.ASM (MASM 6.11) --> SYSDEFS.INC (NASM 2.11)

22 <1> ; ----------------------------------------------------------------------------

23 <1> ;

24 <1> ; Derived from UNIX Operating System (v1.0 for PDP-11)

25 <1> ; (Original) Source Code by Ken Thompson (1971-1972)

26 <1> ; <Bell Laboratories (17/3/1972)>

27 <1> ; <Preliminary Release of UNIX Implementation Document>

28 <1> ;

29 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

30 <1>

31 <1> nproc equ 16 ; number of processes

32 <1> nfiles equ 50

33 <1> ntty equ 8 ; 8+1 -> 8 (10/05/2013)

34 <1> nbuf equ 4 ; 6 ;; 21/08/2015 - 'namei' buffer problem when nbuf > 4

35 <1> ; NOTE: If fd0 super block buffer addres is beyond of the 1st

36 <1> ; 32K, DMA r/w routine or someting else causes a jump to

37 <1> ; kernel panic routine (in 'alloc' routine, in u5.s)

38 <1> ; because of invalid buffer content (r/w error).

39 <1> ; When all buffers are set before the end of the 1st 32k,

40 <1> ; there is no problem!? (14/11/2015)

41 <1>

42 <1> ;csgmnt equ 2000h ; 26/05/2013 (segment of process 1)

43 <1> ;core equ 0 ; 19/04/2013

44 <1> ;ecore equ 32768 - 64 ; 04/06/2013 (24/05/2013)

45 <1> ; (if total size of argument list and arguments is 128 bytes)

46 <1> ; maximum executable file size = 32768-(64+40+128-6) = 32530 bytes

47 <1> ; maximum stack size = 40 bytes (+6 bytes for 'IRET' at 32570)

48 <1> ; initial value of user's stack pointer = 32768-64-128-2 = 32574

49 <1> ; (sp=32768-args\_space-2 at the beginning of execution)

50 <1> ; argument list offset = 32768-64-128 = 32576 (if it is 128 bytes)

51 <1> ; 'u' structure offset (for the '/core' dump file) = 32704

52 <1> ; '/core' dump file size = 32768 bytes

53 <1>

54 <1> ; 08/03/2014

55 <1> ;sdsegmnt equ 6C0h ; 256\*16 bytes (swap data segment size for 16 processes)

56 <1> ; 19/04/2013 Retro UNIX 8086 v1 feaure only !

57 <1> ;;sdsegmnt equ 740h ; swap data segment (for user structures and registers)

58 <1>

59 <1> ; 30/08/2013

60 <1> time\_count equ 4 ; 10 --> 4 01/02/2014

61 <1>

62 <1> ; 05/02/2014

63 <1> ; process status

64 <1> ;SFREE equ 0

65 <1> ;SRUN equ 1

66 <1> ;SWAIT equ 2

67 <1> ;SZOMB equ 3

68 <1> ;SSLEEP equ 4 ; Retro UNIX 8086 V1 extension (for sleep and wakeup)

69 <1>

70 <1> ; 09/03/2015

71 <1> userdata equ 80000h ; user structure data address for current user ; temporary

72 <1> swap\_queue equ 90000h - 2000h ; swap queue address ; temporary

73 <1> swap\_alloc\_table equ 0D0000h ; swap allocation table address ; temporary

74 <1>

75 <1> ; 17/09/2015

76 <1> ESPACE equ 48 ; [u.usp] (at 'sysent') - [u.sp] value for error return

77 <1>

78 <1> ; 31/12/2017

79 <1> ; 19/02/2017

80 <1> ; 15/10/2016

81 <1> ; 20/05/2016

82 <1> ; 19/05/2016

83 <1> ; 18/05/2016

84 <1> ; 29/04/2016

85 <1> ; TRDOS 386 (TRDOS v2.0) system calls - temporary List

86 <1> ; 14/07/2013 - 21/09/2015 (Retro UNIX 8086 & 386 system calls)

87 <1> \_ver equ 0 ; Get TRDOS version (v2.0)

88 <1> \_exit equ 1

89 <1> \_fork equ 2

90 <1> \_read equ 3

91 <1> \_write equ 4

92 <1> \_open equ 5

93 <1> \_close equ 6

94 <1> \_wait equ 7

95 <1> \_creat equ 8

96 <1> \_rename equ 9 ; TRDOS 386, Rename File (31/12/2017)

97 <1> \_delete equ 10 ; TRDOS 386, Delete File (29/12/2017)

98 <1> \_exec equ 11

99 <1> \_chdir equ 12

100 <1> \_time equ 13 ; TRDOS 386, Get Sys Date&Time (30/12/2017)

101 <1> \_mkdir equ 14

102 <1> \_chmod equ 15 ; TRDOS 386, Change Attributes (30/12/2017)

103 <1> \_rmdir equ 16 ; TRDOS 386, Remove Directory (29/12/2017)

104 <1> \_break equ 17

105 <1> \_drive equ 18 ; TRDOS 386, Get/Set Current Drv (30/12/2017)

106 <1> \_seek equ 19

107 <1> \_tell equ 20

108 <1> \_mem equ 21 ; TRDOS 386, Get Total&Free Mem (31/12/2017)

109 <1> \_prompt equ 22 ; TRDOS 386, Change Cmd Prompt (31/12/2017)

110 <1> \_path equ 23 ; TRDOS 386, Get/Set Run Path (31/12/2017)

111 <1> \_env equ 24 ; TRDOS 386, Get/Set Env Vars (31/12/2017)

112 <1> \_stime equ 25 ; TRDOS 386, Set Sys Date&Time (30/12/2017)

113 <1> \_quit equ 26

114 <1> \_intr equ 27

115 <1> \_dir equ 28 ; TRDOS 386, Get Curr Drive&Dir (30/12/2017)

116 <1> \_emt equ 29

117 <1> \_ldrvt equ 30 ; TRDOS 386, Get Logical DOS DDT (30/12/2017)

118 <1> \_video equ 31 ; TRDOS 386 Video Functions (16/05/2016)

119 <1> \_audio equ 32 ; TRDOS 386 Video Functions (16/05/2016)

120 <1> \_timer equ 33 ; TRDOS 386 Timer Functions (18/05/2016)

121 <1> \_sleep equ 34 ; Retro UNIX 8086 v1 feature only !

122 <1> \_msg equ 35 ; Retro UNIX 386 v1 feature only !

123 <1> \_geterr equ 36 ; Retro UNIX 386 v1 feature only !

124 <1> \_fpsave equ 37 ; TRDOS 386 FPU state option (28/02/2017)

125 <1> \_pri equ 38 ; change priority - TRDOS 386 (20/05/2016)

126 <1> \_rele equ 39 ; TRDOS 386 (19/05/2016)

127 <1> \_fff equ 40 ; Find First File - TRDOS 386 (15/10/2016)

128 <1> \_fnf equ 41 ; Find Next File - TRDOS 386 (15/10/2016)

129 <1> \_alloc equ 42 ; Allocate memory - TRDOS 386 (19/02/2017)

130 <1> ; TRDOS 386 (19/02/2017) DMA buff fuctions

131 <1> \_dalloc equ 43 ; Deallocate mem - TRDOS 386 (19/02/2017)

132 <1> \_calbac equ 44 ; Set IRQ callback - TRDOS 386 (20/02/2017)

133 <1> \_dma equ 45 ; DMA service - TRDOS 386 (20/08/2017)

134 <1>

135 <1> %macro sys 1-4

136 <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)

137 <1> ; 03/09/2015

138 <1> ; 13/04/2015

139 <1> ; Retro UNIX 386 v1 system call.

140 <1> %if %0 >= 2

141 <1> mov ebx, %2

142 <1> %if %0 >= 3

143 <1> mov ecx, %3

144 <1> %if %0 = 4

145 <1> mov edx, %4

146 <1> %endif

147 <1> %endif

148 <1> %endif

149 <1> mov eax, %1

150 <1> ;int 30h

151 <1> int 40h ; TRDOS 386 (TRDOS v2.0)

152 <1> %endmacro

153 <1>

154 <1> ; TRDOS 386 system calls, interrupt number

155 <1> ; 25/12/2016

156 <1> SYSCALL\_INT\_NUM equ '40' ; '40h'

157 <1>

158 <1> ; 13/05/2015 - ERROR CODES

159 <1> ERR\_FILE\_NOT\_OPEN equ 10 ; 'file not open !' error

160 <1> ERR\_FILE\_ACCESS equ 11 ; 'permission denied !' error

161 <1> ; 14/05/2015

162 <1> ERR\_DIR\_ACCESS equ 11 ; 'permission denied !' error

163 <1> ERR\_FILE\_NOT\_FOUND equ 12 ; 'file not found !' error

164 <1> ERR\_TOO\_MANY\_FILES equ 13 ; 'too many open files !' error

165 <1> ERR\_DIR\_EXISTS equ 14 ; 'directory already exists !' error

166 <1> ; 16/05/2015

167 <1> ERR\_DRV\_NOT\_RDY equ 15 ; 'drive not ready !' error

168 <1> ; 18/05/2015

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> ; 07/10/2016

255 <1> ;NUMOFDEVICES equ 20 ; max. num of available devices (sys)

256 <1>

2304 %include 'trdosk0.s' ; 04/01/2016

1 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2 <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.0) - DEFINITIONS : trdosk0.s

3 <1> ; ----------------------------------------------------------------------------

4 <1> ; Last Update: 29/02/2016

5 <1> ; ----------------------------------------------------------------------------

6 <1> ; Beginning: 04/01/2016

7 <1> ; ----------------------------------------------------------------------------

8 <1> ; Assembler: NASM version 2.11 (trdos386.s)

9 <1> ; ----------------------------------------------------------------------------

10 <1> ; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan

11 <1> ; TRDOS2.ASM (09/11/2011)

12 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

13 <1> ; TRDOS2.ASM (c) 2004-2011 Erdogan TAN [ 17/01/2004 ] Last Update: 09/11/2011

14 <1> ;

15 <1> ; Masterboot / Partition Table at Beginning+1BEh

16 <1> ptBootable equ 0

17 <1> ptBeginHead equ 1

18 <1> ptBeginSector equ 2

19 <1> ptBeginCylinder equ 3

20 <1> ptFileSystemID equ 4

21 <1> ptEndHead equ 5

22 <1> ptEndSector equ 6

23 <1> ptEndCylinder equ 7

24 <1> ptStartSector equ 8

25 <1> ptSectors equ 12

26 <1>

27 <1> ; Boot Sector Parameters at 7C00h

28 <1> DataArea1 equ -4

29 <1> DataArea2 equ -2

30 <1> BootStart equ 0h

31 <1> OemName equ 03h

32 <1> BytesPerSec equ 0Bh

33 <1> SecPerClust equ 0Dh

34 <1> ResSectors equ 0Eh

35 <1> FATs equ 10h

36 <1> RootDirEnts equ 11h

37 <1> Sectors equ 13h

38 <1> Media equ 15h

39 <1> FATSecs equ 16h

40 <1> SecPerTrack equ 18h

41 <1> Heads equ 1Ah

42 <1> Hidden1 equ 1Ch

43 <1> Hidden2 equ 1Eh

44 <1> HugeSec1 equ 20h

45 <1> HugeSec2 equ 22h

46 <1> DriveNumber equ 24h

47 <1> Reserved1 equ 25h

48 <1> bootsignature equ 26h

49 <1> VolumeID equ 27h

50 <1> VolumeLabel equ 2Bh

51 <1> FileSysType equ 36h

52 <1> Reserved2 equ 3Eh ; Starting cluster of P2000

53 <1>

54 <1> ; FAT32 BPB Structure

55 <1> FAT32\_FAT\_Size equ 36

56 <1> FAT32\_RootFClust equ 44

57 <1> FAT32\_FSInfoSec equ 48

58 <1> FAT32\_DrvNum equ 64

59 <1> FAT32\_BootSig equ 66

60 <1> FAT32\_VolID equ 67

61 <1> FAT32\_VolLab equ 71

62 <1> FAT32\_FilSysType equ 82

63 <1>

64 <1> ; BIOS Disk Parameters

65 <1> DPDiskNumber equ 0h

66 <1> DPDType equ 1h

67 <1> DPReturn equ 2h

68 <1> DPHeads equ 3h

69 <1> DPCylinders equ 4h

70 <1> DPSecPerTrack equ 6h

71 <1> 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 0FFFh

179 <1> FAT16EOC equ 0FFFFh

180 <1> ;FAT32EOC equ 0FFFFFFFh ; It is not direct usable for 8086 code

181 <1> ; BAD Cluster

182 <1> FAT12BADC equ 0FF7h

183 <1> FAT16BADC equ 0FFF7h

184 <1> ;FAT32BADC equ 0FFFFFF7h ; It is not direct usable for 8086 code

185 <1> ; MS-DOS FAT16 FS (Maximum Possible) Last Cluster Number= 0FFF6h

186 <1>

187 <1> ; TRFS

188 <1>

189 <1> bs\_FS\_JmpBoot equ 0 ; jmp short bsBootCode

190 <1> ; db 0EBh, db 3Fh, db 90h

191 <1> bs\_FS\_Identifier equ 3 ; db 'FS', db 0

192 <1> bs\_FS\_BytesPerSec equ 6 ; dw 512

193 <1> bs\_FS\_MediaAttrib equ 8 ; db 3

194 <1> bs\_FS\_PartitionID equ 9 ; db 0A1h

195 <1> bs\_FS\_VersionMaj equ 10 ; db 01h

196 <1> bs\_FS\_VersionMin equ 11 ; db 0

197 <1> bs\_FS\_BeginSector equ 12 ; dd 0

198 <1> bs\_FS\_VolumeSize equ 16 ; dd 2880

199 <1> bs\_FS\_StartupFD equ 20 ; dd 0

200 <1> bs\_FS\_MATLocation equ 24 ; dd 1

201 <1> bs\_FS\_RootDirD equ 28 ; dd 8

202 <1> bs\_FS\_SystemConfFD equ 32 ; dd 0

203 <1> bs\_FS\_SwapFD equ 36 ; dd 0

204 <1> bs\_FS\_UnDelDirD equ 40 ; dd 0

205 <1> bs\_FS\_DriveNumber equ 44 ; db 0

206 <1> bs\_FS\_LBA\_Ready equ 45 ; db 0

207 <1> bs\_FS\_MagicWord equ 46

208 <1> bs\_FS\_SecPerTrack equ 46 ; db 0A1h

209 <1> bs\_FS\_Heads equ 47 ; db 01h

210 <1> bs\_FS\_OperationSys equ 48 ; db "TR-SINGLIX v1.0b"

211 <1> bs\_FS\_Terminator equ 64 ; db 0

212 <1> bs\_FS\_BootCode equ 65

213 <1>

214 <1> FS\_MAT\_DATLocation equ 12

215 <1> FS\_MAT\_DATScount equ 16

216 <1> FS\_MAT\_FreeSectors equ 20

217 <1> FS\_MAT\_FirstFreeSector equ 24

218 <1> FS\_RDT\_VolumeSerialNo equ 28

219 <1> FS\_RDT\_VolumeName equ 64

220 <1>

221 <1> ; FAT12 + FAT16 + FAT32

222 <1> BS\_JmpBoot equ 0

223 <1> BS\_OEMName equ 3

224 <1> BPB\_BytsPerSec equ 11

225 <1> BPB\_SecPerClust equ 13

226 <1> BPB\_RsvdSecCnt equ 14

227 <1> BPB\_NumFATs equ 16

228 <1> BPB\_RootEntCnt equ 17

229 <1> BPB\_TotalSec16 equ 19

230 <1> BPB\_Media equ 21

231 <1> BPB\_FATSz16 equ 22

232 <1> BPB\_SecPerTrk equ 24

233 <1> BPB\_NumHeads equ 26

234 <1> BPB\_HiddSec equ 28

235 <1> BPB\_TotalSec32 equ 32

236 <1>

237 <1> ; FAT12 and FAT16 only

238 <1> BS\_DrvNum equ 36

239 <1> BS\_Reserved1 equ 37

240 <1> BS\_BootSig equ 38

241 <1> BS\_VolID equ 39

242 <1> BS\_VolLab equ 43

243 <1> BS\_FilSysType equ 54 ; 8 bytes

244 <1> BS\_BootCode equ 62

245 <1>

246 <1> ; FAT32 only

247 <1> BPB\_FATSz32 equ 36 ; FAT32, 4 bytes

248 <1> BPB\_ExtFlags equ 40 ; FAT32, 2 bytes

249 <1> BPB\_FSVer equ 42 ; FAT32, 2 bytes

250 <1> BPB\_RootClus equ 44 ; FAT32, 4 bytes

251 <1> BPB\_FSInfo equ 48 ; FAT 32, 2 bytes

252 <1> BPB\_BkBootSec equ 50 ; FAT32, 2 bytes

253 <1> BPB\_Reserved equ 52 ; FAT32, 12 bytes

254 <1> BS\_FAT32\_DrvNum equ 64 ; FAT32, 1 byte

255 <1> BS\_FAT32\_Reserved1 equ 65 ; FAT32, 1 byte

256 <1> BS\_FAT32\_BootSig equ 66 ; FAT32, 1 byte

257 <1> BS\_FAT32\_VolID equ 67 ; FAT32, 4 bytes

258 <1> BS\_FAT32\_VolLab equ 71 ; FAT32, 11 bytes

259 <1> BS\_FAT32\_FilSysType equ 82 ; FAT32, 8 bytes

260 <1> BS\_FAT32\_BootCode equ 90

261 <1>

262 <1> ; 29/02/2016

263 <1> ;(FAT32 Free Cluster Count & First Free Cluster values)

264 <1> ;[BPB\_Reserved] = Free Cluster Count (offset 52)

265 <1> ;[BPB\_Reserved+4] = First Free Cluster (offset 56)

266 <1>

267 <1> BS\_Validation equ 510

268 <1>

269 <1> ; 15/02/2016

270 <1> ; FILE.ASM - 09/10/2011

271 <1> ; Directory Entry Structure

272 <1> ; 29/10/2009 (According to Microsoft FAT32 File System Specification)

273 <1> DirEntry\_Name equ 0

274 <1> DirEntry\_Attr equ 11

275 <1> DirEntry\_NTRes equ 12

276 <1> DirEntry\_CrtTimeTenth equ 13

277 <1> DirEntry\_CrtTime equ 14

278 <1> DirEntry\_CrtDate equ 16

279 <1> DirEntry\_LastAccDate equ 18

280 <1> DirEntry\_FstClusHI equ 20

281 <1> DirEntry\_WrtTime equ 22

282 <1> DirEntry\_WrtDate equ 24

283 <1> DirEntry\_FstClusLO equ 26

284 <1> DirEntry\_FileSize equ 28

2305 %include 'trdosk1.s' ; 04/01/2016

1 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2 <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.0) - SYS INIT : trdosk1.s

3 <1> ; ----------------------------------------------------------------------------

4 <1> ; Last Update: 31/12/2017

5 <1> ; ----------------------------------------------------------------------------

6 <1> ; Beginning: 04/01/2016

7 <1> ; ----------------------------------------------------------------------------

8 <1> ; Assembler: NASM version 2.11 (trdos386.s)

9 <1> ; ----------------------------------------------------------------------------

10 <1> ; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan

11 <1> ; TRDOS2.ASM (09/11/2011)

12 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

13 <1> ; TRDOS2.ASM (c) 2004-2011 Erdogan TAN [ 17/01/2004 ] Last Update: 09/11/2011

14 <1> ;

15 <1>

16 <1> sys\_init:

17 <1> ; 23/01/2017

18 <1> ; 07/05/2016

19 <1> ; 02/05/2016

20 <1> ; 24/04/2016

21 <1> ; 14/04/2016

22 <1> ; 13/04/2016

23 <1> ; 30/03/2016

24 <1> ; 24/01/2016

25 <1> ; 06/01/2016

26 <1> ; 04/01/2016

27 <1>

28 <1> ; 23/01/2017 - reset timer frequency (to 18.2Hz)

29 0000624E B036 <1> mov al, 00110110b ; 36h

30 00006250 E643 <1> out 43h, al

31 00006252 31C0 <1> xor eax, eax ; sub al, al ; 0

32 00006254 E640 <1> out 40h, al ; LB

33 00006256 E640 <1> out 40h, al ; HB

34 <1> ;

35 <1> ; 30/03/2016

36 <1> ; Clear Logical DOS Disk Description Tables Area

37 <1> ;xor eax, eax

38 00006258 BF00010900 <1> mov edi, Logical\_DOSDisks

39 0000625D B980060000 <1> mov ecx, 6656/4 ; 26\*256 = 6656 bytes

40 00006262 F3AB <1> rep stosd ; 1664 times 4 bytes

41 <1>

42 00006264 B83F3A2F00 <1> mov eax, '?:/'

43 00006269 A3[FF580100] <1> mov [Current\_Dir\_Drv], eax

44 <1>

45 <1> ; Logical DRV INIT (only for hard disks)

46 0000626E E803040000 <1> call ldrv\_init ; trdosk2.s

47 <1>

48 <1> ; When floppy\_drv\_init call is disabled

49 <1> ; media changed sign is needed

50 <1> ; for proper drive initialization

51 <1>

52 00006273 BE00010900 <1> mov esi, Logical\_DOSDisks

53 00006278 B001 <1> mov al, 1 ; Initialization sign (invalid\_fd\_parameter)

54 0000627A 83C67E <1> add esi, LD\_MediaChanged ; Media Change Status = 1 (init needed)

55 0000627D 8806 <1> mov [esi], al ; A:

56 0000627F 81C600010000 <1> add esi, 100h

57 00006285 8806 <1> mov [esi], al ; B:

58 <1>

59 <1> \_current\_drive\_bootdisk:

60 00006287 8A15[F25C0000] <1> mov dl, [boot\_drv] ; physical drive number

61 0000628D 80FAFF <1> cmp dl, 0FFh

62 00006290 740A <1> je short \_last\_dos\_diskno\_check

63 <1> \_boot\_drive\_check:

64 00006292 80FA80 <1> cmp dl, 80h

65 00006295 7218 <1> jb short \_current\_drive\_a

66 00006297 80EA7E <1> sub dl, 7Eh ; C = 2 , D = 3

67 0000629A EB13 <1> jmp short \_current\_drive\_a

68 <1>

69 <1> \_last\_dos\_diskno\_check:

70 0000629C 8A15[D20C0100] <1> mov dl, [Last\_DOS\_DiskNo]

71 000062A2 80FA02 <1> cmp dl, 2

72 000062A5 7706 <1> ja short \_current\_drive\_c

73 000062A7 7406 <1> je short \_current\_drive\_a

74 000062A9 30D2 <1> xor dl, dl ; A:

75 000062AB EB02 <1> jmp short \_current\_drive\_a

76 <1>

77 <1> \_current\_drive\_c:

78 000062AD B202 <1> mov dl, 2 ; C:

79 <1>

80 <1> \_current\_drive\_a:

81 000062AF 8815[F35C0000] <1> mov [drv], dl

82 000062B5 BE[D40C0100] <1> mov esi, msg\_CRLF\_temp

83 000062BA E89E000000 <1> call print\_msg

84 <1>

85 000062BF 8A15[F35C0000] <1> mov dl, [drv]

86 000062C5 E8F60B0000 <1> call change\_current\_drive

87 000062CA 730C <1> jnc short \_start\_mainprog

88 <1>

89 <1> \_drv\_not\_ready\_error:

90 000062CC BE[8F0F0100] <1> mov esi, msgl\_drv\_not\_ready

91 000062D1 E887000000 <1> call print\_msg

92 000062D6 EB63 <1> jmp \_end\_of\_mainprog

93 <1>

94 <1> \_start\_mainprog:

95 <1> ; 07/01/2017

96 <1> ; 07/05/2016

97 <1> ; 02/05/2016

98 <1> ; 24/04/2016

99 <1> ; Retro UNIX 386 v1, 'sys\_init' (u0.s)

100 <1> ; 23/06/2015

101 <1>

102 <1> ; 02/05/2016

103 <1> ; 24/04/2016

104 000062D8 66B80100 <1> mov ax, 1

105 000062DC A2[B3030300] <1> mov [u.uno], al

106 000062E1 66A3[4E030300] <1> mov [mpid], ax

107 000062E7 66A3[20000300] <1> mov [p.pid], ax

108 000062ED A2[B0000300] <1> mov [p.stat], al

109 000062F2 C605[A8030300]04 <1> mov byte [u.quant], time\_count ; 07/01/2017

110 <1> ;

111 000062F9 A1[38580100] <1> mov eax, [k\_page\_dir]

112 000062FE A3[B8030300] <1> mov [u.pgdir], eax ; reset

113 <1> ;

114 00006303 E872E8FFFF <1> call allocate\_page

115 00006308 0F82A3000000 <1> jc panic

116 0000630E A3[B4030300] <1> mov [u.upage], eax ; user structure page

117 00006313 A3[C0000300] <1> mov [p.upage], eax

118 00006318 E8D7E8FFFF <1> call clear\_page

119 <1> ;

120 <1> ; 24/08/2015

121 0000631D FE0D[5B030300] <1> dec byte [sysflg] ; FFh = ready for system call

122 <1> ; 0 = executing a system call

123 <1> ; 13/04/2016

124 <1> ; Clear Environment Variables Page/Area

125 00006323 BF00300900 <1> mov edi, Env\_Page ; 93000h

126 00006328 B980000000 <1> mov ecx, Env\_Page\_Size / 4 ; 512/4 (4096/4)

127 0000632D 31C0 <1> xor eax, eax

128 0000632F F3AB <1> rep stosd

129 <1>

130 <1> ; 14/04/2016

131 00006331 E8E1340000 <1> call mainprog\_startup\_configuration

132 <1>

133 00006336 E8C60C0000 <1> call dos\_prompt

134 <1>

135 <1> \_end\_of\_mainprog:

136 0000633B BE[D40C0100] <1> mov esi, msg\_CRLF\_temp

137 00006340 E818000000 <1> call print\_msg

138 00006345 BE[DA0C0100] <1> mov esi, mainprog\_Version

139 0000634A E80E000000 <1> call print\_msg

140 <1> ; 24/01/2016

141 0000634F 28E4 <1> sub ah, ah

142 00006351 E8C0A8FFFF <1> call int16h ; call getch

143 00006356 E9A0ADFFFF <1> jmp cpu\_reset

144 <1>

145 0000635B EBFE <1> infinitiveloop: jmp short infinitiveloop

146 <1>

147 <1> print\_msg:

148 <1> ; 13/05/2016

149 <1> ; 04/01/2016

150 <1> ; 01/07/2015

151 <1> ; 13/03/2015 (Retro UNIX 386 v1)

152 <1> ; 07/03/2014 (Retro UNIX 8086 v1)

153 <1> ; (Modified registers: EAX, EBX, ECX, EDX, ESI, EDI)

154 <1> ;

155 0000635D 8A3D[66580100] <1> mov bh, [ACTIVE\_PAGE] ; 04/01/2016 (ptty)

156 <1> ;mov bl, 07h ; Black background, light gray forecolor

157 <1>

158 00006363 AC <1> lodsb

159 <1> pmsg1:

160 00006364 56 <1> push esi

161 <1> ;mov bh, [ACTIVE\_PAGE] ; 04/01/2016 (ptty)

162 00006365 B307 <1> mov bl, 07h ; Black background, light gray forecolor

163 00006367 E846B9FFFF <1> call \_write\_tty

164 0000636C 5E <1> pop esi

165 0000636D AC <1> lodsb

166 0000636E 20C0 <1> and al, al

167 00006370 75F2 <1> jnz short pmsg1

168 00006372 C3 <1> retn

169 <1>

170 <1> clear\_screen:

171 <1> ; 13/05/2016

172 <1> ; 30/01/2016

173 <1> ; 24/01/2016

174 <1> ; 04/01/2016

175 00006373 0FB61D[66580100] <1> movzx ebx, byte [ACTIVE\_PAGE] ; video page number (0 to 7)

176 0000637A 8AA3[D35E0000] <1> mov ah, [ebx+vmode] ; default = 03h (80x25 text)

177 00006380 80FC04 <1> cmp ah, 4

178 00006383 7205 <1> jb short cls1

179 00006385 80FC07 <1> cmp ah, 7

180 00006388 7526 <1> jne short vga\_clear

181 <1> cls1:

182 <1> ;mov bh, bl

183 <1> ;mov bl, 7

184 0000638A 3A25[C25E0000] <1> cmp ah, [CRT\_MODE] ; current video mode ?

185 <1> ;je short cls2 ; yes (current video mode = 3)

186 <1> ;;call set\_mode\_3 ; set video mode to 3 (& clear screen)

187 <1> ;;retn

188 <1> ;jmp set\_mode\_3

189 00006390 0F8526B9FFFF <1> jne set\_mode\_3

190 <1> cls2:

191 00006396 88DF <1> mov bh, bl ; video page (0 to 7)

192 00006398 B307 <1> mov bl, 07h ; attribute to be used on blanked line

193 0000639A 28C0 <1> sub al, al ; 0 = entire window

194 0000639C 6631C9 <1> xor cx, cx

195 0000639F 66BA4F18 <1> mov dx, 184Fh

196 000063A3 E862B6FFFF <1> call \_scroll\_up ; 24/01/2016

197 <1> ;

198 <1> ;mov bh, [ACTIVE\_PAGE] ; video page number (0 to 7)

199 000063A8 6631D2 <1> xor dx, dx

200 000063AB E898B9FFFF <1> call \_set\_cpos ; 24/01/2016

201 <1> ;retn

202 <1> vga\_clear:

203 000063B0 C3 <1> retn

204 <1>

205 <1> panic:

206 <1> ; 13/05/2016 (TRDOS 386 = TRDOS v2)

207 <1> ; 13/03/2015 (Retro UNIX 386 v1)

208 <1> ; 07/03/2014 (Retro UNIX 8086 v1)

209 000063B1 BE[72190100] <1> mov esi, panic\_msg

210 000063B6 E8A2FFFFFF <1> call print\_msg

211 <1> key\_to\_reboot:

212 <1> ; 24/01/2016

213 000063BB 28E4 <1> sub ah, ah

214 000063BD E854A8FFFF <1> call int16h ; call getch

215 <1> ; wait for a character from the current tty

216 <1> ;

217 000063C2 B00A <1> mov al, 0Ah

218 000063C4 8A3D[66580100] <1> mov bh, [ptty] ; [ACTIVE\_PAGE]

219 000063CA B307 <1> mov bl, 07h ; Black background,

220 <1> ; light gray forecolor

221 000063CC E8E1B8FFFF <1> call \_write\_tty

222 000063D1 E925ADFFFF <1> jmp cpu\_reset

223 <1>

224 <1> ctrlbrk:

225 <1> ; 12/11/2015

226 <1> ; 13/03/2015 (Retro UNIX 386 v1)

227 <1> ; 06/12/2013 (Retro UNIX 8086 v1)

228 <1> ;

229 <1> ; INT 1Bh (control+break) handler

230 <1> ;

231 <1> ; Retro Unix 8086 v1 feature only!

232 <1> ;

233 000063D6 66833D[AA030300]00 <1> cmp word [u.intr], 0

234 000063DE 7645 <1> jna short cbrk4

235 <1> cbrk0:

236 <1> ; 12/11/2015

237 <1> ; 06/12/2013

238 000063E0 66833D[AC030300]00 <1> cmp word [u.quit], 0

239 000063E8 743B <1> jz short cbrk4

240 <1> ;

241 <1> ; 20/09/2013

242 000063EA 6650 <1> push ax

243 000063EC A0[66580100] <1> mov al, [ptty]

244 <1> ;

245 <1> ; 12/11/2015

246 <1> ;

247 <1> ; ctrl+break (EOT, CTRL+D) from serial port

248 <1> ; or ctrl+break from console (pseudo) tty

249 <1> ; (!redirection!)

250 <1> ;

251 000063F1 3C08 <1> cmp al, 8 ; serial port tty nums > 7

252 000063F3 7211 <1> jb short cbrk1 ; console (pseudo) tty

253 <1> ;

254 <1> ; Serial port interrupt handler sets [ptty]

255 <1> ; to the port's tty number (as temporary).

256 <1> ;

257 <1> ; If active process is using a stdin or

258 <1> ; stdout redirection (by the shell),

259 <1> ; console tty keyboard must be available

260 <1> ; to terminate running process,

261 <1> ; in order to prevent a deadlock.

262 <1> ;

263 000063F5 52 <1> push edx

264 000063F6 0FB615[B3030300] <1> movzx edx, byte [u.uno]

265 000063FD 3A82[7F000300] <1> cmp al, [edx+p.ttyc-1] ; console tty (rw)

266 00006403 5A <1> pop edx

267 00006404 7412 <1> je short cbrk2

268 <1> cbrk1:

269 00006406 FEC0 <1> inc al ; [u.ttyp] : 1 based tty number

270 <1> ; 06/12/2013

271 00006408 3A05[94030300] <1> cmp al, [u.ttyp] ; recent open tty (r)

272 0000640E 7408 <1> je short cbrk2

273 00006410 3A05[95030300] <1> cmp al, [u.ttyp+1] ; recent open tty (w)

274 00006416 750B <1> jne short cbrk3

275 <1> cbrk2:

276 <1> ;; 06/12/2013

277 <1> ;mov ax, [u.quit]

278 <1> ;and ax, ax

279 <1> ;jz short cbrk3

280 <1> ;

281 00006418 6631C0 <1> xor ax, ax ; 0

282 0000641B 6648 <1> dec ax

283 <1> ; 0FFFFh = 'ctrl+brk' keystroke

284 0000641D 66A3[AC030300] <1> mov [u.quit], ax

285 <1> cbrk3:

286 00006423 6658 <1> pop ax

287 <1> cbrk4:

288 00006425 C3 <1> retn

289 <1>

290 <1>

291 <1> ; 31/12/2017

292 <1> ; TRDOS 386 - 30/12/2017

293 <1> %define get\_rtc\_date RTC\_40

294 <1> %define get\_rtc\_time RTC\_20

295 <1> %define set\_rtc\_date RTC\_50

296 <1> %define set\_rtc\_time RTC\_30

297 <1> get\_rtc\_date\_time:

298 <1> ; Retro UNIX 8086 v1 - UNIX.ASM (01/09/2014)

299 <1> ;epoch:

300 <1> ; 30/12/2017 (TRDOS 386 = TRDOS v2.0)

301 <1> ; 15/03/2015 (Retro UNIX 386 v1 - 32 bit version)

302 <1> ; 09/04/2013 (Retro UNIX 8086 v1 - UNIX.ASM)

303 <1> ; 'epoch' procedure prototype:

304 <1> ; UNIXCOPY.ASM, 10/03/2013

305 <1> ; 14/11/2012

306 <1> ; unixboot.asm (boot file configuration)

307 <1> ; version of "epoch" procedure in "unixproc.asm"

308 <1> ; 21/7/2012

309 <1> ; 15/7/2012

310 <1> ; 14/7/2012

311 <1> ; Erdogan Tan - RETRO UNIX v0.1

312 <1> ; compute current date and time as UNIX Epoch/Time

313 <1> ; UNIX Epoch: seconds since 1/1/1970 00:00:00

314 <1> ;

315 <1> ; ((Modified registers: EAX, EDX, ECX, EBX))

316 <1> ;

317 <1>

318 00006426 E8C3F5FFFF <1> call get\_rtc\_time ; Return Current Time

319 0000642B 86E9 <1> xchg ch,cl

320 0000642D 66890D[30550100] <1> mov [hour], cx

321 00006434 86F2 <1> xchg dh,dl

322 00006436 668915[34550100] <1> mov [second], dx

323 <1> ;

324 0000643D E81DF6FFFF <1> call get\_rtc\_date ; Return Current Date

325 00006442 86E9 <1> xchg ch,cl

326 00006444 66890D[2A550100] <1> mov [year], cx

327 0000644B 86F2 <1> xchg dh,dl

328 0000644D 668915[2C550100] <1> mov [month], dx

329 <1> ;

330 00006454 66B93030 <1> mov cx, 3030h

331 <1> ;

332 00006458 A0[30550100] <1> mov al, [hour] ; Hour

333 <1> ; AL <= BCD number)

334 0000645D D410 <1> db 0D4h,10h ; Undocumented inst. AAM

335 <1> ; AH = AL / 10h

336 <1> ; AL = AL MOD 10h

337 0000645F D50A <1> aad ; AX= AH\*10+AL

338 00006461 A2[30550100] <1> mov [hour], al

339 00006466 A0[31550100] <1> mov al, [hour+1] ; Minute

340 <1> ; AL <= BCD number)

341 0000646B D410 <1> db 0D4h,10h ; Undocumented inst. AAM

342 <1> ; AH = AL / 10h

343 <1> ; AL = AL MOD 10h

344 0000646D D50A <1> aad ; AX= AH\*10+AL

345 0000646F A2[32550100] <1> mov [minute], al

346 00006474 A0[34550100] <1> mov al, [second] ; Second

347 <1> ; AL <= BCD number)

348 00006479 D410 <1> db 0D4h,10h ; Undocumented inst. AAM

349 <1> ; AH = AL / 10h

350 <1> ; AL = AL MOD 10h

351 0000647B D50A <1> aad ; AX= AH\*10+AL

352 0000647D A2[34550100] <1> mov [second], al

353 00006482 66A1[2A550100] <1> mov ax, [year] ; Year (century)

354 00006488 6650 <1> push ax

355 <1> ; AL <= BCD number)

356 0000648A D410 <1> db 0D4h,10h ; Undocumented inst. AAM

357 <1> ; AH = AL / 10h

358 <1> ; AL = AL MOD 10h

359 0000648C D50A <1> aad ; AX= AH\*10+AL

360 0000648E B464 <1> mov ah, 100

361 00006490 F6E4 <1> mul ah

362 00006492 66A3[2A550100] <1> mov [year], ax

363 00006498 6658 <1> pop ax

364 0000649A 88E0 <1> mov al, ah

365 <1> ; AL <= BCD number)

366 0000649C D410 <1> db 0D4h,10h ; Undocumented inst. AAM

367 <1> ; AH = AL / 10h

368 <1> ; AL = AL MOD 10h

369 0000649E D50A <1> aad ; AX= AH\*10+AL

370 000064A0 660105[2A550100] <1> add [year], ax

371 000064A7 A0[2C550100] <1> mov al, [month] ; Month

372 <1> ; AL <= BCD number)

373 000064AC D410 <1> db 0D4h,10h ; Undocumented inst. AAM

374 <1> ; AH = AL / 10h

375 <1> ; AL = AL MOD 10h

376 000064AE D50A <1> aad ; AX= AH\*10+AL

377 000064B0 A2[2C550100] <1> mov [month], al

378 000064B5 A0[2D550100] <1> mov al, [month+1] ; Day

379 <1> ; AL <= BCD number)

380 000064BA D410 <1> db 0D4h,10h ; Undocumented inst. AAM

381 <1> ; AH = AL / 10h

382 <1> ; AL = AL MOD 10h

383 000064BC D50A <1> aad ; AX= AH\*10+AL

384 000064BE A2[2E550100] <1> mov [day], al

385 <1>

386 000064C3 C3 <1> retn ; 30/12/2017

387 <1>

388 <1> epoch:

389 000064C4 E85DFFFFFF <1> call get\_rtc\_date\_time ; TRDOS 386 - 30/12/2017

390 <1>

391 <1> convert\_to\_epoch:

392 <1> ; 31/12/2017 (TRDOS 386 = TRDOS v2.0)

393 <1> ; 15/03/2015 (Retro UNIX 386 v1 - 32 bit modification)

394 <1> ; 09/04/2013 (Retro UNIX 8086 v1)

395 <1> ;

396 <1> ; ((Modified registers: EAX, EDX, EBX))

397 <1> ;

398 <1> ; Derived from DALLAS Semiconductor

399 <1> ; Application Note 31 (DS1602/DS1603)

400 <1> ; 6 May 1998

401 000064C9 29C0 <1> sub eax, eax

402 000064CB 66A1[2A550100] <1> mov ax, [year]

403 000064D1 662DB207 <1> sub ax, 1970

404 000064D5 BA6D010000 <1> mov edx, 365

405 000064DA F7E2 <1> mul edx

406 000064DC 31DB <1> xor ebx, ebx

407 000064DE 8A1D[2C550100] <1> mov bl, [month]

408 000064E4 FECB <1> dec bl

409 000064E6 D0E3 <1> shl bl, 1

410 <1> ;sub edx, edx

411 000064E8 668B93[36550100] <1> mov dx, [EBX+DMonth]

412 000064EF 8A1D[2E550100] <1> mov bl, [day]

413 000064F5 FECB <1> dec bl

414 000064F7 01D0 <1> add eax, edx

415 000064F9 01D8 <1> add eax, ebx

416 <1> ; EAX = days since 1/1/1970

417 000064FB 668B15[2A550100] <1> mov dx, [year]

418 00006502 6681EAB107 <1> sub dx, 1969

419 00006507 66D1EA <1> shr dx, 1

420 0000650A 66D1EA <1> shr dx, 1

421 <1> ; (year-1969)/4

422 0000650D 01D0 <1> add eax, edx

423 <1> ; + leap days since 1/1/1970

424 0000650F 803D[2C550100]02 <1> cmp byte [month], 2 ; if past february

425 00006516 7610 <1> jna short cte1

426 00006518 668B15[2A550100] <1> mov dx, [year]

427 0000651F 6683E203 <1> and dx, 3 ; year mod 4

428 00006523 7503 <1> jnz short cte1

429 <1> ; and if leap year

430 00006525 83C001 <1> add eax, 1 ; add this year's leap day (february 29)

431 <1> cte1: ; compute seconds since 1/1/1970

432 00006528 BA18000000 <1> mov edx, 24

433 0000652D F7E2 <1> mul edx

434 0000652F 8A15[30550100] <1> mov dl, [hour]

435 00006535 01D0 <1> add eax, edx

436 <1> ; EAX = hours since 1/1/1970 00:00:00

437 <1> ;mov ebx, 60

438 00006537 B33C <1> mov bl, 60

439 00006539 F7E3 <1> mul ebx

440 0000653B 8A15[32550100] <1> mov dl, [minute]

441 00006541 01D0 <1> add eax, edx

442 <1> ; EAX = minutes since 1/1/1970 00:00:00

443 <1> ;mov ebx, 60

444 00006543 F7E3 <1> mul ebx

445 00006545 8A15[34550100] <1> mov dl, [second]

446 0000654B 01D0 <1> add eax, edx

447 <1> ; EAX -> seconds since 1/1/1970 00:00:00

448 0000654D C3 <1> retn

449 <1>

450 <1> ;set\_date\_time:

451 <1> convert\_from\_epoch:

452 <1> ; 31/12/2017

453 <1> ; 30/12/2017 (TRDOS 386 = TRDOS v2.0)

454 <1> ; 15/03/2015 (Retro UNIX 386 v1 - 32 bit version)

455 <1> ; 20/06/2013 (Retro UNIX 8086 v1)

456 <1> ; 'convert\_from\_epoch' procedure prototype:

457 <1> ; UNIXCOPY.ASM, 10/03/2013

458 <1> ;

459 <1> ; ((Modified registers: EAX, EDX, ECX, EBX))

460 <1> ;

461 <1> ; Derived from DALLAS Semiconductor

462 <1> ; Application Note 31 (DS1602/DS1603)

463 <1> ; 6 May 1998

464 <1> ;

465 <1> ; INPUT:

466 <1> ; EAX = Unix (Epoch) Time

467 <1> ;

468 0000654E 31D2 <1> xor edx, edx

469 00006550 B93C000000 <1> mov ecx, 60

470 00006555 F7F1 <1> div ecx

471 <1> ;mov [imin], eax ; whole minutes

472 <1> ; since 1/1/1970

473 00006557 668915[34550100] <1> mov [second], dx ; leftover seconds

474 0000655E 29D2 <1> sub edx, edx

475 00006560 F7F1 <1> div ecx

476 <1> ;mov [ihrs], eax ; whole hours

477 <1> ; ; since 1/1/1970

478 00006562 668915[32550100] <1> mov [minute], dx ; leftover minutes

479 00006569 31D2 <1> xor edx, edx

480 <1> ;mov cx, 24

481 0000656B B118 <1> mov cl, 24

482 0000656D F7F1 <1> div ecx

483 <1> ;mov [iday], ax ; whole days

484 <1> ; since 1/1/1970

485 0000656F 668915[30550100] <1> mov [hour], dx ; leftover hours

486 00006576 05DB020000 <1> add eax, 365+366 ; whole day since

487 <1> ; 1/1/1968

488 <1> ;mov [iday], ax

489 0000657B 50 <1> push eax

490 0000657C 29D2 <1> sub edx, edx

491 0000657E B9B5050000 <1> mov ecx, (4\*365)+1 ; 4 years = 1461 days

492 00006583 F7F1 <1> div ecx

493 00006585 59 <1> pop ecx

494 <1> ;mov [lday], ax ; count of quadyrs (4 years)

495 00006586 6652 <1> push dx

496 <1> ;mov [qday], dx ; days since quadyr began

497 00006588 6683FA3C <1> cmp dx, 31 + 29 ; if past feb 29 then

498 0000658C F5 <1> cmc ; add this quadyr's leap day

499 0000658D 83D000 <1> adc eax, 0 ; to # of qadyrs (leap days)

500 <1> ;mov [lday], ax ; since 1968

501 <1> ;mov cx, [iday]

502 00006590 91 <1> xchg ecx, eax ; ECX = lday, EAX = iday

503 00006591 29C8 <1> sub eax, ecx ; iday - lday

504 00006593 B96D010000 <1> mov ecx, 365

505 00006598 31D2 <1> xor edx, edx

506 <1> ; EAX = iday-lday, EDX = 0

507 0000659A F7F1 <1> div ecx

508 <1> ;mov [iyrs], ax ; whole years since 1968

509 <1> ;jday = iday - (iyrs\*365) - lday

510 <1> ;mov [jday], dx ; days since 1/1 of current year

511 <1> ;add eax, 1968

512 0000659C 6605B007 <1> add ax, 1968 ; compute year

513 000065A0 66A3[2A550100] <1> mov [year], ax

514 000065A6 6689D1 <1> mov cx, dx

515 <1> ;mov dx, [qday]

516 000065A9 665A <1> pop dx

517 000065AB 6681FA6D01 <1> cmp dx, 365 ; if qday <= 365 and qday >= 60

518 000065B0 7709 <1> ja short cfe1 ; jday = jday +1

519 000065B2 6683FA3C <1> cmp dx, 60 ; if past 2/29 and leap year then

520 000065B6 F5 <1> cmc ; add a leap day to the # of whole

521 000065B7 6683D100 <1> adc cx, 0 ; days since 1/1 of current year

522 <1> cfe1:

523 <1> ;mov [jday], cx

524 000065BB 66BB0C00 <1> mov bx, 12 ; estimate month

525 000065BF 66BA6E01 <1> mov dx, 366 ; mday, max. days since 1/1 is 365

526 000065C3 6683E003 <1> and ax, 11b ; year mod 4 (and dx, 3)

527 <1> cfe2: ; Month calculation ; 0 to 11 (11 to 0)

528 000065C7 6639D1 <1> cmp cx, dx ; mday = # of days passed from 1/1

529 000065CA 731D <1> jnb short cfe3

530 000065CC 664B <1> dec bx ; month = month - 1

531 000065CE 66D1E3 <1> shl bx, 1

532 000065D1 668B93[36550100] <1> mov dx, [EBX+DMonth] ; # elapsed days at 1st of month

533 000065D8 66D1EB <1> shr bx, 1 ; bx = month - 1 (0 to 11)

534 000065DB 6683FB01 <1> cmp bx, 1 ; if month > 2 and year mod 4 = 0

535 000065DF 76E6 <1> jna short cfe2 ; then mday = mday + 1

536 000065E1 08C0 <1> or al, al ; if past 2/29 and leap year then

537 000065E3 75E2 <1> jnz short cfe2 ; add leap day (to mday)

538 000065E5 6642 <1> inc dx ; mday = mday + 1

539 000065E7 EBDE <1> jmp short cfe2

540 <1> cfe3:

541 000065E9 6643 <1> inc bx ; -> bx = month, 1 to 12

542 000065EB 66891D[2C550100] <1> mov [month], bx

543 000065F2 6629D1 <1> sub cx, dx ; day = jday - mday + 1

544 000065F5 6641 <1> inc cx

545 000065F7 66890D[2E550100] <1> mov [day], cx

546 <1>

547 <1> ; eax, ebx, ecx, edx is changed at return

548 <1> ; output ->

549 <1> ; [year], [month], [day], [hour], [minute], [second]

550 <1>

551 000065FE C3 <1> retn ; 31/12/2017 (TRDOS 386)

552 <1>

553 <1> set\_rtc\_date\_time:

554 <1> ; 31/12/2017

555 <1> ; 30/12/2017 (TRDOS 386)

556 <1> ; 15/03/2015 (Retro UNIX 386 v1 - 32 bit version)

557 <1> ; 20/06/2013 (Retro UNIX 8086 v1)

558 000065FF E80F000000 <1> call set\_date\_bcd

559 <1> ; Set real-time clock date

560 00006604 E883F4FFFF <1> call set\_rtc\_date ; RTC\_50

561 <1> ; Set real-time clock time

562 00006609 E832000000 <1> call set\_time\_bcd

563 0000660E E90AF4FFFF <1> jmp set\_rtc\_time ; RTC\_30

564 <1>

565 <1> ; 31/12/2017

566 <1> set\_date\_bcd:

567 00006613 A0[2B550100] <1> mov al, [year+1]

568 00006618 D40A <1> aam ; ah = al / 10, al = al mod 10

569 0000661A D510 <1> db 0D5h,10h ; Undocumented inst. AAD

570 <1> ; AL = AH \* 10h + AL

571 0000661C 88C5 <1> mov ch, al ; century (BCD)

572 0000661E A0[2A550100] <1> mov al, [year]

573 00006623 D40A <1> aam ; ah = al / 10, al = al mod 10

574 00006625 D510 <1> db 0D5h,10h ; Undocumented inst. AAD

575 <1> ; AL = AH \* 10h + AL

576 00006627 88C1 <1> mov cl, al ; year (BCD)

577 00006629 A0[2C550100] <1> mov al, [month]

578 0000662E D40A <1> aam ; ah = al / 10, al = al mod 10

579 00006630 D510 <1> db 0D5h,10h ; Undocumented inst. AAD

580 <1> ; AL = AH \* 10h + AL

581 00006632 88C6 <1> mov dh, al ; month (BCD)

582 00006634 A0[2E550100] <1> mov al, [day]

583 00006639 D40A <1> aam ; ah = al / 10, al = al mod 10

584 0000663B D510 <1> db 0D5h,10h ; Undocumented inst. AAD

585 <1> ; AL = AH \* 10h + AL

586 0000663D 88C6 <1> mov dh, al ; day (BCD)

587 0000663F C3 <1> retn ; 30/12/2017

588 <1>

589 <1> ; 31/12/2017

590 <1> set\_time\_bcd:

591 <1> ; Read real-time clock time

592 <1> ; (get day light saving time bit status)

593 00006640 FA <1> cli

594 00006641 E898F5FFFF <1> CALL UPD\_IPR ; CHECK FOR UPDATE IN PROCESS

595 <1> ; cf = 1 -> al = 0

596 00006646 7207 <1> jc short stime1

597 00006648 B00B <1> MOV AL,CMOS\_REG\_B ; ADDRESS ALARM REGISTER

598 0000664A E8AAF5FFFF <1> CALL CMOS\_READ ; READ CURRENT VALUE OF DSE BIT

599 <1> stime1:

600 0000664F FB <1> sti

601 00006650 2401 <1> AND AL,00000001B ; MASK FOR VALID DSE BIT

602 00006652 88C2 <1> MOV DL,AL ; SET [DL] TO ZERO FOR NO DSE BIT

603 <1> ; DL = 1 or 0 (day light saving time)

604 <1> ;

605 00006654 A0[30550100] <1> mov al, [hour]

606 00006659 D40A <1> aam ; ah = al / 10, al = al mod 10

607 0000665B D510 <1> db 0D5h,10h ; Undocumented inst. AAD

608 <1> ; AL = AH \* 10h + AL

609 0000665D 88C5 <1> mov ch, al ; hour (BCD)

610 0000665F A0[32550100] <1> mov al, [minute]

611 00006664 D40A <1> aam ; ah = al / 10, al = al mod 10

612 00006666 D510 <1> db 0D5h,10h ; Undocumented inst. AAD

613 <1> ; AL = AH \* 10h + AL

614 00006668 88C1 <1> mov cl, al ; minute (BCD)

615 0000666A A0[34550100] <1> mov al, [second]

616 0000666F D40A <1> aam ; ah = al / 10, al = al mod 10

617 00006671 D510 <1> db 0D5h,10h ; Undocumented inst. AAD

618 <1> ; AL = AH \* 10h + AL

619 00006673 88C6 <1> mov dh, al ; second (BCD)

620 00006675 C3 <1> retn ; 30/12/2017

2306 %include 'trdosk2.s' ; 04/01/2016

1 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2 <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.0) - DRV INIT : trdosk2.s

3 <1> ; ----------------------------------------------------------------------------

4 <1> ; Last Update: 27/12/2017

5 <1> ; ----------------------------------------------------------------------------

6 <1> ; Beginning: 04/01/2016

7 <1> ; ----------------------------------------------------------------------------

8 <1> ; Assembler: NASM version 2.11 (trdos386.s)

9 <1> ; ----------------------------------------------------------------------------

10 <1> ; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan

11 <1> ; TRDOS2.ASM (09/11/2011)

12 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

13 <1> ; DRV\_INIT.ASM (c) 2009-2011 Erdogan TAN [26/09/2009] Last Update: 07/08/2011

14 <1> ;

15 <1>

16 <1> ldrv\_init: ; Logical Drive Initialization

17 <1> ; 27/12/2017

18 <1> ; 12/02/2016

19 <1> ; 06/01/2016

20 <1> ; ('diskinit.inc', 'diskio.inc' integration)

21 <1> ; 04/01/2016 (TRDOS 386 = TRDOS v2.0)

22 <1> ; 07/08/2011

23 <1> ; 20/09/2009

24 <1> ; 2005

25 00006676 0FB60D[D4580100] <1> movzx ecx, byte [HF\_NUM] ; number of fixed disks

26 0000667D 80F901 <1> cmp cl, 1

27 00006680 7301 <1> jnb short load\_hd\_partition\_tables

28 <1> ; No hard disks

29 00006682 C3 <1> retn

30 <1> load\_hd\_partition\_tables:

31 00006683 8B35[D8580100] <1> mov esi, [HDPM\_TBL\_VEC] ; primary master disk FDPT

32 00006689 BF[FE5C0100] <1> mov edi, PTable\_hd0

33 0000668E B280 <1> mov dl, 80h

34 <1> load\_next\_hd\_partition\_table:

35 00006690 51 <1> push ecx

36 00006691 57 <1> push edi

37 00006692 56 <1> push esi ; FDPT (+ DPTE) address

38 00006693 8A4614 <1> mov al, [esi+20] ; DPTE offset 4

39 00006696 2440 <1> and al, 40h ; LBA bit (bit 6)

40 <1> ;shr al, 6

41 00006698 A2[FF5E0100] <1> mov [HD\_LBA\_yes], al

42 0000669D E82B040000 <1> call load\_masterboot

43 000066A2 7275 <1> jc short pass\_pt\_this\_hard\_disk

44 <1>

45 000066A4 BE[BC5C0100] <1> mov esi, PartitionTable

46 000066A9 89F3 <1> mov ebx, esi

47 <1> ;mov ecx, 16

48 000066AB B110 <1> mov cl, 16

49 000066AD F3A5 <1> rep movsd

50 000066AF 89DE <1> mov esi, ebx

51 000066B1 C605[F55C0000]04 <1> mov byte [hdc], 4 ; 4 - partition index

52 <1> loc\_validate\_hdp\_partition:

53 000066B8 807E0400 <1> cmp byte [esi+ptFileSystemID], 0

54 000066BC 7641 <1> jna short loc\_validate\_next\_hdp\_partition2

55 000066BE 56 <1> push esi ; Masterboot partition table offset

56 000066BF 52 <1> push edx ; dl = Physical drive number

57 000066C0 FE05[005F0100] <1> inc byte [PP\_Counter]

58 000066C6 31FF <1> xor edi, edi ; 0

59 <1> ; Input -> ESI = PartitionTable offset

60 <1> ; DL = Hard disk drive number

61 <1> ; EDI = 0 -> Primary Partition

62 <1> ; EDI > 0 -> Extended Partition's Start Sector

63 000066C8 E879010000 <1> call validate\_hd\_fat\_partition

64 000066CD 730A <1> jnc short loc\_set\_valid\_hdp\_partition\_entry

65 <1> ;pop edx

66 <1> ;push edx

67 000066CF 8B1424 <1> mov edx, [esp]

68 000066D2 E8D4020000 <1> call validate\_hd\_fs\_partition

69 000066D7 7224 <1> jc short loc\_validate\_next\_hdp\_partition1

70 <1> loc\_set\_valid\_hdp\_partition\_entry:

71 000066D9 8A0D[D20C0100] <1> mov cl, [Last\_DOS\_DiskNo]

72 000066DF 80C141 <1> add cl, 'A'

73 <1> ; ESI = Logical dos drive description table address

74 000066E2 880E <1> mov [esi+LD\_Name], cl

75 000066E4 8A6602 <1> mov ah, [esi+LD\_PhyDrvNo]

76 000066E7 88E0 <1> mov al, ah ; Physical drive number

77 000066E9 2C80 <1> sub al, 80h

78 000066EB C0E002 <1> shl al, 2

79 000066EE 0404 <1> add al, 4 ; 0 Based

80 000066F0 2A05[F55C0000] <1> sub al, [hdc] ; 4 - partition index

81 <1> ; AL = Partition entry/index, 0 based

82 <1> ; 0 -> hd 0, Partition Table offset = 0

83 <1> ; 15 -> hd 3, Partition Table offset = 3

84 <1> ;mov [esi+LD\_PartitionEntry], al

85 000066F6 80EC7E <1> sub ah, 7Eh

86 <1> ; AH = Physical drive index, zero based

87 <1> ; 0 for drive A:, 2 for drive C:

88 <1> ;mov [esi+LD\_DParamEntry], ah

89 000066F9 6689467C <1> mov [esi+LD\_PartitionEntry], ax

90 <1> loc\_validate\_next\_hdp\_partition1:

91 000066FD 5A <1> pop edx ; dl = Physical drive number

92 000066FE 5E <1> pop esi ; Masterboot partition table offset

93 <1> loc\_validate\_next\_hdp\_partition2:

94 <1> ; ESI = PartitionTable offset

95 <1> ; DL = Hard/Fixed disk drive number

96 000066FF FE0D[F55C0000] <1> dec byte [hdc] ; 4 - partition index

97 00006705 7412 <1> jz short pass\_pt\_this\_hard\_disk

98 00006707 83C610 <1> add esi, 16 ; 10h

99 0000670A EBAC <1> jmp short loc\_validate\_hdp\_partition

100 <1> loc\_next\_hd\_partition\_table:

101 0000670C FEC2 <1> inc dl

102 0000670E 83C620 <1> add esi, 32 ; next FDPT address

103 00006711 83C740 <1> add edi, 64 ; next partition table destination

104 00006714 E977FFFFFF <1> jmp load\_next\_hd\_partition\_table

105 <1> pass\_pt\_this\_hard\_disk:

106 00006719 5E <1> pop esi ; FDPT (+ DPTE) address

107 0000671A 5F <1> pop edi ; Ptable\_hd?

108 0000671B 59 <1> pop ecx

109 0000671C E2EE <1> loop loc\_next\_hd\_partition\_table

110 0000671E 803D[005F0100]01 <1> cmp byte [PP\_Counter], 1

111 00006725 7301 <1> jnb short load\_extended\_dos\_partitions

112 <1> ; Empty partition table

113 00006727 C3 <1> retn

114 <1> load\_extended\_dos\_partitions:

115 00006728 BE[FE5C0100] <1> mov esi, PTable\_hd0

116 0000672D BF[FE5D0100] <1> mov edi, PTable\_ep0

117 00006732 C605[F55C0000]80 <1> mov byte [hdc], 80h

118 <1> next\_hd\_extd\_partition:

119 00006739 56 <1> push esi ; PTable\_hd? offset

120 0000673A 57 <1> push edi ; PTable\_ep?

121 <1> ;mov ecx, 4

122 0000673B B104 <1> mov cl, 4

123 0000673D 8A15[F55C0000] <1> mov dl, byte [hdc]

124 <1> hd\_check\_fs\_id\_05h:

125 00006743 8A4604 <1> mov al, [esi+ptFileSystemID]

126 00006746 3C05 <1> cmp al, 05h ; Is it an extended dos partition ?

127 00006748 7404 <1> je short loc\_set\_ep\_start\_sector

128 0000674A 3C0F <1> cmp al, 0Fh ; Is it an extended win4 (LBA mode) partition ?

129 0000674C 7546 <1> jne short continue\_to\_check\_ep

130 <1> loc\_set\_ep\_start\_sector:

131 0000674E FE05[015F0100] <1> inc byte [EP\_Counter]

132 00006754 88D4 <1> mov ah, dl ; byte [hdc]

133 00006756 86E0 <1> xchg ah, al ; al = Drv Number, ah = Partition Identifier

134 00006758 50 <1> push eax

135 00006759 30E4 <1> xor ah, ah

136 0000675B 2C80 <1> sub al, 80h

137 0000675D 50 <1> push eax

138 0000675E C0E002 <1> shl al, 2 ; al = al \* 4

139 00006761 0FB6D8 <1> movzx ebx, al

140 00006764 81C3[025F0100] <1> add ebx, EP\_StartSector

141 0000676A 8B4608 <1> mov eax, [esi+ptStartSector]

142 <1> ; EAX = Extended partition's start sector

143 0000676D 8903 <1> mov [ebx], eax

144 0000676F 58 <1> pop eax ; AL = Drv number - 80h, AH = 0

145 00006770 5A <1> pop edx ; DL = Drv number, DH = Partition ID

146 00006771 BB[FE5A0100] <1> mov ebx, MasterBootBuff

147 00006776 803D[FF5E0100]01 <1> cmp byte [HD\_LBA\_yes], 1 ; LBA ready = Yes

148 0000677D 7240 <1> jb short loc\_hd\_load\_ep\_05h

149 0000677F 80FE05 <1> cmp dh, 05h

150 00006782 743B <1> je short loc\_hd\_load\_ep\_05h

151 <1> loc\_hd\_load\_ep\_0Fh:

152 <1> ; 04/01/2016

153 00006784 51 <1> push ecx

154 00006785 8B4E08 <1> mov ecx, [esi+ptStartSector] ; sector number

155 <1> ;mov ebx, MasterBootBuff ; buffer address

156 <1> ; LBA read/write (with private LBA function)

157 <1> ;((Retro UNIX 386 v1 - DISK I/O code by Erdogan Tan))

158 <1> ; dl = physical drive number (0,1, 80h, 81h, 82h, 83h)

159 00006788 B41B <1> mov ah, 1Bh ; LBA read

160 0000678A B001 <1> mov al, 1 ; sector count

161 0000678C E875DAFFFF <1> call int13h

162 00006791 59 <1> pop ecx

163 00006792 733F <1> jnc short loc\_hd\_move\_ep\_table

164 <1> continue\_to\_check\_ep:

165 00006794 83C610 <1> add esi, 16

166 00006797 E2AA <1> loop hd\_check\_fs\_id\_05h

167 <1> continue\_check\_ep\_next\_disk:

168 00006799 5F <1> pop edi ; PTable\_ep?

169 0000679A 5E <1> pop esi ; PTable\_hd?

170 0000679B A0[D4580100] <1> mov al, [HF\_NUM] ; number of hard disks

171 000067A0 047F <1> add al, 7Fh

172 000067A2 3805[F55C0000] <1> cmp [hdc], al

173 000067A8 0F8392000000 <1> jnb loc\_validating\_hd\_partitions\_ok

174 000067AE 83C640 <1> add esi, 64

175 000067B1 83C740 <1> add edi, 64

176 000067B4 FE05[F55C0000] <1> inc byte [hdc]

177 000067BA E97AFFFFFF <1> jmp next\_hd\_extd\_partition

178 <1> loc\_hd\_load\_ep\_05h:

179 000067BF 51 <1> push ecx

180 000067C0 8A7601 <1> mov dh, [esi+ptBeginHead]

181 000067C3 668B4E02 <1> mov cx, word [esi+ptBeginSector]

182 000067C7 66B80102 <1> mov ax, 0201h ; Read 1 sector

183 <1> ;mov ebx, MasterBootBuff

184 000067CB E836DAFFFF <1> call int13h

185 000067D0 59 <1> pop ecx

186 000067D1 72C1 <1> jc short continue\_to\_check\_ep

187 <1> loc\_hd\_move\_ep\_table:

188 <1> ;pop edi

189 <1> ;push edi ; PTable\_ep?

190 000067D3 8B3C24 <1> mov edi, [esp]

191 000067D6 BE[BC5C0100] <1> mov esi, PartitionTable ; Extended

192 000067DB 89F3 <1> mov ebx, esi

193 <1> ;mov ecx, 16

194 000067DD B110 <1> mov cl, 16

195 000067DF F3A5 <1> rep movsd

196 000067E1 89DE <1> mov esi, ebx

197 <1> loc\_set\_hde\_sub\_partition\_count:

198 000067E3 C605[005F0100]04 <1> mov byte [PP\_Counter], 4

199 <1> loc\_validate\_hde\_partition:

200 000067EA 807E0400 <1> cmp byte [esi+ptFileSystemID], 0

201 000067EE 763F <1> jna short loc\_validate\_next\_hde\_partition2

202 000067F0 56 <1> push esi ; Extended partition table offset

203 000067F1 8A15[F55C0000] <1> mov dl, byte [hdc]

204 000067F7 0FB6C2 <1> movzx eax, dl

205 000067FA 2C80 <1> sub al, 80h

206 000067FC C0E002 <1> shl al, 2

207 <1> ; 06/01/2016

208 <1> ; (TRDOS v1.0 had a bug here, in 'DRV\_INIT.ASM')

209 <1> ; BUGFIX \*

210 <1> ;mov ecx, eax

211 000067FF 88C1 <1> mov cl, al

212 00006801 80C104 <1> add cl, 4

213 00006804 2A0D[005F0100] <1> sub cl, [PP\_Counter] ; 4 to 1

214 <1> ; CL = Partition entry/index, 0 based

215 <1> ; 0 -> hd 0, Partition Table offset = 0

216 <1> ; 15 -> hd 3, Partition Table offset = 3

217 0000680A 88D5 <1> mov ch, dl

218 0000680C 80ED7E <1> sub ch, 7Eh ;

219 <1> ; CH = Physical drive index, zero based

220 <1> ; 0 for drive A:, 2 for drive C:

221 <1> ; BUGFIX \*

222 0000680F 51 <1> push ecx ; \*

223 00006810 BF[025F0100] <1> mov edi, EP\_StartSector

224 00006815 01C7 <1> add edi, eax

225 <1> ; Input -> ESI = PartitionTable offset

226 <1> ; DL = Hard disk drive number

227 <1> ; EDI = Extended partition start sector pointer

228 00006817 E82A000000 <1> call validate\_hd\_fat\_partition

229 0000681C 59 <1> pop ecx ; \*

230 0000681D 720F <1> jc short loc\_validate\_next\_hde\_partition1

231 <1> loc\_set\_valid\_hde\_partition\_entry:

232 <1> ; 06/01/2016 (TRDOS v2.0)

233 <1> ; BUGFIX \*

234 <1> ;mov [esi+LD\_PartitionEntry], cl

235 <1> ;mov [esi+LD\_DParamEntry], ch

236 0000681F 66894E7C <1> mov [esi+LD\_PartitionEntry], cx

237 <1> ;

238 00006823 8A0D[D20C0100] <1> mov cl, [Last\_DOS\_DiskNo]

239 00006829 80C141 <1> add cl, 'A'

240 0000682C 880E <1> mov [esi+LD\_Name], cl

241 <1> loc\_validate\_next\_hde\_partition1:

242 0000682E 5E <1> pop esi ; Extended partition table offset

243 <1> loc\_validate\_next\_hde\_partition2:

244 <1> ; ESI = Extended partition table offset

245 <1> ; DL = Hard disk drive number

246 0000682F FE0D[005F0100] <1> dec byte [PP\_Counter]

247 00006835 0F845EFFFFFF <1> jz continue\_check\_ep\_next\_disk

248 0000683B 83C610 <1> add esi, 16 ; 10h

249 0000683E EBAA <1> jmp short loc\_validate\_hde\_partition

250 <1> loc\_validating\_hd\_partitions\_ok:

251 00006840 A0[D20C0100] <1> mov al, [Last\_DOS\_DiskNo]

252 <1> loc\_drv\_init\_retn:

253 00006845 C3 <1> retn

254 <1>

255 <1> validate\_hd\_fat\_partition:

256 <1> ; 27/12/2017

257 <1> ; 12/02/2016

258 <1> ; 07/01/2016 (TRDOS 386 = TRDOS v2.0)

259 <1> ; 07/08/2011

260 <1> ; 23/07/2011

261 <1> ; Input

262 <1> ; DL = Hard/Fixed Disk Drive Number

263 <1> ; ESI = PartitionTable offset

264 <1> ; EDI = Extend. Part. Start Sector Pointer

265 <1> ; EDI = 0 -> Primary Partition

266 <1> ; byte [Last\_DOS\_DiskNo]

267 <1> ; Output

268 <1> ; cf=0 -> Validated

269 <1> ; ESI = Logical dos drv desc. table

270 <1> ; EBX = FAT boot sector buffer

271 <1> ; byte [Last\_DOS\_DiskNo]

272 <1> ; cf=1 -> Not a valid FAT partition

273 <1> ; EAX, EDX, ECX, EDI -> changed

274 <1>

275 <1> ;mov esi, PartitionTable

276 00006846 8A6604 <1> mov ah, [esi+ptFileSystemID]

277 00006849 B002 <1> mov al, 2 ; 27/12/2017

278 0000684B 80FC06 <1> cmp ah, 06h ; FAT16 CHS partition

279 <1> ; 12/02/2016

280 <1> ;jb short loc\_not\_a\_valid\_fat\_partition2

281 0000684E 7310 <1> jnb short vhdp\_FAT16\_32

282 <1> ;

283 <1> ; 27/12/2017

284 00006850 FEC8 <1> dec al ; mov al, 1

285 00006852 38C4 <1> cmp ah, al ; 1 ; FAT12 partition

286 00006854 7421 <1> je short loc\_set\_valid\_hd\_partition\_params

287 <1> ;

288 00006856 FEC0 <1> inc al ; mov al, 2

289 00006858 80FC04 <1> cmp ah, 04h ; FAT16 CHS partition (< 32MB)

290 0000685B 741A <1> je short loc\_set\_valid\_hd\_partition\_params

291 0000685D 7716 <1> ja short loc\_not\_a\_valid\_fat\_partition1

292 <1> ; cf=1

293 0000685F C3 <1> retn

294 <1> vhdp\_FAT16\_32:

295 00006860 80FC0E <1> cmp ah, 0Eh ; FAT16 LBA partition

296 00006863 7710 <1> ja short loc\_not\_a\_valid\_fat\_partition1

297 00006865 7410 <1> je short loc\_set\_valid\_hd\_partition\_params

298 <1> ;mov al, 3

299 00006867 FEC0 <1> inc al

300 00006869 80FC0B <1> cmp ah, 0Bh ; FAT32 CHS partition

301 0000686C 7409 <1> je short loc\_set\_valid\_hd\_partition\_params

302 0000686E 7206 <1> jb short loc\_not\_a\_valid\_fat\_partition2

303 00006870 80FC0C <1> cmp ah, 0Ch ; FAT32 LBA partition

304 00006873 7402 <1> je short loc\_set\_valid\_hd\_partition\_params

305 <1> loc\_not\_a\_valid\_fat\_partition1:

306 00006875 F9 <1> stc

307 <1> loc\_not\_a\_valid\_fat\_partition2:

308 00006876 C3 <1> retn

309 <1>

310 <1> loc\_set\_valid\_hd\_partition\_params:

311 00006877 FE05[D20C0100] <1> inc byte [Last\_DOS\_DiskNo] ; > 1

312 <1> ;

313 0000687D 31DB <1> xor ebx, ebx

314 0000687F 8A3D[D20C0100] <1> mov bh, [Last\_DOS\_DiskNo] ; \* 256

315 00006885 81C300010900 <1> add ebx, Logical\_DOSDisks

316 <1> ;

317 0000688B C6430102 <1> mov byte [ebx+LD\_DiskType], 2

318 0000688F 885302 <1> mov byte [ebx+LD\_PhyDrvNo], dl

319 <1> ;mov byte [ebx+LD\_FATType], al ; 2 or 3

320 <1> ;mov byte [ebx+LD\_FSType], ah ; 06h, 0Eh, 0Bh, 0Ch

321 00006892 66894303 <1> mov word [ebx+LD\_FATType], ax

322 <1> ;

323 00006896 8B4E08 <1> mov ecx, [esi+ptStartSector]

324 00006899 09FF <1> or edi, edi

325 0000689B 7402 <1> jz short pass\_hd\_FAT\_ep\_start\_sector\_adding

326 <1> loc\_add\_hd\_FAT\_ep\_start\_sector:

327 0000689D 030F <1> add ecx, [edi]

328 <1> pass\_hd\_FAT\_ep\_start\_sector\_adding:

329 0000689F 894B6C <1> mov [ebx+LD\_StartSector], ecx

330 <1> loc\_hd\_FAT\_logical\_drv\_init:

331 000068A2 89DD <1> mov ebp, ebx

332 <1> ;mov dl, [ebx+LD\_PhyDrvNo]

333 000068A4 A0[FF5E0100] <1> mov al, [HD\_LBA\_yes] ; 07/01/2016

334 000068A9 884305 <1> mov [ebx+LD\_LBAYes], al

335 000068AC BB[125F0100] <1> mov ebx, DOSBootSectorBuff ; buffer address

336 000068B1 08C0 <1> or al, al

337 000068B3 740C <1> jz short loc\_hd\_FAT\_drv\_init\_load\_bs\_chs

338 <1> loc\_hd\_FAT\_drv\_init\_load\_bs\_lba:

339 <1> ; DL = Physical drive number

340 <1> ;mov ecx, [esi+ptStartSector] ; sector number

341 <1> ;mov ebx, DOSBootSectorBuff ; buffer address

342 <1> ; LBA read/write (with private LBA function)

343 <1> ;((Retro UNIX 386 v1 - DISK I/O code by Erdogan Tan))

344 <1> ; dl = physical drive number (0,1, 80h, 81h, 82h, 83h)

345 000068B5 B41B <1> mov ah, 1Bh ; LBA read

346 000068B7 B001 <1> mov al, 1 ; sector count

347 000068B9 E848D9FFFF <1> call int13h

348 000068BE 7313 <1> jnc short loc\_hd\_drv\_FAT\_boot\_validation

349 <1> loc\_not\_a\_valid\_fat\_partition3:

350 000068C0 C3 <1> retn

351 <1> loc\_hd\_FAT\_drv\_init\_load\_bs\_chs:

352 000068C1 8A7601 <1> mov dh, [esi+ptBeginHead]

353 000068C4 668B4E02 <1> mov cx, [esi+ptBeginSector]

354 000068C8 66B80102 <1> mov ax, 0201h ; Read 1 sector

355 <1> ;mov ebx, DOSBootSectorBuff

356 000068CC E835D9FFFF <1> call int13h

357 000068D1 72ED <1> jc short loc\_not\_a\_valid\_fat\_partition3

358 <1> loc\_hd\_drv\_FAT\_boot\_validation:

359 <1> ;mov esi, DOSBootSectorBuff

360 000068D3 89DE <1> mov esi, ebx

361 000068D5 6681BEFE01000055AA <1> cmp word [esi+BS\_Validation], 0AA55h

362 000068DE 7512 <1> jne short loc\_not\_a\_valid\_fat\_partition4

363 000068E0 807E15F8 <1> cmp byte [esi+BPB\_Media], 0F8h

364 000068E4 750C <1> jne short loc\_not\_a\_valid\_fat\_partition4

365 <1>

366 <1> ; 27/12/2017

367 000068E6 807D0303 <1> cmp byte [ebp+LD\_FATType], 3

368 000068EA 7508 <1> jne short loc\_hd\_FAT16\_BPB

369 <1>

370 <1> loc\_hd\_drv\_FAT32\_boot\_validation:

371 000068EC 807E4229 <1> cmp byte [esi+BS\_FAT32\_BootSig], 29h

372 000068F0 7416 <1> je short loc\_hd\_FAT32\_BPB

373 <1>

374 <1> loc\_not\_a\_valid\_fat\_partition4:

375 000068F2 F9 <1> stc

376 000068F3 C3 <1> retn

377 <1>

378 <1> loc\_hd\_FAT16\_BPB:

379 000068F4 807E2629 <1> cmp byte [esi+BS\_BootSig], 29h

380 000068F8 75F8 <1> jne short loc\_not\_a\_valid\_fat\_partition4

381 <1>

382 000068FA 66837E1600 <1> cmp word [esi+BPB\_FATSz16], 0

383 000068FF 7607 <1> jna short loc\_hd\_big\_FAT16\_BPB

384 00006901 B920000000 <1> mov ecx, 32

385 00006906 EB05 <1> jmp short loc\_hd\_move\_FAT\_BPB

386 <1>

387 <1> loc\_hd\_FAT32\_BPB:

388 <1> ;cmp word [esi+BPB\_FATSz16], 0

389 <1> ;ja short loc\_not\_a\_valid\_fat\_partition4

390 <1> loc\_hd\_big\_FAT16\_BPB:

391 00006908 B92D000000 <1> mov ecx, 45

392 <1> loc\_hd\_move\_FAT\_BPB:

393 0000690D 89EF <1> mov edi, ebp

394 <1> ;mov esi, ebx ; Boot sector

395 0000690F 57 <1> push edi

396 00006910 83C706 <1> add edi, LD\_BPB

397 00006913 F366A5 <1> rep movsw

398 00006916 5E <1> pop esi

399 00006917 0FB74614 <1> movzx eax, word [esi+LD\_BPB+BPB\_RsvdSecCnt]

400 0000691B 03466C <1> add eax, [esi+LD\_StartSector]

401 0000691E 894660 <1> mov [esi+LD\_FATBegin], eax

402 00006921 807E0303 <1> cmp byte [esi+LD\_FATType], 3

403 00006925 7224 <1> jb short loc\_set\_FAT16\_RootDirLoc

404 <1> loc\_set\_FAT32\_RootDirLoc:

405 00006927 8B462A <1> mov eax, [esi+LD\_BPB+BPB\_FATSz32]

406 0000692A 0FB65E16 <1> movzx ebx, byte [esi+LD\_BPB+BPB\_NumFATs]

407 0000692E F7E3 <1> mul ebx

408 00006930 034660 <1> add eax, [esi+LD\_FATBegin]

409 <1> loc\_set\_FAT32\_data\_begin:

410 00006933 894668 <1> mov [esi+LD\_DATABegin], eax

411 00006936 894664 <1> mov [esi+LD\_ROOTBegin], eax

412 <1> ; If Root Directory Cluster <> 2 then

413 <1> ; change the beginning sector value

414 <1> ; of the root dir by adding sector offset.

415 00006939 8B4632 <1> mov eax, [esi+LD\_BPB+BPB\_RootClus]

416 0000693C 83E802 <1> sub eax, 2

417 0000693F 7442 <1> jz short short loc\_set\_32bit\_FAT\_total\_sectors

418 <1> ;movzx ebx, byte [esi+LD\_BPB+BPB\_SecPerClust]

419 00006941 8A5E13 <1> mov bl, byte [esi+LD\_BPB+BPB\_SecPerClust]

420 00006944 F7E3 <1> mul ebx

421 00006946 014664 <1> add [esi+LD\_ROOTBegin], eax

422 00006949 EB38 <1> jmp short loc\_set\_32bit\_FAT\_total\_sectors

423 <1> ;

424 <1> loc\_set\_FAT16\_RootDirLoc:

425 0000694B 0FB64616 <1> movzx eax, byte [esi+LD\_BPB+BPB\_NumFATs]

426 0000694F 0FB7561C <1> movzx edx, word [esi+LD\_BPB+BPB\_FATSz16]

427 00006953 F7E2 <1> mul edx

428 00006955 034660 <1> add eax, [esi+LD\_FATBegin]

429 00006958 894664 <1> mov [esi+LD\_ROOTBegin], eax

430 <1> loc\_set\_FAT16\_data\_begin:

431 0000695B 894668 <1> mov [esi+LD\_DATABegin], eax

432 0000695E B820000000 <1> mov eax, 20h ; Size of a directory entry

433 <1> ;movzx edx, word [esi+LD\_BPB+BPB\_RootEntCnt]

434 00006963 668B5617 <1> mov dx, [esi+LD\_BPB+BPB\_RootEntCnt]

435 00006967 F7E2 <1> mul edx

436 <1> ;mov ecx, 511

437 00006969 66B9FF01 <1> mov cx, 511

438 0000696D 01C8 <1> add eax, ecx

439 0000696F 41 <1> inc ecx ; 512

440 00006970 F7F1 <1> div ecx

441 00006972 014668 <1> add [esi+LD\_DATABegin], eax

442 00006975 0FB74619 <1> movzx eax, word [esi+LD\_BPB+BPB\_TotalSec16]

443 00006979 6685C0 <1> test ax, ax

444 0000697C 7405 <1> jz short loc\_set\_32bit\_FAT\_total\_sectors

445 <1> loc\_set\_16bit\_FAT\_total\_sectors:

446 0000697E 894670 <1> mov [esi+LD\_TotalSectors], eax

447 00006981 EB06 <1> jmp short loc\_set\_hd\_FAT\_cluster\_count

448 <1> loc\_set\_32bit\_FAT\_total\_sectors:

449 00006983 8B4626 <1> mov eax, [esi+LD\_BPB+BPB\_TotalSec32]

450 00006986 894670 <1> mov [esi+LD\_TotalSectors], eax

451 <1> loc\_set\_hd\_FAT\_cluster\_count:

452 00006989 8B5668 <1> mov edx, [esi+LD\_DATABegin]

453 0000698C 2B566C <1> sub edx, [esi+LD\_StartSector]

454 0000698F 29D0 <1> sub eax, edx

455 00006991 31D2 <1> xor edx, edx ; 0

456 00006993 0FB64E13 <1> movzx ecx, byte [esi+LD\_BPB+BPB\_SecPerClust]

457 00006997 F7F1 <1> div ecx

458 00006999 894678 <1> mov [esi+LD\_Clusters], eax

459 <1> ; Maximum Valid Cluster Number= EAX +1

460 <1> ; with 2 reserved clusters= EAX +2

461 <1> loc\_set\_hd\_FAT\_fs\_free\_sectors:

462 <1> ;mov dword [esi+LD\_FreeSectors], 0

463 0000699C E859010000 <1> call get\_free\_FAT\_sectors

464 000069A1 7207 <1> jc short loc\_validate\_hd\_FAT\_partition\_retn

465 000069A3 894674 <1> mov [esi+LD\_FreeSectors], eax

466 000069A6 C6467E06 <1> mov byte [esi+LD\_MediaChanged], 6 ; Volume Name Reset

467 <1> ;mov cl, [Last\_DOS\_DiskNo]

468 <1> ;add cl, 'A'

469 <1> ;mov [esi+LD\_FS\_Name], cl

470 <1>

471 <1> loc\_validate\_hd\_FAT\_partition\_retn:

472 000069AA C3 <1> retn

473 <1>

474 <1> validate\_hd\_fs\_partition:

475 <1> ; 09/12/2017

476 <1> ; 13/02/2016

477 <1> ; 10/01/2016 (TRDOS 386 = TRDOS v2.0)

478 <1> ; 29/01/2011

479 <1> ; 23/07/2011

480 <1> ; Input

481 <1> ; DL = Hard/Fixed Disk Drive Number

482 <1> ; ESI = PartitionTable offset

483 <1> ; byte [Last\_DOS\_DiskNo]

484 <1> ; Output

485 <1> ; cf=0 -> Validated

486 <1> ; ESI = Logical dos drv desc. table

487 <1> ; EBX = Singlix FS boot sector buffer

488 <1> ; byte [Last\_DOS\_DiskNo]

489 <1> ; cf=1 -> Not a valid 'Singlix FS' partition

490 <1> ; EAX, EDX, ECX, EDI -> changed

491 <1>

492 <1> ;mov esi, PartitionTable

493 000069AB 8A6604 <1> mov ah, [esi+ptFileSystemID]

494 000069AE 80FCA1 <1> cmp ah, 0A1h ; SINGLIX FS1 (trfs1) partition

495 000069B1 7549 <1> jne short loc\_validate\_hd\_fs\_partition\_stc\_retn

496 <1> loc\_set\_valid\_hd\_fs\_partition\_params:

497 000069B3 FE05[D20C0100] <1> inc byte [Last\_DOS\_DiskNo] ; > 1

498 000069B9 30C0 <1> xor al, al ; mov al, 0

499 <1> ;mov [drv], dl

500 000069BB 29DB <1> sub ebx, ebx ; 0

501 000069BD 8A3D[D20C0100] <1> mov bh, [Last\_DOS\_DiskNo]

502 000069C3 81C300010900 <1> add ebx, Logical\_DOSDisks

503 000069C9 C6430102 <1> mov byte [ebx+LD\_DiskType], 2

504 000069CD 885302 <1> mov [ebx+LD\_PhyDrvNo], dl

505 <1> ;mov [ebx+LD\_FATType], al ; 0

506 <1> ;mov [ebx+LD\_FSType], ah

507 000069D0 66894303 <1> mov [ebx+LD\_FATType], ax

508 <1> ;mov eax, [esi+ptStartSector]

509 <1> ;mov [ebx+LD\_StartSector], eax

510 <1> loc\_hd\_fs\_logical\_drv\_init:

511 000069D4 89DD <1> mov ebp, ebx ; 10/01/2016

512 <1> ;mov dl, [ebx+LD\_PhyDrvNo]

513 000069D6 A0[FF5E0100] <1> mov al, [HD\_LBA\_yes] ; 10/01/2016

514 000069DB 884305 <1> mov [ebx+LD\_LBAYes], al

515 000069DE 89DE <1> mov esi, ebx

516 000069E0 BB[125F0100] <1> mov ebx, DOSBootSectorBuff ; buffer address

517 000069E5 08C0 <1> or al, al

518 000069E7 7515 <1> jnz short loc\_hd\_fs\_drv\_init\_load\_bs\_lba

519 <1> loc\_hd\_fs\_drv\_init\_load\_bs\_chs:

520 000069E9 8A7601 <1> mov dh, [esi+ptBeginHead]

521 000069EC 668B4E02 <1> mov cx, [esi+ptBeginSector]

522 000069F0 66B80102 <1> mov ax, 0201h ; Read 1 sector

523 <1> ;mov ebx, DOSBootSectorBuff

524 000069F4 E80DD8FFFF <1> call int13h

525 000069F9 7311 <1> jnc short loc\_hd\_drv\_fs\_boot\_validation

526 <1> loc\_validate\_hd\_fs\_partition\_err\_retn:

527 000069FB C3 <1> retn

528 <1> loc\_validate\_hd\_fs\_partition\_stc\_retn:

529 000069FC F9 <1> stc

530 000069FD C3 <1> retn

531 <1> loc\_hd\_fs\_drv\_init\_load\_bs\_lba:

532 <1> ; DL = Physical drive number

533 <1> ;mov esi, ebx

534 000069FE 8B4E08 <1> mov ecx, [esi+ptStartSector] ; sector number

535 <1> ;mov ebx, DOSBootSectorBuff ; buffer address

536 <1> ; LBA read/write (with private LBA function)

537 <1> ;((Retro UNIX 386 v1 - DISK I/O code by Erdogan Tan))

538 <1> ; dl = physical drive number (0,1, 80h, 81h, 82h, 83h)

539 00006A01 B41B <1> mov ah, 1Bh ; LBA read

540 00006A03 B001 <1> mov al, 1 ; sector count

541 00006A05 E8FCD7FFFF <1> call int13h

542 00006A0A 72EF <1> jc short loc\_validate\_hd\_fs\_partition\_err\_retn

543 <1> loc\_hd\_drv\_fs\_boot\_validation:

544 <1> ;mov esi, DOSBootSectorBuff

545 00006A0C 89DE <1> mov esi, ebx ; Boot sector buffer

546 00006A0E 6681BEFE01000055AA <1> cmp word [esi+BS\_Validation], 0AA55h

547 00006A17 75E3 <1> jne short loc\_validate\_hd\_fs\_partition\_stc\_retn

548 <1> ;

549 <1> ;Singlix FS Extensions to TR-DOS (7/6/2009)

550 00006A19 66817E035346 <1> cmp word [esi+bs\_FS\_Identifier], 'SF'

551 00006A1F 75DB <1> jne short loc\_validate\_hd\_fs\_partition\_stc\_retn

552 <1> ;'A1h' check is not necessary

553 <1> ; if 'FS' check is passed as OK/Yes.

554 00006A21 807E09A1 <1> cmp byte [esi+bs\_FS\_PartitionID], 0A1h

555 00006A25 75D5 <1> jne short loc\_validate\_hd\_fs\_partition\_stc\_retn

556 <1> ;

557 00006A27 89EF <1> mov edi, ebp ; 10/01/2016

558 <1> ;

559 00006A29 8A462D <1> mov al, byte [esi+bs\_FS\_LBA\_Ready]

560 00006A2C 884705 <1> mov [edi+LD\_FS\_LBAYes], al

561 <1> ;

562 <1> ; 03/01/2010 CHS -> DOS FAT/BPB compatibility fix

563 00006A2F 8A4608 <1> mov al, [esi+bs\_FS\_MediaAttrib]

564 00006A32 884706 <1> mov byte [edi+LD\_FS\_MediaAttrib], al

565 <1> ;

566 00006A35 8A460A <1> mov al, [esi+bs\_FS\_VersionMaj]

567 00006A38 884707 <1> mov [edi+LD\_FS\_VersionMajor], al

568 <1> ;

569 00006A3B 668B4606 <1> mov ax, [esi+bs\_FS\_BytesPerSec]

570 00006A3F 66894711 <1> mov [edi+LD\_FS\_BytesPerSec], ax

571 00006A43 8A462E <1> mov al, [esi+bs\_FS\_SecPerTrack]

572 00006A46 30E4 <1> xor ah, ah ; 09/12/2017

573 00006A48 6689471E <1> mov [edi+LD\_FS\_SecPerTrack], ax

574 00006A4C 8A462F <1> mov al, [esi+bs\_FS\_Heads]

575 00006A4F 66894720 <1> mov [edi+LD\_FS\_NumHeads], ax

576 <1> ;

577 00006A53 8B4628 <1> mov eax, [esi+bs\_FS\_UnDelDirD]

578 00006A56 894722 <1> mov [edi+LD\_FS\_UnDelDirD], eax

579 00006A59 8B5618 <1> mov edx, [esi+bs\_FS\_MATLocation]

580 00006A5C 89570C <1> mov [edi+LD\_FS\_MATLocation], edx

581 00006A5F 8B461C <1> mov eax, [esi+bs\_FS\_RootDirD]

582 00006A62 894708 <1> mov [edi+LD\_FS\_RootDirD], eax

583 00006A65 8B460C <1> mov eax, [esi+bs\_FS\_BeginSector]

584 00006A68 89476C <1> mov [edi+LD\_FS\_BeginSector], eax

585 00006A6B 8B4710 <1> mov eax, [edi+bs\_FS\_VolumeSize]

586 00006A6E 894770 <1> mov [edi+LD\_FS\_VolumeSize], eax

587 <1> ;

588 00006A71 89D0 <1> mov eax, edx ; [edi+LD\_FS\_MATLocation]

589 00006A73 03476C <1> add eax, [edi+LD\_FS\_BeginSector]

590 00006A76 89FE <1> mov esi, edi

591 <1> mread\_hd\_fs\_MAT\_sector:

592 <1> ;mov ebx, DOSBootSectorBuff

593 00006A78 B901000000 <1> mov ecx, 1

594 00006A7D E8568D0000 <1> call disk\_read

595 00006A82 7248 <1> jc short loc\_validate\_hd\_fs\_partition\_retn

596 <1> ; EDI will not be changed

597 00006A84 89DE <1> mov esi, ebx

598 <1> use\_hdfs\_mat\_sector\_params:

599 00006A86 8B460C <1> mov eax, [esi+FS\_MAT\_DATLocation]

600 00006A89 894714 <1> mov [edi+LD\_FS\_DATLocation], eax

601 00006A8C 8B4610 <1> mov eax, [esi+FS\_MAT\_DATScount]

602 00006A8F 894718 <1> mov [edi+LD\_FS\_DATSectors], eax

603 00006A92 8B4614 <1> mov eax, [esi+FS\_MAT\_FreeSectors]

604 00006A95 894774 <1> mov [edi+LD\_FS\_FreeSectors], eax

605 00006A98 8B4618 <1> mov eax, [esi+FS\_MAT\_FirstFreeSector]

606 00006A9B 894778 <1> mov [edi+LD\_FS\_FirstFreeSector], eax

607 00006A9E 8B4708 <1> mov eax, [edi+LD\_FS\_RootDirD]

608 00006AA1 03476C <1> add eax, [edi+LD\_FS\_BeginSector]

609 00006AA4 89FE <1> mov esi, edi

610 <1> read\_hd\_fs\_RDT\_sector:

611 00006AA6 BB[125F0100] <1> mov ebx, DOSBootSectorBuff

612 <1> ;mov ecx, 1

613 00006AAB B101 <1> mov cl, 1

614 00006AAD E8268D0000 <1> call disk\_read

615 00006AB2 7218 <1> jc short loc\_validate\_hd\_fs\_partition\_retn

616 <1> ; EDI will not be changed

617 00006AB4 89DE <1> mov esi, ebx

618 <1> use\_hdfs\_RDT\_sector\_params:

619 00006AB6 8B461C <1> mov eax, [esi+FS\_RDT\_VolumeSerialNo]

620 00006AB9 894728 <1> mov [edi+LD\_FS\_VolumeSerial], eax

621 00006ABC 57 <1> push edi

622 <1> ;mov ecx, 16

623 00006ABD B110 <1> mov cl, 16

624 00006ABF 83C640 <1> add esi, FS\_RDT\_VolumeName

625 00006AC2 83C72C <1> add edi, LD\_FS\_VolumeName

626 00006AC5 F3A5 <1> rep movsd ; 64 bytes

627 00006AC7 5E <1> pop esi

628 <1> ; Volume Name Reset

629 00006AC8 C6467E06 <1> mov byte [esi+LD\_FS\_MediaChanged], 6

630 <1> ;

631 <1> ;mov cl, [Last\_DOS\_DiskNo]

632 <1> ;add cl, 'A'

633 <1> ;mov [esi+LD\_FS\_Name], cl

634 <1>

635 <1> loc\_validate\_hd\_fs\_partition\_retn:

636 00006ACC C3 <1> retn

637 <1>

638 <1> load\_masterboot:

639 <1> ; 10/01/2016 (TRDOS 386 = TRDOS v2.0)

640 <1> ; 2005 - 2011

641 <1> ; input -> DL = drive number

642 00006ACD B40D <1> mov ah, 0Dh ; Alternate disk reset

643 00006ACF E832D7FFFF <1> call int13h

644 00006AD4 7301 <1> jnc short pass\_reset\_error

645 <1> harddisk\_error:

646 00006AD6 C3 <1> retn

647 <1> pass\_reset\_error:

648 00006AD7 BB[FE5A0100] <1> mov ebx, MasterBootBuff

649 00006ADC 66B80102 <1> mov ax, 0201h

650 00006AE0 66B90100 <1> mov cx, 1

651 00006AE4 30F6 <1> xor dh, dh

652 00006AE6 E81BD7FFFF <1> call int13h

653 00006AEB 72E9 <1> jc short harddisk\_error

654 <1> ;

655 00006AED 66813D[FC5C0100]55- <1> cmp word [MBIDCode], 0AA55h

655 00006AF5 AA <1>

656 00006AF6 7401 <1> je short load\_masterboot\_ok

657 00006AF8 F9 <1> stc

658 <1> load\_masterboot\_ok:

659 00006AF9 C3 <1> retn

660 <1>

661 <1> get\_free\_FAT\_sectors:

662 <1> ; 21/12/2017

663 <1> ; 29/02/2016

664 <1> ; 13/02/2016

665 <1> ; 04/02/2016

666 <1> ; 07/01/2016 (TRDOS 386 = TRDOS v2.0)

667 <1> ; 11/07/2010

668 <1> ; 21/06/2009

669 <1> ; INPUT: ESI = Logical DOS Drive Description Table address

670 <1> ; OUTPUT: STC => Error

671 <1> ; cf = 0 and EAX = Free FAT sectors

672 <1> ; Also, related parameters and FAT buffer will be reset and updated

673 <1>

674 00006AFA 31C0 <1> xor eax, eax

675 <1> ;mov [esi+LD\_FreeSectors], eax ; Reset

676 <1>

677 00006AFC 807E0302 <1> cmp byte [esi+LD\_FATType], 2

678 00006B00 7654 <1> jna short loc\_gfc\_get\_fat\_free\_clusters

679 <1>

680 <1> ; 29/02/2016

681 00006B02 48 <1> dec eax ; 0FFFFFFFFh

682 00006B03 89463A <1> mov [esi+LD\_BPB+BPB\_Reserved], eax ; Free cluster count (reset)

683 00006B06 89463E <1> mov [esi+LD\_BPB+BPB\_Reserved+4], eax ; First Free Cluster (reset)

684 00006B09 40 <1> inc eax ; 0

685 <1> ;

686 00006B0A 668B4636 <1> mov ax, [esi+LD\_BPB+BPB\_FSInfo]

687 00006B0E 03466C <1> add eax, [esi+LD\_StartSector]

688 <1>

689 00006B11 BB[125F0100] <1> mov ebx, DOSBootSectorBuff

690 00006B16 B901000000 <1> mov ecx, 1

691 00006B1B E8B88C0000 <1> call disk\_read

692 00006B20 7301 <1> jnc short loc\_gfc\_check\_fsinfo\_signs

693 <1> retn\_gfc\_get\_fsinfo\_sec:

694 00006B22 C3 <1> retn

695 <1>

696 <1> loc\_gfc\_check\_fsinfo\_signs:

697 00006B23 BB[125F0100] <1> mov ebx, DOSBootSectorBuff ; 13/02/2016

698 00006B28 813B52526141 <1> cmp dword [ebx], 41615252h

699 00006B2E 7524 <1> jne short retn\_gfc\_get\_fsinfo\_stc

700 <1> ;add ebx, 484

701 <1> ;cmp dword [ebx], 61417272h

702 00006B30 81BBE4010000727241- <1> cmp dword [ebx+484], 61417272h

702 00006B39 61 <1>

703 00006B3A 7518 <1> jne short retn\_gfc\_get\_fsinfo\_stc

704 <1> ;add ebx, 4

705 <1> ;mov eax, [ebx]

706 00006B3C 8B83E8010000 <1> mov eax, [ebx+488]

707 <1> ; 29/02/2016

708 00006B42 89463A <1> mov [esi+LD\_BPB+BPB\_Reserved], eax ; Free cluster count

709 00006B45 8B93EC010000 <1> mov edx, [ebx+492]

710 00006B4B 89463E <1> mov [esi+LD\_BPB+BPB\_Reserved+4], eax ; First Free Cluster

711 <1> ; 21/12/2017

712 00006B4E 89C3 <1> mov ebx, eax ; (initial value = 0FFFFFFFFh)

713 00006B50 43 <1> inc ebx ; 0FFFFFFFFh -> 0

714 00006B51 7513 <1> jnz short short retn\_from\_get\_free\_fat32\_clusters

715 00006B53 C3 <1> retn

716 <1>

717 <1> retn\_gfc\_get\_fsinfo\_stc:

718 00006B54 F9 <1> stc

719 00006B55 C3 <1> retn

720 <1>

721 <1> loc\_gfc\_get\_fat\_free\_clusters:

722 <1> ;mov eax, 2

723 00006B56 B002 <1> mov al, 2

724 <1> ;mov [FAT\_CurrentCluster], eax

725 <1> loc\_gfc\_loop\_get\_next\_cluster:

726 00006B58 E8EB4F0000 <1> call get\_next\_cluster

727 00006B5D 730E <1> jnc short loc\_gfc\_free\_fat\_clusters\_cont

728 00006B5F 21C0 <1> and eax, eax

729 00006B61 7411 <1> jz short loc\_gfc\_pass\_inc\_free\_cluster\_count

730 <1>

731 <1> retn\_from\_get\_free\_fat\_clusters:

732 00006B63 8B4674 <1> mov eax, [esi+LD\_FreeSectors] ; Free clusters !

733 <1> retn\_from\_get\_free\_fat32\_clusters:

734 00006B66 0FB65E13 <1> movzx ebx, byte [esi+LD\_BPB+BPB\_SecPerClust]

735 00006B6A F7E3 <1> mul ebx

736 <1> ;mov [esi+LD\_FreeSectors], eax ; Free sectors

737 <1> retn\_get\_free\_sectors\_calc:

738 00006B6C C3 <1> retn

739 <1>

740 <1> loc\_gfc\_free\_fat\_clusters\_cont:

741 00006B6D 09C0 <1> or eax, eax

742 00006B6F 7503 <1> jnz short loc\_gfc\_pass\_inc\_free\_cluster\_count

743 00006B71 FF4674 <1> inc dword [esi+LD\_FreeSectors] ; Free clusters !

744 <1>

745 <1> loc\_gfc\_pass\_inc\_free\_cluster\_count:

746 <1> ;mov eax, [FAT\_CurrentCluster]

747 00006B74 89C8 <1> mov eax, ecx ; [FAT\_CurrentCluster]

748 00006B76 3B4678 <1> cmp eax, [esi+LD\_Clusters]

749 00006B79 77E8 <1> ja short retn\_from\_get\_free\_fat\_clusters

750 00006B7B 40 <1> inc eax

751 <1> ;mov [FAT\_CurrentCluster], eax

752 00006B7C EBDA <1> jmp short loc\_gfc\_loop\_get\_next\_cluster

753 <1>

754 <1> floppy\_drv\_init:

755 <1> ; 09/12/2017

756 <1> ; 06/07/2016

757 <1> ; 10/01/2016 (TRDOS 386 = TRDOS v2.0)

758 <1> ; 24/07/2011

759 <1> ; 04/07/2009

760 <1> ; INPUT ->

761 <1> ; DL = Drive number (0,1)

762 <1> ; OUTPUT ->

763 <1> ; BL = drive name

764 <1> ; BH = drive number

765 <1> ; ESI = Logical DOS drv description table

766 <1> ; EAX = Volume serial number

767 <1>

768 00006B7E BE[F65C0000] <1> mov esi, fd0\_type ; 10/01/2016

769 00006B83 BF00010900 <1> mov edi, Logical\_DOSDisks

770 00006B88 08D2 <1> or dl, dl

771 00006B8A 7407 <1> jz short loc\_drv\_init\_fd0\_fd1

772 00006B8C 81C700010000 <1> add edi, 100h

773 00006B92 46 <1> inc esi ; fd1\_type ; 10/01/2016

774 <1> loc\_drv\_init\_fd0\_fd1:

775 00006B93 C6477E00 <1> mov byte [edi+LD\_MediaChanged], 0

776 00006B97 803E01 <1> cmp byte [esi], 1 ; type (>0 if it is existing)

777 <1> ; 4 = 1.44 MB, 80 track, 3 1/2"

778 00006B9A 7221 <1> jb short read\_fd\_boot\_sector\_retn

779 00006B9C 885702 <1> mov [edi+LD\_PhyDrvNo], dl

780 <1> read\_fd\_boot\_sector:

781 00006B9F 30F6 <1> xor dh, dh

782 00006BA1 B904000000 <1> mov ecx, 4 ; Retry Count

783 <1> read\_fd\_boot\_sector\_again:

784 00006BA6 51 <1> push ecx

785 <1> ;mov cx, 1

786 00006BA7 B101 <1> mov cl, 1

787 00006BA9 66B80102 <1> mov ax, 0201h ; Read 1 sector

788 00006BAD BB[125F0100] <1> mov ebx, DOSBootSectorBuff

789 00006BB2 E84FD6FFFF <1> call int13h

790 00006BB7 59 <1> pop ecx

791 00006BB8 7304 <1> jnc short use\_fd\_boot\_sector\_params

792 00006BBA E2EA <1> loop read\_fd\_boot\_sector\_again

793 <1>

794 <1> read\_fd\_boot\_sector\_stc\_retn:

795 00006BBC F9 <1> stc

796 <1> read\_fd\_boot\_sector\_retn:

797 00006BBD C3 <1> retn

798 <1>

799 <1> use\_fd\_boot\_sector\_params:

800 <1> ;mov esi, DOSBootSectorBuff

801 00006BBE 89DE <1> mov esi, ebx

802 00006BC0 6681BEFE01000055AA <1> cmp word [esi+BS\_Validation], 0AA55h

803 00006BC9 75F1 <1> jne short read\_fd\_boot\_sector\_stc\_retn

804 00006BCB 66817E035346 <1> cmp word [esi+bs\_FS\_Identifier], 'SF'

805 00006BD1 0F85A2000000 <1> jne use\_fd\_fatfs\_boot\_sector\_params

806 <1> ;

807 00006BD7 8A462D <1> mov al, [esi+bs\_FS\_LBA\_Ready]

808 00006BDA 884705 <1> mov [edi+LD\_FS\_LBAYes], al

809 <1> ;

810 <1> ; 03/01/2010 CHS -> DOS FAT/BPB compatibility fix

811 00006BDD 8A4608 <1> mov al, [esi+bs\_FS\_MediaAttrib]

812 00006BE0 884706 <1> mov [edi+LD\_FS\_MediaAttrib], al

813 <1> ;

814 00006BE3 8A460A <1> mov al, [esi+bs\_FS\_VersionMaj]

815 00006BE6 884707 <1> mov byte [edi+LD\_FS\_VersionMajor], al

816 00006BE9 668B4606 <1> mov ax, [esi+bs\_FS\_BytesPerSec]

817 00006BED 66894711 <1> mov [edi+LD\_FS\_BytesPerSec], ax

818 00006BF1 8A462E <1> mov al, [esi+bs\_FS\_SecPerTrack]

819 00006BF4 28E4 <1> sub ah, ah ; 09/12/2017

820 00006BF6 6689471E <1> mov [edi+LD\_FS\_SecPerTrack], ax

821 00006BFA 8A462F <1> mov al, [esi+bs\_FS\_Heads]

822 00006BFD 66894720 <1> mov [edi+LD\_FS\_NumHeads], ax

823 <1> ;

824 00006C01 8B4628 <1> mov eax, [esi+bs\_FS\_UnDelDirD]

825 00006C04 894722 <1> mov [edi+LD\_FS\_UnDelDirD], eax

826 00006C07 8B4618 <1> mov eax, [esi+bs\_FS\_MATLocation]

827 00006C0A 89470C <1> mov [edi+LD\_FS\_MATLocation], eax

828 00006C0D 8B461C <1> mov eax, [esi+bs\_FS\_RootDirD]

829 00006C10 894708 <1> mov [edi+LD\_FS\_RootDirD], eax

830 00006C13 8B460C <1> mov eax, [esi+bs\_FS\_BeginSector]

831 00006C16 89476C <1> mov [edi+LD\_FS\_BeginSector], eax

832 00006C19 8B4610 <1> mov eax, [esi+bs\_FS\_VolumeSize]

833 00006C1C 894770 <1> mov [edi+LD\_FS\_VolumeSize], eax

834 <1> ;

835 00006C1F 89FE <1> mov esi, edi

836 00006C21 8B460C <1> mov eax, [esi+LD\_FS\_MATLocation]

837 <1> ;add eax, [edi+LD\_FS\_BeginSector]

838 <1> read\_fd\_MAT\_sector\_again:

839 <1> ;mov ebx, DOSBootSectorBuff

840 <1> ;mov ecx, 1

841 00006C24 B101 <1> mov cl, 1

842 00006C26 E8B38B0000 <1> call chs\_read

843 00006C2B 89DE <1> mov esi, ebx

844 00006C2D 7301 <1> jnc short use\_fdfs\_mat\_sector\_params

845 <1> ;jmp short read\_fd\_boot\_sector\_retn

846 00006C2F C3 <1> retn

847 <1> use\_fdfs\_mat\_sector\_params:

848 00006C30 8B460C <1> mov eax, [esi+FS\_MAT\_DATLocation]

849 00006C33 894714 <1> mov [edi+LD\_FS\_DATLocation], eax

850 00006C36 8B4610 <1> mov eax, [esi+FS\_MAT\_DATScount]

851 00006C39 894718 <1> mov [edi+LD\_FS\_DATSectors], eax

852 00006C3C 8B4714 <1> mov eax, [edi+FS\_MAT\_FreeSectors]

853 00006C3F 894774 <1> mov [edi+LD\_FS\_FreeSectors], eax

854 00006C42 8B4618 <1> mov eax, [esi+FS\_MAT\_FirstFreeSector]

855 00006C45 894778 <1> mov [edi+LD\_FS\_FirstFreeSector], eax

856 <1> ;

857 00006C48 89FE <1> mov esi, edi

858 00006C4A 8B4608 <1> mov eax, [esi+LD\_FS\_RootDirD]

859 <1> read\_fd\_RDT\_sector\_again:

860 <1> ;mov ebx, DOSBootSectorBuff

861 <1> ;mov cx, 1

862 00006C4D B101 <1> mov cl, 1

863 00006C4F E88A8B0000 <1> call chs\_read

864 00006C54 89DE <1> mov esi, ebx

865 00006C56 7220 <1> jc short read\_fd\_RDT\_sector\_retn

866 <1> use\_fdfs\_RDT\_sector\_params:

867 00006C58 8B461C <1> mov eax, [esi+FS\_RDT\_VolumeSerialNo]

868 00006C5B 894728 <1> mov [edi+LD\_FS\_VolumeSerial], eax

869 00006C5E 57 <1> push edi

870 <1> ;mov ecx, 16

871 00006C5F B110 <1> mov cl, 16

872 00006C61 83C640 <1> add esi, FS\_RDT\_VolumeName

873 00006C64 83C72C <1> add edi, LD\_FS\_VolumeName

874 00006C67 F3A5 <1> rep movsd ; 64 bytes

875 00006C69 5E <1> pop esi

876 00006C6A C6460300 <1> mov byte [esi+LD\_FATType], 0

877 00006C6E C64604A1 <1> mov byte [esi+LD\_FSType], 0A1h

878 00006C72 E9A5000000 <1> jmp loc\_cont\_use\_fd\_boot\_sector\_params

879 <1>

880 <1> read\_fd\_RDT\_sector\_stc\_retn:

881 00006C77 F9 <1> stc

882 <1> read\_fd\_RDT\_sector\_retn:

883 00006C78 C3 <1> retn

884 <1>

885 <1> use\_fd\_fatfs\_boot\_sector\_params:

886 00006C79 807E2629 <1> cmp byte [esi+BS\_BootSig], 29h

887 00006C7D 75F8 <1> jne short read\_fd\_RDT\_sector\_stc\_retn

888 00006C7F 807E15F0 <1> cmp byte [esi+BPB\_Media], 0F0h

889 00006C83 72F3 <1> jb short read\_fd\_RDT\_sector\_retn

890 00006C85 57 <1> push edi

891 00006C86 83C706 <1> add edi, LD\_BPB

892 <1> ;mov ecx, 16

893 00006C89 B110 <1> mov cl, 16

894 00006C8B F3A5 <1> rep movsd ; 64 bytes

895 00006C8D 5E <1> pop esi

896 00006C8E 31C0 <1> xor eax, eax

897 00006C90 89466C <1> mov [esi+LD\_StartSector], eax ; 0

898 00006C93 668B461C <1> mov ax, [esi+LD\_BPB+BPB\_FATSz16]

899 00006C97 8A4E16 <1> mov cl, [esi+LD\_BPB+BPB\_NumFATs]

900 00006C9A F7E1 <1> mul ecx

901 <1> ; edx = 0 !

902 00006C9C 668B5614 <1> mov dx, [esi+LD\_BPB+BPB\_RsvdSecCnt]

903 00006CA0 66895660 <1> mov [esi+LD\_FATBegin], dx

904 <1> ;add eax, edx

905 00006CA4 6601D0 <1> add ax, dx

906 00006CA7 894664 <1> mov [esi+LD\_ROOTBegin], eax

907 00006CAA 894668 <1> mov [esi+LD\_DATABegin], eax

908 00006CAD 668B5617 <1> mov dx, [esi+LD\_BPB+BPB\_RootEntCnt]

909 <1> ;;shl edx, 5 ; \* 32 (Size of a directory entry)

910 <1> ;shl dx, 5

911 <1> ;;add edx, 511

912 <1> ;add dx, 511

913 <1> ;;shr edx, 9 ; edx = ((edx\*32)+511) / 512

914 <1> ;shr dx, 9

915 00006CB1 6683C20F <1> add dx, 15 ; 06/07/2016 (+(512/32)-1)

916 00006CB5 66C1EA04 <1> shr dx, 4 ; / 16 (==16 entries per sector)

917 00006CB9 015668 <1> add [esi+LD\_DATABegin], edx ; + rd sectors

918 <1> ;movzx eax, word [esi+LD\_BPB+BPB\_TotalSec16]

919 00006CBC 668B4619 <1> mov ax, [esi+LD\_BPB+BPB\_TotalSec16]

920 00006CC0 894670 <1> mov [esi+LD\_TotalSectors], eax

921 00006CC3 2B4668 <1> sub eax, [esi+LD\_DATABegin]

922 <1> ;movzx ecx, byte [esi+LD\_BPB+BPB\_SecPerClust]

923 00006CC6 8A4E13 <1> mov cl, [esi+LD\_BPB+BPB\_SecPerClust]

924 00006CC9 80F901 <1> cmp cl, 1

925 00006CCC 7605 <1> jna short save\_fd\_fatfs\_cluster\_count

926 <1> ;sub edx, edx

927 00006CCE 6629D2 <1> sub dx, dx ; 0

928 <1> ;sub dl, dl ; 06/07/2016

929 00006CD1 F7F1 <1> div ecx

930 <1> save\_fd\_fatfs\_cluster\_count:

931 00006CD3 894678 <1> mov [esi+LD\_Clusters], eax

932 <1>

933 <1> ; Maximum Valid Cluster Number = EAX +1

934 <1> ; with 2 reserved clusters= EAX +2

935 <1>

936 <1> reset\_FAT\_buffer\_decriptors:

937 00006CD6 29C0 <1> sub eax, eax ; 0

938 00006CD8 A2[16610100] <1> mov [FAT\_BuffValidData], al ; 0

939 00006CDD A2[17610100] <1> mov [FAT\_BuffDrvName], al ; 0

940 00006CE2 A3[1A610100] <1> mov [FAT\_BuffSector], eax ; 0

941 <1>

942 <1> read\_fd\_FAT\_sectors:

943 00006CE7 BB001C0900 <1> mov ebx, FAT\_Buffer

944 00006CEC 668B4614 <1> mov ax, [esi+LD\_BPB+BPB\_RsvdSecCnt]

945 <1> ;mov ecx, 3

946 00006CF0 B103 <1> mov cl, 3 ; 3 sectors

947 00006CF2 E8E78A0000 <1> call chs\_read

948 00006CF7 7240 <1> jc short read\_fd\_FAT\_sectors\_retn

949 <1> use\_fd\_FAT\_sectors:

950 00006CF9 8A4602 <1> mov al, [esi+LD\_PhyDrvNo]

951 00006CFC 0441 <1> add al, 'A'

952 00006CFE A2[17610100] <1> mov [FAT\_BuffDrvName], al

953 00006D03 C605[16610100]01 <1> mov byte [FAT\_BuffValidData], 1

954 00006D0A E82B000000 <1> call fd\_init\_calculate\_free\_clusters

955 00006D0F 7228 <1> jc short read\_fd\_FAT\_sectors\_retn

956 <1>

957 <1> loc\_use\_fd\_boot\_sector\_params\_FAT:

958 00006D11 C6460301 <1> mov byte [esi+LD\_FATType], 1 ; FAT 12

959 00006D15 C6460401 <1> mov byte [esi+LD\_FSType], 1

960 00006D19 8B462D <1> mov eax, [esi+LD\_BPB+VolumeID]

961 <1> loc\_cont\_use\_fd\_boot\_sector\_params:

962 00006D1C 8A7E02 <1> mov bh, [esi+LD\_PhyDrvNo]

963 00006D1F 887E7D <1> mov [esi+LD\_DParamEntry], bh

964 00006D22 88FB <1> mov bl, bh

965 00006D24 80C341 <1> add bl, 'A'

966 00006D27 881E <1> mov byte [esi+LD\_Name], bl

967 00006D29 C6460101 <1> mov byte [esi+LD\_DiskType], 1

968 00006D2D C6460500 <1> mov byte [esi+LD\_LBAYes], 0

969 00006D31 C6467C00 <1> mov byte [esi+LD\_PartitionEntry], 0

970 00006D35 C6467E06 <1> mov byte [esi+LD\_MediaChanged], 6 ; Volume Name Reset

971 <1>

972 <1> read\_fd\_FAT\_sectors\_retn:

973 00006D39 C3 <1> retn

974 <1>

975 <1> fd\_init\_calculate\_free\_clusters:

976 <1> ; 09/12/2017

977 <1> ; 10/01/2016 (TRDOS 386 = TRDOS v2.0)

978 <1> ; 04/07/2009

979 <1> ; INPUT ->

980 <1> ; ESI = Logical DOS drive description table address

981 <1> ; OUTPUT ->

982 <1> ; [ESI+LD\_FreeSectors] will be set

983 <1>

984 00006D3A 29C0 <1> sub eax, eax

985 00006D3C 894674 <1> mov [esi+LD\_FreeSectors], eax ; 0

986 00006D3F B002 <1> mov al, 2 ; eax = 2

987 <1>

988 <1> fd\_init\_loop\_get\_next\_cluster:

989 00006D41 E830000000 <1> call fd\_init\_get\_next\_cluster

990 00006D46 722D <1> jc short fd\_init\_calculate\_free\_clusters\_retn

991 <1>

992 <1> fd\_init\_free\_fat\_clusters:

993 <1> ;cmp eax, 0

994 <1> ;ja short fd\_init\_pass\_inc\_free\_cluster\_count

995 <1> ;and eax, eax

996 <1> ;jnz short fd\_init\_pass\_inc\_free\_cluster\_count

997 00006D48 6621C0 <1> and ax, ax

998 00006D4B 7504 <1> jnz short fd\_init\_pass\_inc\_free\_cluster\_count

999 <1> ;inc dword [esi+LD\_FreeSectors]

1000 00006D4D 66FF4674 <1> inc word [esi+LD\_FreeSectors]

1001 <1>

1002 <1> fd\_init\_pass\_inc\_free\_cluster\_count:

1003 <1> ;mov eax, [FAT\_CurrentCluster]

1004 00006D51 66A1[12610100] <1> mov ax, [FAT\_CurrentCluster]

1005 <1> ;cmp eax, [esi+LD\_Clusters]

1006 00006D57 663B4678 <1> cmp ax, [esi+LD\_Clusters]

1007 00006D5B 7704 <1> ja short short retn\_from\_fd\_init\_calculate\_free\_clusters

1008 <1> ;inc eax

1009 00006D5D 6640 <1> inc ax

1010 00006D5F EBE0 <1> jmp short fd\_init\_loop\_get\_next\_cluster

1011 <1>

1012 <1> retn\_from\_fd\_init\_calculate\_free\_clusters:

1013 00006D61 8A4613 <1> mov al, [esi+LD\_BPB+BPB\_SecPerClust]

1014 00006D64 3C01 <1> cmp al, 1

1015 00006D66 760D <1> jna short fd\_init\_calculate\_free\_clusters\_retn

1016 <1> ;movzx eax, al

1017 00006D68 30E4 <1> xor ah, ah ; 09/12/2017

1018 <1> ;mov ecx, [esi+LD\_FreeSectors]

1019 00006D6A 668B4E74 <1> mov cx, [esi+LD\_FreeSectors] ; Count of free clusters

1020 <1> ;mul ecx

1021 00006D6E 66F7E1 <1> mul cx

1022 <1> ;mov [esi+LD\_FreeSectors], eax

1023 00006D71 66894674 <1> mov [esi+LD\_FreeSectors], ax

1024 <1> fd\_init\_calculate\_free\_clusters\_retn:

1025 00006D75 C3 <1> retn

1026 <1>

1027 <1> fd\_init\_get\_next\_cluster:

1028 <1> ; 04/02/2016

1029 <1> ; 02/02/2016

1030 <1> ; 10/01/2016 (TRDOS 386 = TRDOS v2.0)

1031 <1> ; 04/07/2009

1032 <1> ; INPUT ->

1033 <1> ; EAX = Current cluster

1034 <1> ; ESI = Logical DOS drive description table address

1035 <1> ; EDX = 0

1036 <1> ; OUTPUT ->

1037 <1> ; EAX = Next cluster

1038 <1>

1039 00006D76 A3[12610100] <1> mov [FAT\_CurrentCluster], eax

1040 <1> fd\_init\_get\_next\_cluster\_readnext:

1041 00006D7B 29D2 <1> sub edx, edx ; 0

1042 00006D7D BB00040000 <1> mov ebx, 1024 ; 400h

1043 00006D82 F7F3 <1> div ebx

1044 <1> ; EAX = Count of 3 FAT sectors

1045 <1> ; EDX = Buffer entry index

1046 00006D84 89C1 <1> mov ecx, eax

1047 <1> ;mov eax, 3

1048 00006D86 B003 <1> mov al, 3

1049 00006D88 F7E2 <1> mul edx ; Multiply by 3

1050 00006D8A 66D1E8 <1> shr ax, 1 ; Divide by 2

1051 00006D8D 89C3 <1> mov ebx, eax ; Buffer byte offset

1052 00006D8F 81C3001C0900 <1> add ebx, FAT\_Buffer

1053 00006D95 89C8 <1> mov eax, ecx

1054 <1> ;mov edx, 3

1055 00006D97 66BA0300 <1> mov dx, 3

1056 00006D9B F7E2 <1> mul edx

1057 <1> ; EAX = FAT Beginning Sector

1058 <1> ; EDX = 0

1059 00006D9D 8A0E <1> mov cl, [esi+LD\_Name]

1060 <1> ;cmp byte [FAT\_BuffValidData], 0

1061 <1> ;jna short fd\_init\_load\_FAT\_sectors0

1062 00006D9F 3A0D[17610100] <1> cmp cl, [FAT\_BuffDrvName]

1063 00006DA5 751E <1> jne short fd\_init\_load\_FAT\_sectors0

1064 00006DA7 3B05[1A610100] <1> cmp eax, [FAT\_BuffSector]

1065 00006DAD 751C <1> jne short fd\_init\_load\_FAT\_sectors1

1066 <1> ;mov eax, [FAT\_CurrentCluster]

1067 00006DAF A0[12610100] <1> mov al, [FAT\_CurrentCluster]

1068 <1> ;shr eax, 1

1069 00006DB4 D0E8 <1> shr al, 1

1070 00006DB6 668B03 <1> mov ax, [ebx]

1071 00006DB9 7306 <1> jnc short fd\_init\_gnc\_even

1072 00006DBB 66C1E804 <1> shr ax, 4

1073 <1> fd\_init\_gnc\_clc\_retn:

1074 00006DBF F8 <1> clc

1075 00006DC0 C3 <1> retn

1076 <1>

1077 <1> fd\_init\_gnc\_even:

1078 00006DC1 80E40F <1> and ah, 0Fh

1079 00006DC4 C3 <1> retn

1080 <1>

1081 <1> fd\_init\_load\_FAT\_sectors0:

1082 00006DC5 880D[17610100] <1> mov [FAT\_BuffDrvName], cl

1083 <1> fd\_init\_load\_FAT\_sectors1:

1084 00006DCB C605[16610100]00 <1> mov byte [FAT\_BuffValidData], 0

1085 00006DD2 A3[1A610100] <1> mov [FAT\_BuffSector], eax

1086 00006DD7 034660 <1> add eax, [esi+LD\_FATBegin]

1087 00006DDA BB001C0900 <1> mov ebx, FAT\_Buffer

1088 <1> ;movzx ecx, word [esi+LD\_BPB+BPB\_FATSz16]

1089 00006DDF 668B4E1C <1> mov cx, [esi+LD\_BPB+BPB\_FATSz16]

1090 00006DE3 662B0D[1A610100] <1> sub cx, [FAT\_BuffSector]

1091 <1> ;cmp ecx, 3

1092 00006DEA 6683F903 <1> cmp cx, 3

1093 00006DEE 7605 <1> jna short fdinit\_pass\_fix\_sector\_count\_3

1094 <1> ;mov ecx, 3

1095 00006DF0 B903000000 <1> mov ecx, 3

1096 <1> fdinit\_pass\_fix\_sector\_count\_3:

1097 00006DF5 E8E4890000 <1> call chs\_read

1098 00006DFA 730D <1> jnc short fd\_init\_FAT\_sectors\_no\_load\_error

1099 00006DFC C605[16610100]00 <1> mov byte [FAT\_BuffValidData], 0

1100 <1> ; Drv not ready or read Error !

1101 00006E03 B80F000000 <1> mov eax, ERR\_DRV\_NOT\_RDY ; 15

1102 <1> ;xor edx, edx

1103 00006E08 C3 <1> retn

1104 <1>

1105 <1> fd\_init\_FAT\_sectors\_no\_load\_error:

1106 00006E09 C605[16610100]01 <1> mov byte [FAT\_BuffValidData], 1

1107 00006E10 A1[12610100] <1> mov eax, [FAT\_CurrentCluster]

1108 00006E15 E961FFFFFF <1> jmp fd\_init\_get\_next\_cluster\_readnext

1109 <1>

1110 <1> get\_FAT\_volume\_name:

1111 <1> ; 10/01/2016 (TRDOS 386 = TRDOS v2.0)

1112 <1> ; 12/09/2009

1113 <1> ; INPUT ->

1114 <1> ; BH = Logical DOS drive number (0,1,2,3,4 ...)

1115 <1> ; BL = 0

1116 <1> ; OUTPUT ->

1117 <1> ; CF = 0 -> ESI = Volume name address

1118 <1> ; CF = 1 -> Root volume name not found

1119 <1>

1120 <1> ;mov ah, 0FFh

1121 <1> ;mov al, [Last\_Dos\_DiskNo]

1122 <1> ;cmp al, bh

1123 <1> ;jb short loc\_gfvn\_dir\_load\_err

1124 <1>

1125 00006E1A 89DE <1> mov esi, ebx

1126 00006E1C 81E600FF0000 <1> and esi, 0FF00h ; esi = bh

1127 00006E22 81C600010900 <1> add esi, Logical\_DOSDisks

1128 00006E28 8A06 <1> mov al, [esi+LD\_Name]

1129 00006E2A 8A6603 <1> mov ah, [esi+LD\_FATType]

1130 00006E2D 80FC01 <1> cmp ah, 1

1131 00006E30 7210 <1> jb short loc\_gfvn\_dir\_load\_err

1132 00006E32 3C41 <1> cmp al, 'A'

1133 00006E34 720C <1> jb short loc\_gfvn\_dir\_load\_err

1134 00006E36 80FC02 <1> cmp ah, 2

1135 00006E39 7708 <1> ja short get\_FAT32\_root\_cluster

1136 <1>

1137 00006E3B E8634E0000 <1> call load\_FAT\_root\_directory

1138 00006E40 730B <1> jnc short loc\_get\_volume\_name

1139 <1>

1140 <1> loc\_gfvn\_dir\_load\_err:

1141 00006E42 C3 <1> retn

1142 <1>

1143 <1> get\_FAT32\_root\_cluster:

1144 00006E43 8B4632 <1> mov eax, [esi+LD\_BPB+BPB\_RootClus]

1145 00006E46 E8E34E0000 <1> call load\_FAT\_sub\_directory

1146 00006E4B 7224 <1> jc short loc\_get\_volume\_name\_retn

1147 <1>

1148 <1> loc\_get\_volume\_name:

1149 00006E4D BE00000800 <1> mov esi, Directory\_Buffer

1150 00006E52 6631C9 <1> xor cx, cx ; 0

1151 <1> check\_root\_volume\_name:

1152 00006E55 8A06 <1> mov al, [esi]

1153 00006E57 08C0 <1> or al, al

1154 00006E59 7416 <1> jz short loc\_get\_volume\_name\_retn

1155 00006E5B 807E0B08 <1> cmp byte [esi+0Bh], 08h

1156 00006E5F 7410 <1> je short loc\_get\_volume\_name\_retn

1157 00006E61 663B0D[2B610100] <1> cmp cx, [DirBuff\_LastEntry]

1158 00006E68 7308 <1> jnb short pass\_check\_root\_volume\_name

1159 00006E6A 6641 <1> inc cx

1160 00006E6C 83C620 <1> add esi, 32

1161 00006E6F EBE4 <1> jmp short check\_root\_volume\_name

1162 <1>

1163 <1> loc\_get\_volume\_name\_retn:

1164 00006E71 C3 <1> retn

1165 <1>

1166 <1> pass\_check\_root\_volume\_name:

1167 00006E72 803D[27610100]03 <1> cmp byte [DirBuff\_FATType], 3

1168 00006E79 7230 <1> jb short loc\_get\_volume\_name\_retn\_xor

1169 <1>

1170 00006E7B BB001C0900 <1> mov ebx, FAT\_Buffer

1171 00006E80 BE00010900 <1> mov esi, Logical\_DOSDisks

1172 00006E85 31C0 <1> xor eax, eax

1173 00006E87 8A25[26610100] <1> mov ah, [DirBuff\_DRV]

1174 00006E8D 80EC41 <1> sub ah, 'A'

1175 00006E90 01C6 <1> add esi, eax

1176 00006E92 A1[2D610100] <1> mov eax, [DirBuff\_Cluster]

1177 00006E97 E8AC4C0000 <1> call get\_next\_cluster

1178 00006E9C 7305 <1> jnc short loc\_gfvn\_load\_FAT32\_dir\_cluster

1179 <1>

1180 00006E9E 83F801 <1> cmp eax, 1

1181 00006EA1 F5 <1> cmc

1182 00006EA2 C3 <1> retn

1183 <1>

1184 <1> loc\_gfvn\_load\_FAT32\_dir\_cluster:

1185 00006EA3 E8864E0000 <1> call load\_FAT\_sub\_directory

1186 00006EA8 73A3 <1> jnc short loc\_get\_volume\_name

1187 00006EAA C3 <1> retn

1188 <1>

1189 <1> loc\_get\_volume\_name\_retn\_xor:

1190 00006EAB 31C0 <1> xor eax, eax

1191 00006EAD C3 <1> retn

1192 <1>

1193 <1> get\_media\_change\_status:

1194 <1> ; 10/01/2016 (TRDOS 386 = TRDOS v2.0)

1195 <1> ; 09/09/2009

1196 <1> ; INPUT:

1197 <1> ; DL = Drive number (physical)

1198 <1> ; OUTPUT: clc & AH = 6 media changed

1199 <1> ; clc & AH = 0 media not changed

1200 <1> ; stc -> Drive not ready or an error

1201 <1>

1202 00006EAE B416 <1> mov ah, 16h

1203 00006EB0 E851D3FFFF <1> call int13h

1204 00006EB5 80FC06 <1> cmp ah, 06h

1205 00006EB8 7405 <1> je short loc\_gmc\_status\_retn

1206 00006EBA 08E4 <1> or ah, ah

1207 00006EBC 7401 <1> jz short loc\_gmc\_status\_retn

1208 <1> loc\_gmc\_status\_stc\_retn:

1209 00006EBE F9 <1> stc

1210 <1> loc\_gmc\_status\_retn:

1211 00006EBF C3 <1> retn

2307 %include 'trdosk3.s' ; 06/01/2016

1 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2 <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.0) - MAIN PROGRAM : trdosk3.s

3 <1> ; ----------------------------------------------------------------------------

4 <1> ; Last Update: 31/12/2017

5 <1> ; ----------------------------------------------------------------------------

6 <1> ; Beginning: 06/01/2016

7 <1> ; ----------------------------------------------------------------------------

8 <1> ; Assembler: NASM version 2.11 (trdos386.s)

9 <1> ; ----------------------------------------------------------------------------

10 <1> ; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan

11 <1> ; MAINPROG.ASM (09/11/2011)

12 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

13 <1> ; MAINPROG.ASM [ TRDOS KERNEL - COMMAND EXECUTER SECTION - MAIN PROGRAM ]

14 <1> ; (c) 2004-2011 Erdogan TAN [ 17/01/2004 ] Last Update: 09/11/2011

15 <1> ; CMD\_INTR.ASM [ TRDOS Command Interpreter Procedure ] Last Update: 09/11/2011

16 <1> ; DIR.ASM [ DIRECTORY FUNCTIONS ] Last Update: 09/10/2011

17 <1> ; FILE.ASM [ FILE FUNCTIONS ] Last Update: 09/10/2011

18 <1>

19 <1> change\_current\_drive:

20 <1> ; 16/10/2016

21 <1> ; 02/02/2016

22 <1> ; 15/01/2016 (TRDOS 386 = TRDOS v2.0)

23 <1> ; 18/08/2011

24 <1> ; 09/09/2009

25 <1> ; INPUT:

26 <1> ; DL = Logical DOS Drive Number

27 <1> ; OUTPUT:

28 <1> ; cf=1 -> Not successful

29 <1> ; EAX = Error code

30 <1> ; cf=0 ->

31 <1> ; EAX = 0 (successful)

32 <1>

33 00006EC0 31DB <1> xor ebx, ebx

34 00006EC2 88D7 <1> mov bh, dl

35 <1>

36 <1> ;cmp dl, 1

37 <1> ;jna short loc\_ccdrv\_initial\_media\_change\_check

38 <1> ;cmp bh, [Last\_Dos\_DiskNo]

39 <1> ;ja short loc\_ccdrv\_drive\_not\_ready\_err

40 <1>

41 <1> loc\_ccdrv\_initial\_media\_change\_check:

42 00006EC4 BE00010900 <1> mov esi, Logical\_DOSDisks

43 00006EC9 01DE <1> add esi, ebx

44 <1> loc\_ccdrv\_dos\_drive\_name\_check:

45 00006ECB 80FA02 <1> cmp dl, 2

46 00006ECE 720F <1> jb short loc\_ccdrv\_dos\_drive\_name\_check\_ok

47 <1>

48 00006ED0 8A06 <1> mov al, [esi+LD\_Name]

49 00006ED2 2C41 <1> sub al, 'A'

50 00006ED4 38D0 <1> cmp al, dl

51 00006ED6 7407 <1> je short loc\_ccdrv\_dos\_drive\_name\_check\_ok

52 <1>

53 <1> loc\_ccdrv\_drive\_not\_ready\_err:

54 <1> ; 16/10/2016 (15h -> 15)

55 00006ED8 B80F000000 <1> mov eax, 15 ; Drive not ready

56 <1> loc\_change\_current\_drive\_stc\_retn:

57 00006EDD F9 <1> stc

58 00006EDE C3 <1> retn

59 <1>

60 <1> loc\_ccdrv\_dos\_drive\_name\_check\_ok:

61 00006EDF 8A667E <1> mov ah, [esi+LD\_MediaChanged]

62 00006EE2 80FC06 <1> cmp ah, 6 ; VOLUME NAME CHECK/MOVE SIGN

63 00006EE5 7455 <1> je short loc\_ccdrv\_get\_FAT\_volume\_name\_0

64 <1>

65 00006EE7 80FA01 <1> cmp dl, 1

66 00006EEA 777D <1> ja short loc\_gmcs\_init\_drv\_hd

67 <1>

68 <1> loc\_gmcs\_init\_drv\_fd:

69 00006EEC 08E4 <1> or ah, ah

70 <1> ; AH = 1 is initialization sign (invalid\_fd\_parameter)

71 00006EEE 7517 <1> jnz short loc\_ccdrv\_call\_fd\_init

72 <1>

73 00006EF0 E8B9FFFFFF <1> call get\_media\_change\_status

74 00006EF5 72E1 <1> jc short loc\_ccdrv\_drive\_not\_ready\_err

75 <1>

76 00006EF7 20E4 <1> and ah, ah

77 00006EF9 7476 <1> jz short loc\_change\_current\_drv3

78 <1>

79 00006EFB 80F406 <1> xor ah, 6

80 00006EFE 75D8 <1> jnz short loc\_ccdrv\_drive\_not\_ready\_err

81 <1>

82 <1> loc\_ccdrv\_call\_fd\_init\_check\_vol\_id:

83 00006F00 E8440A0000 <1> call get\_volume\_serial\_number

84 00006F05 730D <1> jnc short loc\_ccdrv\_check\_vol\_serial

85 <1>

86 <1> loc\_ccdrv\_call\_fd\_init:

87 00006F07 E872FCFFFF <1> call floppy\_drv\_init

88 00006F0C 731A <1> jnc short loc\_reset\_drv\_fd\_current\_dir

89 <1>

90 <1> loc\_ccdrv\_fdinit\_fail\_retn:

91 <1> ; 16/10/2016

92 00006F0E B80F000000 <1> mov eax, 15 ; Drive not ready

93 00006F13 C3 <1> retn

94 <1>

95 <1> loc\_ccdrv\_check\_vol\_serial:

96 00006F14 A3[F4580100] <1> mov [Current\_VolSerial], eax

97 <1> ;mov dl, bh

98 00006F19 E860FCFFFF <1> call floppy\_drv\_init

99 00006F1E 72EE <1> jc short loc\_ccdrv\_fdinit\_fail\_retn

100 <1>

101 00006F20 3B05[F4580100] <1> cmp eax, [Current\_VolSerial]

102 00006F26 7445 <1> je short loc\_change\_current\_drv2

103 <1>

104 <1> loc\_reset\_drv\_fd\_current\_dir:

105 00006F28 31C0 <1> xor eax, eax

106 00006F2A 88467F <1> mov [esi+LD\_CDirLevel], al

107 00006F2D 89F7 <1> mov edi, esi

108 00006F2F 81C780000000 <1> add edi, LD\_CurrentDirectory

109 00006F35 B920000000 <1> mov ecx, 32

110 00006F3A F3AB <1> rep stosd

111 <1>

112 <1> loc\_ccdrv\_get\_FAT\_volume\_name\_0:

113 00006F3C 8A4603 <1> mov al, [esi+LD\_FATType]

114 00006F3F 08C0 <1> or al, al

115 00006F41 742A <1> jz short loc\_change\_current\_drv2

116 <1>

117 00006F43 56 <1> push esi

118 00006F44 3C02 <1> cmp al, 2

119 00006F46 7705 <1> ja short loc\_ccdrv\_get\_FAT32\_vol\_name

120 <1>

121 <1> loc\_ccdrv\_get\_FAT2\_16\_vol\_name:

122 00006F48 83C631 <1> add esi, LD\_BPB + VolumeLabel

123 00006F4B EB03 <1> jmp short loc\_ccdrv\_get\_FAT\_volume\_name\_1

124 <1>

125 <1> loc\_ccdrv\_get\_FAT32\_vol\_name:

126 00006F4D 83C64D <1> add esi, LD\_BPB + FAT32\_VolLab

127 <1> loc\_ccdrv\_get\_FAT\_volume\_name\_1:

128 00006F50 53 <1> push ebx

129 00006F51 56 <1> push esi

130 00006F52 E8C3FEFFFF <1> call get\_FAT\_volume\_name

131 00006F57 5F <1> pop edi

132 00006F58 5B <1> pop ebx

133 <1> ; BL = 0

134 00006F59 720B <1> jc short loc\_change\_current\_drv1

135 00006F5B 20C0 <1> and al, al

136 00006F5D 7407 <1> jz short loc\_change\_current\_drv1

137 <1>

138 <1> loc\_ccdrv\_move\_FAT\_volume\_name:

139 00006F5F B90B000000 <1> mov ecx, 11

140 00006F64 F3A4 <1> rep movsb

141 <1>

142 <1> loc\_change\_current\_drv1:

143 00006F66 5E <1> pop esi

144 00006F67 EB04 <1> jmp short loc\_change\_current\_drv2

145 <1>

146 <1> loc\_gmcs\_init\_drv\_hd:

147 00006F69 08E4 <1> or ah, ah

148 00006F6B 7404 <1> jz short loc\_change\_current\_drv3

149 <1> ; BL = 0, BH = Logical DOS drive number

150 <1> loc\_change\_current\_drv2:

151 00006F6D C6467E00 <1> mov byte [esi+LD\_MediaChanged], 0

152 <1> loc\_change\_current\_drv3:

153 00006F71 883D[FE580100] <1> mov [Current\_Drv], bh

154 <1>

155 <1> ;call restore\_current\_directory

156 <1> ;retn

157 <1>

158 <1> restore\_current\_directory:

159 <1> ; 11/02/2016

160 <1> ; 15/01/2016 (TRDOS 386 = TRDOS v2.0)

161 <1> ; 25/01/2010

162 <1> ; 12/10/2009

163 <1> ;

164 <1> ; INPUT:

165 <1> ; ESI = Logical DOS Drive Description Table

166 <1> ;

167 <1> ; OUTPUT:

168 <1> ; ESI = Logical DOS Drive Description Table

169 <1> ; EDI = offset Current\_Dir\_Drv

170 <1>

171 00006F77 8A4603 <1> mov al, [esi+LD\_FATType]

172 00006F7A A2[FD580100] <1> mov [Current\_FATType], al

173 <1>

174 00006F7F 8A26 <1> mov ah, [esi+LD\_Name]

175 00006F81 8825[FF580100] <1> mov [Current\_Dir\_Drv], ah

176 <1>

177 00006F87 20C0 <1> and al, al

178 00006F89 741D <1> jz short loc\_restore\_FS\_current\_directory

179 <1>

180 <1> loc\_restore\_FAT\_current\_directory:

181 00006F8B 8A667F <1> mov ah, [esi+LD\_CDirLevel]

182 00006F8E 8825[FC580100] <1> mov [Current\_Dir\_Level], ah

183 00006F94 08E4 <1> or ah, ah

184 00006F96 7416 <1> jz short loc\_ccdrv\_reset\_cdir\_FAT\_12\_16\_32\_fcluster

185 <1>

186 00006F98 0FB6D4 <1> movzx edx, ah

187 00006F9B C0E204 <1> shl dl, 4 ; \* 16

188 00006F9E 01F2 <1> add edx, esi

189 00006FA0 8B828C000000 <1> mov eax, [edx+LD\_CurrentDirectory+12]

190 00006FA6 EB2C <1> jmp short loc\_ccdrv\_reset\_cdir\_FAT\_fcluster

191 <1>

192 <1> loc\_restore\_FS\_current\_directory:

193 00006FA8 E8BC4D0000 <1> call load\_current\_FS\_directory

194 00006FAD C3 <1> retn

195 <1>

196 <1> loc\_ccdrv\_reset\_cdir\_FAT\_12\_16\_32\_fcluster:

197 00006FAE 3C03 <1> cmp al, 3

198 00006FB0 7205 <1> jb short loc\_ccdrv\_reset\_cdir\_FAT\_12\_16\_fcluster

199 <1> loc\_ccdrv\_reset\_cdir\_FAT32\_fcluster:

200 00006FB2 8B4632 <1> mov eax, [esi+LD\_BPB+FAT32\_RootFClust]

201 00006FB5 EB04 <1> jmp short loc\_ccdrv\_check\_rootdir\_sign

202 <1> loc\_ccdrv\_reset\_cdir\_FAT\_12\_16\_fcluster:

203 00006FB7 30C0 <1> xor al, al ; xor eax, eax

204 00006FB9 31D2 <1> xor edx, edx

205 <1> loc\_ccdrv\_check\_rootdir\_sign:

206 00006FBB 80BE8000000000 <1> cmp byte [esi+LD\_CurrentDirectory], 0

207 00006FC2 7510 <1> jne short loc\_ccdrv\_reset\_cdir\_FAT\_fcluster

208 <1> loc\_ccdrv\_set\_rootdir\_FAT\_fcluster:

209 00006FC4 89868C000000 <1> mov [esi+LD\_CurrentDirectory+12], eax

210 00006FCA C78680000000524F4F- <1> mov dword [esi+LD\_CurrentDirectory], 'ROOT'

210 00006FD3 54 <1>

211 <1>

212 <1> loc\_ccdrv\_reset\_cdir\_FAT\_fcluster:

213 00006FD4 A3[F8580100] <1> mov [Current\_Dir\_FCluster], eax

214 <1>

215 00006FD9 BF[5F610100] <1> mov edi, PATH\_Array

216 00006FDE 89F2 <1> mov edx, esi

217 00006FE0 81C680000000 <1> add esi, LD\_CurrentDirectory

218 00006FE6 B920000000 <1> mov ecx, 32

219 00006FEB F3A5 <1> rep movsd

220 <1>

221 00006FED E84C2D0000 <1> call change\_prompt\_dir\_string

222 <1>

223 00006FF2 89D6 <1> mov esi, edx

224 <1>

225 00006FF4 29C0 <1> sub eax, eax

226 <1> ;sub edx, edx

227 00006FF6 BF[FF580100] <1> mov edi, Current\_Dir\_Drv

228 <1>

229 00006FFB A2[D30C0100] <1> mov [Restore\_CDIR], al ; 0

230 00007000 C3 <1> retn

231 <1>

232 <1> dos\_prompt:

233 <1> ; 06/05/2016

234 <1> ; 30/01/2016

235 <1> ; 29/01/2016

236 <1> ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)

237 <1> ; 15/09/2011

238 <1> ; 13/09/2009

239 <1> ; 2004-2005

240 <1>

241 <1> ; 06/05/2016

242 00007001 C705[BC650100]- <1> mov dword [mainprog\_return\_addr], return\_from\_cmd\_interpreter

242 00007007 [B5700000] <1>

243 <1>

244 <1> loc\_TRDOS\_prompt:

245 0000700B BF[FE590100] <1> mov edi, TextBuffer

246 00007010 C6075B <1> mov byte [edi], "["

247 00007013 47 <1> inc edi

248 00007014 BE[260D0100] <1> mov esi, TRDOSPromptLabel

249 <1> get\_next\_prompt\_label\_char:

250 00007019 803E20 <1> cmp byte [esi], 20h

251 0000701C 7203 <1> jb short pass\_prompt\_label

252 0000701E A4 <1> movsb

253 0000701F EBF8 <1> jmp short get\_next\_prompt\_label\_char

254 <1> pass\_prompt\_label:

255 00007021 C6075D <1> mov byte [edi], "]"

256 00007024 47 <1> inc edi

257 00007025 C60720 <1> mov byte [edi], 20h

258 00007028 47 <1> inc edi

259 00007029 BE[FF580100] <1> mov esi, Current\_Dir\_Drv

260 0000702E 66A5 <1> movsw

261 00007030 A4 <1> movsb

262 <1> loc\_prompt\_current\_directory:

263 00007031 803E20 <1> cmp byte [esi], 20h

264 00007034 7203 <1> jb short pass\_prompt\_current\_directory

265 00007036 A4 <1> movsb

266 00007037 EBF8 <1> jmp short loc\_prompt\_current\_directory

267 <1> pass\_prompt\_current\_directory:

268 00007039 C6073E <1> mov byte [edi], '>'

269 0000703C 47 <1> inc edi

270 0000703D C60700 <1> mov byte [edi], 0

271 00007040 BE[FE590100] <1> mov esi, TextBuffer

272 00007045 E813F3FFFF <1> call print\_msg

273 <1>

274 <1> ;sub bh, bh ; video page = 0

275 <1> ;call get\_cpos ; get cursor position

276 0000704A 668B15[56580100] <1> mov dx, [CURSOR\_POSN] ; video page 0

277 00007051 8815[5E590100] <1> mov [CursorColumn], dl

278 <1>

279 <1> ; 30/01/2016 (to show cursor on the row, again)

280 <1> ; (Initial color attributes of video page 0 is 0)

281 <1> ; (see: 'StartPMP' in trdos386.s)

282 <1> ;

283 <1> ;mov edi, 0B8000h ; start of video page 0

284 <1> ;movzx ecx, dl ; column

285 <1> ;mov al, 80

286 <1> ;mul dh

287 <1> ;add ax, cx

288 <1> ;shl ax, 1 ; character + attribute

289 <1> ;add di, ax ; (2\*80\*row) + (2\*column)

290 <1> ;neg cl

291 <1> ;add cl, 80

292 <1> ;mov ax, 700h ; ah = 7 (color attribute)

293 <1> ;rep stosw

294 <1>

295 <1> loc\_rw\_char:

296 00007057 E899000000 <1> call rw\_char

297 <1> loc\_move\_command:

298 0000705C BE[AE590100] <1> mov esi, CommandBuffer

299 00007061 89F7 <1> mov edi, esi

300 00007063 31C9 <1> xor ecx, ecx

301 <1> first\_command\_char:

302 00007065 AC <1> lodsb

303 00007066 3C20 <1> cmp al, 20h

304 00007068 772E <1> ja short pass\_space\_control

305 0000706A 7241 <1> jb short loc\_move\_cmd\_arguments\_ok

306 0000706C 81FE[FD590100] <1> cmp esi, CommandBuffer + 79

307 00007072 72F1 <1> jb short first\_command\_char

308 00007074 EB37 <1> jmp short loc\_move\_cmd\_arguments\_ok

309 <1>

310 <1> next\_command\_char:

311 00007076 AC <1> lodsb

312 00007077 3C20 <1> cmp al, 20h

313 00007079 771D <1> ja short pass\_space\_control

314 0000707B 7230 <1> jb short loc\_move\_cmd\_arguments\_ok

315 <1>

316 <1> loc\_1st\_cmd\_arg: ; 30/01/2016

317 0000707D AC <1> lodsb

318 0000707E 3C20 <1> cmp al, 20h

319 00007080 74FB <1> je short loc\_1st\_cmd\_arg

320 00007082 7229 <1> jb short loc\_move\_cmd\_arguments\_ok

321 <1>

322 00007084 C60700 <1> mov byte [edi], 0

323 00007087 47 <1> inc edi

324 <1>

325 <1> loc\_move\_cmd\_arguments:

326 00007088 AA <1> stosb

327 00007089 81FE[FD590100] <1> cmp esi, CommandBuffer + 79

328 0000708F 731C <1> jnb short loc\_move\_cmd\_arguments\_ok

329 00007091 AC <1> lodsb

330 00007092 3C20 <1> cmp al, 20h

331 00007094 73F2 <1> jnb short loc\_move\_cmd\_arguments

332 00007096 EB15 <1> jmp short loc\_move\_cmd\_arguments\_ok

333 <1>

334 <1> pass\_space\_control:

335 00007098 3C61 <1> cmp al, 61h

336 0000709A 7206 <1> jb short pass\_capitalize

337 0000709C 3C7A <1> cmp al, 7Ah

338 0000709E 7702 <1> ja short pass\_capitalize

339 000070A0 24DF <1> and al, 0DFh

340 <1> pass\_capitalize:

341 000070A2 AA <1> stosb

342 000070A3 FEC1 <1> inc cl

343 000070A5 81FE[FD590100] <1> cmp esi, CommandBuffer + 79

344 000070AB 72C9 <1> jb short next\_command\_char

345 <1>

346 <1> loc\_move\_cmd\_arguments\_ok:

347 000070AD C60700 <1> mov byte [edi], 0

348 <1>

349 <1> call\_command\_interpreter:

350 000070B0 E8CF080000 <1> call command\_interpreter

351 <1>

352 <1> return\_from\_cmd\_interpreter:

353 000070B5 B950000000 <1> mov ecx, 80

354 <1> ;mov cx, 80

355 000070BA BF[AE590100] <1> mov edi, CommandBuffer

356 000070BF 30C0 <1> xor al, al

357 000070C1 F3AA <1> rep stosb

358 <1> ;cmp byte [Program\_Exit], 0

359 <1> ;ja short loc\_terminate\_trdos

360 <1>

361 <1> ; 16/01/2016

362 000070C3 803D[C25E0000]03 <1> cmp byte [CRT\_MODE], 3 ; 80\*25 color

363 000070CA 741D <1> je short pass\_set\_txt\_mode

364 <1>

365 000070CC E892A4FFFF <1> call set\_txt\_mode ; set vide mode to 03h

366 <1> ; 07/01/2017

367 000070D1 30C0 <1> xor al, al

368 <1>

369 <1> loc\_check\_active\_page:

370 <1> ;xor al, al

371 000070D3 3805[66580100] <1> cmp [ACTIVE\_PAGE], al ; 0

372 000070D9 0F842CFFFFFF <1> je loc\_TRDOS\_prompt

373 <1> ; AL = 0 = video page 0

374 000070DF E898A8FFFF <1> call set\_active\_page

375 000070E4 E922FFFFFF <1> jmp loc\_TRDOS\_prompt ; infinitive loop

376 <1>

377 <1> pass\_set\_txt\_mode:

378 000070E9 BE[6F190100] <1> mov esi, nextline

379 000070EE E86AF2FFFF <1> call print\_msg

380 000070F3 EBDE <1> jmp short loc\_check\_active\_page

381 <1>

382 <1> rw\_char:

383 <1> ; 13/05/2016

384 <1> ; 30/01/2016

385 <1> ; 29/01/2016

386 <1> ; 17/01/2016 (TRDOS 386 = TRDOS v2.0)

387 <1> ; 2004-2005

388 <1>

389 <1> ; DH = cursor row, DL = cursor column

390 <1> ; BH = 0 = video page number (active page)

391 <1>

392 <1> ;xor bh, bh ; 0 = video page 0

393 <1>

394 <1> readnextchar:

395 000070F5 30E4 <1> xor ah, ah

396 000070F7 E81A9BFFFF <1> call int16h

397 000070FC 20C0 <1> and al, al

398 000070FE 7432 <1> jz short loc\_arrow

399 00007100 3CE0 <1> cmp al, 0E0h

400 00007102 742E <1> je short loc\_arrow

401 00007104 3C08 <1> cmp al, 08h

402 00007106 7542 <1> jne short char\_return

403 <1> loc\_back:

404 00007108 3A15[5E590100] <1> cmp dl, [CursorColumn]

405 0000710E 76E5 <1> jna short readnextchar

406 <1> prev\_column:

407 00007110 FECA <1> dec dl

408 <1> set\_cursor\_pos:

409 <1> ;push dx

410 00007112 52 <1> push edx ; 29/12/2017

411 <1> ;xor bh, bh ; 0 = video page 0

412 <1> ; DH = Row, DL = Column

413 00007113 E830ACFFFF <1> call \_set\_cpos ; 17/01/2016

414 00007118 5A <1> pop edx ; 29/12/2017

415 <1> ;pop dx

416 <1> ;movzx ebx, dl

417 00007119 88D3 <1> mov bl, dl

418 0000711B 2A1D[5E590100] <1> sub bl, [CursorColumn]

419 00007121 B020 <1> mov al, 20h

420 00007123 8883[AE590100] <1> mov [CommandBuffer+ebx], al

421 <1> ;sub bh, bh ; video page 0

422 <1> ;mov cx, 1

423 00007129 B307 <1> mov bl, 7 ; color attribute

424 0000712B E809ABFFFF <1> call \_write\_c\_current ; 17/01/2016

425 <1> ;mov dx, [CURSOR\_POSN]

426 00007130 EBC3 <1> jmp short readnextchar

427 <1> loc\_arrow:

428 00007132 80FC4B <1> cmp ah, 4Bh

429 00007135 74D1 <1> je short loc\_back

430 00007137 80FC53 <1> cmp ah, 53h

431 0000713A 74CC <1> je short loc\_back

432 0000713C 80FC4D <1> cmp ah, 4Dh

433 0000713F 75B4 <1> jne short readnextchar

434 00007141 80FA4F <1> cmp dl, 79

435 00007144 73AF <1> jnb short readnextchar

436 00007146 FEC2 <1> inc dl

437 00007148 EBC8 <1> jmp short set\_cursor\_pos

438 <1> char\_return:

439 0000714A 0FB6DA <1> movzx ebx, dl

440 0000714D 2A1D[5E590100] <1> sub bl, [CursorColumn]

441 00007153 3C20 <1> cmp al, 20h

442 00007155 721D <1> jb short loc\_escape

443 00007157 8883[AE590100] <1> mov [CommandBuffer+ebx], al

444 0000715D 80FA4F <1> cmp dl, 79

445 00007160 7393 <1> jnb short readnextchar

446 00007162 66BB0700 <1> mov bx, 7 ; color attribute

447 00007166 E847ABFFFF <1> call \_write\_tty

448 0000716B 668B15[56580100] <1> mov dx, [CURSOR\_POSN] ; video page 0

449 00007172 EB81 <1> jmp readnextchar

450 <1> loc\_escape:

451 00007174 3C1B <1> cmp al, 1Bh

452 00007176 7418 <1> je short rw\_char\_retn

453 <1> ;

454 00007178 3C0D <1> cmp al, 0Dh ; CR

455 0000717A 0F8575FFFFFF <1> jne readnextchar

456 <1> ; 13/05/2016

457 00007180 66BB0700 <1> mov bx, 7 ; attribute/color (bl)

458 <1> ; video page 0 (bh=0)

459 00007184 E829ABFFFF <1> call \_write\_tty

460 <1> ;mov bx, 7 ; attribute/color

461 <1> ; video page 0 (bh=0)

462 00007189 B00A <1> mov al, 0Ah ; LF

463 0000718B E822ABFFFF <1> call \_write\_tty

464 <1> rw\_char\_retn:

465 00007190 C3 <1> retn

466 <1>

467 <1> show\_date:

468 <1> ; 18/01/2016 (TRDOS 386 = TRDOS v2.0)

469 <1> ; 2004-2005

470 <1>

471 <1> ;mov ah, 04h

472 <1> ;call int1Ah

473 00007191 E8C9E8FFFF <1> call RTC\_40 ; GET RTC DATE

474 <1>

475 00007196 88D0 <1> mov al, dl

476 00007198 E8709AFFFF <1> call bcd\_to\_ascii

477 0000719D 66A3[120E0100] <1> mov [Day], ax

478 <1>

479 000071A3 88F0 <1> mov al, dh

480 000071A5 E8639AFFFF <1> call bcd\_to\_ascii

481 000071AA 66A3[150E0100] <1> mov [Month], ax

482 <1>

483 000071B0 88E8 <1> mov al, ch

484 000071B2 E8569AFFFF <1> call bcd\_to\_ascii

485 000071B7 66A3[180E0100] <1> mov [Century], ax

486 <1>

487 000071BD 88C8 <1> mov al, cl

488 000071BF E8499AFFFF <1> call bcd\_to\_ascii

489 000071C4 66A3[1A0E0100] <1> mov word [Year], ax

490 <1>

491 000071CA BE[020E0100] <1> mov esi, Msg\_Show\_Date

492 000071CF E889F1FFFF <1> call print\_msg

493 <1>

494 000071D4 C3 <1> retn

495 <1>

496 <1> set\_date:

497 <1> ; 13/05/2016

498 <1> ; 18/01/2016 (TRDOS 386 = TRDOS v2.0)

499 <1> ; 2004-2005

500 <1>

501 000071D5 BE[E60D0100] <1> mov esi, Msg\_Enter\_Date

502 000071DA E87EF1FFFF <1> call print\_msg

503 <1>

504 <1> loc\_enter\_day\_1:

505 000071DF 30E4 <1> xor ah, ah

506 000071E1 E8309AFFFF <1> call int16h

507 <1> ; AL = ASCII Code of the Character

508 000071E6 3C0D <1> cmp al, 13

509 000071E8 0F84B7010000 <1> je loc\_set\_date\_retn

510 000071EE 3C1B <1> cmp al, 27

511 000071F0 0F84AF010000 <1> je loc\_set\_date\_retn

512 000071F6 A2[120E0100] <1> mov [Day], al

513 000071FB 3C30 <1> cmp al, '0'

514 000071FD 0F82AD010000 <1> jb loc\_set\_date\_stc\_0

515 00007203 3C33 <1> cmp al, '3'

516 00007205 0F87A5010000 <1> ja loc\_set\_date\_stc\_0

517 <1> ; 13/05/2016

518 <1> ;mov bx, 7 ; attribute/color (bl)

519 <1> ; video page 0 (bh)

520 0000720B B307 <1> mov bl, 7

521 0000720D E8A0AAFFFF <1> call \_write\_tty

522 <1> loc\_enter\_day\_2:

523 00007212 30E4 <1> xor ah, ah

524 00007214 E8FD99FFFF <1> call int16h

525 <1> ; AL = ASCII Code of the Character

526 00007219 3C1B <1> cmp al, 27

527 0000721B 0F8484010000 <1> je loc\_set\_date\_retn

528 00007221 A2[130E0100] <1> mov [Day+1], al

529 00007226 3C30 <1> cmp al, '0'

530 00007228 0F828C010000 <1> jb loc\_set\_date\_stc\_1

531 0000722E 3C39 <1> cmp al, '9'

532 00007230 0F8784010000 <1> ja loc\_set\_date\_stc\_1

533 00007236 803D[120E0100]33 <1> cmp byte [Day], '3'

534 0000723D 7208 <1> jb short pass\_set\_day\_31

535 0000723F 3C31 <1> cmp al, '1'

536 00007241 0F8773010000 <1> ja loc\_set\_date\_stc\_1

537 <1> pass\_set\_day\_31:

538 <1> ; 13/05/2016

539 <1> ;mov bx, 7 ; attribute/color (bl)

540 <1> ; video page 0 (bh)

541 00007247 B307 <1> mov bl, 7

542 00007249 E864AAFFFF <1> call \_write\_tty

543 <1> loc\_enter\_separator\_1:

544 0000724E 28E4 <1> sub ah, ah ; 0

545 00007250 E8C199FFFF <1> call int16h

546 <1> ; AL = ASCII Code of the Character

547 00007255 3C1B <1> cmp al, 27

548 00007257 0F8448010000 <1> je loc\_set\_date\_retn

549 0000725D 3C2D <1> cmp al, '-'

550 0000725F 7408 <1> je short pass\_set\_date\_separator\_1

551 00007261 3C2F <1> cmp al, '/'

552 00007263 0F856C010000 <1> jne loc\_set\_date\_stc\_2

553 <1> pass\_set\_date\_separator\_1:

554 <1> ; 13/05/2016

555 <1> ;mov bx, 7 ; attribute/color (bl)

556 <1> ; video page 0 (bh)

557 00007269 B307 <1> mov bl, 7

558 0000726B E842AAFFFF <1> call \_write\_tty

559 <1> loc\_enter\_month\_1:

560 00007270 30E4 <1> xor ah, ah ; 0

561 00007272 E89F99FFFF <1> call int16h

562 <1> ; AL = ASCII Code of the Character

563 00007277 3C1B <1> cmp al, 27

564 00007279 0F8426010000 <1> je loc\_set\_date\_retn

565 0000727F A2[150E0100] <1> mov [Month], al

566 00007284 3C30 <1> cmp al, '0'

567 00007286 0F8264010000 <1> jb loc\_set\_date\_stc\_3

568 0000728C 3C31 <1> cmp al, '1'

569 0000728E 0F875C010000 <1> ja loc\_set\_date\_stc\_3

570 <1> ; 13/05/2016

571 <1> ;mov bx, 7 ; attribute/color (bl)

572 <1> ; video page 0 (bh)

573 00007294 B307 <1> mov bl, 7

574 00007296 E817AAFFFF <1> call \_write\_tty

575 <1> loc\_enter\_month\_2:

576 0000729B 30E4 <1> xor ah, ah

577 0000729D E87499FFFF <1> call int16h

578 <1> ; AL = ASCII Code of the Character

579 000072A2 3C1B <1> cmp al, 27

580 000072A4 0F84FB000000 <1> je loc\_set\_date\_retn

581 000072AA A2[160E0100] <1> mov [Month+1], al

582 000072AF 3C30 <1> cmp al, '0'

583 000072B1 0F8254010000 <1> jb loc\_set\_date\_stc\_4

584 000072B7 3C39 <1> cmp al, '9'

585 000072B9 0F874C010000 <1> ja loc\_set\_date\_stc\_4

586 000072BF 803D[150E0100]31 <1> cmp byte [Month], '1'

587 000072C6 7208 <1> jb short pass\_set\_month\_12

588 000072C8 3C32 <1> cmp al, '2'

589 000072CA 0F873B010000 <1> ja loc\_set\_date\_stc\_4

590 <1> pass\_set\_month\_12:

591 <1> ; 13/05/2016

592 <1> ;mov bx, 7 ; attribute/color (bl)

593 <1> ; video page 0 (bh)

594 000072D0 B307 <1> mov bl, 7

595 000072D2 E8DBA9FFFF <1> call \_write\_tty

596 <1> loc\_enter\_separator\_2:

597 000072D7 28E4 <1> sub ah, ah

598 000072D9 E83899FFFF <1> call int16h

599 <1> ; AL = ASCII Code of the Character

600 000072DE 3C1B <1> cmp al, 27

601 000072E0 0F84BF000000 <1> je loc\_set\_date\_retn

602 000072E6 3C2D <1> cmp al, '-'

603 000072E8 7408 <1> je short pass\_set\_date\_separator\_2

604 000072EA 3C2F <1> cmp al, '/'

605 000072EC 0F8534010000 <1> jne loc\_set\_date\_stc\_5

606 <1> pass\_set\_date\_separator\_2:

607 <1> ; 13/05/2016

608 <1> ;mov bx, 7 ; attribute/color (bl)

609 <1> ; video page 0 (bh)

610 000072F2 B307 <1> mov bl, 7

611 000072F4 E8B9A9FFFF <1> call \_write\_tty

612 <1> loc\_enter\_year\_1:

613 000072F9 30E4 <1> xor ah, ah

614 000072FB E81699FFFF <1> call int16h

615 <1> ; AL = ASCII Code of the Character

616 00007300 3C1B <1> cmp al, 27

617 00007302 0F849D000000 <1> je loc\_set\_date\_retn

618 00007308 A2[1A0E0100] <1> mov [Year], al

619 0000730D 3C30 <1> cmp al, '0'

620 0000730F 0F822C010000 <1> jb loc\_set\_date\_stc\_6

621 00007315 3C39 <1> cmp al, '9'

622 00007317 0F8724010000 <1> ja loc\_set\_date\_stc\_6

623 <1> ; 13/05/2016

624 <1> ;mov bx, 7 ; attribute/color (bl)

625 <1> ; video page 0 (bh)

626 0000731D B307 <1> mov bl, 7

627 0000731F E88EA9FFFF <1> call \_write\_tty

628 <1> loc\_enter\_year\_2:

629 00007324 30E4 <1> xor ah, ah

630 00007326 E8EB98FFFF <1> call int16h

631 <1> ; AL = ASCII Code of the Character

632 0000732B 3C1B <1> cmp al, 27

633 0000732D 7476 <1> je short loc\_set\_date\_retn

634 0000732F A2[1B0E0100] <1> mov byte [Year+1], al

635 00007334 3C30 <1> cmp al, '0'

636 00007336 0F8220010000 <1> jb loc\_set\_date\_stc\_7

637 0000733C 3C39 <1> cmp al, '9'

638 0000733E 0F8718010000 <1> ja loc\_set\_date\_stc\_7

639 <1> ; 13/05/2016

640 <1> ;mov bx, 7 ; attribute/color (bl)

641 <1> ; video page 0 (bh)

642 00007344 B307 <1> mov bl, 7

643 00007346 E867A9FFFF <1> call \_write\_tty

644 <1> loc\_set\_date\_get\_lchar\_again:

645 0000734B 28E4 <1> sub ah, ah ; 0

646 0000734D E8C498FFFF <1> call int16h

647 <1> ; AL = ASCII Code of the Character

648 00007352 3C0D <1> cmp al, 13 ; ENTER key

649 00007354 7412 <1> je short loc\_set\_date\_progress

650 00007356 3C1B <1> cmp al, 27 ; ESC key

651 00007358 744B <1> je short loc\_set\_date\_retn

652 <1> ;

653 0000735A E82A010000 <1> call check\_for\_backspace

654 0000735F 75EA <1> jne short loc\_set\_date\_get\_lchar\_again

655 <1>

656 <1> loc\_set\_date\_bs\_8:

657 00007361 E811010000 <1> call write\_backspace

658 00007366 EBBC <1> jmp short loc\_enter\_year\_2

659 <1>

660 <1> loc\_set\_date\_progress:

661 <1> ; Get Current Date

662 <1> ;mov ah, 04h

663 <1> ;call int1Ah

664 00007368 E8F2E6FFFF <1> call RTC\_40 ; GET RTC DATE

665 <1> ; CH = century (in BCD)

666 <1>

667 0000736D 66A1[1A0E0100] <1> mov ax, [Year]

668 00007373 662D3030 <1> sub ax, '00'

669 00007377 C0E004 <1> shl al, 4 ; \* 16

670 0000737A 88C1 <1> mov cl, al

671 0000737C 00E1 <1> add cl, ah

672 0000737E 66A1[150E0100] <1> mov ax, [Month]

673 00007384 662D3030 <1> sub ax, '00'

674 00007388 C0E004 <1> shl al, 4 ; \* 16

675 0000738B 88C6 <1> mov dh, al

676 0000738D 00E6 <1> add dh, ah

677 0000738F 66A1[120E0100] <1> mov ax, [Day]

678 00007395 662D3030 <1> sub ax, '00'

679 00007399 C0E004 <1> shl al, 4 ; \* 16

680 0000739C 88C2 <1> mov dl, al

681 0000739E 00E2 <1> add dl, ah

682 <1>

683 <1> ;mov ah, 05h

684 <1> ;call int1Ah

685 000073A0 E8E7E6FFFF <1> call RTC\_50 ; SET RTC DATE

686 <1>

687 <1> loc\_set\_date\_retn:

688 000073A5 BE[6F190100] <1> mov esi, nextline

689 000073AA E8AEEFFFFF <1> call print\_msg

690 000073AF C3 <1> retn

691 <1>

692 <1> loc\_set\_date\_stc\_0:

693 <1> ;xor bh, bh ; video page 0

694 000073B0 E8DDA9FFFF <1> call beeper ; BEEP !

695 000073B5 E925FEFFFF <1> jmp loc\_enter\_day\_1

696 <1> loc\_set\_date\_stc\_1:

697 000073BA E8CA000000 <1> call check\_for\_backspace

698 000073BF 740A <1> je short loc\_set\_date\_bs\_1

699 <1> ;xor bh, bh ; video page 0

700 000073C1 E8CCA9FFFF <1> call beeper ; BEEP !

701 000073C6 E947FEFFFF <1> jmp loc\_enter\_day\_2

702 <1> loc\_set\_date\_bs\_1:

703 000073CB E8A7000000 <1> call write\_backspace

704 000073D0 E90AFEFFFF <1> jmp loc\_enter\_day\_1

705 <1> loc\_set\_date\_stc\_2:

706 000073D5 E8AF000000 <1> call check\_for\_backspace

707 000073DA 740A <1> je short loc\_set\_date\_bs\_2

708 <1> ;xor bh, bh ; video page 0

709 000073DC E8B1A9FFFF <1> call beeper ; BEEP !

710 000073E1 E968FEFFFF <1> jmp loc\_enter\_separator\_1

711 <1> loc\_set\_date\_bs\_2:

712 000073E6 E88C000000 <1> call write\_backspace

713 000073EB E922FEFFFF <1> jmp loc\_enter\_day\_2

714 <1> loc\_set\_date\_stc\_3:

715 000073F0 E894000000 <1> call check\_for\_backspace

716 000073F5 740A <1> je short loc\_set\_date\_bs\_3

717 <1> ;xor bh, bh ; video page 0

718 000073F7 E896A9FFFF <1> call beeper ; BEEP !

719 000073FC E96FFEFFFF <1> jmp loc\_enter\_month\_1

720 <1> loc\_set\_date\_bs\_3:

721 00007401 E871000000 <1> call write\_backspace

722 00007406 E943FEFFFF <1> jmp loc\_enter\_separator\_1

723 <1> loc\_set\_date\_stc\_4:

724 0000740B E879000000 <1> call check\_for\_backspace

725 00007410 740A <1> je short loc\_set\_date\_bs\_4

726 <1> ;xor bh, bh ; video page 0

727 00007412 E87BA9FFFF <1> call beeper ; BEEP !

728 00007417 E97FFEFFFF <1> jmp loc\_enter\_month\_2

729 <1> loc\_set\_date\_bs\_4:

730 0000741C E856000000 <1> call write\_backspace

731 00007421 E94AFEFFFF <1> jmp loc\_enter\_month\_1

732 <1> loc\_set\_date\_stc\_5:

733 00007426 E85E000000 <1> call check\_for\_backspace

734 0000742B 740A <1> je short loc\_set\_date\_bs\_5

735 <1> ;xor bh, bh ; video page 0

736 0000742D E860A9FFFF <1> call beeper ; BEEP !

737 00007432 E9A0FEFFFF <1> jmp loc\_enter\_separator\_2

738 <1> loc\_set\_date\_bs\_5:

739 00007437 E83B000000 <1> call write\_backspace

740 0000743C E95AFEFFFF <1> jmp loc\_enter\_month\_2

741 <1> loc\_set\_date\_stc\_6:

742 00007441 E843000000 <1> call check\_for\_backspace

743 00007446 740A <1> je short loc\_set\_date\_bs\_6

744 <1> ;xor bh, bh ; video page 0

745 00007448 E845A9FFFF <1> call beeper ; BEEP !

746 0000744D E9A7FEFFFF <1> jmp loc\_enter\_year\_1

747 <1> loc\_set\_date\_bs\_6:

748 00007452 E820000000 <1> call write\_backspace

749 00007457 E97BFEFFFF <1> jmp loc\_enter\_separator\_2

750 <1> loc\_set\_date\_stc\_7:

751 0000745C E828000000 <1> call check\_for\_backspace

752 00007461 740A <1> je short loc\_set\_date\_bs\_7

753 <1> ;xor bh, bh ; video page 0

754 00007463 E82AA9FFFF <1> call beeper ; BEEP !

755 00007468 E9B7FEFFFF <1> jmp loc\_enter\_year\_2

756 <1> loc\_set\_date\_bs\_7:

757 0000746D E805000000 <1> call write\_backspace

758 00007472 E982FEFFFF <1> jmp loc\_enter\_year\_1

759 <1>

760 <1> write\_backspace:

761 <1> ; 18/01/2016 (TRDOS 386 = TRDOS v2.0)

762 00007477 B008 <1> mov al, 08h ; BACKSPACE

763 <1> ; 13/05/2016

764 00007479 66BB0700 <1> mov bx, 7 ; bl = attribute/color

765 <1> ; bh = video page = 0

766 0000747D E830A8FFFF <1> call \_write\_tty

767 00007482 B020 <1> mov al, 20h ; BLANK/SPACE char

768 <1> ;mov bx, 7 ; attribute/color

769 <1> ;call \_write\_c\_current

770 <1> ;retn

771 00007484 E9B0A7FFFF <1> jmp \_write\_c\_current

772 <1>

773 <1> check\_for\_backspace:

774 <1> ; 18/01/2016 (TRDOS 386 = TRDOS v2.0)

775 00007489 663D080E <1> cmp ax, 0E08h

776 0000748D 7410 <1> je short cfbs\_retn

777 0000748F 663DE04B <1> cmp ax, 4BE0h

778 00007493 740A <1> je short cfbs\_retn

779 00007495 663D004B <1> cmp ax, 4B00h

780 00007499 7404 <1> je short cfbs\_retn

781 0000749B 663DE053 <1> cmp ax, 53E0h

782 <1> cfbs\_retn:

783 0000749F C3 <1> retn

784 <1>

785 <1> show\_time:

786 <1> ; 18/01/2016 (TRDOS 386 = TRDOS v2.0)

787 <1> ; 2004-2005

788 <1>

789 <1> ;mov ah, 02h

790 <1> ;call int1Ah

791 000074A0 E849E5FFFF <1> call RTC\_20 ; GET RTC TIME

792 <1>

793 000074A5 88E8 <1> mov al, ch

794 000074A7 E86197FFFF <1> call bcd\_to\_ascii

795 000074AC 66A3[400E0100] <1> mov [Hour], ax

796 <1>

797 000074B2 88C8 <1> mov al, cl

798 000074B4 E85497FFFF <1> call bcd\_to\_ascii

799 000074B9 66A3[430E0100] <1> mov [Minute], ax

800 <1>

801 000074BF 88F0 <1> mov al, dh

802 000074C1 E84797FFFF <1> call bcd\_to\_ascii

803 000074C6 66A3[460E0100] <1> mov [Second], ax

804 <1>

805 000074CC BE[300E0100] <1> mov esi, Msg\_Show\_Time

806 000074D1 E887EEFFFF <1> call print\_msg

807 000074D6 C3 <1> retn

808 <1>

809 <1> set\_time:

810 <1> ; 13/05/2016

811 <1> ; 18/01/2016 (TRDOS 386 = TRDOS v2.0)

812 <1> ; 2004-2005

813 <1>

814 000074D7 BE[1F0E0100] <1> mov esi, Msg\_Enter\_Time

815 000074DC E87CEEFFFF <1> call print\_msg

816 <1>

817 <1> loc\_enter\_hour\_1:

818 000074E1 30E4 <1> xor ah, ah

819 000074E3 E82E97FFFF <1> call int16h

820 <1> ; AL = ASCII Code of the Character

821 000074E8 3C0D <1> cmp al, 13 ; ENTER key

822 000074EA 0F84AE010000 <1> je loc\_set\_time\_retn

823 000074F0 3C1B <1> cmp al, 27 ; ESC key

824 000074F2 0F84A6010000 <1> je loc\_set\_time\_retn

825 000074F8 A2[400E0100] <1> mov [Hour], al

826 000074FD 3C30 <1> cmp al, '0'

827 000074FF 0F82A4010000 <1> jb loc\_set\_time\_stc\_0

828 00007505 3C32 <1> cmp al, '2'

829 00007507 0F879C010000 <1> ja loc\_set\_time\_stc\_0

830 <1> ; 13/05/2016

831 <1> ;mov bx, 7 ; attribute/color (bl)

832 <1> ; video page 0 (bh)

833 0000750D B307 <1> mov bl, 7

834 0000750F E89EA7FFFF <1> call \_write\_tty

835 <1> loc\_enter\_hour\_2:

836 00007514 30E4 <1> xor ah, ah

837 00007516 E8FB96FFFF <1> call int16h

838 <1> ; AL = ASCII Code of the Character

839 0000751B 3C1B <1> cmp al, 27

840 0000751D 0F847B010000 <1> je loc\_set\_time\_retn

841 00007523 A2[410E0100] <1> mov [Hour+1], al

842 00007528 3C30 <1> cmp al, '0'

843 0000752A 0F8283010000 <1> jb loc\_set\_time\_stc\_1

844 00007530 3C39 <1> cmp al, '9'

845 00007532 0F877B010000 <1> ja loc\_set\_time\_stc\_1

846 00007538 803D[400E0100]32 <1> cmp byte [Hour], '2'

847 0000753F 7208 <1> jb short pass\_set\_time\_24

848 00007541 3C34 <1> cmp al, '4'

849 00007543 0F876A010000 <1> ja loc\_set\_time\_stc\_1

850 <1> pass\_set\_time\_24:

851 <1> ; 13/05/2016

852 <1> ;mov bx, 7 ; attribute/color (bl)

853 <1> ; video page 0 (bh)

854 00007549 B307 <1> mov bl, 7

855 0000754B E862A7FFFF <1> call \_write\_tty

856 <1> loc\_enter\_time\_separator\_1:

857 00007550 28E4 <1> sub ah, ah ; 0

858 00007552 E8BF96FFFF <1> call int16h

859 <1> ; AL = ASCII Code of the Character

860 00007557 3C1B <1> cmp al, 27

861 00007559 0F843F010000 <1> je loc\_set\_time\_retn

862 0000755F 3C3A <1> cmp al, ':'

863 00007561 0F8567010000 <1> jne loc\_set\_time\_stc\_2

864 <1> ; 13/05/2016

865 <1> ;mov bx, 7 ; attribute/color (bl)

866 <1> ; video page 0 (bh)

867 00007567 B307 <1> mov bl, 7

868 00007569 E844A7FFFF <1> call \_write\_tty

869 <1> loc\_enter\_minute\_1:

870 0000756E 30E4 <1> xor ah, ah

871 00007570 E8A196FFFF <1> call int16h

872 <1> ; AL = ASCII Code of the Character

873 00007575 3C1B <1> cmp al, 27

874 00007577 0F8421010000 <1> je loc\_set\_time\_retn

875 0000757D A2[430E0100] <1> mov [Minute], al

876 00007582 3C30 <1> cmp al, '0'

877 00007584 0F825F010000 <1> jb loc\_set\_time\_stc\_3

878 0000758A 3C35 <1> cmp al, '5'

879 0000758C 0F8757010000 <1> ja loc\_set\_time\_stc\_3

880 <1> ; 13/05/2016

881 <1> ;mov bx, 7 ; attribute/color (bl)

882 <1> ; video page 0 (bh)

883 00007592 B307 <1> mov bl, 7

884 00007594 E819A7FFFF <1> call \_write\_tty

885 <1> loc\_enter\_minute\_2:

886 00007599 30E4 <1> xor ah, ah

887 0000759B E87696FFFF <1> call int16h

888 <1> ; AL = ASCII Code of the Character

889 000075A0 3C1B <1> cmp al, 27

890 000075A2 0F84F6000000 <1> je loc\_set\_time\_retn

891 000075A8 A2[440E0100] <1> mov [Minute+1], al

892 000075AD 3C30 <1> cmp al, '0'

893 000075AF 0F824F010000 <1> jb loc\_set\_time\_stc\_4

894 000075B5 3C39 <1> cmp al, '9'

895 000075B7 0F8747010000 <1> ja loc\_set\_time\_stc\_4

896 <1> ; 13/05/2016

897 <1> ;mov bx, 7 ; attribute/color (bl)

898 <1> ; video page 0 (bh)

899 000075BD B307 <1> mov bl, 7

900 000075BF E8EEA6FFFF <1> call \_write\_tty

901 <1> loc\_enter\_time\_separator\_2:

902 000075C4 66C705[460E0100]30- <1> mov word [Second], 3030h

902 000075CC 30 <1>

903 000075CD 28E4 <1> sub ah, ah

904 000075CF E84296FFFF <1> call int16h

905 <1> ; AL = ASCII Code of the Character

906 000075D4 3C0D <1> cmp al, 13

907 000075D6 0F8485000000 <1> je loc\_set\_time\_progress

908 000075DC 3C1B <1> cmp al, 27

909 000075DE 0F84BA000000 <1> je loc\_set\_time\_retn

910 000075E4 3C3A <1> cmp al, ':'

911 000075E6 0F8533010000 <1> jne loc\_set\_time\_stc\_5

912 <1> ; 13/05/2016

913 <1> ;mov bx, 7 ; attribute/color (bl)

914 <1> ; video page 0 (bh)

915 000075EC B307 <1> mov bl, 7

916 000075EE E8BFA6FFFF <1> call \_write\_tty

917 <1> loc\_enter\_second\_1:

918 000075F3 30E4 <1> xor ah, ah

919 000075F5 E81C96FFFF <1> call int16h

920 <1> ; AL = ASCII Code of the Character

921 000075FA 3C0D <1> cmp al, 13

922 000075FC 7463 <1> je short loc\_set\_time\_progress

923 000075FE 3C1B <1> cmp al, 27

924 00007600 0F8498000000 <1> je loc\_set\_time\_retn

925 00007606 A2[460E0100] <1> mov [Second], al

926 0000760B 3C30 <1> cmp al, '0'

927 0000760D 0F8227010000 <1> jb loc\_set\_time\_stc\_6

928 00007613 3C35 <1> cmp al, '5'

929 00007615 0F871F010000 <1> ja loc\_set\_time\_stc\_6

930 <1> ; 13/05/2016

931 <1> ;mov bx, 7 ; attribute/color (bl)

932 <1> ; video page 0 (bh)

933 0000761B B307 <1> mov bl, 7

934 0000761D E890A6FFFF <1> call \_write\_tty

935 <1> loc\_enter\_second\_2:

936 00007622 30E4 <1> xor ah, ah

937 00007624 E8ED95FFFF <1> call int16h

938 <1> ; AL = ASCII Code of the Character

939 00007629 3C1B <1> cmp al, 27

940 0000762B 7471 <1> je short loc\_set\_time\_retn

941 0000762D 3C30 <1> cmp al, '0'

942 0000762F 0F8229010000 <1> jb loc\_set\_time\_stc\_7

943 00007635 3C39 <1> cmp al, '9'

944 00007637 0F8721010000 <1> ja loc\_set\_time\_stc\_7

945 <1> ; 13/05/2016

946 <1> ;mov bx, 7 ; attribute/color (bl)

947 <1> ; video page 0 (bh)

948 0000763D B307 <1> mov bl, 7

949 0000763F E86EA6FFFF <1> call \_write\_tty

950 <1> loc\_set\_time\_get\_lchar\_again:

951 00007644 28E4 <1> sub ah, ah ; 0

952 00007646 E8CB95FFFF <1> call int16h

953 <1> ; AL = ASCII Code of the Character

954 0000764B 3C0D <1> cmp al, 13

955 0000764D 7412 <1> je short loc\_set\_time\_progress

956 0000764F 3C1B <1> cmp al, 27

957 00007651 744B <1> je short loc\_set\_time\_retn

958 <1> ;

959 00007653 E831FEFFFF <1> call check\_for\_backspace

960 00007658 75EA <1> jne short loc\_set\_time\_get\_lchar\_again

961 <1>

962 <1> loc\_set\_time\_bs\_8:

963 0000765A E818FEFFFF <1> call write\_backspace

964 0000765F EBC1 <1> jmp short loc\_enter\_second\_2

965 <1>

966 <1> loc\_set\_time\_progress:

967 <1> ; Get Current Time

968 <1> ;mov ah, 02h

969 <1> ;call int1Ah

970 00007661 E888E3FFFF <1> call RTC\_20 ; GET RTC TIME

971 <1> ;DL = Daylight Savings Enable option (0-1)

972 <1>

973 00007666 66A1[400E0100] <1> mov ax, [Hour]

974 0000766C 662D3030 <1> sub ax, '00'

975 00007670 C0E004 <1> shl al, 4 ; \* 16

976 00007673 88C5 <1> mov ch, al

977 00007675 00E5 <1> add ch, ah

978 00007677 66A1[430E0100] <1> mov ax, [Minute]

979 0000767D 662D3030 <1> sub ax, '00'

980 00007681 C0E004 <1> shl al, 4 ; \* 16

981 00007684 88C1 <1> mov cl, al

982 00007686 00E1 <1> add cl, ah

983 00007688 66A1[460E0100] <1> mov ax, [Second]

984 0000768E 662D3030 <1> sub ax, '00'

985 00007692 C0E004 <1> shl al, 4 ; \* 16

986 00007695 88C6 <1> mov dh, al

987 00007697 00E6 <1> add dh, ah

988 <1>

989 <1> ;mov ah, 03h

990 <1> ;call int1Ah

991 00007699 E87FE3FFFF <1> call RTC\_30 ; SET RTC TIME

992 <1>

993 <1> loc\_set\_time\_retn:

994 0000769E BE[6F190100] <1> mov esi, nextline

995 000076A3 E8B5ECFFFF <1> call print\_msg

996 000076A8 C3 <1> retn

997 <1>

998 <1> loc\_set\_time\_stc\_0:

999 <1> ;xor bh, bh ; video page 0

1000 000076A9 E8E4A6FFFF <1> call beeper ; BEEP !

1001 000076AE E92EFEFFFF <1> jmp loc\_enter\_hour\_1

1002 <1> loc\_set\_time\_stc\_1:

1003 000076B3 E8D1FDFFFF <1> call check\_for\_backspace

1004 000076B8 740A <1> je short loc\_set\_time\_bs\_1

1005 <1> ;xor bh, bh ; video page 0

1006 000076BA E8D3A6FFFF <1> call beeper ; BEEP !

1007 000076BF E950FEFFFF <1> jmp loc\_enter\_hour\_2

1008 <1> loc\_set\_time\_bs\_1:

1009 000076C4 E8AEFDFFFF <1> call write\_backspace

1010 000076C9 E913FEFFFF <1> jmp loc\_enter\_hour\_1

1011 <1> loc\_set\_time\_stc\_2:

1012 000076CE E8B6FDFFFF <1> call check\_for\_backspace

1013 000076D3 740A <1> je short loc\_set\_time\_bs\_2

1014 <1> ;xor bh, bh ; video page 0

1015 000076D5 E8B8A6FFFF <1> call beeper ; BEEP !

1016 000076DA E971FEFFFF <1> jmp loc\_enter\_time\_separator\_1

1017 <1> loc\_set\_time\_bs\_2:

1018 000076DF E893FDFFFF <1> call write\_backspace

1019 000076E4 E92BFEFFFF <1> jmp loc\_enter\_hour\_2

1020 <1> loc\_set\_time\_stc\_3:

1021 000076E9 E89BFDFFFF <1> call check\_for\_backspace

1022 000076EE 740A <1> je short loc\_set\_time\_bs\_3

1023 <1> ;xor bh, bh ; video page 0

1024 000076F0 E89DA6FFFF <1> call beeper ; BEEP !6

1025 000076F5 E974FEFFFF <1> jmp loc\_enter\_minute\_1

1026 <1> loc\_set\_time\_bs\_3:

1027 000076FA E878FDFFFF <1> call write\_backspace

1028 000076FF E94CFEFFFF <1> jmp loc\_enter\_time\_separator\_1

1029 <1> loc\_set\_time\_stc\_4:

1030 00007704 E880FDFFFF <1> call check\_for\_backspace

1031 00007709 740A <1> je short loc\_set\_time\_bs\_4

1032 <1> ;xor bh, bh ; video page 0

1033 0000770B E882A6FFFF <1> call beeper ; BEEP !

1034 00007710 E984FEFFFF <1> jmp loc\_enter\_minute\_2

1035 <1> loc\_set\_time\_bs\_4:

1036 00007715 E85DFDFFFF <1> call write\_backspace

1037 0000771A E94FFEFFFF <1> jmp loc\_enter\_minute\_1

1038 <1> loc\_set\_time\_stc\_5:

1039 0000771F E865FDFFFF <1> call check\_for\_backspace

1040 00007724 740A <1> je short loc\_set\_time\_bs\_5

1041 <1> ;xor bh, bh ; video page 0

1042 00007726 E867A6FFFF <1> call beeper ; BEEP !

1043 0000772B E994FEFFFF <1> jmp loc\_enter\_time\_separator\_2

1044 <1> loc\_set\_time\_bs\_5:

1045 00007730 E842FDFFFF <1> call write\_backspace

1046 00007735 E95FFEFFFF <1> jmp loc\_enter\_minute\_2

1047 <1> loc\_set\_time\_stc\_6:

1048 0000773A E84AFDFFFF <1> call check\_for\_backspace

1049 0000773F 7413 <1> je short loc\_set\_time\_bs\_6

1050 <1> ;xor bh, bh ; video page 0

1051 00007741 E84CA6FFFF <1> call beeper ; BEEP !

1052 00007746 66C705[460E0100]30- <1> mov word [Second], 3030h

1052 0000774E 30 <1>

1053 0000774F E99FFEFFFF <1> jmp loc\_enter\_second\_1

1054 <1> loc\_set\_time\_bs\_6:

1055 00007754 E81EFDFFFF <1> call write\_backspace

1056 00007759 E966FEFFFF <1> jmp loc\_enter\_time\_separator\_2

1057 <1> loc\_set\_time\_stc\_7:

1058 0000775E E826FDFFFF <1> call check\_for\_backspace

1059 00007763 740A <1> je short loc\_set\_time\_bs\_7

1060 <1> ;xor bh, bh ; video page 0

1061 00007765 E828A6FFFF <1> call beeper ; BEEP !

1062 0000776A E9B3FEFFFF <1> jmp loc\_enter\_second\_2

1063 <1> loc\_set\_time\_bs\_7:

1064 0000776F E803FDFFFF <1> call write\_backspace

1065 00007774 E97AFEFFFF <1> jmp loc\_enter\_second\_1

1066 <1>

1067 <1> print\_volume\_info:

1068 <1> ; 01/03/2016

1069 <1> ; 08/02/2016

1070 <1> ; 06/02/2016

1071 <1> ; 04/02/2016

1072 <1> ; 18/01/2016 (TRDOS 386 = TRDOS v2.0)

1073 <1> ; 25/10/2009

1074 <1> ;

1075 <1> ; "Volume Serial No: "

1076 <1> ;

1077 <1> ; INPUT : AL = DOS Drive Number

1078 <1> ; OUTPUT : AH = FS Type

1079 <1> ; AL = DOS Drive Name

1080 <1> ; CF = 0 -> OK

1081 <1> ; CF = 1 -> Drive not ready

1082 <1>

1083 00007779 88C4 <1> mov ah, al

1084 0000777B 28C0 <1> sub al, al

1085 0000777D 0FB7F0 <1> movzx esi, ax

1086 00007780 81C600010900 <1> add esi, Logical\_DOSDisks

1087 00007786 8A06 <1> mov al, [esi]

1088 00007788 3C41 <1> cmp al, 'A'

1089 0000778A 7304 <1> jnb short loc\_pvi\_set\_vol\_name

1090 0000778C 8A6604 <1> mov ah, [esi+LD\_FSType]

1091 0000778F C3 <1> retn

1092 <1>

1093 <1> loc\_pvi\_set\_vol\_name:

1094 00007790 A2[7A0E0100] <1> mov [Vol\_Drv\_Name], al

1095 00007795 56 <1> push esi

1096 00007796 E858010000 <1> call move\_volume\_name\_and\_serial\_no ;;;

1097 0000779B 7302 <1> jnc short loc\_pvi\_mvn\_ok

1098 0000779D 5E <1> pop esi

1099 0000779E C3 <1> retn

1100 <1>

1101 <1> loc\_pvi\_mvn\_ok:

1102 0000779F 8B3424 <1> mov esi, [esp]

1103 000077A2 807E04A1 <1> cmp byte [esi+LD\_FSType], 0A1h

1104 000077A6 7509 <1> jne short loc\_pvi\_fat\_vol\_size

1105 000077A8 8B4670 <1> mov eax, [esi+LD\_FS\_VolumeSize]

1106 000077AB 0FB75E11 <1> movzx ebx, word [esi+LD\_FS\_BytesPerSec]

1107 000077AF EB07 <1> jmp short loc\_vol\_size\_mul32

1108 <1> loc\_pvi\_fat\_vol\_size:

1109 000077B1 8B4670 <1> mov eax, [esi+LD\_TotalSectors]

1110 000077B4 0FB75E11 <1> movzx ebx, word [esi+LD\_BPB+BPB\_BytsPerSec]

1111 <1> loc\_vol\_size\_mul32:

1112 000077B8 F7E3 <1> mul ebx

1113 000077BA 09D2 <1> or edx, edx

1114 000077BC 7507 <1> jnz short loc\_vol\_size\_in\_kbytes

1115 <1> loc\_vol\_size\_in\_bytes:

1116 000077BE B9[580E0100] <1> mov ecx, VolSize\_Bytes

1117 000077C3 EB0D <1> jmp short loc\_write\_vol\_size\_str

1118 <1> loc\_vol\_size\_in\_kbytes:

1119 000077C5 66BB0004 <1> mov bx, 1024

1120 000077C9 F7F3 <1> div ebx

1121 000077CB B9[4B0E0100] <1> mov ecx, VolSize\_KiloBytes

1122 000077D0 31D2 <1> xor edx, edx ; 0

1123 <1> loc\_write\_vol\_size\_str:

1124 000077D2 890D[37610100] <1> mov [VolSize\_Unit1], ecx

1125 <1> ;

1126 000077D8 BF[4D610100] <1> mov edi, Vol\_Tot\_Sec\_Str\_End

1127 <1> ;mov byte [edi], 0

1128 000077DD B90A000000 <1> mov ecx, 10

1129 <1> loc\_write\_vol\_size\_chr:

1130 000077E2 F7F1 <1> div ecx

1131 000077E4 80C230 <1> add dl, '0'

1132 000077E7 4F <1> dec edi

1133 000077E8 8817 <1> mov [edi], dl

1134 000077EA 85C0 <1> test eax, eax

1135 000077EC 7404 <1> jz short loc\_write\_vol\_size\_str\_ok

1136 000077EE 28D2 <1> sub dl, dl ; 0

1137 000077F0 EBF0 <1> jmp short loc\_write\_vol\_size\_chr

1138 <1>

1139 <1> loc\_write\_vol\_size\_str\_ok:

1140 000077F2 893D[3F610100] <1> mov [Vol\_Tot\_Sec\_Str\_Start], edi

1141 <1> ;

1142 000077F8 BF[630E0100] <1> mov edi, Vol\_FS\_Name

1143 000077FD 8A4E03 <1> mov cl, [esi+LD\_FATType]

1144 00007800 20C9 <1> and cl, cl ; 0 ?

1145 00007802 7515 <1> jnz short loc\_write\_vol\_FAT\_str\_1

1146 00007804 66C7075452 <1> mov word [edi], 'TR'

1147 00007809 C7470420465331 <1> mov dword [edi+4], ' FS1'

1148 <1> ;movzx ebx, word [esi+LD\_FS\_BytesPerSec]

1149 00007810 668B5E11 <1> mov bx, [esi+LD\_FS\_BytesPerSec]

1150 00007814 8B4674 <1> mov eax, [esi+LD\_FS\_FreeSectors]

1151 00007817 EB36 <1> jmp short loc\_vol\_freespace\_mul32

1152 <1>

1153 <1> loc\_write\_vol\_FAT\_str\_1:

1154 00007819 66B83332 <1> mov ax, '32' ; FAT32

1155 0000781D 80F902 <1> cmp cl, 2 ; [esi+LD\_FATType]

1156 00007820 7708 <1> ja short loc\_write\_vol\_FAT\_str\_2

1157 00007822 66B83132 <1> mov ax, '12' ; FAT12

1158 00007826 7202 <1> jb short loc\_write\_vol\_FAT\_str\_2

1159 00007828 B436 <1> mov ah, '6' ; FAT16

1160 <1> loc\_write\_vol\_FAT\_str\_2:

1161 0000782A C70746415420 <1> mov dword [edi], 'FAT '

1162 00007830 66894704 <1> mov word [edi+4], ax

1163 <1> ;

1164 <1> ;movzx ebx, word [esi+LD\_BPB+BPB\_BytsPerSec]

1165 00007834 668B5E11 <1> mov bx, [esi+LD\_BPB+BPB\_BytsPerSec]

1166 00007838 8B4674 <1> mov eax, [esi+LD\_FreeSectors]

1167 <1>

1168 <1> loc\_vol\_freespace\_recalc0:

1169 <1> ; 01/03/2016

1170 0000783B 83F8FF <1> cmp eax, 0FFFFFFFFh

1171 0000783E 720F <1> jb short loc\_vol\_freespace\_mul32

1172 <1> ;inc eax ; 0

1173 00007840 20C9 <1> and cl, cl ; byte [esi+LD\_FATType]

1174 00007842 740B <1> jz short loc\_vol\_freespace\_mul32

1175 00007844 53 <1> push ebx

1176 00007845 66BB00FF <1> mov bx, 0FF00h ; recalculate free sectors

1177 00007849 E876490000 <1> call calculate\_fat\_freespace

1178 0000784E 5B <1> pop ebx

1179 <1>

1180 <1> loc\_vol\_freespace\_mul32:

1181 0000784F F7E3 <1> mul ebx

1182 00007851 09D2 <1> or edx, edx

1183 00007853 7507 <1> jnz short loc\_vol\_fspace\_in\_kbytes

1184 <1> loc\_vol\_fspace\_in\_bytes:

1185 00007855 B9[580E0100] <1> mov ecx, VolSize\_Bytes

1186 0000785A EB0D <1> jmp short loc\_write\_vol\_fspace\_str

1187 <1> loc\_vol\_fspace\_in\_kbytes:

1188 0000785C 66BB0004 <1> mov bx, 1024

1189 00007860 F7F3 <1> div ebx

1190 00007862 B9[4B0E0100] <1> mov ecx, VolSize\_KiloBytes

1191 00007867 31D2 <1> xor edx, edx ; 0

1192 <1> loc\_write\_vol\_fspace\_str:

1193 00007869 890D[3B610100] <1> mov [VolSize\_Unit2], ecx

1194 <1> ;

1195 0000786F BF[5D610100] <1> mov edi, Vol\_Free\_Sectors\_Str\_End

1196 <1> ;mov byte [edi], 0

1197 00007874 B90A000000 <1> mov ecx, 10

1198 <1> loc\_write\_vol\_fspace\_chr:

1199 00007879 F7F1 <1> div ecx

1200 0000787B 80C230 <1> add dl, '0'

1201 0000787E 4F <1> dec edi

1202 0000787F 8817 <1> mov [edi], dl

1203 00007881 85C0 <1> test eax, eax

1204 00007883 7404 <1> jz short loc\_write\_vol\_fspace\_str\_ok

1205 00007885 28D2 <1> sub dl, dl ; 0

1206 00007887 EBF0 <1> jmp short loc\_write\_vol\_fspace\_chr

1207 <1>

1208 <1> loc\_write\_vol\_fspace\_str\_ok:

1209 00007889 893D[4F610100] <1> mov [Vol\_Free\_Sectors\_Str\_Start], edi

1210 <1> ;

1211 0000788F BE[610E0100] <1> mov esi, Volume\_in\_drive

1212 00007894 E8C4EAFFFF <1> call print\_msg

1213 00007899 BE[A10E0100] <1> mov esi, Vol\_Name

1214 0000789E E8BAEAFFFF <1> call print\_msg

1215 000078A3 BE[6F190100] <1> mov esi, nextline

1216 000078A8 E8B0EAFFFF <1> call print\_msg

1217 <1> ;

1218 000078AD BE[020F0100] <1> mov esi, Vol\_Total\_Sector\_Header

1219 000078B2 E8A6EAFFFF <1> call print\_msg

1220 000078B7 8B35[3F610100] <1> mov esi, [Vol\_Tot\_Sec\_Str\_Start]

1221 000078BD E89BEAFFFF <1> call print\_msg

1222 000078C2 8B35[37610100] <1> mov esi, [VolSize\_Unit1]

1223 000078C8 E890EAFFFF <1> call print\_msg

1224 <1> ;

1225 000078CD BE[130F0100] <1> mov esi, Vol\_Free\_Sectors\_Header

1226 000078D2 E886EAFFFF <1> call print\_msg

1227 000078D7 8B35[4F610100] <1> mov esi, [Vol\_Free\_Sectors\_Str\_Start]

1228 000078DD E87BEAFFFF <1> call print\_msg

1229 000078E2 8B35[3B610100] <1> mov esi, [VolSize\_Unit2]

1230 000078E8 E870EAFFFF <1> call print\_msg

1231 <1> ;

1232 000078ED 5E <1> pop esi

1233 <1>

1234 <1> ;mov ah, [esi+LD\_FSType]

1235 <1> ;mov al, [esi+LD\_FATType]

1236 000078EE 668B4603 <1> mov ax, [esi+LD\_FATType]

1237 <1>

1238 000078F2 C3 <1> retn

1239 <1>

1240 <1> move\_volume\_name\_and\_serial\_no:

1241 <1> ; 08/02/2016 (TRDOS 386 = TRDOS v2.0)

1242 <1> ; this routine will be called by

1243 <1> ; "print\_volume\_info" and "print\_directory"

1244 <1> ; INPUT ->

1245 <1> ; ESI = Logical DOS drv descripton table address

1246 <1> ; OUTPUT ->

1247 <1> ; \*Volume name will be moved to text area

1248 <1> ; \*Volume serial number will be converted to

1249 <1> ; text and will be moved to text area

1250 <1> ; cf = 1 -> invalid/unknown dos drive

1251 <1> ; cf = 0 -> ecx = 0

1252 <1> ;

1253 <1> ; (eax, edx, ecx, esi, edi will be changed)

1254 <1>

1255 000078F3 BF[A10E0100] <1> mov edi, Vol\_Name

1256 <1>

1257 <1> ;mov ah, [esi+LD\_FSType]

1258 <1> ;mov al, [esi+LD\_FATType]

1259 000078F8 668B4603 <1> mov ax, [esi+LD\_FATType]

1260 000078FC 80FCA1 <1> cmp ah, 0A1h

1261 000078FF 7418 <1> je short mvn\_2

1262 00007901 08E4 <1> or ah, ah

1263 00007903 7404 <1> jz short mvn\_0

1264 00007905 08C0 <1> or al, al

1265 00007907 7504 <1> jnz short mvn\_1

1266 <1> mvn\_0:

1267 00007909 8A06 <1> mov al, [esi]

1268 0000790B F9 <1> stc

1269 0000790C C3 <1> retn

1270 <1> mvn\_1:

1271 0000790D 3C02 <1> cmp al, 2

1272 0000790F 7717 <1> ja short mvn\_3

1273 <1> ;or al, al

1274 <1> ;jz short mvn\_2

1275 00007911 8B462D <1> mov eax, [esi+LD\_BPB+VolumeID]

1276 00007914 83C631 <1> add esi, LD\_BPB+VolumeLabel

1277 00007917 EB15 <1> jmp short mvn\_4

1278 <1> mvn\_2:

1279 00007919 8B4628 <1> mov eax, [esi+LD\_FS\_VolumeSerial]

1280 0000791C 83C62C <1> add esi, LD\_FS\_VolumeName

1281 0000791F B910000000 <1> mov ecx, 16

1282 00007924 F3A5 <1> rep movsd

1283 00007926 EB10 <1> jmp short mvn\_5

1284 <1> mvn\_3:

1285 00007928 8B4649 <1> mov eax, [esi+LD\_BPB+FAT32\_VolID]

1286 0000792B 83C64D <1> add esi, LD\_BPB+FAT32\_VolLab

1287 <1> mvn\_4:

1288 0000792E B90B000000 <1> mov ecx, 11

1289 00007933 F3A4 <1> rep movsb

1290 00007935 C60700 <1> mov byte [edi], 0

1291 <1> mvn\_5:

1292 <1> ;mov [Current\_VolSerial], eax

1293 00007938 E8CCB9FFFF <1> call dwordtohex

1294 0000793D 8915[F60E0100] <1> mov [Vol\_Serial1], edx

1295 00007943 A3[FB0E0100] <1> mov [Vol\_Serial2], eax

1296 <1> ; ecx = 0

1297 00007948 C3 <1> retn

1298 <1>

1299 <1> get\_volume\_serial\_number:

1300 <1> ; 19/01/2016 (TRDOS 386 = TRDOS v2.0)

1301 <1> ; 08/08/2010

1302 <1> ;

1303 <1> ; INPUT -> DL = Logical DOS Drive number

1304 <1> ; OUTPUT -> EAX = Volume serial number

1305 <1> ; BL= FAT Type

1306 <1> ; BH = Logical DOS drv Number (DL input)

1307 <1> ; cf = 1 -> Drive not ready

1308 <1>

1309 00007949 31DB <1> xor ebx, ebx

1310 0000794B 88D7 <1> mov bh, dl

1311 0000794D 3815[D20C0100] <1> cmp [Last\_DOS\_DiskNo], dl

1312 00007953 7304 <1> jnb short loc\_gvsn\_start

1313 <1> loc\_gvsn\_stc\_retn:

1314 00007955 31C0 <1> xor eax, eax

1315 00007957 F9 <1> stc

1316 00007958 C3 <1> retn

1317 <1> loc\_gvsn\_start:

1318 00007959 56 <1> push esi

1319 0000795A BE00010900 <1> mov esi, Logical\_DOSDisks

1320 0000795F 01DE <1> add esi, ebx

1321 00007961 8A5E03 <1> mov bl, [esi+LD\_FATType]

1322 00007964 20DB <1> and bl, bl

1323 00007966 740F <1> jz short loc\_gvsn\_fs

1324 00007968 80FB02 <1> cmp bl, 2

1325 0000796B 7705 <1> ja short loc\_gvsn\_fat32

1326 <1> loc\_gvsn\_fat:

1327 0000796D 83C62D <1> add esi, LD\_BPB + VolumeID

1328 00007970 EB0E <1> jmp short loc\_gvsn\_return

1329 <1> loc\_gvsn\_fat32:

1330 00007972 83C649 <1> add esi, LD\_BPB + FAT32\_VolID

1331 00007975 EB09 <1> jmp short loc\_gvsn\_return

1332 <1> loc\_gvsn\_fs:

1333 00007977 807E04A1 <1> cmp byte [esi+LD\_FSType], 0A1h

1334 0000797B 75D8 <1> jne short loc\_gvsn\_stc\_retn

1335 0000797D 83C628 <1> add esi, LD\_FS\_VolumeSerial

1336 <1> loc\_gvsn\_return:

1337 00007980 8B06 <1> mov eax, [esi]

1338 00007982 5E <1> pop esi

1339 00007983 C3 <1> retn

1340 <1>

1341 <1> ; CMD\_INTR.ASM [ TRDOS Command Interpreter Procedure ]

1342 <1> ; 09/11/2011

1343 <1> ; 29/01/2005

1344 <1>

1345 <1> command\_interpreter:

1346 <1> ; 16/10/2016

1347 <1> ; 12/10/2016

1348 <1> ; 13/05/2016

1349 <1> ; 07/05/2016

1350 <1> ; 04/03/2016

1351 <1> ; 04/02/2016

1352 <1> ; 03/02/2016

1353 <1> ; 30/01/2016

1354 <1> ; 29/01/2016 (TRDOS 386 = TRDOS 2.0)

1355 <1> ; 15/09/2011

1356 <1> ; 29/01/2005

1357 <1>

1358 <1> ; Input: ecx = command word length (CL)

1359 <1> ; CommandBuffer = Command string offset

1360 <1>

1361 00007984 C605[F0610100]00 <1> mov byte [Program\_Exit],0

1362 0000798B 80F904 <1> cmp cl, 4

1363 0000798E 0F87B5020000 <1> ja c\_6

1364 00007994 0F8237010000 <1> jb c\_2

1365 <1> c\_4:

1366 <1>

1367 <1> cmp\_cmd\_exit:

1368 0000799A BF[400D0100] <1> mov edi, Cmd\_Exit

1369 0000799F E8C2030000 <1> call cmp\_cmd

1370 000079A4 7208 <1> jc short cmp\_cmd\_date

1371 <1>

1372 000079A6 C605[F0610100]01 <1> mov byte [Program\_Exit], 1

1373 000079AD C3 <1> retn

1374 <1>

1375 <1> cmp\_cmd\_date:

1376 000079AE B104 <1> mov cl, 4

1377 000079B0 BF[5C0D0100] <1> mov edi, Cmd\_Date

1378 000079B5 E8AC030000 <1> call cmp\_cmd

1379 000079BA 720B <1> jc short cmp\_cmd\_time

1380 <1>

1381 000079BC E8D0F7FFFF <1> call show\_date

1382 000079C1 E80FF8FFFF <1> call set\_date

1383 000079C6 C3 <1> retn

1384 <1>

1385 <1> cmp\_cmd\_time:

1386 000079C7 B104 <1> mov cl, 4

1387 000079C9 BF[610D0100] <1> mov edi, Cmd\_Time

1388 000079CE E893030000 <1> call cmp\_cmd

1389 000079D3 720B <1> jc short cmp\_cmd\_show

1390 <1>

1391 000079D5 E8C6FAFFFF <1> call show\_time

1392 000079DA E8F8FAFFFF <1> call set\_time

1393 000079DF C3 <1> retn

1394 <1>

1395 <1> cmp\_cmd\_show:

1396 000079E0 B104 <1> mov cl, 4

1397 000079E2 BF[720D0100] <1> mov edi, Cmd\_Show

1398 000079E7 E87A030000 <1> call cmp\_cmd

1399 000079EC 0F83050A0000 <1> jnc show\_file

1400 <1>

1401 <1> cmp\_cmd\_echo:

1402 000079F2 B104 <1> mov cl, 4

1403 000079F4 BF[AE0D0100] <1> mov edi, Cmd\_Echo

1404 000079F9 E868030000 <1> call cmp\_cmd

1405 000079FE 7224 <1> jc short cmp\_cmd\_copy

1406 <1>

1407 <1> ; 22/11/2017

1408 <1> ; AL = 0

1409 00007A00 803E20 <1> cmp byte [esi], 20h

1410 00007A03 7215 <1> jb short cmd\_echo\_nextline

1411 <1> ; 14/04/2016

1412 00007A05 56 <1> push esi

1413 <1> cmd\_echo\_asciiz:

1414 <1> ;inc esi

1415 <1> ;mov al, [esi]

1416 <1> ; 22/11/2017

1417 00007A06 AC <1> lodsb

1418 00007A07 3C20 <1> cmp al, 20h

1419 00007A09 73FB <1> jnb short cmd\_echo\_asciiz

1420 00007A0B 4E <1> dec esi

1421 00007A0C C60600 <1> mov byte [esi], 0

1422 00007A0F 5E <1> pop esi

1423 00007A10 89F7 <1> mov edi, esi

1424 00007A12 E846E9FFFF <1> call print\_msg

1425 00007A17 C60700 <1> mov byte [edi], 0

1426 <1> cmd\_echo\_nextline:

1427 00007A1A BE[B8190100] <1> mov esi, NextLine

1428 <1> ;call print\_msg

1429 <1> ;retn

1430 00007A1F E939E9FFFF <1> jmp print\_msg

1431 <1>

1432 <1> cmp\_cmd\_copy:

1433 00007A24 B104 <1> mov cl, 4

1434 00007A26 BF[950D0100] <1> mov edi, Cmd\_Copy

1435 00007A2B E836030000 <1> call cmp\_cmd

1436 00007A30 0F83CC170000 <1> jnc copy\_file

1437 <1>

1438 <1> cmp\_cmd\_move:

1439 00007A36 B104 <1> mov cl, 4

1440 00007A38 BF[9A0D0100] <1> mov edi, Cmd\_Move

1441 00007A3D E824030000 <1> call cmp\_cmd

1442 00007A42 0F836E160000 <1> jnc move\_file

1443 <1>

1444 <1> cmp\_cmd\_path:

1445 00007A48 B104 <1> mov cl, 4

1446 00007A4A BF[9F0D0100] <1> mov edi, Cmd\_Path

1447 00007A4F E812030000 <1> call cmp\_cmd

1448 00007A54 0F83F0190000 <1> jnc set\_get\_path

1449 <1>

1450 <1> cmp\_cmd\_beep:

1451 00007A5A B104 <1> mov cl, 4

1452 00007A5C BF[CC0D0100] <1> mov edi, Cmd\_Beep

1453 00007A61 E800030000 <1> call cmp\_cmd

1454 00007A66 720B <1> jc short cmp\_cmd\_find

1455 <1> ; 13/05/2016

1456 00007A68 8A3D[66580100] <1> mov bh, [ptty] ; [ACTIVE\_PAGE]

1457 00007A6E E91FA3FFFF <1> jmp beeper

1458 <1>

1459 <1> cmp\_cmd\_find:

1460 00007A73 B104 <1> mov cl, 4

1461 00007A75 BF[A90D0100] <1> mov edi, Cmd\_Find

1462 00007A7A E8E7020000 <1> call cmp\_cmd

1463 00007A7F 0F82C4020000 <1> jc cmp\_cmd\_external

1464 <1>

1465 <1> ;call find\_and\_list\_files

1466 00007A85 E9AF220000 <1> jmp find\_and\_list\_files

1467 <1> ;retn

1468 <1>

1469 <1> c\_1:

1470 00007A8A AD <1> lodsd

1471 <1> cmp\_cmd\_help:

1472 00007A8B 3C3F <1> cmp al, '?'

1473 00007A8D 751D <1> jne short cmp\_cmd\_remark

1474 <1>

1475 00007A8F BE[320D0100] <1> mov esi, Command\_List

1476 <1> cmd\_help\_next\_w:

1477 00007A94 E8C4E8FFFF <1> call print\_msg

1478 <1>

1479 00007A99 803E20 <1> cmp byte [esi], 20h ; 0

1480 00007A9C 7232 <1> jb short cmd\_help\_retn

1481 <1>

1482 00007A9E 56 <1> push esi

1483 00007A9F BE[6F190100] <1> mov esi, nextline

1484 00007AA4 E8B4E8FFFF <1> call print\_msg

1485 00007AA9 5E <1> pop esi

1486 00007AAA EBE8 <1> jmp short cmd\_help\_next\_w

1487 <1>

1488 <1> cmp\_cmd\_remark:

1489 00007AAC 3C2A <1> cmp al, '\*'

1490 00007AAE 0F8595020000 <1> jne cmp\_cmd\_external

1491 00007AB4 46 <1> inc esi

1492 00007AB5 BF[60590100] <1> mov edi, Remark

1493 00007ABA 8A06 <1> mov al, [esi]

1494 00007ABC 3C20 <1> cmp al, 20h

1495 00007ABE 7707 <1> ja short cmd\_remark\_write

1496 00007AC0 89FE <1> mov esi, edi ; Remark

1497 00007AC2 E996E8FFFF <1> jmp print\_msg

1498 <1>

1499 <1> cmd\_remark\_write:

1500 00007AC7 AA <1> stosb

1501 00007AC8 AC <1> lodsb

1502 00007AC9 3C20 <1> cmp al, 20h

1503 00007ACB 73FA <1> jnb short cmd\_remark\_write

1504 00007ACD C60700 <1> mov byte [edi], 0

1505 <1>

1506 <1> cmd\_help\_retn:

1507 <1> cmd\_remark\_retn:

1508 <1> cd\_retn:

1509 00007AD0 C3 <1> retn

1510 <1>

1511 <1> c\_2:

1512 00007AD1 80F902 <1> cmp cl, 2

1513 00007AD4 0F87AF000000 <1> ja c\_3

1514 00007ADA BE[AE590100] <1> mov esi, CommandBuffer

1515 00007ADF 72A9 <1> jb short c\_1

1516 <1>

1517 <1> cmp\_cmd\_cd:

1518 00007AE1 66AD <1> lodsw

1519 00007AE3 663D4344 <1> cmp ax, 'CD'

1520 00007AE7 7551 <1> jne short cmp\_cmd\_drive

1521 00007AE9 46 <1> inc esi

1522 <1> cd\_0:

1523 00007AEA 668B06 <1> mov ax, [esi]

1524 00007AED 3C20 <1> cmp al, 20h

1525 00007AEF 76DF <1> jna short cd\_retn

1526 <1> ; 10/02/2016

1527 00007AF1 80FC3A <1> cmp ah, ':'

1528 00007AF4 7504 <1> jne short cd\_1

1529 00007AF6 46 <1> inc esi

1530 00007AF7 46 <1> inc esi

1531 00007AF8 EB49 <1> jmp short cd\_2

1532 <1>

1533 <1> cd\_1: ; change current directory

1534 <1> ; 29/11/2009

1535 <1> ; AH = CDh ; to separate 'CD' command from others

1536 <1> ; for restoring current directory

1537 <1> ; 0CDh sign is for saving cdir into

1538 <1> ; DOS drv description table cdir area

1539 <1>

1540 00007AFA B4CD <1> mov ah, 0CDh ; mov byte [CD\_COMMAND], 0CDh

1541 <1>

1542 00007AFC E81D230000 <1> call change\_current\_directory

1543 00007B01 0F8337220000 <1> jnc change\_prompt\_dir\_string

1544 <1>

1545 <1> cd\_error\_messages:

1546 00007B07 3C03 <1> cmp al, 3

1547 00007B09 740C <1> je short cd\_path\_not\_found

1548 <1> ; 16/10/2016 (15h -> 15)

1549 00007B0B 3C0F <1> cmp al, 15 ; drive not ready error

1550 00007B0D 7459 <1> je short cd\_drive\_not\_ready

1551 00007B0F 3C11 <1> cmp al, 17 ; read error

1552 00007B11 7455 <1> je short cd\_drive\_not\_ready

1553 00007B13 3C13 <1> cmp al, 19 ; ; Bad directory/path name

1554 00007B15 7466 <1> je short cd\_command\_failed

1555 <1>

1556 <1> cd\_path\_not\_found:

1557 00007B17 50 <1> push eax ; 29/12/2017

1558 <1> ;push ax

1559 00007B18 BE[D50F0100] <1> mov esi, Msg\_Dir\_Not\_Found

1560 00007B1D E83BE8FFFF <1> call print\_msg

1561 <1> ;pop ax

1562 00007B22 58 <1> pop eax ; 29/12/2017

1563 00007B23 3A25[FC580100] <1> cmp ah, [Current\_Dir\_Level]

1564 00007B29 0F830F220000 <1> jnb change\_prompt\_dir\_string

1565 00007B2F 8825[FC580100] <1> mov [Current\_Dir\_Level], ah

1566 00007B35 E904220000 <1> jmp change\_prompt\_dir\_string

1567 <1>

1568 <1> cmp\_cmd\_drive: ; change current drive

1569 <1> ; C:, D:, E: etc.

1570 00007B3A 80FC3A <1> cmp ah, ':'

1571 00007B3D 0F8506020000 <1> jne cmp\_cmd\_external

1572 <1>

1573 <1> cd\_2: ; 'CD C:', 'CD D:' ...

1574 00007B43 803E20 <1> cmp byte [esi], 20h

1575 00007B46 0F8707020000 <1> ja loc\_cmd\_failed

1576 <1>

1577 00007B4C 24DF <1> and al, 0DFh

1578 00007B4E 2C41 <1> sub al, 'A'

1579 00007B50 0F82FD010000 <1> jc loc\_cmd\_failed

1580 <1>

1581 00007B56 3A05[D20C0100] <1> cmp al, [Last\_DOS\_DiskNo]

1582 00007B5C 770A <1> ja short cd\_drive\_not\_ready

1583 <1>

1584 00007B5E 88C2 <1> mov dl, al

1585 00007B60 E85BF3FFFF <1> call change\_current\_drive

1586 00007B65 7201 <1> jc short cd\_drive\_not\_ready

1587 00007B67 C3 <1> retn

1588 <1>

1589 <1> cd\_drive\_not\_ready:

1590 00007B68 BE[920F0100] <1> mov esi, Msg\_Not\_Ready\_Read\_Err

1591 00007B6D E8EBE7FFFF <1> call print\_msg

1592 <1>

1593 <1> cd\_fail\_drive\_restart:

1594 00007B72 8A15[FE580100] <1> mov dl, [Current\_Drv]

1595 <1> ;call change\_current\_drive

1596 00007B78 E943F3FFFF <1> jmp change\_current\_drive

1597 <1> ;retn

1598 <1>

1599 <1> cd\_command\_failed:

1600 00007B7D BE[730F0100] <1> mov esi, Msg\_Bad\_Command

1601 00007B82 E8D6E7FFFF <1> call print\_msg

1602 00007B87 EBE9 <1> jmp short cd\_fail\_drive\_restart

1603 <1>

1604 <1> c\_3:

1605 <1> cmp\_cmd\_dir:

1606 00007B89 BF[320D0100] <1> mov edi, Cmd\_Dir

1607 00007B8E E8D3010000 <1> call cmp\_cmd

1608 00007B93 0F8380020000 <1> jnc print\_directory\_list

1609 <1>

1610 <1> cmp\_cmd\_cls:

1611 00007B99 B103 <1> mov cl, 3

1612 00007B9B BF[6E0D0100] <1> mov edi, Cmd\_Cls

1613 00007BA0 E8C1010000 <1> call cmp\_cmd

1614 00007BA5 0F83C8E7FFFF <1> jnc clear\_screen

1615 <1>

1616 <1> cmp\_cmd\_ver:

1617 00007BAB B103 <1> mov cl, 3

1618 00007BAD BF[3C0D0100] <1> mov edi, Cmd\_Ver

1619 00007BB2 E8AF010000 <1> call cmp\_cmd

1620 00007BB7 720A <1> jc short cmp\_cmd\_mem

1621 <1>

1622 00007BB9 BE[DA0C0100] <1> mov esi, mainprog\_Version

1623 <1> ;call print\_msg

1624 00007BBE E99AE7FFFF <1> jmp print\_msg

1625 <1> ;retn

1626 <1>

1627 <1> cmp\_cmd\_mem:

1628 00007BC3 B103 <1> mov cl, 3

1629 00007BC5 BF[A40D0100] <1> mov edi, Cmd\_Mem

1630 00007BCA E897010000 <1> call cmp\_cmd

1631 00007BCF 0F837FB6FFFF <1> jnc memory\_info

1632 <1>

1633 <1> cmp\_cmd\_del:

1634 00007BD5 B103 <1> mov cl, 3

1635 00007BD7 BF[770D0100] <1> mov edi, Cmd\_Del

1636 00007BDC E885010000 <1> call cmp\_cmd

1637 00007BE1 0F83280F0000 <1> jnc delete\_file

1638 <1>

1639 <1> cmp\_cmd\_set:

1640 00007BE7 B103 <1> mov cl, 3

1641 00007BE9 BF[6A0D0100] <1> mov edi, Cmd\_Set

1642 00007BEE E873010000 <1> call cmp\_cmd

1643 00007BF3 0F83C9170000 <1> jnc set\_get\_env

1644 <1>

1645 <1> cmp\_cmd\_run:

1646 00007BF9 B103 <1> mov cl, 3

1647 00007BFB BF[660D0100] <1> mov edi, Cmd\_Run

1648 00007C00 E861010000 <1> call cmp\_cmd

1649 <1> ; 07/05/2016

1650 00007C05 0F823E010000 <1> jc cmp\_cmd\_external

1651 00007C0B E90F1E0000 <1> jmp load\_and\_execute\_file

1652 <1> c\_5:

1653 <1> cmp\_cmd\_mkdir:

1654 00007C10 BF[8F0D0100] <1> mov edi, Cmd\_Mkdir

1655 00007C15 E84C010000 <1> call cmp\_cmd

1656 00007C1A 0F83990A0000 <1> jnc make\_directory

1657 <1>

1658 <1> cmp\_cmd\_rmdir:

1659 00007C20 B105 <1> mov cl, 5

1660 00007C22 BF[890D0100] <1> mov edi, Cmd\_Rmdir

1661 00007C27 E83A010000 <1> call cmp\_cmd

1662 00007C2C 0F83AA0B0000 <1> jnc delete\_directory

1663 <1>

1664 <1> cmp\_cmd\_chdir:

1665 00007C32 B105 <1> mov cl, 5

1666 00007C34 BF[C60D0100] <1> mov edi, Cmd\_Chdir

1667 00007C39 E828010000 <1> call cmp\_cmd

1668 00007C3E 0F8205010000 <1> jc cmp\_cmd\_external

1669 <1>

1670 00007C44 E9A1FEFFFF <1> jmp cd\_0

1671 <1>

1672 <1> c\_6:

1673 00007C49 80F906 <1> cmp cl, 6

1674 00007C4C 0F87E0000000 <1> ja c\_8

1675 00007C52 72BC <1> jb short c\_5

1676 <1> cmp\_cmd\_prompt:

1677 00007C54 BF[450D0100] <1> mov edi, Cmd\_Prompt

1678 00007C59 E808010000 <1> call cmp\_cmd

1679 00007C5E 722F <1> jc short cmp\_cmd\_volume

1680 <1> get\_prompt\_name\_fchar:

1681 00007C60 AC <1> lodsb

1682 00007C61 3C20 <1> cmp al, 20h

1683 00007C63 74FB <1> je short get\_prompt\_name\_fchar

1684 00007C65 7713 <1> ja short loc\_change\_prompt\_label

1685 <1> default\_command\_prompt: ; 31/12/2017 ('sysprompt')

1686 00007C67 BE[260D0100] <1> mov esi, TRDOSPromptLabel

1687 00007C6C C7065452444F <1> mov dword [esi], "TRDO"

1688 00007C72 66C746045300 <1> mov word [esi+4], "S"

1689 <1> loc\_cmd\_prompt\_return:

1690 00007C78 C3 <1> retn

1691 <1>

1692 <1> set\_command\_prompt: ; 31/12/2017 ('sysprompt')

1693 00007C79 AC <1> lodsb

1694 <1> loc\_change\_prompt\_label:

1695 00007C7A 66B90B00 <1> mov cx, 11

1696 00007C7E BF[260D0100] <1> mov edi, TRDOSPromptLabel

1697 <1> put\_char\_new\_prompt\_label:

1698 00007C83 AA <1> stosb

1699 00007C84 AC <1> lodsb

1700 00007C85 3C20 <1> cmp al, 20h

1701 00007C87 7202 <1> jb short pass\_put\_new\_prompt\_label

1702 00007C89 E2F8 <1> loop put\_char\_new\_prompt\_label

1703 <1> pass\_put\_new\_prompt\_label:

1704 00007C8B C60700 <1> mov byte [edi], 0

1705 00007C8E C3 <1> retn

1706 <1>

1707 <1> cmp\_cmd\_volume:

1708 00007C8F B106 <1> mov cl, 6

1709 00007C91 BF[4C0D0100] <1> mov edi, Cmd\_Volume

1710 00007C96 E8CB000000 <1> call cmp\_cmd

1711 00007C9B 7255 <1> jc short cmp\_cmd\_attrib

1712 <1>

1713 <1> cmd\_vol1:

1714 00007C9D AC <1> lodsb

1715 00007C9E 3C20 <1> cmp al, 20h

1716 00007CA0 7707 <1> ja short cmd\_vol2

1717 00007CA2 A0[FE580100] <1> mov al, [Current\_Drv]

1718 00007CA7 EB3D <1> jmp short cmd\_vol4

1719 <1> cmd\_vol2:

1720 00007CA9 3C41 <1> cmp al, 'A'

1721 00007CAB 0F82A2000000 <1> jb loc\_cmd\_failed

1722 00007CB1 3C7A <1> cmp al, 'z'

1723 00007CB3 0F879A000000 <1> ja loc\_cmd\_failed

1724 00007CB9 3C5A <1> cmp al, 'Z'

1725 00007CBB 760A <1> jna short cmd\_vol3

1726 00007CBD 3C61 <1> cmp al, 'a'

1727 00007CBF 0F828E000000 <1> jb loc\_cmd\_failed

1728 00007CC5 24DF <1> and al, 0DFh

1729 <1> cmd\_vol3:

1730 00007CC7 8A26 <1> mov ah, [esi]

1731 00007CC9 80FC3A <1> cmp ah, ':'

1732 00007CCC 0F8581000000 <1> jne loc\_cmd\_failed

1733 00007CD2 2C41 <1> sub al, 'A'

1734 00007CD4 3A05[D20C0100] <1> cmp al, [Last\_DOS\_DiskNo]

1735 00007CDA 760A <1> jna short cmd\_vol4

1736 <1>

1737 00007CDC BE[920F0100] <1> mov esi, Msg\_Not\_Ready\_Read\_Err

1738 00007CE1 E977E6FFFF <1> jmp print\_msg

1739 <1>

1740 <1> cmd\_vol4:

1741 00007CE6 E88EFAFFFF <1> call print\_volume\_info

1742 00007CEB 0F8277FEFFFF <1> jc cd\_drive\_not\_ready

1743 00007CF1 C3 <1> retn

1744 <1>

1745 <1> cmp\_cmd\_attrib:

1746 00007CF2 B106 <1> mov cl, 6

1747 00007CF4 BF[7B0D0100] <1> mov edi, Cmd\_Attrib

1748 00007CF9 E868000000 <1> call cmp\_cmd

1749 00007CFE 0F831D0F0000 <1> jnc set\_file\_attributes

1750 <1>

1751 <1> cmp\_cmd\_rename:

1752 00007D04 B106 <1> mov cl, 6

1753 00007D06 BF[820D0100] <1> mov edi, Cmd\_Rename

1754 00007D0B E856000000 <1> call cmp\_cmd

1755 00007D10 0F8353110000 <1> jnc rename\_file

1756 <1>

1757 <1> cmp\_cmd\_device:

1758 00007D16 B106 <1> mov cl, 6

1759 00007D18 BF[B70D0100] <1> mov edi, Cmd\_Device

1760 00007D1D E844000000 <1> call cmp\_cmd

1761 00007D22 7225 <1> jc short cmp\_cmd\_external

1762 <1>

1763 00007D24 C3 <1> retn

1764 <1>

1765 <1> c\_7:

1766 <1> cmp\_cmd\_devlist:

1767 00007D25 BF[BE0D0100] <1> mov edi, Cmd\_DevList

1768 00007D2A E837000000 <1> call cmp\_cmd

1769 00007D2F 7218 <1> jc short cmp\_cmd\_external

1770 <1>

1771 <1> loc\_cmd\_return:

1772 00007D31 C3 <1> retn

1773 <1>

1774 <1> c\_8:

1775 00007D32 80F908 <1> cmp cl, 8

1776 00007D35 7712 <1> ja short cmp\_cmd\_external

1777 00007D37 72EC <1> jb short c\_7

1778 <1>

1779 <1> cmp\_cmd\_longname:

1780 00007D39 BF[530D0100] <1> mov edi, Cmd\_LongName

1781 00007D3E E823000000 <1> call cmp\_cmd

1782 00007D43 0F8350060000 <1> jnc get\_and\_print\_longname

1783 <1>

1784 <1> cmp\_cmd\_external:

1785 <1> ; 07/05/2016

1786 <1> ; 22/04/2016

1787 00007D49 BE[AE590100] <1> mov esi, CommandBuffer

1788 00007D4E E9CC1C0000 <1> jmp loc\_run\_check\_filename

1789 <1>

1790 <1> loc\_cmd\_failed:

1791 00007D53 803D[AE590100]20 <1> cmp byte [CommandBuffer], 20h

1792 00007D5A 76D5 <1> jna short loc\_cmd\_return

1793 00007D5C BE[730F0100] <1> mov esi, Msg\_Bad\_Command

1794 <1> ; call print\_msg

1795 <1> ;loc\_cmd\_return:

1796 <1> ; retn

1797 00007D61 E9F7E5FFFF <1> jmp print\_msg

1798 <1>

1799 <1> cmp\_cmd:

1800 <1> ; 29/01/2016 (TRDOS 386 = TRDOS v2.0)

1801 00007D66 BE[AE590100] <1> mov esi, CommandBuffer

1802 <1> ; edi = internal command word (ASCIIZ)

1803 <1> ; ecx = command length (<=8)

1804 <1> cmp\_cmd\_1:

1805 00007D6B AC <1> lodsb

1806 00007D6C AE <1> scasb

1807 00007D6D 750D <1> jne short cmp\_cmd\_3

1808 00007D6F E2FA <1> loop cmp\_cmd\_1

1809 00007D71 AC <1> lodsb

1810 00007D72 3C20 <1> cmp al, 20h

1811 00007D74 7703 <1> ja short cmp\_cmd\_2

1812 00007D76 30C0 <1> xor al, al

1813 <1> ; ZF = 1 -> internal command word matches

1814 00007D78 C3 <1> retn

1815 <1> cmp\_cmd\_2:

1816 <1> ; ZF = 0 (CF = 0) -> external command word

1817 00007D79 58 <1> pop eax ; no return to the caller from here

1818 00007D7A EBCD <1> jmp cmp\_cmd\_external

1819 <1> cmp\_cmd\_3:

1820 00007D7C F9 <1> stc

1821 <1> ; CF = 1 -> internal command word does not match

1822 00007D7D C3 <1> retn

1823 <1>

1824 <1> loc\_run\_cmd\_failed:

1825 <1> ; 15/03/2016

1826 <1> ; 15/02/2016 (TRDOS 386 = TRDOS v2.0)

1827 <1> ; 07/12/2009 (CMD\_INTR.ASM)

1828 <1> ; 29/11/2009

1829 <1>

1830 00007D7E E863000000 <1> call restore\_cdir\_after\_cmd\_fail

1831 <1>

1832 <1> loc\_run\_cmd\_failed\_cmp\_al:

1833 <1> ; End of Restore\_CDIR code (29/11/2009)

1834 <1>

1835 00007D83 3C01 <1> cmp al, 1 ; Bad command or file name

1836 00007D85 74CC <1> je loc\_cmd\_failed

1837 <1> loc\_run\_dir\_not\_found:

1838 00007D87 3C03 <1> cmp al, 3

1839 00007D89 750A <1> jne short loc\_run\_file\_notfound\_msg

1840 <1> ; Path not found (MS-DOS Error Code = 3)

1841 00007D8B BE[D50F0100] <1> mov esi, Msg\_Dir\_Not\_Found

1842 00007D90 E9C8E5FFFF <1> jmp print\_msg

1843 <1>

1844 <1> loc\_run\_file\_notfound\_msg:

1845 00007D95 3C02 <1> cmp al, 2 ; File not found

1846 00007D97 750A <1> jne short loc\_run\_file\_drv\_read\_err

1847 <1>

1848 <1> loc\_print\_file\_notfound\_msg:

1849 00007D99 BE[EC0F0100] <1> mov esi, Msg\_File\_Not\_Found

1850 <1> ;call proc\_printmsg

1851 <1> ;retn

1852 00007D9E E9BAE5FFFF <1> jmp print\_msg

1853 <1>

1854 <1> loc\_run\_file\_drv\_read\_err:

1855 <1> ; Err: 17 (Read fault)

1856 00007DA3 3C11 <1> cmp al, 17 ; Drive not ready or read error

1857 00007DA5 7404 <1> je short loc\_run\_file\_print\_drv\_read\_err

1858 <1> ;

1859 00007DA7 3C0F <1> cmp al, 15 ; Drive not ready (or read error)

1860 00007DA9 750A <1> jne short loc\_run\_file\_toobig

1861 <1>

1862 <1> loc\_run\_file\_print\_drv\_read\_err:

1863 00007DAB BE[920F0100] <1> mov esi, Msg\_Not\_Ready\_Read\_Err

1864 00007DB0 E9A8E5FFFF <1> jmp print\_msg

1865 <1>

1866 <1> loc\_run\_file\_toobig:

1867 00007DB5 3C08 <1> cmp al, 8 ; Not enough free memory to load&run file

1868 00007DB7 750A <1> jne short loc\_run\_file\_perm\_denied

1869 00007DB9 BE[37100100] <1> mov esi, Msg\_Insufficient\_Memory

1870 00007DBE E99AE5FFFF <1> jmp print\_msg

1871 <1>

1872 <1> loc\_run\_file\_perm\_denied:

1873 <1> ; 29/12/2017

1874 00007DC3 3C0B <1> cmp al, ERR\_PERM\_DENIED ; 11 ; Permission denied

1875 00007DC5 750A <1> jne short loc\_run\_misc\_error

1876 00007DC7 BE[CC110100] <1> mov esi, Msg\_Permission\_Denied

1877 00007DCC E98CE5FFFF <1> jmp print\_msg

1878 <1>

1879 <1> ; 15/03/2016

1880 <1> print\_misc\_error\_msg:

1881 <1> loc\_run\_misc\_error:

1882 <1> ; AL = Error code

1883 00007DD1 E8F3B4FFFF <1> call bytetohex

1884 00007DD6 66A3[6B100100] <1> mov [error\_code\_hex], ax

1885 <1>

1886 00007DDC BE[4E100100] <1> mov esi, Msg\_Error\_Code

1887 <1> ;call print\_msg

1888 <1> ;retn

1889 <1>

1890 00007DE1 E977E5FFFF <1> jmp print\_msg

1891 <1>

1892 <1> restore\_cdir\_after\_cmd\_fail:

1893 <1> ; 15/02/2016 (TRDOS 386 = TRDOS v2.0)

1894 00007DE6 50 <1> push eax

1895 00007DE7 8A3D[5E610100] <1> mov bh, [RUN\_CDRV] ; it is set at the beginning

1896 <1> ; of the 'run' command.

1897 00007DED 3A3D[FE580100] <1> cmp bh, [Current\_Drv]

1898 00007DF3 7409 <1> je short loc\_run\_restore\_cdir

1899 00007DF5 88FA <1> mov dl, bh

1900 00007DF7 E8C4F0FFFF <1> call change\_current\_drive

1901 00007DFC EB19 <1> jmp short loc\_run\_err\_pass\_restore\_cdir

1902 <1>

1903 <1> loc\_run\_restore\_cdir:

1904 00007DFE 803D[D30C0100]00 <1> cmp byte [Restore\_CDIR], 0

1905 00007E05 7610 <1> jna short loc\_run\_err\_pass\_restore\_cdir

1906 00007E07 30DB <1> xor bl, bl

1907 00007E09 0FB7F3 <1> movzx esi, bx

1908 00007E0C 81C600010900 <1> add esi, Logical\_DOSDisks

1909 00007E12 E860F1FFFF <1> call restore\_current\_directory

1910 <1>

1911 <1> loc\_run\_err\_pass\_restore\_cdir:

1912 00007E17 58 <1> pop eax

1913 00007E18 C3 <1> retn

1914 <1>

1915 <1> print\_directory\_list:

1916 <1> ; 10/02/2016

1917 <1> ; 08/02/2016 (TRDOS 386 = TRDOS v2.0)

1918 <1> ; 06/12/2009 ('cmp\_cmd\_dir')

1919 <1> ;

1920 00007E19 66C705[A0620100]00- <1> mov word [AttributesMask], 0800h ; ..except volume names..

1920 00007E21 08 <1>

1921 00007E22 A0[FE580100] <1> mov al, [Current\_Drv]

1922 00007E27 A2[5E610100] <1> mov [RUN\_CDRV], al

1923 <1> get\_dfname\_fchar:

1924 00007E2C AC <1> lodsb

1925 00007E2D 3C20 <1> cmp al, 20h

1926 00007E2F 74FB <1> je short get\_dfname\_fchar

1927 00007E31 0F82A4000000 <1> jb loc\_print\_dir\_call\_all

1928 00007E37 3C2D <1> cmp al, '-'

1929 00007E39 7542 <1> jne short loc\_print\_dir\_call\_flt

1930 <1> get\_next\_attr\_char:

1931 00007E3B AC <1> lodsb

1932 00007E3C 3C20 <1> cmp al, 20h

1933 00007E3E 74FB <1> je short get\_next\_attr\_char

1934 00007E40 0F820DFFFFFF <1> jb loc\_cmd\_failed

1935 00007E46 24DF <1> and al, 0DFh

1936 00007E48 3C44 <1> cmp al, 'D' ; directories only ?

1937 00007E4A 7512 <1> jne short pass\_only\_directories

1938 00007E4C AC <1> lodsb

1939 00007E4D 3C20 <1> cmp al, 20h

1940 00007E4F 0F87FEFEFFFF <1> ja loc\_cmd\_failed

1941 00007E55 800D[A0620100]10 <1> or byte [AttributesMask], 10h ; ..directory..

1942 00007E5C EB18 <1> jmp short get\_dfname\_fchar\_attr

1943 <1> pass\_only\_directories:

1944 00007E5E 3C46 <1> cmp al, 'F' ; files only ?

1945 00007E60 0F85B0000000 <1> jne check\_attr\_s

1946 00007E66 AC <1> lodsb

1947 00007E67 3C20 <1> cmp al, 20h

1948 00007E69 0F87E4FEFFFF <1> ja loc\_cmd\_failed

1949 00007E6F 800D[A1620100]10 <1> or byte [AttributesMask+1], 10h ; ..except directories..

1950 <1> get\_dfname\_fchar\_attr:

1951 00007E76 AC <1> lodsb

1952 00007E77 3C20 <1> cmp al, 20h

1953 00007E79 74FB <1> je short get\_dfname\_fchar\_attr

1954 00007E7B 725E <1> jb short loc\_print\_dir\_call\_all

1955 <1>

1956 <1> loc\_print\_dir\_call\_flt:

1957 00007E7D 4E <1> dec esi

1958 00007E7E BF[A2620100] <1> mov edi, FindFile\_Drv

1959 00007E83 E8AC250000 <1> call parse\_path\_name

1960 00007E88 7308 <1> jnc short loc\_print\_dir\_change\_drv\_1

1961 00007E8A 3C01 <1> cmp al, 1

1962 00007E8C 0F87ECFEFFFF <1> ja loc\_run\_cmd\_failed

1963 <1>

1964 <1> loc\_print\_dir\_change\_drv\_1:

1965 00007E92 8A15[A2620100] <1> mov dl, [FindFile\_Drv]

1966 <1> loc\_print\_dir\_change\_drv\_2:

1967 00007E98 3A15[5E610100] <1> cmp dl, [RUN\_CDRV]

1968 00007E9E 740B <1> je short loc\_print\_dir\_change\_directory

1969 00007EA0 E81BF0FFFF <1> call change\_current\_drive

1970 00007EA5 0F82D3FEFFFF <1> jc loc\_run\_cmd\_failed

1971 <1> loc\_print\_dir\_change\_directory:

1972 00007EAB 803D[A3620100]20 <1> cmp byte [FindFile\_Directory], 20h ; 0 or 20h ?

1973 00007EB2 761D <1> jna short pass\_print\_dir\_change\_directory

1974 <1>

1975 00007EB4 FE05[D30C0100] <1> inc byte [Restore\_CDIR]

1976 00007EBA BE[A3620100] <1> mov esi, FindFile\_Directory

1977 00007EBF 30E4 <1> xor ah, ah ; CD\_COMMAND sign -> 0

1978 00007EC1 E8581F0000 <1> call change\_current\_directory

1979 00007EC6 0F82B2FEFFFF <1> jc loc\_run\_cmd\_failed

1980 <1>

1981 <1> loc\_print\_dir\_change\_prompt\_dir\_string:

1982 00007ECC E86D1E0000 <1> call change\_prompt\_dir\_string

1983 <1>

1984 <1> pass\_print\_dir\_change\_directory:

1985 00007ED1 BE[E4620100] <1> mov esi, FindFile\_Name

1986 00007ED6 803E20 <1> cmp byte [esi], 20h ; ; 0 or 20h ?

1987 00007ED9 7706 <1> ja short loc\_print\_dir\_call

1988 <1>

1989 <1> loc\_print\_dir\_call\_all:

1990 00007EDB C7062A2E2A00 <1> mov dword [esi], '\*.\*'

1991 <1> loc\_print\_dir\_call:

1992 00007EE1 E87E000000 <1> call print\_directory

1993 <1>

1994 00007EE6 8A15[5E610100] <1> mov dl, [RUN\_CDRV] ; it is set at the beginning

1995 00007EEC 3A15[FE580100] <1> cmp dl, [Current\_Drv]

1996 00007EF2 7406 <1> je short loc\_print\_dir\_call\_restore\_cdir\_retn

1997 00007EF4 E8C7EFFFFF <1> call change\_current\_drive

1998 00007EF9 C3 <1> retn

1999 <1>

2000 <1> loc\_print\_dir\_call\_restore\_cdir\_retn:

2001 00007EFA 803D[D30C0100]00 <1> cmp byte [Restore\_CDIR], 0

2002 00007F01 7610 <1> jna short pass\_print\_dir\_call\_restore\_cdir\_retn

2003 <1>

2004 00007F03 BE00010900 <1> mov esi, Logical\_DOSDisks

2005 00007F08 31C0 <1> xor eax, eax

2006 00007F0A 88D4 <1> mov ah, dl

2007 00007F0C 01C6 <1> add esi, eax

2008 <1>

2009 00007F0E E864F0FFFF <1> call restore\_current\_directory

2010 <1>

2011 <1> pass\_print\_dir\_call\_restore\_cdir\_retn:

2012 00007F13 C3 <1> retn

2013 <1>

2014 <1> check\_attr\_s\_cap:

2015 00007F14 24DF <1> and al, 0DFh

2016 <1> check\_attr\_s:

2017 00007F16 3C53 <1> cmp al, 'S'

2018 00007F18 7514 <1> jne short pass\_attr\_s

2019 00007F1A 800D[A0620100]04 <1> or byte [AttributesMask], 4 ; system

2020 00007F21 AC <1> lodsb

2021 00007F22 3C20 <1> cmp al, 20h

2022 00007F24 0F844CFFFFFF <1> je get\_dfname\_fchar\_attr

2023 00007F2A 72AF <1> jb short loc\_print\_dir\_call\_all

2024 00007F2C 24DF <1> and al, 0DFh

2025 <1> pass\_attr\_s:

2026 00007F2E 3C48 <1> cmp al, 'H'

2027 00007F30 7514 <1> jne short pass\_attr\_h

2028 00007F32 800D[A0620100]02 <1> or byte [AttributesMask], 2 ; hidden

2029 <1> pass\_attr\_shr:

2030 00007F39 AC <1> lodsb

2031 00007F3A 3C20 <1> cmp al, 20h

2032 00007F3C 0F8434FFFFFF <1> je get\_dfname\_fchar\_attr

2033 00007F42 7297 <1> jb short loc\_print\_dir\_call\_all

2034 00007F44 EBCE <1> jmp short check\_attr\_s\_cap

2035 <1>

2036 <1> pass\_attr\_h:

2037 00007F46 3C52 <1> cmp al, 'R'

2038 00007F48 7509 <1> jne short pass\_attr\_r

2039 00007F4A 800D[A0620100]01 <1> or byte [AttributesMask], 1 ; read only

2040 00007F51 EBE6 <1> jmp short pass\_attr\_shr

2041 <1>

2042 <1> pass\_attr\_r:

2043 00007F53 3C41 <1> cmp al, 'A'

2044 00007F55 0F85F8FDFFFF <1> jne loc\_cmd\_failed

2045 00007F5B 800D[A0620100]20 <1> or byte [AttributesMask], 20h ; archive

2046 00007F62 EBD5 <1> jmp short pass\_attr\_shr

2047 <1>

2048 <1> print\_directory:

2049 <1> ; 13/05/2016

2050 <1> ; 11/02/2016

2051 <1> ; 10/02/2016

2052 <1> ; 08/02/2016 (TRDOS 386 = TRDOS v2.0)

2053 <1> ; 30/10/2010 ('proc\_print\_directory')

2054 <1> ; 19/09/2009

2055 <1> ; 2005

2056 <1> ; INPUT ->

2057 <1> ; ESI = Asciiz File/Dir Name Address

2058 <1>

2059 00007F64 56 <1> push esi

2060 <1>

2061 00007F65 29C0 <1> sub eax, eax

2062 <1>

2063 00007F67 66A3[2C630100] <1> mov word [Dir\_Count], ax ; 0

2064 00007F6D 66A3[2A630100] <1> mov word [File\_Count], ax ; 0

2065 00007F73 A3[2E630100] <1> mov dword [Total\_FSize], eax ; 0

2066 <1>

2067 00007F78 E8F6E3FFFF <1> call clear\_screen

2068 <1>

2069 00007F7D 31C9 <1> xor ecx, ecx

2070 00007F7F 8A2D[FE580100] <1> mov ch, [Current\_Drv] ; DirBuff\_Drv - 'A'

2071 00007F85 A0[FF580100] <1> mov al, [Current\_Dir\_Drv]

2072 00007F8A A2[900E0100] <1> mov [Dir\_Drive\_Name], al

2073 00007F8F BE00010900 <1> mov esi, Logical\_DOSDisks

2074 00007F94 01CE <1> add esi, ecx

2075 <1>

2076 00007F96 E858F9FFFF <1> call move\_volume\_name\_and\_serial\_no

2077 00007F9B 730C <1> jnc short print\_dir\_strlen\_check

2078 <1>

2079 00007F9D 5E <1> pop esi

2080 00007F9E 8A3D[66580100] <1> mov bh, [ptty] ; [ACTIVE\_PAGE]

2081 <1> ;call beeper

2082 <1> ;retn

2083 00007FA4 E9E99DFFFF <1> jmp beeper ; beep ! and return

2084 <1>

2085 <1> print\_dir\_strlen\_check:

2086 00007FA9 BE[01590100] <1> mov esi, Current\_Dir\_Root

2087 00007FAE BF[2D0F0100] <1> mov edi, Dir\_Str\_Root

2088 <1>

2089 <1> ;xor ecx, ecx

2090 00007FB3 8A0D[5D590100] <1> mov cl, [Current\_Dir\_StrLen]

2091 00007FB9 FEC1 <1> inc cl

2092 00007FBB 80F940 <1> cmp cl, 64

2093 00007FBE 760D <1> jna short pass\_print\_dir\_strlen\_shorting

2094 00007FC0 46 <1> inc esi

2095 00007FC1 01CE <1> add esi, ecx

2096 00007FC3 83EE40 <1> sub esi, 64

2097 00007FC6 47 <1> inc edi

2098 00007FC7 B82E2E2E20 <1> mov eax, '... '

2099 00007FCC AB <1> stosd

2100 <1>

2101 <1> pass\_print\_dir\_strlen\_shorting:

2102 00007FCD F3A4 <1> rep movsb

2103 <1>

2104 00007FCF BE[830E0100] <1> mov esi, Dir\_Drive\_Str

2105 00007FD4 E884E3FFFF <1> call print\_msg

2106 <1>

2107 00007FD9 BE[E20E0100] <1> mov esi, Vol\_Serial\_Header

2108 00007FDE E87AE3FFFF <1> call print\_msg

2109 <1>

2110 00007FE3 BE[220F0100] <1> mov esi, Dir\_Str\_Header

2111 00007FE8 E870E3FFFF <1> call print\_msg

2112 <1>

2113 00007FED BE[6D190100] <1> mov esi, next2line

2114 00007FF2 E866E3FFFF <1> call print\_msg

2115 <1>

2116 <1> loc\_print\_dir\_first\_file:

2117 00007FF7 C605[41630100]10 <1> mov byte [PrintDir\_RowCounter], 16

2118 00007FFE 66A1[A0620100] <1> mov ax, [AttributesMask]

2119 00008004 5E <1> pop esi

2120 <1>

2121 00008005 E859020000 <1> call find\_first\_file

2122 0000800A 0F826F010000 <1> jc loc\_dir\_ok

2123 <1>

2124 <1> loc\_dfname\_use\_this:

2125 <1> ; bl = File Attributes (bh = Long Name Entry Length)

2126 00008010 F6C310 <1> test bl, 10h ; Is it a directory?

2127 00008013 741B <1> jz short loc\_not\_dir

2128 <1>

2129 00008015 66FF05[2C630100] <1> inc word [Dir\_Count]

2130 0000801C 89F2 <1> mov edx, esi ; FindFile\_DirEntry address

2131 0000801E BE[72100100] <1> mov esi, Type\_Dir ; '<DIR> '

2132 00008023 BF[89100100] <1> mov edi, Dir\_Or\_FileSize

2133 <1> ; move 10 bytes

2134 00008028 A5 <1> movsd

2135 00008029 A5 <1> movsd

2136 0000802A 66A5 <1> movsw

2137 0000802C 89D6 <1> mov esi, edx

2138 0000802E EB36 <1> jmp short loc\_dir\_attribute

2139 <1>

2140 <1> loc\_not\_dir:

2141 00008030 66FF05[2A630100] <1> inc word [File\_Count]

2142 00008037 0105[2E630100] <1> add [Total\_FSize], eax

2143 <1>

2144 0000803D B90A000000 <1> mov ecx, 10 ; 32 bit divisor

2145 00008042 89CF <1> mov edi, ecx

2146 00008044 81C7[89100100] <1> add edi, Dir\_Or\_FileSize

2147 <1> loc\_dir\_rdivide:

2148 0000804A 29D2 <1> sub edx, edx

2149 0000804C F7F1 <1> div ecx ; remainder in dl (< 10)

2150 0000804E 80C230 <1> add dl, '0' ; to make visible (ascii)

2151 00008051 4F <1> dec edi

2152 00008052 8817 <1> mov [edi], dl

2153 00008054 21C0 <1> and eax, eax

2154 00008056 75F2 <1> jnz short loc\_dir\_rdivide

2155 <1>

2156 <1> loc\_dir\_fill\_space:

2157 00008058 81FF[89100100] <1> cmp edi, Dir\_Or\_FileSize

2158 0000805E 7606 <1> jna short loc\_dir\_attribute

2159 00008060 4F <1> dec edi

2160 00008061 C60720 <1> mov byte [edi], 20h

2161 00008064 EBF2 <1> jmp short loc\_dir\_fill\_space

2162 <1>

2163 <1> loc\_dir\_attribute:

2164 00008066 C705[94100100]2020- <1> mov dword [File\_Attribute], 20202020h

2164 0000806E 2020 <1>

2165 <1>

2166 00008070 80FB20 <1> cmp bl, 20h ; Is it an archive file?

2167 00008073 7207 <1> jb short loc\_dir\_pass\_arch

2168 00008075 C605[97100100]41 <1> mov byte [File\_Attribute+3], 'A'

2169 <1>

2170 <1> loc\_dir\_pass\_arch:

2171 0000807C 80E307 <1> and bl, 7

2172 0000807F 7428 <1> jz short loc\_dir\_file\_name

2173 00008081 88DF <1> mov bh, bl

2174 00008083 80E303 <1> and bl, 3

2175 00008086 38DF <1> cmp bh, bl

2176 00008088 7607 <1> jna short loc\_dir\_pass\_s

2177 0000808A C605[94100100]53 <1> mov byte [File\_Attribute], 'S'

2178 <1>

2179 <1> loc\_dir\_pass\_s:

2180 00008091 80E302 <1> and bl,2

2181 00008094 7407 <1> jz short loc\_dir\_pass\_h

2182 00008096 C605[95100100]48 <1> mov byte [File\_Attribute+1], 'H'

2183 <1> loc\_dir\_pass\_h:

2184 0000809D 80E701 <1> and bh,1

2185 000080A0 7407 <1> jz short loc\_dir\_file\_name

2186 000080A2 C605[96100100]52 <1> mov byte [File\_Attribute+2], 'R'

2187 <1> loc\_dir\_file\_name:

2188 <1> ;mov bx, [esi+18h] ; Date

2189 <1> ;mov dx, [esi+16h] ; Time

2190 000080A9 8B5E16 <1> mov ebx, [esi+16h]

2191 000080AC 89F1 <1> mov ecx, esi ; FindFile\_DirEntry address

2192 000080AE BF[7C100100] <1> mov edi, File\_Name

2193 <1> ; move 8 bytes

2194 000080B3 A5 <1> movsd

2195 000080B4 A5 <1> movsd

2196 000080B5 C60720 <1> mov byte [edi], 20h

2197 000080B8 47 <1> inc edi

2198 <1> ; move 3 bytes

2199 000080B9 66A5 <1> movsw

2200 000080BB A4 <1> movsb

2201 000080BC 89CE <1> mov esi, ecx

2202 <1>

2203 <1> Dir\_Time\_start:

2204 <1> ;mov ax, dx ; Time

2205 000080BE 6689D8 <1> mov ax, bx

2206 000080C1 66C1E805 <1> shr ax, 5 ; shift right 5 times

2207 000080C5 6683E03F <1> and ax, 0000111111b ; Minute Mask

2208 000080C9 D40A <1> aam ; Q([AL]/10)->AH

2209 <1> ; R([AL]/10)->AL

2210 <1> ; [AL]+[AH]= Minute as BCD

2211 000080CB 660D3030 <1> or ax, '00' ; Convert to ASCII

2212 000080CF 86E0 <1> xchg ah, al

2213 000080D1 66A3[A7100100] <1> mov [File\_Minute], ax

2214 <1>

2215 <1> ;mov al, dh

2216 000080D7 88F8 <1> mov al, bh

2217 000080D9 C0E803 <1> shr al, 3 ; shift right 3 times

2218 000080DC D40A <1> aam ; [AL]+[AH]= Hours as BCD

2219 000080DE 660D3030 <1> or ax, '00'

2220 000080E2 86E0 <1> xchg ah, al

2221 000080E4 66A3[A4100100] <1> mov [File\_Hour], ax

2222 <1>

2223 000080EA C1EB10 <1> shr ebx, 16 ; BX = Date

2224 <1>

2225 <1> Dir\_Date\_start:

2226 000080ED 6689D8 <1> mov ax, bx ; Date

2227 000080F0 6683E01F <1> and ax, 00011111b ; Day Mask

2228 000080F4 D40A <1> aam ; Q([AL]/10)->AH

2229 <1> ; R([AL]/10)->AL

2230 <1> ; [AL]+[AH]= Day as BCD

2231 000080F6 660D3030 <1> or ax, '00' ; Convert to ASCII

2232 000080FA 86C4 <1> xchg al, ah

2233 <1>

2234 000080FC 66A3[99100100] <1> mov [File\_Day], ax

2235 <1>

2236 00008102 6689D8 <1> mov ax, bx

2237 00008105 66C1E805 <1> shr ax, 5 ; shift right 5 times

2238 00008109 6683E00F <1> and ax, 00001111b ; Month Mask

2239 0000810D D40A <1> aam

2240 0000810F 660D3030 <1> or ax, '00'

2241 00008113 86E0 <1> xchg ah, al

2242 00008115 66A3[9C100100] <1> mov [File\_Month], ax

2243 <1>

2244 0000811B 6689D8 <1> mov ax, bx

2245 0000811E 66C1E809 <1> shr ax, 9

2246 00008122 6683E07F <1> and ax, 01111111b ; Result = Year - 1980

2247 00008126 6605BC07 <1> add ax, 1980

2248 <1>

2249 0000812A B10A <1> mov cl, 10

2250 0000812C F6F1 <1> div cl ; Q -> AL, R -> AH

2251 0000812E 80CC30 <1> or ah, '0'

2252 00008131 8825[A2100100] <1> mov [File\_Year+3], ah

2253 00008137 D40A <1> aam

2254 00008139 86E0 <1> xchg ah, al

2255 0000813B 80CC30 <1> or ah, '0' ; Convert to ASCII

2256 0000813E 8825[A1100100] <1> mov [File\_Year+2], ah

2257 00008144 D40A <1> aam

2258 00008146 86C4 <1> xchg al, ah

2259 00008148 660D3030 <1> or ax, '00'

2260 0000814C 66A3[9F100100] <1> mov [File\_Year], ax

2261 <1>

2262 <1> loc\_show\_line:

2263 00008152 56 <1> push esi

2264 00008153 BE[7C100100] <1> mov esi, File\_Name

2265 00008158 E800E2FFFF <1> call print\_msg

2266 0000815D BE[6F190100] <1> mov esi, nextline

2267 00008162 E8F6E1FFFF <1> call print\_msg

2268 00008167 5E <1> pop esi

2269 <1>

2270 00008168 FE0D[41630100] <1> dec byte [PrintDir\_RowCounter]

2271 0000816E 0F84D4000000 <1> jz pause\_dir\_scroll

2272 <1>

2273 <1> loc\_next\_entry:

2274 00008174 E899010000 <1> call find\_next\_file

2275 00008179 0F8391FEFFFF <1> jnc loc\_dfname\_use\_this

2276 <1>

2277 <1> loc\_dir\_ok:

2278 0000817F B90A000000 <1> mov ecx, 10

2279 00008184 66A1[2C630100] <1> mov ax, [Dir\_Count]

2280 0000818A BF[BD100100] <1> mov edi, Decimal\_Dir\_Count

2281 0000818F 6639C8 <1> cmp ax, cx ; 10

2282 00008192 7216 <1> jb short pass\_ddc

2283 00008194 47 <1> inc edi

2284 00008195 6683F864 <1> cmp ax, 100

2285 00008199 720F <1> jb short pass\_ddc

2286 0000819B 47 <1> inc edi

2287 0000819C 663DE803 <1> cmp ax, 1000

2288 000081A0 7208 <1> jb short pass\_ddc

2289 000081A2 47 <1> inc edi

2290 000081A3 663D1027 <1> cmp ax, 10000

2291 000081A7 7201 <1> jb short pass\_ddc

2292 000081A9 47 <1> inc edi

2293 <1> pass\_ddc:

2294 000081AA 886F01 <1> mov [edi+1], ch ; 0

2295 <1> loc\_ddc\_rediv:

2296 000081AD 31D2 <1> xor edx, edx

2297 000081AF 66F7F1 <1> div cx ; 10

2298 000081B2 80C230 <1> add dl, '0'

2299 000081B5 8817 <1> mov [edi], dl

2300 000081B7 4F <1> dec edi

2301 000081B8 6609C0 <1> or ax, ax

2302 000081BB 75F0 <1> jnz short loc\_ddc\_rediv

2303 <1>

2304 000081BD 66A1[2A630100] <1> mov ax, [File\_Count]

2305 000081C3 BF[AC100100] <1> mov edi, Decimal\_File\_Count

2306 000081C8 6639C8 <1> cmp ax, cx ; 10

2307 000081CB 7216 <1> jb short pass\_dfc

2308 000081CD 47 <1> inc edi

2309 000081CE 6683F864 <1> cmp ax, 100

2310 000081D2 720F <1> jb short pass\_dfc

2311 000081D4 47 <1> inc edi

2312 000081D5 663DE803 <1> cmp ax, 1000

2313 000081D9 7208 <1> jb short pass\_dfc

2314 000081DB 47 <1> inc edi

2315 000081DC 663D1027 <1> cmp ax, 10000

2316 000081E0 7201 <1> jb short pass\_dfc

2317 000081E2 47 <1> inc edi

2318 <1> pass\_dfc:

2319 <1> ;mov cx, 10

2320 000081E3 886F01 <1> mov [edi+1], ch ; 00

2321 <1> loc\_dfc\_rediv:

2322 <1> ;xor dx, dx

2323 000081E6 30D2 <1> xor dl, dl

2324 000081E8 66F7F1 <1> div cx

2325 000081EB 80C230 <1> add dl, '0'

2326 000081EE 8817 <1> mov [edi], dl

2327 000081F0 4F <1> dec edi

2328 000081F1 6609C0 <1> or ax, ax

2329 000081F4 75F0 <1> jnz short loc\_dfc\_rediv

2330 <1>

2331 000081F6 BF[40630100] <1> mov edi, TFS\_Dec\_End

2332 <1> ;mov byte [edi], 0

2333 000081FB A1[2E630100] <1> mov eax, [Total\_FSize]

2334 <1> ;mov ecx, 10

2335 <1> rediv\_tfs\_hex:

2336 <1> ;sub edx, edx

2337 00008200 28D2 <1> sub dl, dl

2338 00008202 F7F1 <1> div ecx

2339 00008204 80C230 <1> add dl, '0'

2340 00008207 4F <1> dec edi

2341 00008208 8817 <1> mov [edi], dl

2342 0000820A 21C0 <1> and eax, eax

2343 0000820C 75F2 <1> jnz short rediv\_tfs\_hex

2344 <1>

2345 0000820E 893D[32630100] <1> mov [TFS\_Dec\_Begin], edi

2346 00008214 BE[AA100100] <1> mov esi, Decimal\_File\_Count\_Header

2347 00008219 E83FE1FFFF <1> call print\_msg

2348 0000821E BE[B2100100] <1> mov esi, str\_files

2349 00008223 E835E1FFFF <1> call print\_msg

2350 00008228 BE[C3100100] <1> mov esi, str\_dirs

2351 0000822D E82BE1FFFF <1> call print\_msg

2352 00008232 8B35[32630100] <1> mov esi, [TFS\_Dec\_Begin]

2353 00008238 E820E1FFFF <1> call print\_msg

2354 0000823D BE[D4100100] <1> mov esi, str\_bytes

2355 00008242 E816E1FFFF <1> call print\_msg

2356 <1>

2357 00008247 C3 <1> retn

2358 <1>

2359 <1> pause\_dir\_scroll:

2360 00008248 28E4 <1> sub ah, ah

2361 0000824A E8C789FFFF <1> call int16h

2362 0000824F 3C1B <1> cmp al, 1Bh

2363 00008251 0F8428FFFFFF <1> je loc\_dir\_ok

2364 00008257 C605[41630100]10 <1> mov byte [PrintDir\_RowCounter], 16 ; Reset counter

2365 0000825E E911FFFFFF <1> jmp loc\_next\_entry

2366 <1>

2367 <1> find\_first\_file:

2368 <1> ; 11/02/2016

2369 <1> ; 10/02/2016

2370 <1> ; 08/02/2016 (TRDOS 386 = TRDOS v2.0)

2371 <1> ; 09/10/2011

2372 <1> ; 17/09/2009

2373 <1> ; 2005

2374 <1> ; INPUT ->

2375 <1> ; ESI = ASCIIZ File/Dir Name Address (in Current Directory)

2376 <1> ; AL = Attributes AND mask (The AND result must be equal to AL)

2377 <1> ; bit 0 = Read Only

2378 <1> ; bir 1 = Hidden

2379 <1> ; bit 2 = System

2380 <1> ; bit 3 = Volume Label

2381 <1> ; bit 4 = Directory

2382 <1> ; bit 5 = Archive

2383 <1> ; bit 6 = Reserved, must be 0

2384 <1> ; bit 7 = Reserved, must be 0

2385 <1> ; AH = Attributes Negative AND mask (The AND result must be ZERO)

2386 <1> ;

2387 <1> ; OUTPUT ->

2388 <1> ; CF = 1 -> Error, Error Code in EAX (AL)

2389 <1> ; CF = 0 ->

2390 <1> ; ESI = Directory Entry (FindFile\_DirEntry) Location

2391 <1> ; EDI = Directory Buffer Directory Entry Location

2392 <1> ; EAX = File Size

2393 <1> ; BL = Attributes of The File/Directory

2394 <1> ; BH = Long Name Yes/No Status (>0 is YES)

2395 <1> ; DX > 0 : Ambiguous filename chars are used

2396 <1> ;

2397 <1> ; (EAX, EBX, ECX, EDX, ESI, EDI will be changed)

2398 <1>

2399 00008263 66A3[F2620100] <1> mov [FindFile\_AttributesMask], ax

2400 00008269 BF[F4620100] <1> mov edi, FindFile\_DirEntry ; TR-DOS Fullfilename formatted buffer

2401 0000826E 31C0 <1> xor eax, eax

2402 00008270 B90B000000 <1> mov ecx, 11

2403 00008275 F3AB <1> rep stosd ; 44 bytes

2404 <1> ;stosw ; +2 bytes

2405 <1>

2406 00008277 BF[E4620100] <1> mov edi, FindFile\_Name ; FFF structure, offset 66

2407 0000827C 39FE <1> cmp esi, edi

2408 0000827E 7408 <1> je short loc\_fff\_mfn\_ok

2409 00008280 89FA <1> mov edx, edi

2410 <1> ; move 13 bytes

2411 00008282 A5 <1> movsd

2412 00008283 A5 <1> movsd

2413 00008284 A5 <1> movsd

2414 00008285 AA <1> stosb

2415 00008286 89D6 <1> mov esi, edx

2416 <1> loc\_fff\_mfn\_ok:

2417 00008288 BF[93620100] <1> mov edi, Dir\_Entry\_Name ; Dir Entry Format File Name

2418 0000828D E8D7200000 <1> call convert\_file\_name

2419 00008292 89FE <1> mov esi, edi ; offset Dir\_Entry\_Name

2420 <1>

2421 00008294 66A1[F2620100] <1> mov ax, [FindFile\_AttributesMask]

2422 <1> ;xor ecx, ecx

2423 0000829A 30C9 <1> xor cl, cl

2424 0000829C E8D11D0000 <1> call locate\_current\_dir\_file

2425 000082A1 726E <1> jc short loc\_fff\_retn

2426 <1> ; EDI = Directory Entry

2427 <1> ; EBX = Directory Buffer Entry Index/Number

2428 <1>

2429 <1> loc\_fff\_fnf\_ln\_check:

2430 000082A3 30ED <1> xor ch, ch

2431 000082A5 80F60F <1> xor dh, 0Fh

2432 000082A8 7408 <1> jz short loc\_fff\_longname\_yes

2433 000082AA 882D[F1620100] <1> mov [FindFile\_LongNameYes], ch ; 0

2434 000082B0 EB0C <1> jmp short loc\_fff\_longname\_no

2435 <1>

2436 <1> loc\_fff\_longname\_yes:

2437 <1> ;inc byte [FindFile\_LongNameYes]

2438 000082B2 8A0D[FE610100] <1> mov cl, [LFN\_EntryLength]

2439 000082B8 880D[F1620100] <1> mov [FindFile\_LongNameEntryLength], cl ; FindFile\_LongNameYes

2440 <1>

2441 <1> loc\_fff\_longname\_no:

2442 <1> ;mov bx, [DirBuff\_CurrentEntry]

2443 000082BE 66891D[1C630100] <1> mov [FindFile\_DirEntryNumber], bx

2444 000082C5 6689C2 <1> mov dx, ax ; Ambiguous Filename chars used sign > 0

2445 <1>

2446 000082C8 A0[FE580100] <1> mov al, [Current\_Drv]

2447 000082CD A2[A2620100] <1> mov [FindFile\_Drv], al

2448 <1>

2449 000082D2 A1[F8580100] <1> mov eax, [Current\_Dir\_FCluster]

2450 000082D7 A3[14630100] <1> mov [FindFile\_DirFirstCluster], eax

2451 <1>

2452 000082DC A1[2D610100] <1> mov eax, [DirBuff\_Cluster]

2453 000082E1 A3[18630100] <1> mov [FindFile\_DirCluster], eax

2454 <1>

2455 000082E6 66FF05[1E630100] <1> inc word [FindFile\_MatchCounter]

2456 <1>

2457 000082ED 89FB <1> mov ebx, edi

2458 000082EF 89FE <1> mov esi, edi

2459 000082F1 BF[F4620100] <1> mov edi, FindFile\_DirEntry

2460 000082F6 89F8 <1> mov eax, edi

2461 000082F8 B108 <1> mov cl, 8

2462 000082FA F3A5 <1> rep movsd

2463 000082FC 89C6 <1> mov esi, eax

2464 000082FE 89DF <1> mov edi, ebx

2465 <1>

2466 00008300 A1[10630100] <1> mov eax, [FindFile\_DirEntry+28] ; File Size

2467 <1>

2468 00008305 8A1D[FF620100] <1> mov bl, [FindFile\_DirEntry+11] ; File Attributes

2469 0000830B 8A3D[F1620100] <1> mov bh, [FindFile\_LongNameYes]

2470 <1>

2471 <1> ;mov cx, [DirBuff\_EntryCounter]

2472 <1> ;mov [FindFile\_DirEntryNumber], cx

2473 <1> ;mov cx, [FindFile\_DirEntryNumber]

2474 <1> ; ecx = 0

2475 <1>

2476 <1> loc\_fff\_retn:

2477 00008311 C3 <1> retn

2478 <1>

2479 <1> find\_next\_file:

2480 <1> ; 15/10/2016

2481 <1> ; 10/02/2016

2482 <1> ; 08/02/2016 (TRDOS 386 = TRDOS v2.0)

2483 <1> ; 06/02/2011

2484 <1> ; 17/09/2009

2485 <1> ; 2005

2486 <1> ; INPUT ->

2487 <1> ; NONE, Find First File Parameters

2488 <1> ; OUTPUT ->

2489 <1> ; CF = 1 -> Error, Error Code in EAX (AL)

2490 <1> ; CF = 0 ->

2491 <1> ; ESI = Directory Entry (FindFile\_DirEntry) Location

2492 <1> ; EDI = Directory Buffer Directory Entry Location

2493 <1> ; EAX = File Size

2494 <1> ; BL = Attributes of The File/Directory

2495 <1> ; BH = Long Name Yes/No Status (>0 is YES)

2496 <1> ; DX > 0 : Ambiguous filename chars are used

2497 <1> ;

2498 <1> ; (EAX, EBX, ECX, EDX, ESI, EDI will be changed)

2499 <1>

2500 00008312 66833D[1E630100]00 <1> cmp word [FindFile\_MatchCounter], 0

2501 0000831A 7707 <1> ja short loc\_start\_search\_next\_file

2502 <1>

2503 <1> loc\_fnf\_stc\_retn:

2504 0000831C F9 <1> stc

2505 <1> loc\_fnf\_ax12h\_retn:

2506 0000831D B80C000000 <1> mov eax, 12 ; No More files

2507 <1> ;loc\_fnf\_retn:

2508 00008322 C3 <1> retn

2509 <1>

2510 <1> loc\_start\_search\_next\_file:

2511 00008323 668B1D[1C630100] <1> mov bx, [FindFile\_DirEntryNumber]

2512 0000832A 6643 <1> inc bx

2513 0000832C 663B1D[2B610100] <1> cmp bx, [DirBuff\_LastEntry]

2514 00008333 7719 <1> ja short loc\_cont\_search\_next\_file

2515 <1>

2516 <1> loc\_fnf\_search:

2517 00008335 BE[93620100] <1> mov esi, Dir\_Entry\_Name

2518 0000833A 66A1[F2620100] <1> mov ax, [FindFile\_AttributesMask]

2519 00008340 6631C9 <1> xor cx, cx

2520 00008343 E82E1E0000 <1> call find\_directory\_entry

2521 00008348 0F8355FFFFFF <1> jnc loc\_fff\_fnf\_ln\_check

2522 <1>

2523 <1> loc\_cont\_search\_next\_file:

2524 0000834E 31DB <1> xor ebx, ebx

2525 00008350 8A3D[FE580100] <1> mov bh, [Current\_Drv]

2526 00008356 BE00010900 <1> mov esi, Logical\_DOSDisks

2527 0000835B 01DE <1> add esi, ebx

2528 <1>

2529 0000835D 803D[FC580100]00 <1> cmp byte [Current\_Dir\_Level], 0

2530 00008364 7608 <1> jna short loc\_fnf\_check\_FAT\_type

2531 00008366 807E0301 <1> cmp byte [esi+LD\_FATType], 1

2532 0000836A 72B1 <1> jb short loc\_fnf\_ax12h\_retn

2533 0000836C EB06 <1> jmp short loc\_fnf\_check\_next\_cluster

2534 <1>

2535 <1> loc\_fnf\_check\_FAT\_type:

2536 0000836E 807E0303 <1> cmp byte [esi+LD\_FATType], 3

2537 00008372 72A9 <1> jb short loc\_fnf\_ax12h\_retn

2538 <1>

2539 <1> loc\_fnf\_check\_next\_cluster:

2540 00008374 A1[2D610100] <1> mov eax, [DirBuff\_Cluster]

2541 00008379 E8CA370000 <1> call get\_next\_cluster

2542 0000837E 7306 <1> jnc short loc\_fnf\_load\_next\_dir\_cluster

2543 00008380 09C0 <1> or eax, eax

2544 00008382 7498 <1> jz short loc\_fnf\_stc\_retn

2545 <1> ;mov eax, 17 ;Drive not ready or read error

2546 00008384 F5 <1> cmc ;stc

2547 <1> loc\_fnf\_retn:

2548 00008385 C3 <1> retn

2549 <1>

2550 <1> loc\_fnf\_load\_next\_dir\_cluster:

2551 00008386 E8A3390000 <1> call load\_FAT\_sub\_directory

2552 0000838B 72F8 <1> jc short loc\_fnf\_retn

2553 0000838D 6631DB <1> xor bx, bx

2554 00008390 66891D[1C630100] <1> mov [FindFile\_DirEntryNumber], bx

2555 00008397 EB9C <1> jmp short loc\_fnf\_search

2556 <1>

2557 <1> get\_and\_print\_longname:

2558 <1> ; 16/10/2016

2559 <1> ; 13/02/2016 (TRDOS 386 = TRDOS v2.0)

2560 <1> ; 24/01/2010

2561 <1> ; 17/10/2009 (CMD\_INTR.ASM, 'cmp\_cmd\_longname')

2562 <1> get\_longname\_fchar:

2563 00008399 803E20 <1> cmp byte [esi], 20h

2564 0000839C 7701 <1> ja short loc\_find\_longname

2565 <1> ;jb short loc\_longname\_retn

2566 <1> ;inc esi

2567 <1> ;je short get\_longname\_fchar

2568 <1> ;loc\_longname\_retn:

2569 0000839E C3 <1> retn

2570 <1> loc\_find\_longname:

2571 0000839F E839210000 <1> call find\_longname

2572 000083A4 7328 <1> jnc short loc\_print\_longname

2573 <1>

2574 000083A6 08C0 <1> or al, al

2575 000083A8 741A <1> jz short loc\_longname\_not\_found

2576 <1>

2577 <1> ; 16/10/2016 (15h -> 15, 17)

2578 000083AA 3C0F <1> cmp al, 15

2579 000083AC 0F84B6F7FFFF <1> je cd\_drive\_not\_ready ; drive not ready

2580 <1> ; or

2581 000083B2 3C11 <1> cmp al, 17 ; read error

2582 000083B4 0F84AEF7FFFF <1> je cd\_drive\_not\_ready

2583 <1>

2584 <1> loc\_ln\_file\_dir\_not\_found:

2585 000083BA BE[FE0F0100] <1> mov esi, Msg\_File\_Directory\_Not\_Found

2586 <1> ;call print\_msg

2587 <1> ;retn

2588 000083BF E999DFFFFF <1> jmp print\_msg

2589 <1>

2590 <1> loc\_longname\_not\_found:

2591 000083C4 BE[1D100100] <1> mov esi, Msg\_LongName\_Not\_Found

2592 <1> ;call print\_msg

2593 <1> ;retn

2594 000083C9 E98FDFFFFF <1> jmp print\_msg

2595 <1>

2596 <1> loc\_print\_longname:

2597 <1> ;mov esi, LongFileName

2598 000083CE BF[FE590100] <1> mov edi, TextBuffer

2599 000083D3 57 <1> push edi

2600 000083D4 3C00 <1> cmp al, 0

2601 000083D6 7708 <1> ja short loc\_print\_longname\_1

2602 <1> loc\_print\_FS\_longname: ; Singlix FS (64 byte ASCIIZ file name)

2603 000083D8 AC <1> lodsb

2604 000083D9 AA <1> stosb

2605 000083DA 08C0 <1> or al, al

2606 000083DC 75FA <1> jnz short loc\_print\_FS\_longname

2607 000083DE EB07 <1> jmp short loc\_print\_longname\_2

2608 <1> ;

2609 <1> loc\_print\_longname\_1: ; MS Windows long name (UNICODE chars)

2610 000083E0 66AD <1> lodsw

2611 000083E2 AA <1> stosb

2612 000083E3 08C0 <1> or al, al

2613 000083E5 75F9 <1> jnz short loc\_print\_longname\_1

2614 <1> ;

2615 <1> loc\_print\_longname\_2:

2616 000083E7 5E <1> pop esi

2617 000083E8 E870DFFFFF <1> call print\_msg

2618 000083ED BE[6F190100] <1> mov esi, nextline

2619 <1> ;call print\_msg

2620 <1> ;retn

2621 000083F2 E966DFFFFF <1> jmp print\_msg

2622 <1>

2623 <1> show\_file:

2624 <1> ; 18/02/2016

2625 <1> ; 17/02/2016

2626 <1> ; 15/02/2016 (TRDOS 386 = TRDOS v2.0)

2627 <1> ; 13/09/2011 (CMD\_INTR.ASM, 'cmp\_cmd\_show')

2628 <1> ; 08/11/2009

2629 <1>

2630 <1> loc\_show\_parse\_path\_name:

2631 000083F7 BF[A2620100] <1> mov edi, FindFile\_Drv

2632 000083FC E833200000 <1> call parse\_path\_name

2633 00008401 0F824CF9FFFF <1> jc loc\_cmd\_failed

2634 <1>

2635 <1> loc\_show\_check\_filename\_exists:

2636 00008407 BE[E4620100] <1> mov esi, FindFile\_Name

2637 0000840C 803E20 <1> cmp byte [esi], 20h

2638 0000840F 0F863EF9FFFF <1> jna loc\_cmd\_failed

2639 <1>

2640 <1> ; 15/02/2016 (invalid file name check)

2641 00008415 E807020000 <1> call check\_filename

2642 0000841A 730A <1> jnc short loc\_show\_change\_drv

2643 <1>

2644 0000841C BE[EA100100] <1> mov esi, Msg\_invalid\_name\_chars

2645 00008421 E937DFFFFF <1> jmp print\_msg

2646 <1>

2647 <1> loc\_show\_change\_drv:

2648 00008426 8A35[FE580100] <1> mov dh, [Current\_Drv]

2649 0000842C 8835[5E610100] <1> mov [RUN\_CDRV], dh

2650 00008432 8A15[A2620100] <1> mov dl, [FindFile\_Drv]

2651 00008438 38F2 <1> cmp dl, dh

2652 0000843A 740B <1> je short loc\_show\_change\_directory

2653 0000843C E87FEAFFFF <1> call change\_current\_drive

2654 <1> ;jc loc\_file\_rw\_cmd\_failed

2655 00008441 0F8237F9FFFF <1> jc loc\_run\_cmd\_failed

2656 <1>

2657 <1> loc\_show\_change\_directory:

2658 00008447 803D[A3620100]20 <1> cmp byte [FindFile\_Directory], 20h

2659 0000844E 7618 <1> jna short loc\_findload\_showfile

2660 <1>

2661 00008450 FE05[D30C0100] <1> inc byte [Restore\_CDIR]

2662 00008456 BE[A3620100] <1> mov esi, FindFile\_Directory

2663 0000845B 30E4 <1> xor ah, ah ; CD\_COMMAND sign -> 0

2664 0000845D E8BC190000 <1> call change\_current\_directory

2665 <1> ;jc loc\_file\_rw\_cmd\_failed

2666 00008462 0F8216F9FFFF <1> jc loc\_run\_cmd\_failed

2667 <1>

2668 <1> ;loc\_show\_change\_prompt\_dir\_string:

2669 <1> ;call change\_prompt\_dir\_string

2670 <1>

2671 <1> loc\_findload\_showfile:

2672 <1> ; 15/02/2016

2673 00008468 BE[E4620100] <1> mov esi, FindFile\_Name

2674 0000846D BF[93620100] <1> mov edi, Dir\_Entry\_Name ; Dir Entry Format File Name

2675 00008472 E8F21E0000 <1> call convert\_file\_name

2676 00008477 89FE <1> mov esi, edi ; offset Dir\_Entry\_Name

2677 <1>

2678 00008479 28C0 <1> sub al, al ; Attrib AND mask = 0

2679 <1> ; Directory attribute : 10h

2680 <1> ; Volume name attribute: 8h

2681 0000847B B418 <1> mov ah, 00011000b ; 18h (Attrib NAND, AND --> zero mask)

2682 <1> ;

2683 0000847D 6631C9 <1> xor cx, cx

2684 00008480 E8ED1B0000 <1> call locate\_current\_dir\_file

2685 <1> ;jc loc\_file\_rw\_cmd\_failed

2686 00008485 0F82F3F8FFFF <1> jc loc\_run\_cmd\_failed

2687 <1>

2688 <1> loc\_show\_load\_file:

2689 <1> ; EDI = Directory Entry

2690 0000848B 668B4714 <1> mov ax, [edi+DirEntry\_FstClusHI] ; First Cluster High Word

2691 0000848F C1E010 <1> shl eax, 16

2692 00008492 668B471A <1> mov ax, [edi+DirEntry\_FstClusLO] ; First Cluster Low Word

2693 00008496 A3[4C630100] <1> mov [Show\_Cluster], eax

2694 0000849B 8B471C <1> mov eax, [edi+DirEntry\_FileSize] ; File Size

2695 0000849E 21C0 <1> and eax, eax ; Empty file !

2696 000084A0 0F8491000000 <1> jz end\_of\_show\_file

2697 000084A6 A3[50630100] <1> mov [Show\_FileSize], eax

2698 000084AB 31C0 <1> xor eax, eax

2699 000084AD A3[54630100] <1> mov [Show\_FilePointer], eax ; 0

2700 000084B2 66A3[58630100] <1> mov [Show\_ClusterPointer], ax ; 0

2701 000084B8 29DB <1> sub ebx, ebx

2702 000084BA 8A3D[FE580100] <1> mov bh, [Current\_Drv]

2703 000084C0 BE00010900 <1> mov esi, Logical\_DOSDisks

2704 000084C5 01DE <1> add esi, ebx

2705 000084C7 8935[48630100] <1> mov [Show\_LDDDT], esi ; Logical DOS Drv Description Table addr

2706 <1>

2707 000084CD 807E0300 <1> cmp byte [esi+LD\_FATType], 0

2708 000084D1 7713 <1> ja short loc\_show\_calculate\_cluster\_size

2709 <1> ; Singlix FS

2710 <1> ; First Cluster Number is FDT number (in compatibility buffer)

2711 000084D3 8B15[4C630100] <1> mov edx, [Show\_Cluster] ; Compatibility dir. buffer value (FDT)

2712 000084D9 8915[44630100] <1> mov [Show\_FDT], edx

2713 000084DF 31C0 <1> xor eax, eax

2714 000084E1 A3[4C630100] <1> mov [Show\_Cluster], eax ; Sector index = 0

2715 <1> ; (next time it will be 1)

2716 <1> loc\_show\_calculate\_cluster\_size:

2717 000084E6 668B5E11 <1> mov bx, [esi+LD\_BPB+BPB\_BytsPerSec] ; FAT 12-16-32 (512)

2718 <1> ; BX = 512 = [esi+LD\_FS\_BytesPerSec] ; Singlix FS

2719 000084EA 8A4613 <1> mov al, [esi+LD\_BPB+BPB\_SecPerClust] ; FAT 12-16-32 (<= 128)

2720 <1> ; AL = 1 = [esi+LD\_FS\_Reserved2] ; SectPerClust for Singlix FS

2721 000084ED F7E3 <1> mul ebx

2722 <1>

2723 <1> ;cmp eax, 65536 ; non-compatible (very big) cluster size

2724 <1> ;ja short end\_of\_show\_file

2725 000084EF 66A3[5A630100] <1> mov [Show\_ClusterSize], ax

2726 <1>

2727 <1> loc\_start\_show\_file:

2728 000084F5 BE[6F190100] <1> mov esi, nextline

2729 000084FA E85EDEFFFF <1> call print\_msg

2730 <1>

2731 000084FF A1[4C630100] <1> mov eax, [Show\_Cluster]

2732 00008504 C605[5C630100]17 <1> mov byte [Show\_RowCount], 23

2733 <1>

2734 <1> ; 17/02/2016

2735 0000850B 8B35[48630100] <1> mov esi, [Show\_LDDDT]

2736 <1>

2737 <1> loc\_show\_next\_cluster:

2738 <1> ; 15/02/2016

2739 00008511 BB00000700 <1> mov ebx, Cluster\_Buffer ; 70000h (for current TRDOS 386 version)

2740 <1> ; ESI = Logical DOS drv description table address

2741 00008516 E851380000 <1> call read\_cluster

2742 <1> ;jc loc\_file\_rw\_cmd\_failed

2743 0000851B 0F825DF8FFFF <1> jc loc\_run\_cmd\_failed

2744 <1>

2745 00008521 31DB <1> xor ebx, ebx

2746 <1> loc\_show\_next\_byte:

2747 00008523 803D[5C630100]00 <1> cmp byte [Show\_RowCount], 0

2748 0000852A 7521 <1> jne short pass\_show\_wait\_for\_key

2749 0000852C 30E4 <1> xor ah, ah

2750 0000852E E8E386FFFF <1> call int16h

2751 00008533 3C1B <1> cmp al, 1Bh

2752 00008535 750F <1> jne short pass\_exit\_show

2753 <1> end\_of\_show\_file:

2754 <1> pass\_show\_file:

2755 00008537 BE[6F190100] <1> mov esi, nextline

2756 0000853C E81CDEFFFF <1> call print\_msg

2757 00008541 E94B010000 <1> jmp loc\_file\_rw\_restore\_retn

2758 <1>

2759 <1> pass\_exit\_show:

2760 00008546 C605[5C630100]14 <1> mov byte [Show\_RowCount], 20

2761 <1> pass\_show\_wait\_for\_key:

2762 0000854D 81C300000700 <1> add ebx, Cluster\_Buffer

2763 00008553 8A03 <1> mov al, [ebx]

2764 00008555 3C0D <1> cmp al, 0Dh

2765 00008557 0F8590000000 <1> jne loc\_show\_check\_tab\_space

2766 0000855D FE0D[5C630100] <1> dec byte [Show\_RowCount]

2767 <1> pass\_show\_dec\_rowcount:

2768 00008563 B307 <1> mov bl, 7 ; (light gray character color, black background)

2769 00008565 8A3D[66580100] <1> mov bh, [ACTIVE\_PAGE] ; [ptty]

2770 0000856B E84297FFFF <1> call \_write\_tty

2771 <1> loc\_show\_check\_eof:

2772 00008570 FF05[54630100] <1> inc dword [Show\_FilePointer]

2773 00008576 A1[54630100] <1> mov eax, [Show\_FilePointer]

2774 0000857B 3B05[50630100] <1> cmp eax, [Show\_FileSize]

2775 00008581 73B4 <1> jnb short end\_of\_show\_file

2776 00008583 66FF05[58630100] <1> inc word [Show\_ClusterPointer]

2777 0000858A 0FB71D[58630100] <1> movzx ebx, word [Show\_ClusterPointer]

2778 <1>

2779 <1> ; 17/02/2016

2780 <1> ; (sector boundary -9 bits- check, 512 = 0)

2781 00008591 66F7C3FF01 <1> test bx, 1FFh ; 1 to 511

2782 00008596 758B <1> jnz short loc\_show\_next\_byte

2783 <1>

2784 <1> ; 16/02/2016

2785 00008598 8B35[48630100] <1> mov esi, [Show\_LDDDT]

2786 <1> ;

2787 0000859E 807E0300 <1> cmp byte [esi+LD\_FATType], 0

2788 000085A2 7719 <1> ja short loc\_show\_check\_fat\_cluster\_size

2789 <1>

2790 <1> ; Singlix FS

2791 <1> ; 1 sector, more... (cluster size = 1 sector)

2792 000085A4 A1[4C630100] <1> mov eax, [Show\_Cluster]

2793 000085A9 40 <1> inc eax

2794 000085AA A3[4C630100] <1> mov [Show\_Cluster], eax

2795 <1>

2796 000085AF 6621DB <1> and bx, bx ; 65536 -> 0

2797 000085B2 0F856BFFFFFF <1> jnz loc\_show\_next\_byte

2798 000085B8 E954FFFFFF <1> jmp loc\_show\_next\_cluster

2799 <1>

2800 <1> loc\_show\_check\_fat\_cluster\_size:

2801 <1> ; 17/02/2016

2802 000085BD 663B1D[5A630100] <1> cmp bx, [Show\_ClusterSize] ; cluster size in bytes

2803 000085C4 0F8259FFFFFF <1> jb loc\_show\_next\_byte

2804 000085CA 66C705[58630100]00- <1> mov word [Show\_ClusterPointer], 0

2804 000085D2 00 <1>

2805 <1>

2806 000085D3 A1[4C630100] <1> mov eax, [Show\_Cluster]

2807 <1> ;mov esi, [Show\_LDDDT]

2808 <1> loc\_show\_get\_next\_cluster:

2809 000085D8 E86B350000 <1> call get\_next\_cluster

2810 <1> ;jc loc\_file\_rw\_cmd\_failed

2811 000085DD 0F829BF7FFFF <1> jc loc\_run\_cmd\_failed

2812 <1> loc\_show\_update\_ccluster:

2813 000085E3 A3[4C630100] <1> mov [Show\_Cluster], eax

2814 000085E8 E924FFFFFF <1> jmp loc\_show\_next\_cluster

2815 <1>

2816 <1> loc\_show\_check\_tab\_space:

2817 000085ED 3C09 <1> cmp al, 09h

2818 000085EF 0F856EFFFFFF <1> jne pass\_show\_dec\_rowcount

2819 <1> loc\_show\_put\_tab\_space:

2820 000085F5 8A3D[66580100] <1> mov bh, [ACTIVE\_PAGE] ; [ptty]

2821 000085FB E84193FFFF <1> call get\_cpos

2822 <1> ; dl = cursor column

2823 00008600 80E207 <1> and dl, 7 ; 18/02/2016

2824 <1> ;shr bh, 1 ; [ACTIVE\_PAGE]

2825 00008603 8A3D[66580100] <1> mov bh, [ACTIVE\_PAGE]

2826 00008609 B307 <1> mov bl, 7 ; color attribute

2827 <1> loc\_show\_put\_space\_chars:

2828 0000860B B020 <1> mov al, 20h ; space

2829 <1> ;mov bh, [ACTIVE\_PAGE] ; [ptty]

2830 <1> ;mov bl, 7 ; color attribute

2831 <1> ;push dx

2832 0000860D 52 <1> push edx ; 29/12/2017

2833 0000860E E89F96FFFF <1> call \_write\_tty

2834 00008613 5A <1> pop edx ; 29/12/2017

2835 <1> ;pop dx

2836 <1> ; 18/02/2016

2837 00008614 80FA07 <1> cmp dl, 7

2838 00008617 0F8353FFFFFF <1> jnb loc\_show\_check\_eof

2839 0000861D FEC2 <1> inc dl

2840 0000861F EBEA <1> jmp short loc\_show\_put\_space\_chars

2841 <1>

2842 <1> check\_filename:

2843 <1> ; 10/10/2016

2844 <1> ; 15/02/2016 (TRDOS 386 = TRDOS v2.0)

2845 <1> ; 07/08/2010 (FILE.ASM, 'proc\_check\_filename')

2846 <1> ; 10/07/2010

2847 <1> ; Derived from 'proc\_check\_filename'

2848 <1> ; in the old TRDOS.ASM (09/02/2005).

2849 <1> ;

2850 <1> ; INPUT ->

2851 <1> ; ESI = Dot File Name Location

2852 <1> ; OUTPUT ->

2853 <1> ; cf = 1 -> error code in AL

2854 <1> ; AL = ERR\_INV\_FILE\_NAME (=26)

2855 <1> ; Invalid file name chars

2856 <1> ; cf = 0 -> valid file name

2857 <1> ;

2858 <1> ;(EAX, ECX, EDI will be changed)

2859 <1>

2860 <1> check\_invalid\_filename\_chars:

2861 <1> ; 15/02/2016 (TRDOS 386 = TRDOS v2.0)

2862 <1> ; 10/07/2010 (FILE.ASM, 'proc\_check\_invalid\_filename\_chars')

2863 <1> ; 10/02/2010

2864 <1> ; Derived from 'proc\_check\_invalid\_filename\_chars'

2865 <1> ; in the old TRDOS.ASM (09/02/2005).

2866 <1> ;

2867 <1> ; INPUT ->

2868 <1> ; ESI = ASCIIZ FileName

2869 <1> ; OUTPUT ->

2870 <1> ; cf = 1 -> invalid

2871 <1> ; cf = 0 -> valid

2872 <1> ;

2873 <1> ;(EAX, ECX, EDI will be changed)

2874 <1>

2875 00008621 56 <1> push esi

2876 <1>

2877 00008622 BF[D20D0100] <1> mov edi, invalid\_fname\_chars

2878 00008627 AC <1> lodsb

2879 <1> check\_filename\_next\_char:

2880 00008628 B914000000 <1> mov ecx, sizeInvFnChars

2881 0000862D BF[D20D0100] <1> mov edi, invalid\_fname\_chars

2882 <1> loc\_scan\_invalid\_filename\_char:

2883 00008632 AE <1> scasb

2884 00008633 741F <1> je short loc\_invalid\_filename\_stc

2885 00008635 E2FB <1> loop loc\_scan\_invalid\_filename\_char

2886 00008637 AC <1> lodsb

2887 00008638 3C1F <1> cmp al, 1Fh ; 20h and above

2888 0000863A 77EC <1> ja short check\_filename\_next\_char

2889 <1>

2890 <1> check\_filename\_dot:

2891 0000863C 8B3424 <1> mov esi, [esp]

2892 <1>

2893 0000863F B421 <1> mov ah, 21h

2894 00008641 B908000000 <1> mov ecx, 8

2895 <1> loc\_check\_filename\_next\_char:

2896 00008646 AC <1> lodsb

2897 00008647 3C2E <1> cmp al, 2Eh

2898 00008649 7511 <1> jne short pass\_check\_fn\_dot\_check

2899 <1> loc\_check\_filename\_ext\_0:

2900 0000864B AC <1> lodsb

2901 0000864C 38E0 <1> cmp al, ah ; 21h

2902 0000864E 7205 <1> jb short loc\_invalid\_filename

2903 00008650 3C2E <1> cmp al, 2Eh

2904 00008652 7519 <1> jne short loc\_check\_filename\_ext\_1

2905 <1>

2906 <1> loc\_invalid\_filename\_stc:

2907 <1> loc\_check\_fn\_stc\_rtn:

2908 00008654 F9 <1> stc

2909 <1> loc\_invalid\_filename:

2910 <1> ; 10/10/2016 (0Bh -> 26)

2911 00008655 B81A000000 <1> mov eax, ERR\_INV\_FILE\_NAME ; (=26)

2912 <1> ; Invalid file name chars

2913 <1> loc\_check\_fn\_rtn:

2914 0000865A 5E <1> pop esi

2915 0000865B C3 <1> retn

2916 <1>

2917 <1> pass\_check\_fn\_dot\_check:

2918 0000865C 38E0 <1> cmp al, ah ; 21h

2919 0000865E 7224 <1> jb short loc\_check\_fn\_clc\_rtn

2920 00008660 E2E4 <1> loop loc\_check\_filename\_next\_char

2921 00008662 AC <1> lodsb

2922 00008663 38E0 <1> cmp al, ah ; 21h

2923 00008665 721D <1> jb short loc\_check\_fn\_clc\_rtn

2924 00008667 3C2E <1> cmp al, 2Eh

2925 00008669 75E9 <1> jne short loc\_check\_fn\_stc\_rtn

2926 0000866B EBDE <1> jmp short loc\_check\_filename\_ext\_0

2927 <1>

2928 <1> loc\_check\_filename\_ext\_1:

2929 0000866D AC <1> lodsb

2930 0000866E 38E0 <1> cmp al, ah ; 21h

2931 00008670 7212 <1> jb short loc\_check\_fn\_clc\_rtn

2932 00008672 3C2E <1> cmp al, 2Eh

2933 00008674 74DE <1> je short loc\_check\_fn\_stc\_rtn

2934 00008676 AC <1> lodsb

2935 00008677 38E0 <1> cmp al, ah ; 21h

2936 00008679 7209 <1> jb short loc\_check\_fn\_clc\_rtn

2937 0000867B 3C2E <1> cmp al, 2Eh

2938 0000867D 74D5 <1> je short loc\_check\_fn\_stc\_rtn

2939 0000867F AC <1> lodsb

2940 00008680 38E0 <1> cmp al, ah ; 21h

2941 00008682 73D0 <1> jnb short loc\_check\_fn\_stc\_rtn

2942 <1>

2943 <1> loc\_check\_fn\_clc\_rtn:

2944 00008684 5E <1> pop esi

2945 00008685 F8 <1> clc

2946 00008686 C3 <1> retn

2947 <1>

2948 <1> loc\_print\_deleted\_message:

2949 00008687 BE[BF110100] <1> mov esi, Msg\_Deleted

2950 0000868C E8CCDCFFFF <1> call print\_msg

2951 <1>

2952 <1> ;clc

2953 <1>

2954 <1> loc\_file\_rw\_restore\_retn:

2955 <1> ; 15/02/2016 (TRDOS 386 = TRDOS v2.0)

2956 <1> ; 28/02/2010 (CMD\_INTR.ASM)

2957 <1> loc\_file\_rw\_cmd\_failed:

2958 00008691 9C <1> pushf

2959 00008692 E84FF7FFFF <1> call restore\_cdir\_after\_cmd\_fail

2960 00008697 9D <1> popf

2961 00008698 720D <1> jc short loc\_file\_rw\_check\_write\_fault

2962 0000869A C3 <1> retn

2963 <1>

2964 <1> loc\_permission\_denied:

2965 <1> ; 27/02/2016

2966 0000869B BE[CC110100] <1> mov esi, Msg\_Permission\_Denied

2967 000086A0 E8B8DCFFFF <1> call print\_msg

2968 000086A5 EBEA <1> jmp short loc\_file\_rw\_restore\_retn

2969 <1>

2970 <1> loc\_file\_rw\_check\_write\_fault:

2971 <1> ;cmp al, 1Dh ; Write Fault

2972 000086A7 3C12 <1> cmp al, 18 ; 05/11/2016

2973 000086A9 0F85D4F6FFFF <1> jne loc\_run\_cmd\_failed\_cmp\_al

2974 000086AF BE[B30F0100] <1> mov esi, Msg\_Not\_Ready\_Write\_Err

2975 <1> ;call print\_msg

2976 <1> ;retn

2977 000086B4 E9A4DCFFFF <1> jmp print\_msg

2978 <1>

2979 <1> make\_directory:

2980 <1> ; 21/02/2016 (TRDOS 386 = TRDOS v2.0)

2981 <1> ; 12/03/2011 (CMD\_INTR.ASM, 'cmp\_cmd\_mkdir')

2982 <1> ; 14/08/2010

2983 <1> ; 10/07/2010

2984 <1> ; 29/11/2009

2985 <1> ;

2986 <1> get\_mkdir\_fchar:

2987 <1> ; esi = directory name

2988 000086B9 803E20 <1> cmp byte [esi], 20h

2989 000086BC 7701 <1> ja short loc\_mkdir\_parse\_path\_name

2990 <1>

2991 <1> loc\_mkdir\_nodirname\_retn:

2992 000086BE C3 <1> retn

2993 <1>

2994 <1> loc\_mkdir\_parse\_path\_name:

2995 000086BF BF[A2620100] <1> mov edi, FindFile\_Drv

2996 000086C4 E86B1D0000 <1> call parse\_path\_name

2997 000086C9 0F8284F6FFFF <1> jc loc\_cmd\_failed

2998 <1>

2999 <1> loc\_mkdir\_check\_dirname\_exists:

3000 000086CF BE[E4620100] <1> mov esi, FindFile\_Name

3001 000086D4 803E20 <1> cmp byte [esi], 20h

3002 000086D7 0F8676F6FFFF <1> jna loc\_cmd\_failed

3003 000086DD 8935[60630100] <1> mov [DelFile\_FNPointer], esi

3004 000086E3 E839FFFFFF <1> call check\_filename

3005 000086E8 7259 <1> jc short loc\_mkdir\_invalid\_dir\_name\_chars

3006 <1>

3007 <1> loc\_mkdir\_drv:

3008 000086EA 8A35[FE580100] <1> mov dh, [Current\_Drv]

3009 000086F0 8835[5E610100] <1> mov [RUN\_CDRV], dh

3010 <1>

3011 000086F6 8A15[A2620100] <1> mov dl, [FindFile\_Drv]

3012 000086FC 38F2 <1> cmp dl, dh

3013 000086FE 7407 <1> je short loc\_mkdir\_change\_directory

3014 <1>

3015 00008700 E8BBE7FFFF <1> call change\_current\_drive

3016 00008705 728A <1> jc loc\_file\_rw\_cmd\_failed

3017 <1>

3018 <1> loc\_mkdir\_change\_directory:

3019 00008707 803D[A3620100]20 <1> cmp byte [FindFile\_Directory], 20h

3020 0000870E 7614 <1> jna short loc\_mkdir\_find\_directory

3021 <1>

3022 00008710 FE05[D30C0100] <1> inc byte [Restore\_CDIR]

3023 00008716 BE[A3620100] <1> mov esi, FindFile\_Directory

3024 0000871B 30E4 <1> xor ah, ah ; CD\_COMMAND sign -> 0

3025 0000871D E8FC160000 <1> call change\_current\_directory

3026 00008722 722E <1> jc short loc\_mkdir\_check\_error\_code

3027 <1>

3028 <1> ;loc\_mkdir\_change\_prompt\_dir\_string:

3029 <1> ;call change\_prompt\_dir\_string

3030 <1>

3031 <1> loc\_mkdir\_find\_directory:

3032 <1> ;mov esi, FindFile\_Name

3033 00008724 8B35[60630100] <1> mov esi, [DelFile\_FNPointer]

3034 <1> ;xor eax, eax

3035 0000872A 6631C0 <1> xor ax, ax ; any name (dir, file, volume)

3036 0000872D E831FBFFFF <1> call find\_first\_file

3037 00008732 721E <1> jc short loc\_mkdir\_check\_error\_code

3038 <1>

3039 <1> loc\_mkdir\_directory\_found:

3040 00008734 BE[17110100] <1> mov esi, Msg\_Name\_Exists

3041 00008739 E81FDCFFFF <1> call print\_msg

3042 <1>

3043 0000873E E94EFFFFFF <1> jmp loc\_file\_rw\_restore\_retn

3044 <1>

3045 <1> loc\_mkdir\_invalid\_dir\_name\_chars:

3046 00008743 BE[EA100100] <1> mov esi, Msg\_invalid\_name\_chars

3047 00008748 E810DCFFFF <1> call print\_msg

3048 <1>

3049 0000874D E93FFFFFFF <1> jmp loc\_file\_rw\_restore\_retn

3050 <1>

3051 <1> loc\_mkdir\_check\_error\_code:

3052 00008752 3C02 <1> cmp al, 2

3053 <1> ;je short loc\_mkdir\_directory\_not\_found

3054 00008754 7406 <1> je short loc\_mkdir\_ask\_for\_yes\_no

3055 00008756 F9 <1> stc

3056 00008757 E935FFFFFF <1> jmp loc\_file\_rw\_cmd\_failed

3057 <1>

3058 <1> loc\_mkdir\_directory\_not\_found:

3059 <1> loc\_mkdir\_ask\_for\_yes\_no:

3060 0000875C BE[38110100] <1> mov esi, Msg\_DoYouWantMkdir

3061 00008761 E8F7DBFFFF <1> call print\_msg

3062 00008766 8B35[60630100] <1> mov esi, [DelFile\_FNPointer]

3063 0000876C E8ECDBFFFF <1> call print\_msg

3064 00008771 BE[57110100] <1> mov esi, Msg\_YesNo

3065 00008776 E8E2DBFFFF <1> call print\_msg

3066 <1>

3067 0000877B C605[61110100]20 <1> mov byte [Y\_N\_nextline], 20h

3068 <1>

3069 <1> loc\_mkdir\_ask\_again:

3070 00008782 30E4 <1> xor ah, ah

3071 00008784 E88D84FFFF <1> call int16h

3072 00008789 3C1B <1> cmp al, 1Bh

3073 <1> ;je short loc\_do\_not\_make\_directory

3074 0000878B 7439 <1> je short loc\_mkdir\_y\_n\_escape

3075 0000878D 24DF <1> and al, 0DFh ; y -> Y, n -> N

3076 0000878F 3C59 <1> cmp al, 'Y' ; 'yes'

3077 00008791 7404 <1> je short loc\_mkdir\_yes\_make\_directory

3078 00008793 3C4E <1> cmp al, 'N' ; 'no'

3079 00008795 75EB <1> jne short loc\_mkdir\_ask\_again

3080 <1>

3081 <1> loc\_do\_not\_make\_directory:

3082 <1> loc\_mkdir\_yes\_make\_directory:

3083 00008797 E82E000000 <1> call y\_n\_answer ; 29/12/2017

3084 <1> ;cmp al, 'Y' ; 'yes'

3085 <1> ;cmc

3086 <1> ;jnc loc\_file\_rw\_restore\_retn

3087 0000879C 3C4E <1> cmp al, 'N' ; 'no'

3088 0000879E 0F84EDFEFFFF <1> je loc\_file\_rw\_restore\_retn

3089 <1>

3090 <1> loc\_mkdir\_call\_make\_sub\_directory:

3091 000087A4 8B35[60630100] <1> mov esi, [DelFile\_FNPointer]

3092 000087AA B110 <1> mov cl, 10h ; Directory attributes

3093 000087AC E8821D0000 <1> call make\_sub\_directory

3094 <1> loc\_rename\_file\_ok: ; 06/03/2016

3095 000087B1 0F82DAFEFFFF <1> jc loc\_file\_rw\_cmd\_failed

3096 <1> move\_source\_file\_to\_destination\_OK:

3097 000087B7 BE[65110100] <1> mov esi, Msg\_OK

3098 000087BC E89CDBFFFF <1> call print\_msg

3099 000087C1 E9CBFEFFFF <1> jmp loc\_file\_rw\_restore\_retn

3100 <1>

3101 <1> loc\_mkdir\_y\_n\_escape:

3102 000087C6 B04E <1> mov al, 'N' ; 'no'

3103 000087C8 EBCD <1> jmp short loc\_do\_not\_make\_directory

3104 <1>

3105 <1> y\_n\_answer:

3106 <1> ; 29/12/2017

3107 000087CA A2[61110100] <1> mov [Y\_N\_nextline], al

3108 <1> ;push ax

3109 000087CF 50 <1> push eax

3110 000087D0 BE[61110100] <1> mov esi, Y\_N\_nextline

3111 000087D5 E883DBFFFF <1> call print\_msg

3112 000087DA 58 <1> pop eax

3113 <1> ;pop ax

3114 000087DB C3 <1> retn

3115 <1>

3116 <1> delete\_directory:

3117 <1> ; 29/12/2017

3118 <1> ; 15/10/2016

3119 <1> ; 01/03/2016, 06/03/2016

3120 <1> ; 27/02/2016, 28/02/2016, 29/02/2016

3121 <1> ; 26/02/2016 (TRDOS 386 = TRDOS v2.0)

3122 <1> ; 16/10/2010 (CMD\_INTR.ASM, 'cmp\_cmd\_rmdir')

3123 <1> ; 05/06/2010

3124 <1> ;

3125 <1> get\_fchar:

3126 <1> ; esi = directory name

3127 000087DC 803E20 <1> cmp byte [esi], 20h

3128 000087DF 7701 <1> ja short loc\_rmdir\_parse\_path\_name

3129 <1>

3130 <1> loc\_rmdir\_nodirname\_retn:

3131 000087E1 C3 <1> retn

3132 <1>

3133 <1> loc\_rmdir\_parse\_path\_name:

3134 000087E2 BF[A2620100] <1> mov edi, FindFile\_Drv

3135 000087E7 E8481C0000 <1> call parse\_path\_name

3136 000087EC 0F8261F5FFFF <1> jc loc\_cmd\_failed

3137 <1>

3138 <1> loc\_rmdir\_check\_dirname\_exists:

3139 000087F2 BE[E4620100] <1> mov esi, FindFile\_Name

3140 000087F7 803E20 <1> cmp byte [esi], 20h

3141 000087FA 0F8653F5FFFF <1> jna loc\_cmd\_failed

3142 00008800 8935[60630100] <1> mov [DelFile\_FNPointer], esi

3143 <1>

3144 <1> loc\_rmdir\_drv:

3145 00008806 8A35[FE580100] <1> mov dh, [Current\_Drv]

3146 0000880C 8835[5E610100] <1> mov [RUN\_CDRV], dh

3147 <1>

3148 00008812 8A15[A2620100] <1> mov dl, [FindFile\_Drv]

3149 00008818 38F2 <1> cmp dl, dh

3150 0000881A 740B <1> je short loc\_rmdir\_change\_directory

3151 <1>

3152 0000881C E89FE6FFFF <1> call change\_current\_drive

3153 00008821 0F826AFEFFFF <1> jc loc\_file\_rw\_cmd\_failed

3154 <1>

3155 <1> loc\_rmdir\_change\_directory:

3156 00008827 803D[A3620100]20 <1> cmp byte [FindFile\_Directory], 20h

3157 0000882E 7614 <1> jna short loc\_rmdir\_find\_directory

3158 <1>

3159 00008830 FE05[D30C0100] <1> inc byte [Restore\_CDIR]

3160 00008836 BE[A3620100] <1> mov esi, FindFile\_Directory

3161 0000883B 30E4 <1> xor ah, ah ; CD\_COMMAND sign -> 0

3162 0000883D E8DC150000 <1> call change\_current\_directory

3163 00008842 7211 <1> jc short loc\_rmdir\_check\_error\_code

3164 <1>

3165 <1> ;loc\_rmdir\_change\_prompt\_dir\_string:

3166 <1> ;call change\_prompt\_dir\_string

3167 <1>

3168 <1> loc\_rmdir\_find\_directory:

3169 <1> ;mov esi, FindFile\_Name

3170 00008844 8B35[60630100] <1> mov esi, [DelFile\_FNPointer]

3171 0000884A 66B81008 <1> mov ax, 0810h ; Only directories

3172 0000884E E810FAFFFF <1> call find\_first\_file

3173 00008853 730A <1> jnc short loc\_rmdir\_ambgfn\_check

3174 <1>

3175 <1> loc\_rmdir\_check\_error\_code:

3176 00008855 3C02 <1> cmp al, 2

3177 00008857 740B <1> je short loc\_rmdir\_directory\_not\_found

3178 00008859 F9 <1> stc

3179 0000885A E932FEFFFF <1> jmp loc\_file\_rw\_cmd\_failed

3180 <1>

3181 <1> loc\_rmdir\_ambgfn\_check:

3182 0000885F 6621D2 <1> and dx, dx ; Ambiguous filename chars used sign (DX>0)

3183 00008862 740F <1> jz short loc\_rmdir\_directory\_found

3184 <1>

3185 <1> loc\_rmdir\_directory\_not\_found:

3186 00008864 BE[D50F0100] <1> mov esi, Msg\_Dir\_Not\_Found

3187 00008869 E8EFDAFFFF <1> call print\_msg

3188 <1>

3189 0000886E E91EFEFFFF <1> jmp loc\_file\_rw\_restore\_retn

3190 <1>

3191 <1> loc\_rmdir\_directory\_found:

3192 00008873 80E307 <1> and bl, 07h ; Attributes

3193 00008876 0F851FFEFFFF <1> jnz loc\_permission\_denied

3194 <1>

3195 <1> loc\_rmdir\_save\_lnel: ; 28/02/2016

3196 <1> ;mov bh, [LongName\_EntryLength]

3197 0000887C 883D[6A630100] <1> mov [DelFile\_LNEL], bh ; Long name entry length (if > 0)

3198 <1> ; edi = Directory Entry Offset (DirBuff)

3199 <1> ; esi = Directory Entry (FFF Structure)

3200 <1> ;mov [DelFile\_DirEntryAddr], edi ; not required

3201 <1> ;mov ax, [edi+20] ; First Cluster High Word

3202 <1> ;shl eax, 16

3203 <1> ;mov ax, [edi+26] ; First Cluster Low Word

3204 <1> ; ROOT Dir First Cluster = 0

3205 <1> ;cmp eax, 2

3206 <1> ;jb loc\_update\_direntry\_1

3207 <1>

3208 <1> pass\_rmdir\_fc\_check:

3209 00008882 57 <1> push edi ; \* (29/02/2016)

3210 <1>

3211 00008883 BE[6B110100] <1> mov esi, Msg\_DoYouWantRmDir

3212 00008888 E8D0DAFFFF <1> call print\_msg

3213 0000888D 8B35[60630100] <1> mov esi, [DelFile\_FNPointer]

3214 00008893 E8C5DAFFFF <1> call print\_msg

3215 00008898 BE[57110100] <1> mov esi, Msg\_YesNo

3216 0000889D E8BBDAFFFF <1> call print\_msg

3217 <1>

3218 <1> loc\_rmdir\_ask\_again:

3219 000088A2 30E4 <1> xor ah, ah

3220 000088A4 E86D83FFFF <1> call int16h

3221 000088A9 3C1B <1> cmp al, 1Bh

3222 <1> ;je short loc\_do\_not\_delete\_directory

3223 000088AB 7433 <1> je loc\_rmdir\_y\_n\_escape ; 06/03/2016

3224 000088AD 24DF <1> and al, 0DFh

3225 000088AF A2[61110100] <1> mov [Y\_N\_nextline], al

3226 000088B4 3C59 <1> cmp al, 'Y'

3227 000088B6 7404 <1> je short loc\_rmdir\_yes\_delete\_directory

3228 000088B8 3C4E <1> cmp al, 'N'

3229 000088BA 75E6 <1> jne short loc\_rmdir\_ask\_again

3230 <1>

3231 <1> loc\_do\_not\_delete\_directory:

3232 <1> loc\_rmdir\_yes\_delete\_directory:

3233 000088BC E809FFFFFF <1> call y\_n\_answer ; 29/12/2017

3234 000088C1 5F <1> pop edi ; \* (29/02/2016)

3235 <1> ;cmp al, 'Y' ; 'yes'

3236 <1> ;cmc

3237 <1> ;jnc loc\_file\_rw\_restore\_retn

3238 000088C2 3C4E <1> cmp al, 'N' ; 'no'

3239 000088C4 0F84C7FDFFFF <1> je loc\_file\_rw\_restore\_retn

3240 <1>

3241 <1> ; 29/12/2017

3242 000088CA E869000000 <1> call delete\_sub\_directory

3243 000088CF 7213 <1> jc short loc\_rmdir\_cmd\_failed

3244 <1>

3245 <1> loc\_rmdir\_ok:

3246 000088D1 BE[65110100] <1> mov esi, Msg\_OK

3247 000088D6 E882DAFFFF <1> call print\_msg

3248 000088DB E9B1FDFFFF <1> jmp loc\_file\_rw\_restore\_retn

3249 <1>

3250 <1> loc\_rmdir\_y\_n\_escape:

3251 000088E0 B04E <1> mov al, 'N' ; 'no'

3252 000088E2 EBD8 <1> jmp loc\_do\_not\_delete\_directory

3253 <1>

3254 <1> loc\_rmdir\_cmd\_failed:

3255 <1> ; 29/12/2017

3256 000088E4 09C0 <1> or eax, eax ; EAX = 0 -> Directory not empty!

3257 000088E6 7426 <1> jz short loc\_rmdir\_directory\_not\_empty

3258 <1>

3259 <1> ; EAX > 0 -> Error code in AL (or AX or EAX)

3260 <1>

3261 000088E8 833D[1E610100]01 <1> cmp dword [FAT\_ClusterCounter], 1

3262 000088EF 0F829CFDFFFF <1> jb loc\_file\_rw\_cmd\_failed

3263 000088F5 F9 <1> stc

3264 <1> loc\_rmdir\_cmd\_return:

3265 <1> ; 01/03/2016

3266 000088F6 9C <1> pushf

3267 <1> ; ESI = Logical DOS Drive Description Table address

3268 000088F7 66BB00FF <1> mov bx, 0FF00h ; BH = FFh -> use ESI for Drive parameters

3269 <1> ; BL = 0 -> Recalculate free cluster count

3270 000088FB 50 <1> push eax

3271 000088FC E8C3380000 <1> call calculate\_fat\_freespace

3272 00008901 58 <1> pop eax

3273 00008902 9D <1> popf

3274 00008903 0F8288FDFFFF <1> jc loc\_file\_rw\_cmd\_failed

3275 00008909 E983FDFFFF <1> jmp loc\_file\_rw\_restore\_retn

3276 <1>

3277 <1> loc\_rmdir\_directory\_not\_empty:

3278 0000890E BE[8C110100] <1> mov esi, Msg\_Dir\_Not\_Empty

3279 00008913 E845DAFFFF <1> call print\_msg

3280 <1> ; 01/03/2016

3281 00008918 A1[1E610100] <1> mov eax, [FAT\_ClusterCounter]

3282 0000891D 09C0 <1> or eax, eax ; 0 ?

3283 0000891F 0F846CFDFFFF <1> jz loc\_file\_rw\_restore\_retn

3284 <1> ; ESI = Logical DOS Drive Description Table address

3285 00008925 66BB01FF <1> mov bx, 0FF01h ; BH = FFh -> use ESI for Drive parameters

3286 <1> ; BL = 1 -> add free clusters

3287 00008929 E896380000 <1> call calculate\_fat\_freespace

3288 0000892E 09C9 <1> or ecx, ecx

3289 00008930 0F845BFDFFFF <1> jz loc\_file\_rw\_restore\_retn ; ecx = 0 -> OK

3290 <1> ; ecx > 0 -> Error (Recalculation is needed)

3291 00008936 EBBE <1> jmp short loc\_rmdir\_cmd\_return

3292 <1>

3293 <1>

3294 <1> delete\_sub\_directory:

3295 <1> ; 29/12/2017

3296 <1> ; (moved here from 'delete\_directory' for 'sysrmdir' )

3297 <1>

3298 <1> ; EDI = Directory buffer entry offset/address

3299 <1>

3300 <1> loc\_rmdir\_delete\_short\_name\_check\_dir\_empty:

3301 00008938 668B4714 <1> mov ax, [edi+20] ; First Cluster High Word

3302 0000893C C1E010 <1> shl eax, 16

3303 0000893F 668B471A <1> mov ax, [edi+26] ; First Cluster Low Word

3304 <1>

3305 <1> ;mov [DelFile\_FCluster], eax

3306 <1>

3307 <1> ;;mov bx, [DirBuff\_EntryCounter]

3308 <1> ;mov bx, [FindFile\_DirEntryNumber] ; 27/02/2016

3309 <1> ;mov [DelFile\_EntryCounter], bx

3310 <1>

3311 00008943 29DB <1> sub ebx, ebx

3312 <1> ; 29/12/2017

3313 00008945 891D[1E610100] <1> mov [FAT\_ClusterCounter], ebx ; 0 ; Reset

3314 <1>

3315 0000894B 8A3D[A2620100] <1> mov bh, [FindFile\_Drv]

3316 00008951 BE00010900 <1> mov esi, Logical\_DOSDisks

3317 00008956 01DE <1> add esi, ebx

3318 <1>

3319 00008958 66817F0CA101 <1> cmp word [edi+DirEntry\_NTRes], 01A1h

3320 0000895E 745A <1> je short loc\_rmdir\_check\_fs\_directory

3321 <1>

3322 <1> ;cmp byte [esi+LD\_FATType], 1

3323 <1> ;jb short loc\_rmdir\_get\_\_last\_cluster\_0

3324 <1>

3325 <1> ; 29/12/2017

3326 00008960 83F802 <1> cmp eax, 2

3327 00008963 7306 <1> jnb short loc\_rmdir\_get\_last\_cluster\_1

3328 <1> ; eax < 2

3329 <1> loc\_rmdir\_get\_last\_cluster\_0:

3330 <1> ;mov eax, ERR\_INV\_FORMAT ; invalid format!

3331 00008965 B813000000 <1> mov eax, ERR\_NOT\_DIR ; not a valid directory!

3332 <1> ;stc

3333 0000896A C3 <1> retn

3334 <1>

3335 <1> loc\_rmdir\_get\_last\_cluster\_1:

3336 0000896B 807E0303 <1> cmp byte [esi+LD\_FATType], 3 ; FAT32

3337 0000896F 750C <1> jne short loc\_rmdir\_get\_last\_cluster\_2

3338 <1>

3339 <1> ; is it root directory ?

3340 00008971 3B4632 <1> cmp eax, [esi+LD\_BPB+BPB\_RootClus]

3341 00008974 7507 <1> jne short loc\_rmdir\_get\_last\_cluster\_2

3342 <1>

3343 <1> ; root directory can not be deleted !!

3344 <1> loc\_rmdir\_permission\_denied:

3345 00008976 B80B000000 <1> mov eax, ERR\_PERM\_DENIED ; permission denied!

3346 0000897B F9 <1> stc

3347 0000897C C3 <1> retn

3348 <1>

3349 <1> loc\_rmdir\_get\_last\_cluster\_2:

3350 <1> ; 29/12/2017

3351 0000897D A3[64630100] <1> mov [DelFile\_FCluster], eax

3352 <1>

3353 <1> ;mov dx, [DirBuff\_EntryCounter]

3354 00008982 668B15[1C630100] <1> mov dx, [FindFile\_DirEntryNumber] ; 27/02/2016

3355 00008989 668915[68630100] <1> mov [DelFile\_EntryCounter], dx

3356 <1>

3357 00008990 8B15[2D610100] <1> mov edx, [DirBuff\_Cluster]

3358 00008996 8915[94630100] <1> mov [RmDir\_ParentDirCluster], edx

3359 <1>

3360 0000899C 893D[90630100] <1> mov [RmDir\_DirEntryOffset], edi

3361 <1>

3362 <1> ; 01/03/2016

3363 <1> ;mov dword [FAT\_ClusterCounter], 0 ; Reset

3364 <1>

3365 <1> loc\_rmdir\_get\_last\_cluster\_3:

3366 000089A2 E89C390000 <1> call get\_last\_cluster

3367 <1> ;jc loc\_rmdir\_cmd\_failed

3368 000089A7 721E <1> jc short loc\_delete\_sub\_dir\_retn ; 29/12/2017

3369 <1>

3370 000089A9 3B05[64630100] <1> cmp eax, [DelFile\_FCluster]

3371 000089AF 7517 <1> jne short loc\_rmdir\_multi\_dir\_clusters

3372 <1>

3373 000089B1 C605[8F630100]00 <1> mov byte [RmDir\_MultiClusters], 0

3374 000089B8 EB15 <1> jmp short pass\_rmdir\_multi\_dir\_clusters

3375 <1>

3376 <1> loc\_rmdir\_check\_fs\_directory:

3377 <1> ; 29/12/2017

3378 000089BA 807E04A1 <1> cmp byte [esi+LD\_FSType], 0A1h

3379 000089BE 75B6 <1> jne short loc\_rmdir\_permission\_denied

3380 <1>

3381 <1> loc\_rmdir\_delete\_fs\_directory:

3382 000089C0 E876130000 <1> call delete\_fs\_directory

3383 <1> ;jnc loc\_print\_deleted\_message

3384 000089C5 7300 <1> jnc short loc\_delete\_sub\_dir\_retn ; 29/12/2017

3385 <1>

3386 <1> ; EAX=0 -> Directory not empty !

3387 <1> ; EAX>0 -> Disk r/w error or another (misc) error

3388 <1>

3389 <1> ;or eax, eax

3390 <1> ;jz loc\_rmdir\_directory\_not\_empty\_2

3391 <1> ;;stc

3392 <1> ;;jmp loc\_file\_rw\_cmd\_failed

3393 <1>

3394 <1> loc\_delete\_sub\_dir\_retn:

3395 000089C7 C3 <1> retn

3396 <1>

3397 <1> loc\_rmdir\_multi\_dir\_clusters:

3398 000089C8 C605[8F630100]01 <1> mov byte [RmDir\_MultiClusters], 1

3399 <1>

3400 <1> pass\_rmdir\_multi\_dir\_clusters:

3401 000089CF A3[98630100] <1> mov [RmDir\_DirLastCluster], eax

3402 000089D4 890D[9C630100] <1> mov [RmDir\_PreviousCluster], ecx

3403 <1>

3404 <1> loc\_rmdir\_load\_fat\_sub\_directory:

3405 000089DA E84F330000 <1> call load\_FAT\_sub\_directory

3406 <1> ;jc loc\_rmdir\_cmd\_failed

3407 000089DF 72E6 <1> jc short loc\_delete\_sub\_dir\_retn

3408 <1>

3409 <1> loc\_rmdir\_find\_last\_dir\_entry:

3410 000089E1 56 <1> push esi

3411 000089E2 BE[86620100] <1> mov esi, Dir\_File\_Name

3412 000089E7 C6062A <1> mov byte [esi], '\*'

3413 000089EA C646082A <1> mov byte [esi+8], '\*'

3414 000089EE 31DB <1> xor ebx, ebx ; Entry offset = 0

3415 <1> loc\_rmdir\_find\_last\_dir\_entry\_next:

3416 000089F0 66B80008 <1> mov ax, 0800h ; Except volume/long names

3417 000089F4 6631C9 <1> xor cx, cx ; 0 = Find a valid file or dir name

3418 000089F7 E87A170000 <1> call find\_directory\_entry

3419 000089FC 7225 <1> jc short loc\_rmdir\_empty\_dir\_cluster

3420 000089FE 83FB01 <1> cmp ebx, 1

3421 00008A01 771B <1> ja short loc\_rmdir\_directory\_not\_empty\_1

3422 <1> loc\_rmdir\_dot\_entry\_check:

3423 00008A03 80FD2E <1> cmp ch, '.' ; The first char of the dir entry

3424 00008A06 7516 <1> jne short loc\_rmdir\_directory\_not\_empty\_1

3425 00008A08 08DB <1> or bl, bl

3426 00008A0A 7506 <1> jnz short loc\_rmdir\_dotdot\_entry\_check

3427 00008A0C 807F0120 <1> cmp byte [edi+1], 20h

3428 00008A10 EB06 <1> jmp short pass\_rmdir\_dot\_entry\_check

3429 <1>

3430 <1> loc\_rmdir\_dotdot\_entry\_check:

3431 00008A12 66817F012E20 <1> cmp word [edi+1], '. '

3432 <1> pass\_rmdir\_dot\_entry\_check:

3433 00008A18 7504 <1> jne short loc\_rmdir\_directory\_not\_empty\_1

3434 00008A1A FEC3 <1> inc bl

3435 00008A1C EBD2 <1> jmp short loc\_rmdir\_find\_last\_dir\_entry\_next

3436 <1>

3437 <1> loc\_rmdir\_directory\_not\_empty\_1:

3438 00008A1E 58 <1> pop eax ; pushed esi

3439 00008A1F 31C0 <1> xor eax, eax ; 0

3440 <1> loc\_rmdir\_directory\_not\_empty\_2:

3441 <1> loc\_delete\_sub\_dir\_stc\_retn:

3442 00008A21 F9 <1> stc

3443 00008A22 C3 <1> retn

3444 <1>

3445 <1> loc\_rmdir\_empty\_dir\_cluster:

3446 00008A23 5E <1> pop esi

3447 <1>

3448 <1> loc\_rmdir\_set\_prev\_cluster\_dir\_last\_cluster:

3449 00008A24 803D[8F630100]00 <1> cmp byte [RmDir\_MultiClusters], 0

3450 00008A2B 7613 <1> jna short loc\_rmdir\_unlink\_dir\_last\_cluster

3451 <1>

3452 00008A2D A1[9C630100] <1> mov eax, [RmDir\_PreviousCluster]

3453 <1> ;xor ecx, ecx

3454 00008A32 49 <1> dec ecx ; FFFFFFFFh

3455 00008A33 E83A340000 <1> call update\_cluster

3456 00008A38 7306 <1> jnc short loc\_rmdir\_unlink\_dir\_last\_cluster

3457 <1>

3458 <1> ; 01/03/2016

3459 <1> ;cmp eax, 1 ; eax = 0 -> end of cluster chain

3460 <1> ;cmc

3461 <1> ;jc short loc\_rmdir\_cmd\_failed

3462 <1> ;jmp short loc\_rmdir\_save\_fat\_buffer

3463 <1> ; 29/12/2017

3464 00008A3A 21C0 <1> and eax, eax

3465 00008A3C 75E3 <1> jnz short loc\_delete\_sub\_dir\_stc\_retn

3466 00008A3E EB12 <1> jmp short loc\_rmdir\_save\_fat\_buffer

3467 <1>

3468 <1> loc\_rmdir\_unlink\_dir\_last\_cluster:

3469 00008A40 A1[98630100] <1> mov eax, [RmDir\_DirLastCluster]

3470 00008A45 31C9 <1> xor ecx, ecx ; 0

3471 00008A47 E826340000 <1> call update\_cluster

3472 00008A4C 7327 <1> jnc short loc\_rmdir\_unlink\_stc\_retn\_0Bh

3473 <1> ; Because of it is the last cluster

3474 <1> ; 'update\_cluster' must return with eocc error

3475 00008A4E 09C0 <1> or eax, eax

3476 <1> ;jz short loc\_rmdir\_save\_fat\_buffer ; eocc

3477 <1> ;stc

3478 <1> ;jmp short loc\_rmdir\_cmd\_failed

3479 <1> ; 29/12/2017

3480 00008A50 75CF <1> jnz short loc\_delete\_sub\_dir\_stc\_retn

3481 <1>

3482 <1> loc\_rmdir\_save\_fat\_buffer:

3483 00008A52 803D[16610100]02 <1> cmp byte [FAT\_BuffValidData], 2

3484 00008A59 7528 <1> jne short loc\_rmdir\_calculate\_FAT\_freespace

3485 00008A5B E8CF360000 <1> call save\_fat\_buffer

3486 <1> ;jc short loc\_rmdir\_cmd\_failed

3487 <1> ; 29/12/2017

3488 00008A60 7219 <1> jc short loc\_rmdir\_unlink\_error\_retn

3489 <1>

3490 <1> ; 01/03/2016

3491 00008A62 803D[8F630100]00 <1> cmp byte [RmDir\_MultiClusters], 0

3492 00008A69 7618 <1> jna short loc\_rmdir\_calculate\_FAT\_freespace

3493 <1>

3494 00008A6B A1[64630100] <1> mov eax, [DelFile\_FCluster]

3495 00008A70 E92DFFFFFF <1> jmp loc\_rmdir\_get\_last\_cluster\_3

3496 <1>

3497 <1> loc\_rmdir\_unlink\_stc\_retn\_0Bh:

3498 <1> ; 15/10/2016 (0Bh -> 28)

3499 00008A75 B81C000000 <1> mov eax, ERR\_INV\_FORMAT ; 28 = Invalid format

3500 <1> loc\_rmdir\_unlink\_stc\_retn:

3501 00008A7A F9 <1> stc

3502 <1> loc\_rmdir\_unlink\_error\_retn:

3503 00008A7B C3 <1> retn

3504 <1>

3505 <1> loc\_rmdir\_delete\_short\_name\_invalid\_data:

3506 00008A7C B81D000000 <1> mov eax, 29 ; Invalid data (15/10/2016)

3507 <1> ;stc

3508 <1> ;jmp loc\_rmdir\_cmd\_failed

3509 <1> ; 29/12/2017

3510 00008A81 EBF7 <1> jmp short loc\_rmdir\_unlink\_stc\_retn

3511 <1>

3512 <1> loc\_rmdir\_calculate\_FAT\_freespace:

3513 <1> ;mov eax, [FAT\_ClusterCounter]

3514 <1> ; 29/12/2017

3515 00008A83 29C0 <1> sub eax, eax ; 0

3516 00008A85 8705[1E610100] <1> xchg eax, [FAT\_ClusterCounter]

3517 <1> ;

3518 00008A8B 66BB01FF <1> mov bx, 0FF01h

3519 <1> ; BL = 1 -> Add EAX to free space count

3520 <1> ; BH = FFh ->

3521 <1> ; ESI = Logical DOS Drive Description Table address

3522 00008A8F E830370000 <1> call calculate\_fat\_freespace

3523 <1>

3524 00008A94 21C9 <1> and ecx, ecx ; ecx = 0 -> valid free sector count

3525 00008A96 7409 <1> jz short loc\_rmdir\_delete\_short\_name\_continue

3526 <1>

3527 <1> loc\_rmdir\_recalculate\_FAT\_freespace:

3528 00008A98 66BB00FF <1> mov bx, 0FF00h ; BL = 0 -> Recalculate free space

3529 00008A9C E823370000 <1> call calculate\_fat\_freespace

3530 <1>

3531 <1> loc\_rmdir\_delete\_short\_name\_continue:

3532 00008AA1 A1[94630100] <1> mov eax, [RmDir\_ParentDirCluster]

3533 00008AA6 83F802 <1> cmp eax, 2

3534 00008AA9 7309 <1> jnb short loc\_rmdir\_del\_short\_name\_load\_sub\_dir

3535 00008AAB E8F3310000 <1> call load\_FAT\_root\_directory

3536 <1> ;jc loc\_file\_rw\_cmd\_failed

3537 <1> ; 29/12/2017

3538 00008AB0 72C9 <1> jc short loc\_rmdir\_unlink\_error\_retn

3539 00008AB2 EB07 <1> jmp short loc\_rmdir\_del\_short\_name\_ld\_chk\_fclust

3540 <1>

3541 <1> loc\_rmdir\_del\_short\_name\_load\_sub\_dir:

3542 00008AB4 E875320000 <1> call load\_FAT\_sub\_directory

3543 <1> ;jc loc\_file\_rw\_cmd\_failed

3544 <1> ; 29/12/2017

3545 00008AB9 72C0 <1> jc short loc\_rmdir\_unlink\_error\_retn

3546 <1>

3547 <1> loc\_rmdir\_del\_short\_name\_ld\_chk\_fclust:

3548 00008ABB 0FB73D[90630100] <1> movzx edi, word [RmDir\_DirEntryOffset]

3549 00008AC2 81C700000800 <1> add edi, Directory\_Buffer

3550 <1>

3551 00008AC8 668B4714 <1> mov ax, [edi+20] ; First Cluster High Word

3552 00008ACC C1E010 <1> shl eax, 16

3553 00008ACF 668B471A <1> mov ax, [edi+26] ; First Cluster Low Word

3554 <1> ; Not necessary...

3555 00008AD3 3B05[64630100] <1> cmp eax, [DelFile\_FCluster]

3556 00008AD9 75A1 <1> jne short loc\_rmdir\_delete\_short\_name\_invalid\_data

3557 <1> ;

3558 00008ADB C607E5 <1> mov byte [edi], 0E5h ; 'Deleted' sign

3559 <1> ; 27/02/2016

3560 <1> ; TRDOS v1 has a bug here! it does not set

3561 <1> ; 'DirBuff\_ValidData' to 2; as result of this bug,

3562 <1> ; 'save\_directory\_buffer' would not save the change !

3563 00008ADE C605[28610100]02 <1> mov byte [DirBuff\_ValidData], 2 ; change sign

3564 <1> ;

3565 00008AE5 E8AE1D0000 <1> call save\_directory\_buffer

3566 <1> ;jc loc\_file\_rw\_cmd\_failed

3567 <1> ; 29/12/2017

3568 00008AEA 728F <1> jc short loc\_rmdir\_unlink\_error\_retn

3569 <1>

3570 <1> loc\_rmdir\_del\_long\_name:

3571 00008AEC 0FB615[6A630100] <1> movzx edx, byte [DelFile\_LNEL]

3572 00008AF3 08D2 <1> or dl, dl

3573 00008AF5 7410 <1> jz short loc\_rmdir\_update\_parent\_dir\_lmdt

3574 <1>

3575 00008AF7 0FB705[68630100] <1> movzx eax, word [DelFile\_EntryCounter]

3576 00008AFE 29D0 <1> sub eax, edx

3577 <1> ; 29/12/2017

3578 00008B00 7205 <1> jc short loc\_rmdir\_update\_parent\_dir\_lmdt

3579 <1>

3580 <1> ; EAX = Directory Entry Number of the long name last entry

3581 00008B02 E8EF1E0000 <1> call delete\_longname

3582 <1>

3583 <1> loc\_rmdir\_update\_parent\_dir\_lmdt:

3584 00008B07 E8271E0000 <1> call update\_parent\_dir\_lmdt

3585 <1> ;jc short loc\_file\_rw\_cmd\_failed

3586 <1> ; 29/12/2017

3587 <1> ;jc short loc\_rmdir\_unlink\_error\_retn

3588 <1>

3589 <1> loc\_delete\_sub\_directory\_ok:

3590 <1> ; 29/12/2017

3591 00008B0C 31C0 <1> xor eax, eax ; 0 ; cf = 0

3592 00008B0E C3 <1> retn

3593 <1>

3594 <1>

3595 <1> delete\_file:

3596 <1> ; 29/02/2016

3597 <1> ; 28/02/2016 (TRDOS 386 = TRDOS v2.0)

3598 <1> ; 09/08/2010 (CMD\_INTR.ASM, 'cmp\_cmd\_del')

3599 <1> ; 28/02/2010

3600 <1>

3601 <1> get\_delfile\_fchar:

3602 <1> ; esi = file name

3603 00008B0F 803E20 <1> cmp byte [esi], 20h

3604 00008B12 7701 <1> ja short loc\_delfile\_parse\_path\_name

3605 <1>

3606 <1> loc\_delfile\_nofilename\_retn:

3607 00008B14 C3 <1> retn

3608 <1>

3609 <1> loc\_delfile\_parse\_path\_name:

3610 00008B15 BF[A2620100] <1> mov edi, FindFile\_Drv

3611 00008B1A E815190000 <1> call parse\_path\_name

3612 00008B1F 0F822EF2FFFF <1> jc loc\_cmd\_failed

3613 <1>

3614 <1> loc\_delfile\_check\_filename\_exists:

3615 00008B25 BE[E4620100] <1> mov esi, FindFile\_Name

3616 00008B2A 803E20 <1> cmp byte [esi], 20h

3617 00008B2D 0F8620F2FFFF <1> jna loc\_cmd\_failed

3618 00008B33 8935[60630100] <1> mov [DelFile\_FNPointer], esi

3619 <1>

3620 <1> loc\_delfile\_drv:

3621 00008B39 8A15[A2620100] <1> mov dl, [FindFile\_Drv]

3622 00008B3F 8A35[FE580100] <1> mov dh, [Current\_Drv]

3623 00008B45 8835[5E610100] <1> mov [RUN\_CDRV], dh

3624 00008B4B 38F2 <1> cmp dl, dh

3625 00008B4D 740B <1> je short loc\_delfile\_change\_directory

3626 <1>

3627 00008B4F E86CE3FFFF <1> call change\_current\_drive

3628 00008B54 0F8237FBFFFF <1> jc loc\_file\_rw\_cmd\_failed

3629 <1>

3630 <1> loc\_delfile\_change\_directory:

3631 00008B5A 803D[A3620100]20 <1> cmp byte [FindFile\_Directory], 20h

3632 00008B61 7618 <1> jna short loc\_delfile\_find

3633 <1>

3634 00008B63 FE05[D30C0100] <1> inc byte [Restore\_CDIR]

3635 00008B69 BE[A3620100] <1> mov esi, FindFile\_Directory

3636 00008B6E 30E4 <1> xor ah, ah ; CD\_COMMAND sign -> 0

3637 00008B70 E8A9120000 <1> call change\_current\_directory

3638 00008B75 0F8216FBFFFF <1> jc loc\_file\_rw\_cmd\_failed

3639 <1>

3640 <1> ;loc\_delfile\_change\_prompt\_dir\_string:

3641 <1> ;call change\_prompt\_dir\_string

3642 <1>

3643 <1> loc\_delfile\_find:

3644 <1> ;mov esi, FindFile\_Name

3645 00008B7B 8B35[60630100] <1> mov esi, [DelFile\_FNPointer]

3646 00008B81 66B80018 <1> mov ax, 1800h ; Except volume label and dirs

3647 00008B85 E8D9F6FFFF <1> call find\_first\_file

3648 00008B8A 0F8201FBFFFF <1> jc loc\_file\_rw\_cmd\_failed

3649 <1>

3650 <1> loc\_delfile\_ambgfn\_check:

3651 00008B90 6621D2 <1> and dx, dx ; Ambiguous filename chars used sign (DX>0)

3652 00008B93 740B <1> jz short loc\_delfile\_found

3653 <1>

3654 <1> loc\_file\_not\_found:

3655 00008B95 B802000000 <1> mov eax, 2 ; File not found sign

3656 00008B9A F9 <1> stc

3657 00008B9B E9F1FAFFFF <1> jmp loc\_file\_rw\_cmd\_failed

3658 <1>

3659 <1> loc\_delfile\_found:

3660 00008BA0 80E307 <1> and bl, 07h ; Attributes

3661 00008BA3 0F85F2FAFFFF <1> jnz loc\_permission\_denied

3662 <1>

3663 <1> ;loc\_delfile\_found\_save\_lnel:

3664 <1> ; mov [DelFile\_LNEL], bh ; Long name entry length (if > 0)

3665 <1>

3666 <1> loc\_delfile\_ask\_for\_delete:

3667 00008BA9 57 <1> push edi ; \* (29/02/2016)

3668 <1>

3669 00008BAA BE[A3110100] <1> mov esi, Msg\_DoYouWantDelete

3670 00008BAF E8A9D7FFFF <1> call print\_msg

3671 00008BB4 8B35[60630100] <1> mov esi, [DelFile\_FNPointer]

3672 00008BBA E89ED7FFFF <1> call print\_msg

3673 00008BBF BE[57110100] <1> mov esi, Msg\_YesNo

3674 00008BC4 E894D7FFFF <1> call print\_msg

3675 <1>

3676 <1> loc\_delfile\_ask\_again:

3677 00008BC9 30E4 <1> xor ah, ah

3678 00008BCB E84680FFFF <1> call int16h

3679 00008BD0 3C1B <1> cmp al, 1Bh

3680 <1> ;je short loc\_do\_not\_delete\_file

3681 00008BD2 7449 <1> je short loc\_delfile\_y\_n\_escape ; 06/03/2016

3682 00008BD4 24DF <1> and al, 0DFh

3683 00008BD6 A2[61110100] <1> mov [Y\_N\_nextline], al

3684 00008BDB 3C59 <1> cmp al, 'Y'

3685 00008BDD 7404 <1> je short loc\_yes\_delete\_file

3686 00008BDF 3C4E <1> cmp al, 'N'

3687 00008BE1 75E6 <1> jne short loc\_delfile\_ask\_again

3688 <1>

3689 <1> loc\_do\_not\_delete\_file:

3690 <1> loc\_yes\_delete\_file:

3691 00008BE3 E8E2FBFFFF <1> call y\_n\_answer ; 29/12/2017

3692 00008BE8 5F <1> pop edi ; \* (29/02/2016)

3693 <1> ;cmp al, 'Y' ; 'yes'

3694 <1> ;cmc

3695 <1> ;jnc loc\_file\_rw\_restore\_retn

3696 00008BE9 3C4E <1> cmp al, 'N' ; 'no'

3697 00008BEB 0F84A0FAFFFF <1> je loc\_file\_rw\_restore\_retn

3698 <1>

3699 <1> loc\_delete\_file:

3700 00008BF1 8A3D[A2620100] <1> mov bh, [FindFile\_Drv]

3701 <1> ;mov bl, [DelFile\_LNEL]

3702 00008BF7 8A1D[F1620100] <1> mov bl, [FindFile\_LongNameEntryLength]

3703 <1> ;mov cx, [DirBuff\_EntryCounter]

3704 00008BFD 668B0D[1C630100] <1> mov cx, [FindFile\_DirEntryNumber]

3705 <1> ; (\*) EDI = Directory buffer entry offset/address

3706 00008C04 E8D71F0000 <1> call remove\_file ; (FILE.ASM, 'proc\_delete\_file')

3707 00008C09 0F8378FAFFFF <1> jnc loc\_print\_deleted\_message

3708 <1>

3709 <1> ;cmp al, 05h

3710 00008C0F 3C0B <1> cmp al, ERR\_PERM\_DENIED ; 29/12/2017 (5 -> 11)

3711 00008C11 0F8484FAFFFF <1> je loc\_permission\_denied

3712 00008C17 F9 <1> stc

3713 00008C18 E974FAFFFF <1> jmp loc\_file\_rw\_cmd\_failed

3714 <1>

3715 <1> loc\_delfile\_y\_n\_escape:

3716 00008C1D B04E <1> mov al, 'N' ; 'no'

3717 00008C1F EBC2 <1> jmp short loc\_do\_not\_delete\_file

3718 <1>

3719 <1> set\_file\_attributes:

3720 <1> ; 06/03/2016

3721 <1> ; 04/03/2016 (TRDOS 386 = TRDOS v2.0)

3722 <1> ; 10/07/2010 (TRDOS v1, CMD\_INTR.ASM, 'cmp\_cmd\_attrib')

3723 <1> ; 23/05/2010

3724 <1> ; 17/12/2000 (P2000.ASM)

3725 <1>

3726 <1> ; esi = file or directory name

3727 00008C21 6631C0 <1> xor ax, ax

3728 00008C24 66A3[F4110100] <1> mov [Attr\_Chars], ax

3729 00008C2A A2[B8630100] <1> mov [Attributes], al

3730 <1>

3731 <1> get\_attrib\_fchar:

3732 <1> ; esi = file name

3733 00008C2F 8A06 <1> mov al, [esi]

3734 00008C31 3C20 <1> cmp al, 20h

3735 00008C33 7623 <1> jna short loc\_attr\_file\_nofilename\_retn

3736 <1>

3737 <1> loc\_scan\_attrib\_params:

3738 00008C35 3C2D <1> cmp al, '-'

3739 00008C37 0F871C010000 <1> ja loc\_attr\_file\_parse\_path\_name

3740 00008C3D 7408 <1> je short loc\_attr\_space

3741 <1>

3742 00008C3F 3C2B <1> cmp al, '+'

3743 00008C41 0F850CF1FFFF <1> jne loc\_cmd\_failed

3744 <1>

3745 <1> loc\_attr\_space:

3746 00008C47 8A6601 <1> mov ah, [esi+1]

3747 00008C4A 80FC20 <1> cmp ah, 20h

3748 00008C4D 770A <1> ja short pass\_attr\_space

3749 00008C4F 0F82FEF0FFFF <1> jb loc\_cmd\_failed

3750 00008C55 46 <1> inc esi

3751 00008C56 EBEF <1> jmp short loc\_attr\_space

3752 <1>

3753 <1> loc\_attr\_file\_nofilename\_retn:

3754 00008C58 C3 <1> retn

3755 <1>

3756 <1> pass\_attr\_space:

3757 00008C59 80E4DF <1> and ah, 0DFh

3758 00008C5C 80FC53 <1> cmp ah, 'S'

3759 00008C5F 0F87EEF0FFFF <1> ja loc\_cmd\_failed

3760 00008C65 7204 <1> jb short pass\_attr\_system

3761 00008C67 B404 <1> mov ah, 04h ; System

3762 00008C69 EB21 <1> jmp short pass\_attr\_archive

3763 <1>

3764 <1> pass\_attr\_system:

3765 00008C6B 80FC48 <1> cmp ah, 'H'

3766 00008C6E 7706 <1> ja short pass\_attr\_hidden

3767 00008C70 7213 <1> jb short pass\_attr\_read\_only

3768 00008C72 B402 <1> mov ah, 02h ; Hidden

3769 00008C74 EB16 <1> jmp short pass\_attr\_archive

3770 <1>

3771 <1> pass\_attr\_hidden:

3772 00008C76 80FC52 <1> cmp ah, 'R'

3773 00008C79 0F87D4F0FFFF <1> ja loc\_cmd\_failed

3774 00008C7F 7204 <1> jb short pass\_attr\_read\_only ; Read only

3775 00008C81 B401 <1> mov ah, 01h

3776 00008C83 EB07 <1> jmp short pass\_attr\_archive

3777 <1>

3778 <1> pass\_attr\_read\_only:

3779 00008C85 80FC41 <1> cmp ah, 'A'

3780 00008C88 753B <1> jne short loc\_chk\_attr\_enter

3781 00008C8A B420 <1> mov ah, 20h ; Archive

3782 <1>

3783 <1> pass\_attr\_archive:

3784 00008C8C 3C2D <1> cmp al, '-'

3785 00008C8E 7508 <1> jne short pass\_reducing\_attributes

3786 00008C90 0825[F4110100] <1> or [Attr\_Chars], ah

3787 00008C96 EB06 <1> jmp short loc\_change\_attributes\_inc

3788 <1>

3789 <1> pass\_reducing\_attributes:

3790 00008C98 0825[F5110100] <1> or [Attr\_Chars+1], ah

3791 <1>

3792 <1> loc\_change\_attributes\_inc:

3793 00008C9E 46 <1> inc esi

3794 00008C9F 8A6601 <1> mov ah, [esi+1]

3795 00008CA2 80FC20 <1> cmp ah, 20h

3796 00008CA5 7227 <1> jb short pass\_change\_attr

3797 00008CA7 74F5 <1> je short loc\_change\_attributes\_inc

3798 00008CA9 80FC2D <1> cmp ah, '-'

3799 00008CAC 770D <1> ja short loc\_chk\_next\_attr\_char1

3800 00008CAE 7405 <1> je short loc\_chk\_next\_attr\_char0

3801 00008CB0 80FC2B <1> cmp ah, '+'

3802 00008CB3 7506 <1> jne short loc\_chk\_next\_attr\_char1

3803 <1>

3804 <1> loc\_chk\_next\_attr\_char0:

3805 00008CB5 46 <1> inc esi

3806 00008CB6 668B06 <1> mov ax, [esi]

3807 00008CB9 EB9E <1> jmp short pass\_attr\_space

3808 <1>

3809 <1> loc\_chk\_next\_attr\_char1:

3810 00008CBB 803E2D <1> cmp byte [esi], '-'

3811 00008CBE 7799 <1> ja short pass\_attr\_space

3812 00008CC0 E988000000 <1> jmp loc\_attr\_file\_check\_fname\_fchar

3813 <1>

3814 <1> loc\_chk\_attr\_enter:

3815 00008CC5 80FC0D <1> cmp ah, 0Dh

3816 00008CC8 0F8585F0FFFF <1> jne loc\_cmd\_failed

3817 <1>

3818 <1> pass\_change\_attr:

3819 00008CCE A0[F4110100] <1> mov al, [Attr\_Chars]

3820 00008CD3 F6D0 <1> not al

3821 00008CD5 2005[B8630100] <1> and [Attributes], al

3822 00008CDB A0[F5110100] <1> mov al, [Attr\_Chars+1]

3823 00008CE0 0805[B8630100] <1> or [Attributes], al

3824 <1>

3825 <1> loc\_show\_attributes:

3826 00008CE6 BE[6F190100] <1> mov esi, nextline

3827 00008CEB E86DD6FFFF <1> call print\_msg

3828 <1>

3829 <1> loc\_show\_attributes\_no\_nextline:

3830 00008CF0 C705[F4110100]4E4F- <1> mov dword [Attr\_Chars], 'NORM'

3830 00008CF8 524D <1>

3831 00008CFA 66C705[F8110100]41- <1> mov word [Attr\_Chars+4], 'AL'

3831 00008D02 4C <1>

3832 00008D03 BE[F4110100] <1> mov esi, Attr\_Chars

3833 00008D08 A0[B8630100] <1> mov al, [Attributes]

3834 00008D0D A804 <1> test al, 04h

3835 00008D0F 7406 <1> jz short pass\_put\_attr\_s

3836 00008D11 66C7065300 <1> mov word [esi], 0053h ; S

3837 00008D16 46 <1> inc esi

3838 <1>

3839 <1> pass\_put\_attr\_s:

3840 00008D17 A802 <1> test al, 02h

3841 00008D19 7406 <1> jz short pass\_put\_attr\_h

3842 00008D1B 66C7064800 <1> mov word [esi], 0048h ; H

3843 00008D20 46 <1> inc esi

3844 <1>

3845 <1> pass\_put\_attr\_h:

3846 00008D21 A801 <1> test al, 01h

3847 00008D23 7406 <1> jz short pass\_put\_attr\_r

3848 00008D25 66C7065200 <1> mov word [esi], 0052h ; R

3849 00008D2A 46 <1> inc esi

3850 <1>

3851 <1> pass\_put\_attr\_r:

3852 00008D2B 3C20 <1> cmp al, 20h

3853 00008D2D 7205 <1> jb short pass\_put\_attr\_a

3854 00008D2F 66C7064100 <1> mov word [esi], 0041h ; A

3855 <1>

3856 <1> pass\_put\_attr\_a:

3857 00008D34 BE[E7110100] <1> mov esi, Str\_Attributes

3858 00008D39 E81FD6FFFF <1> call print\_msg

3859 00008D3E BE[6F190100] <1> mov esi, nextline

3860 00008D43 E815D6FFFF <1> call print\_msg

3861 00008D48 E944F9FFFF <1> jmp loc\_file\_rw\_restore\_retn

3862 <1>

3863 <1> loc\_attr\_file\_check\_fname\_fchar:

3864 00008D4D 46 <1> inc esi

3865 00008D4E 803E20 <1> cmp byte [esi], 20h

3866 00008D51 74FA <1> je short loc\_attr\_file\_check\_fname\_fchar

3867 00008D53 0F8275FFFFFF <1> jb pass\_change\_attr

3868 <1>

3869 <1> loc\_attr\_file\_parse\_path\_name:

3870 00008D59 BF[A2620100] <1> mov edi, FindFile\_Drv

3871 00008D5E E8D1160000 <1> call parse\_path\_name

3872 00008D63 0F82EAEFFFFF <1> jc loc\_cmd\_failed

3873 <1>

3874 <1> loc\_attr\_file\_check\_filename\_exists:

3875 00008D69 BE[E4620100] <1> mov esi, FindFile\_Name

3876 00008D6E 803E20 <1> cmp byte [esi], 20h

3877 00008D71 0F86DCEFFFFF <1> jna loc\_cmd\_failed

3878 00008D77 8935[60630100] <1> mov [DelFile\_FNPointer], esi

3879 <1>

3880 <1> loc\_attr\_file\_drv:

3881 00008D7D 8A35[FE580100] <1> mov dh, [Current\_Drv]

3882 00008D83 8835[5E610100] <1> mov [RUN\_CDRV], dh

3883 <1>

3884 00008D89 8A15[A2620100] <1> mov dl, [FindFile\_Drv]

3885 00008D8F 38F2 <1> cmp dl, dh

3886 00008D91 740B <1> je short loc\_attr\_file\_change\_directory

3887 <1>

3888 00008D93 E828E1FFFF <1> call change\_current\_drive

3889 00008D98 0F82F3F8FFFF <1> jc loc\_file\_rw\_cmd\_failed

3890 <1>

3891 <1> loc\_attr\_file\_change\_directory:

3892 00008D9E 803D[A3620100]20 <1> cmp byte [FindFile\_Directory], 20h

3893 00008DA5 7618 <1> jna short loc\_attr\_file\_find

3894 <1>

3895 00008DA7 FE05[D30C0100] <1> inc byte [Restore\_CDIR]

3896 <1>

3897 00008DAD BE[A3620100] <1> mov esi, FindFile\_Directory

3898 00008DB2 30E4 <1> xor ah, ah ; CD\_COMMAND sign -> 0

3899 00008DB4 E865100000 <1> call change\_current\_directory

3900 00008DB9 0F82D2F8FFFF <1> jc loc\_file\_rw\_cmd\_failed

3901 <1>

3902 <1> ;loc\_attr\_file\_change\_prompt\_dir\_string:

3903 <1> ;call change\_prompt\_dir\_string

3904 <1>

3905 <1> loc\_attr\_file\_find:

3906 <1> ;mov esi, FindFile\_Name

3907 00008DBF 8B35[60630100] <1> mov esi, [DelFile\_FNPointer]

3908 00008DC5 66B80008 <1> mov ax, 0800h ; Except volume labels

3909 00008DC9 E895F4FFFF <1> call find\_first\_file

3910 00008DCE 0F82BDF8FFFF <1> jc loc\_file\_rw\_cmd\_failed

3911 <1>

3912 <1> loc\_attr\_file\_ambgfn\_check:

3913 00008DD4 6609D2 <1> or dx, dx ; Ambiguous filename chars used sign (DX>0)

3914 <1> ; (Note: It was BX in TRDOS v1)

3915 <1> ;jz short loc\_attr\_file\_found

3916 00008DD7 0F85B8FDFFFF <1> jnz loc\_file\_not\_found ; 06/03/2016

3917 <1>

3918 <1> ;mov eax, 2 ; File not found sign

3919 <1> ;stc

3920 <1> ;jmp loc\_file\_rw\_cmd\_failed

3921 <1>

3922 <1> loc\_attr\_file\_found:

3923 <1> ; EDI = Directory buffer entry offset/address

3924 <1> ; BL = File (or Directory) Attributes

3925 <1> ; (Note: It was 'CL' in TRDOS v1)

3926 <1> ; mov bl, [EDI+0Bh]

3927 <1>

3928 00008DDD 66833D[F4110100]00 <1> cmp word [Attr\_Chars], 0

3929 00008DE5 770B <1> ja short loc\_attr\_file\_change\_attributes

3930 00008DE7 881D[B8630100] <1> mov [Attributes], bl

3931 00008DED E9F4FEFFFF <1> jmp loc\_show\_attributes

3932 <1>

3933 <1> loc\_attr\_file\_change\_attributes:

3934 00008DF2 A0[F4110100] <1> mov al, [Attr\_Chars]

3935 00008DF7 F6D0 <1> not al

3936 00008DF9 20C3 <1> and bl, al

3937 00008DFB A0[F5110100] <1> mov al, [Attr\_Chars+1]

3938 00008E00 08C3 <1> or bl, al

3939 <1>

3940 00008E02 66817F0CA101 <1> cmp word [edi+DirEntry\_NTRes], 01A1h ; Singlix FS

3941 00008E08 741D <1> je short loc\_attr\_file\_fs\_check

3942 <1>

3943 00008E0A 881D[B8630100] <1> mov [Attributes], bl

3944 00008E10 885F0B <1> mov [edi+0Bh], bl ; Attributes (New!)

3945 <1>

3946 <1> ; 04/03/2016

3947 <1> ; TRDOS v1 has a bug here! it does not set

3948 <1> ; 'DirBuff\_ValidData' to 2; as result of this bug,

3949 <1> ; 'save\_directory\_buffer' would not save the new attributes !

3950 <1>

3951 00008E13 C605[28610100]02 <1> mov byte [DirBuff\_ValidData], 2

3952 <1>

3953 00008E1A E8791A0000 <1> call save\_directory\_buffer

3954 00008E1F 0F826CF8FFFF <1> jc loc\_file\_rw\_cmd\_failed

3955 <1>

3956 00008E25 EB33 <1> jmp short loc\_print\_attr\_changed\_message

3957 <1>

3958 <1> loc\_attr\_file\_fs\_check:

3959 00008E27 29C0 <1> sub eax, eax

3960 00008E29 8A25[26610100] <1> mov ah, [DirBuff\_DRV]

3961 00008E2F BE00010900 <1> mov esi, Logical\_DOSDisks

3962 00008E34 01C6 <1> add esi, eax

3963 00008E36 807E04A1 <1> cmp byte [esi+LD\_FSType], 0A1h

3964 00008E3A 7309 <1> jnc short loc\_attr\_file\_change\_fs\_file\_attributes

3965 <1> ; 29/12/2017 (0Dh -> 29)

3966 00008E3C 66B81D00 <1> mov ax, 29 ; Invalid Data

3967 00008E40 E94CF8FFFF <1> jmp loc\_file\_rw\_cmd\_failed

3968 <1>

3969 <1> loc\_attr\_file\_change\_fs\_file\_attributes:

3970 <1> ; BL = New MS-DOS File Attributes

3971 00008E45 88D8 <1> mov al, bl ; File/Directory Attributes

3972 00008E47 30E4 <1> xor ah, ah ; Attributes in MS-DOS format sign

3973 00008E49 E873050000 <1> call change\_fs\_file\_attributes

3974 00008E4E 0F823DF8FFFF <1> jc loc\_file\_rw\_cmd\_failed

3975 <1>

3976 00008E54 881D[B8630100] <1> mov [Attributes], bl

3977 <1>

3978 <1> loc\_print\_attr\_changed\_message:

3979 00008E5A BE[E2110100] <1> mov esi, Msg\_New

3980 00008E5F E8F9D4FFFF <1> call print\_msg

3981 00008E64 E987FEFFFF <1> jmp loc\_show\_attributes\_no\_nextline

3982 <1>

3983 <1> rename\_file:

3984 <1> ; 13/11/2017

3985 <1> ; 06/11/2016

3986 <1> ; 05/11/2016

3987 <1> ; 16/10/2016

3988 <1> ; 08/03/2016

3989 <1> ; 06/03/2016 (TRDOS 386 = TRDOS v2.0)

3990 <1> ; 20/11/2010 (TRDOS v1, CMD\_INTR.ASM, 'cmp\_cmd\_rename')

3991 <1> ; 16/11/2010

3992 <1>

3993 <1> get\_rename\_source\_fchar:

3994 <1> ; esi = file name

3995 00008E69 803E20 <1> cmp byte [esi], 20h

3996 00008E6C 7614 <1> jna short loc\_rename\_nofilename\_retn

3997 <1>

3998 00008E6E 8935[E0630100] <1> mov [SourceFilePath], esi

3999 <1>

4000 <1> rename\_scan\_source\_file:

4001 00008E74 46 <1> inc esi

4002 00008E75 803E20 <1> cmp byte [esi], 20h

4003 00008E78 7409 <1> je short rename\_scan\_destination\_file\_1

4004 <1> ;jb short loc\_rename\_nofilename\_retn

4005 00008E7A 0F82D3EEFFFF <1> jb loc\_cmd\_failed

4006 00008E80 EBF2 <1> jmp short rename\_scan\_source\_file

4007 <1>

4008 <1> loc\_rename\_nofilename\_retn: ; 08/03/2016

4009 00008E82 C3 <1> retn

4010 <1>

4011 <1> rename\_scan\_destination\_file\_1:

4012 00008E83 C60600 <1> mov byte [esi], 0

4013 <1>

4014 <1> rename\_scan\_destination\_file\_2:

4015 00008E86 46 <1> inc esi

4016 00008E87 803E20 <1> cmp byte [esi], 20h

4017 00008E8A 74FA <1> je short rename\_scan\_destination\_file\_2

4018 <1> ;jb short loc\_rename\_nofilename\_retn

4019 00008E8C 0F82C1EEFFFF <1> jb loc\_cmd\_failed

4020 <1>

4021 00008E92 8935[E4630100] <1> mov [DestinationFilePath], esi

4022 <1>

4023 <1> rename\_scan\_destination\_file\_3:

4024 00008E98 46 <1> inc esi

4025 00008E99 803E20 <1> cmp byte [esi], 20h

4026 00008E9C 77FA <1> ja short rename\_scan\_destination\_file\_3

4027 <1>

4028 00008E9E C60600 <1> mov byte [esi], 0

4029 <1>

4030 <1> loc\_rename\_save\_current\_drive:

4031 00008EA1 8A35[FE580100] <1> mov dh, [Current\_Drv]

4032 00008EA7 8835[5E610100] <1> mov byte [RUN\_CDRV], dh

4033 <1>

4034 <1> loc\_rename\_sf\_parse\_path\_name:

4035 00008EAD 8B35[E0630100] <1> mov esi, [SourceFilePath]

4036 00008EB3 BF[A2620100] <1> mov edi, FindFile\_Drv

4037 00008EB8 E877150000 <1> call parse\_path\_name

4038 00008EBD 0F8290EEFFFF <1> jc loc\_cmd\_failed

4039 <1>

4040 <1> loc\_rename\_sf\_check\_filename\_exists:

4041 00008EC3 BE[E4620100] <1> mov esi, FindFile\_Name

4042 00008EC8 803E20 <1> cmp byte [esi], 20h

4043 00008ECB 0F8682EEFFFF <1> jna loc\_cmd\_failed

4044 <1>

4045 <1> ;mov [DelFile\_FNPointer], esi

4046 <1>

4047 <1> loc\_rename\_sf\_drv:

4048 <1> ;mov dh, [Current\_Drv]

4049 <1> ;mov [RUN\_CDRV], dh

4050 <1>

4051 00008ED1 8A15[A2620100] <1> mov dl, [FindFile\_Drv]

4052 00008ED7 38F2 <1> cmp dl, dh ; dh = [Current\_Drv]

4053 00008ED9 740B <1> je short rename\_sf\_change\_directory

4054 <1>

4055 00008EDB E8E0DFFFFF <1> call change\_current\_drive

4056 00008EE0 0F82ABF7FFFF <1> jc loc\_file\_rw\_cmd\_failed

4057 <1>

4058 <1> rename\_sf\_change\_directory:

4059 00008EE6 803D[A3620100]20 <1> cmp byte [FindFile\_Directory], 20h

4060 00008EED 7618 <1> jna short rename\_sf\_find

4061 <1>

4062 00008EEF FE05[D30C0100] <1> inc byte [Restore\_CDIR]

4063 00008EF5 BE[A3620100] <1> mov esi, FindFile\_Directory

4064 00008EFA 30E4 <1> xor ah, ah ; CD\_COMMAND sign -> 0

4065 00008EFC E81D0F0000 <1> call change\_current\_directory

4066 00008F01 0F828AF7FFFF <1> jc loc\_file\_rw\_cmd\_failed

4067 <1>

4068 <1> ;rename\_sf\_change\_prompt\_dir\_string:

4069 <1> ;call change\_prompt\_dir\_string

4070 <1>

4071 <1> rename\_sf\_find:

4072 <1> ;mov esi, [DelFile\_FNPointer]

4073 00008F07 BE[E4620100] <1> mov esi, FindFile\_Name

4074 <1>

4075 00008F0C 66B80008 <1> mov ax, 0800h ; Except volume labels

4076 00008F10 E84EF3FFFF <1> call find\_first\_file

4077 00008F15 0F8276F7FFFF <1> jc loc\_file\_rw\_cmd\_failed

4078 <1>

4079 <1> loc\_rename\_sf\_ambgfn\_check:

4080 00008F1B 6621D2 <1> and dx, dx ; Ambiguous filename chars used sign (DX>0)

4081 <1> ; (Note: It was BX in TRDOS v1)

4082 <1> ;jz short loc\_rename\_sf\_found

4083 00008F1E 0F8571FCFFFF <1> jnz loc\_file\_not\_found

4084 <1>

4085 <1> ;mov eax, 2 ; File not found sign

4086 <1> ;stc

4087 <1> ;jmp loc\_file\_rw\_cmd\_failed

4088 <1>

4089 <1> loc\_rename\_sf\_found:

4090 <1> ; EDI = Directory buffer entry offset/address

4091 <1> ; BL = File (or Directory) Attributes

4092 <1> ; (Note: It was 'CL' in TRDOS v1)

4093 <1> ; mov bl, [EDI+0Bh]

4094 <1>

4095 00008F24 F6C307 <1> test bl, 07h ; Attributes, S-H-R

4096 00008F27 0F856EF7FFFF <1> jnz loc\_permission\_denied

4097 <1>

4098 00008F2D BE[A2620100] <1> mov esi, FindFile\_Drv

4099 00008F32 BF[E8630100] <1> mov edi, SourceFile\_Drv

4100 00008F37 B920000000 <1> mov ecx, 32

4101 00008F3C F3A5 <1> rep movsd

4102 <1>

4103 <1> loc\_rename\_df\_parse\_path\_name:

4104 00008F3E 8B35[E4630100] <1> mov esi, [DestinationFilePath]

4105 00008F44 BF[A2620100] <1> mov edi, FindFile\_Drv

4106 00008F49 E8E6140000 <1> call parse\_path\_name

4107 00008F4E 7219 <1> jc short loc\_rename\_df\_cmd\_failed

4108 <1>

4109 <1> ;mov dh, [RUN\_CDRV]

4110 00008F50 8A35[FE580100] <1> mov dh, [Current\_Drv]

4111 <1>

4112 <1> ; 'rename' command is valid only for same dos drive and same dir!

4113 <1> ; ('move' command must be used if source file and destination file

4114 <1> ; directories are not same!)

4115 00008F56 8A15[A2620100] <1> mov dl, [FindFile\_Drv]

4116 00008F5C 38F2 <1> cmp dl, dh ; are source and destination drives different ?!

4117 00008F5E 7509 <1> jne short loc\_rename\_df\_cmd\_failed ; yes!

4118 <1>

4119 <1> rename\_df\_check\_dirname\_exists:

4120 00008F60 803D[A3620100]00 <1> cmp byte [FindFile\_Directory], 0

4121 00008F67 760B <1> jna short rename\_df\_check\_filename\_exists

4122 <1>

4123 <1> ; different source file and destination file directories !

4124 <1> loc\_rename\_df\_cmd\_failed:

4125 00008F69 B801000000 <1> mov eax, 1 ; TRDOS 'Bad command or file name' error

4126 00008F6E F9 <1> stc

4127 00008F6F E91DF7FFFF <1> jmp loc\_file\_rw\_cmd\_failed

4128 <1>

4129 <1> rename\_df\_check\_filename\_exists:

4130 00008F74 BE[E4620100] <1> mov esi, FindFile\_Name

4131 00008F79 E8A3F6FFFF <1> call check\_filename

4132 00008F7E 0F82BFF7FFFF <1> jc loc\_mkdir\_invalid\_dir\_name\_chars

4133 <1>

4134 <1> ;mov [DelFile\_FNPointer], esi

4135 <1> ;cmp byte [esi], 20h

4136 <1> ;ja short loc\_rename\_df\_find

4137 <1>

4138 <1> ;mov dh, [Current\_Drv] ; dh has not been changed

4139 <1>

4140 <1> rename\_df\_drv\_check\_writable:

4141 00008F84 0FB6F6 <1> movzx esi, dh

4142 <1> ;movzx esi, byte [Current\_Drv]

4143 00008F87 81C600010900 <1> add esi, Logical\_DOSDisks

4144 <1>

4145 00008F8D 88F2 <1> mov dl, dh ; dl = [Current\_Drv]

4146 00008F8F 8A7601 <1> mov dh, [esi+LD\_DiskType]

4147 <1>

4148 00008F92 80FE01 <1> cmp dh, 1 ; 0 = Invalid

4149 00008F95 7310 <1> jnb short rename\_df\_compare\_sf\_df\_name

4150 <1>

4151 <1> ; 16/10/2016 (13h -> 30)

4152 00008F97 B81E000000 <1> mov eax, 30 ; 'Disk write-protected' error

4153 00008F9C 8B1D[E4630100] <1> mov ebx, [DestinationFilePath]

4154 00008FA2 E9EAF6FFFF <1> jmp loc\_file\_rw\_cmd\_failed

4155 <1>

4156 <1> rename\_df\_compare\_sf\_df\_name:

4157 00008FA7 BE[E4620100] <1> mov esi, FindFile\_Name

4158 00008FAC BF[2A640100] <1> mov edi, SourceFile\_Name

4159 00008FB1 B90C000000 <1> mov ecx, 12

4160 <1> rename\_df\_compare\_sf\_df\_name\_next:

4161 00008FB6 AC <1> lodsb

4162 00008FB7 AE <1> scasb

4163 00008FB8 7506 <1> jne short loc\_rename\_df\_find

4164 00008FBA 08C0 <1> or al, al

4165 00008FBC 74AB <1> jz short loc\_rename\_df\_cmd\_failed

4166 00008FBE E2F6 <1> loop rename\_df\_compare\_sf\_df\_name\_next

4167 <1>

4168 <1> loc\_rename\_df\_find:

4169 <1> ;mov esi, [DelFile\_FNPointer]

4170 00008FC0 BE[E4620100] <1> mov esi, FindFile\_Name

4171 <1>

4172 00008FC5 6631C0 <1> xor ax, ax ; Any

4173 00008FC8 E896F2FFFF <1> call find\_first\_file

4174 <1> ;jnc short loc\_rename\_df\_found

4175 <1> ; 29/12/2017

4176 00008FCD 0F83C8F6FFFF <1> jnc loc\_permission\_denied

4177 <1>

4178 <1> loc\_rename\_df\_check\_error\_code:

4179 <1> ;cmp eax, 2

4180 00008FD3 3C02 <1> cmp al, 2 ; Not found error

4181 00008FD5 7406 <1> je short rename\_df\_move\_find\_struct\_to\_dest

4182 00008FD7 F9 <1> stc

4183 00008FD8 E9B4F6FFFF <1> jmp loc\_file\_rw\_cmd\_failed

4184 <1>

4185 <1> ;loc\_rename\_df\_found:

4186 <1> ; 05/11/2016

4187 <1> ; Permission denied error

4188 <1> ;mov eax, ERR\_PERM\_DENIED ; 29/12/2017

4189 <1> ;stc

4190 <1> ;jmp loc\_permission\_denied ; 06/11/2016

4191 <1>

4192 <1> rename\_df\_move\_find\_struct\_to\_dest:

4193 00008FDD BE[A2620100] <1> mov esi, FindFile\_Drv

4194 00008FE2 BF[68640100] <1> mov edi, DestinationFile\_Drv

4195 00008FE7 B920000000 <1> mov ecx, 32

4196 00008FEC F3A5 <1> rep movsd

4197 <1>

4198 <1> loc\_rename\_df\_process\_q\_sf:

4199 <1> ;mov ecx, 12

4200 00008FEE B10C <1> mov cl, 12

4201 00008FF0 BE[2A640100] <1> mov esi, SourceFile\_Name

4202 00008FF5 BF[23120100] <1> mov edi, Rename\_OldName

4203 <1> rename\_df\_process\_q\_nml\_1\_sf:

4204 00008FFA AC <1> lodsb

4205 00008FFB 3C20 <1> cmp al, 20h

4206 00008FFD 7603 <1> jna short rename\_df\_process\_q\_nml\_2\_sf

4207 00008FFF AA <1> stosb

4208 00009000 E2F8 <1> loop rename\_df\_process\_q\_nml\_1\_sf

4209 <1>

4210 <1> rename\_df\_process\_q\_nml\_2\_sf:

4211 00009002 C60700 <1> mov byte [edi], 0

4212 <1>

4213 <1> loc\_rename\_df\_process\_q\_df:

4214 <1> ;mov ecx, 12

4215 00009005 B10C <1> mov cl, 12

4216 00009007 BE[AA640100] <1> mov esi, DestinationFile\_Name

4217 0000900C BF[34120100] <1> mov edi, Rename\_NewName

4218 <1> rename\_df\_process\_q\_nml\_1\_df:

4219 00009011 AC <1> lodsb

4220 00009012 3C20 <1> cmp al, 20h

4221 00009014 7603 <1> jna short loc\_rename\_df\_process\_q\_nml\_2\_df

4222 00009016 AA <1> stosb

4223 00009017 E2F8 <1> loop rename\_df\_process\_q\_nml\_1\_df

4224 <1>

4225 <1> loc\_rename\_df\_process\_q\_nml\_2\_df:

4226 00009019 C60700 <1> mov byte [edi], 0

4227 <1>

4228 <1> loc\_rename\_confirmation\_question:

4229 0000901C BE[FB110100] <1> mov esi, Msg\_DoYouWantRename

4230 00009021 E837D3FFFF <1> call print\_msg

4231 <1>

4232 00009026 A0[45640100] <1> mov al, [SourceFile\_DirEntry+11] ; Attributes

4233 0000902B 2410 <1> and al, 10h

4234 0000902D 750C <1> jnz short rename\_confirmation\_question\_dir

4235 <1>

4236 <1> rename\_confirmation\_question\_file:

4237 0000902F BE[12120100] <1> mov esi, Rename\_File

4238 00009034 E824D3FFFF <1> call print\_msg

4239 00009039 EB0A <1> jmp short rename\_confirmation\_question\_as

4240 <1>

4241 <1> rename\_confirmation\_question\_dir:

4242 0000903B BE[18120100] <1> mov esi, Rename\_Directory

4243 00009040 E818D3FFFF <1> call print\_msg

4244 <1>

4245 <1> rename\_confirmation\_question\_as:

4246 00009045 BE[23120100] <1> mov esi, Rename\_OldName

4247 0000904A E80ED3FFFF <1> call print\_msg

4248 0000904F BE[30120100] <1> mov esi, Msg\_File\_rename\_as

4249 00009054 E804D3FFFF <1> call print\_msg

4250 00009059 BE[57110100] <1> mov esi, Msg\_YesNo

4251 0000905E E8FAD2FFFF <1> call print\_msg

4252 <1>

4253 <1> loc\_rename\_ask\_again:

4254 00009063 30E4 <1> xor ah, ah

4255 00009065 E8AC7BFFFF <1> call int16h

4256 0000906A 3C1B <1> cmp al, 1Bh

4257 0000906C 740F <1> je short loc\_do\_not\_rename\_file

4258 0000906E 24DF <1> and al, 0DFh

4259 00009070 A2[61110100] <1> mov [Y\_N\_nextline], al

4260 00009075 3C59 <1> cmp al, 'Y'

4261 00009077 7404 <1> je short loc\_yes\_rename\_file

4262 00009079 3C4E <1> cmp al, 'N'

4263 0000907B 75E6 <1> jne short loc\_rename\_ask\_again

4264 <1>

4265 <1> loc\_do\_not\_rename\_file:

4266 <1> loc\_yes\_rename\_file:

4267 0000907D E848F7FFFF <1> call y\_n\_answer ; 29/12/2017

4268 <1> ;cmp al, 'Y' ; 'yes'

4269 <1> ;cmc

4270 <1> ;jnc loc\_file\_rw\_restore\_retn

4271 00009082 3C4E <1> cmp al, 'N' ; 'no'

4272 00009084 0F8407F6FFFF <1> je loc\_file\_rw\_restore\_retn

4273 <1>

4274 0000908A BE[34120100] <1> mov esi, Rename\_NewName

4275 0000908F 668B0D[62640100] <1> mov cx, [SourceFile\_DirEntryNumber]

4276 00009096 66A1[4E640100] <1> mov ax, [SourceFile\_DirEntry+20] ; First Cluster, HW

4277 0000909C C1E010 <1> shl eax, 16 ; 13/11/2017

4278 0000909F 66A1[54640100] <1> mov ax, [SourceFile\_DirEntry+26] ; First Cluster, LW

4279 <1>

4280 000090A5 0FB61D[37640100] <1> movzx ebx, byte [SourceFile\_LongNameEntryLength]

4281 000090AC E8CB1B0000 <1> call rename\_directory\_entry

4282 000090B1 E9FBF6FFFF <1> jmp loc\_rename\_file\_ok

4283 <1> ;loc\_rename\_file\_ok:

4284 <1> ; jc loc\_run\_cmd\_failed

4285 <1> ; mov esi, Msg\_OK

4286 <1> ; call proc\_printmsg

4287 <1> ; jmp loc\_file\_rw\_restore\_retn

4288 <1>

4289 <1> move\_file:

4290 <1> ; 11/03/2016

4291 <1> ; 09/03/2016

4292 <1> ; 08/03/2016 (TRDOS 386 = TRDOS v2.0)

4293 <1> ; 21/05/2011 (TRDOS v1, CMD\_INTR.ASM, 'cmp\_cmd\_move')

4294 <1> ; 23/04/2011

4295 <1>

4296 <1> get\_move\_source\_fchar:

4297 <1> ; esi = file name

4298 000090B6 803E20 <1> cmp byte [esi], 20h

4299 000090B9 7614 <1> jna short loc\_move\_nofilename\_retn

4300 <1>

4301 000090BB 8935[E0630100] <1> mov [SourceFilePath], esi

4302 <1>

4303 <1> move\_scan\_source\_file:

4304 000090C1 46 <1> inc esi

4305 000090C2 803E20 <1> cmp byte [esi], 20h

4306 000090C5 7409 <1> je short move\_scan\_destination\_1

4307 <1> ;jb short loc\_move\_nofilename\_retn

4308 000090C7 0F8286ECFFFF <1> jb loc\_cmd\_failed

4309 000090CD EBF2 <1> jmp short move\_scan\_source\_file

4310 <1>

4311 <1> loc\_move\_nofilename\_retn:

4312 000090CF C3 <1> retn

4313 <1>

4314 <1> move\_scan\_destination\_1:

4315 000090D0 C60600 <1> mov byte [esi], 0

4316 <1>

4317 <1> move\_scan\_destination\_2:

4318 000090D3 46 <1> inc esi

4319 000090D4 803E20 <1> cmp byte [esi], 20h

4320 000090D7 74FA <1> je short move\_scan\_destination\_2

4321 <1> ;jb short loc\_move\_nofilename\_retn

4322 000090D9 0F8274ECFFFF <1> jb loc\_cmd\_failed

4323 <1>

4324 000090DF 8935[E4630100] <1> mov [DestinationFilePath], esi

4325 <1>

4326 <1> move\_scan\_destination\_3:

4327 000090E5 46 <1> inc esi

4328 000090E6 803E20 <1> cmp byte [esi], 20h

4329 000090E9 77FA <1> ja short move\_scan\_destination\_3

4330 000090EB C60600 <1> mov byte [esi], 0

4331 <1>

4332 <1> loc\_move\_scan\_destination\_OK:

4333 000090EE 8B35[E0630100] <1> mov esi, [SourceFilePath]

4334 000090F4 8B3D[E4630100] <1> mov edi, [DestinationFilePath]

4335 <1>

4336 000090FA B001 <1> mov al, 1 ; move procedure Phase 1

4337 000090FC E8F71B0000 <1> call move\_source\_file\_to\_destination\_file

4338 00009101 7328 <1> jnc short move\_source\_file\_to\_destination\_question

4339 <1>

4340 <1> loc\_move\_cmd\_failed\_1:

4341 00009103 08C0 <1> or al, al

4342 00009105 0F8448ECFFFF <1> jz loc\_cmd\_failed

4343 0000910B 3C11 <1> cmp al, 11h

4344 0000910D 740D <1> je short loc\_msg\_not\_same\_device

4345 <1> ;cmp al, 05h

4346 <1> ;cmp al, ERR\_PERM\_DENIED ; 29/12/2017

4347 <1> ;jne loc\_run\_cmd\_failed

4348 <1> ;jmp loc\_permission\_denied

4349 0000910F 3C0B <1> cmp al, ERR\_PERM\_DENIED

4350 00009111 0F8484F5FFFF <1> je loc\_permission\_denied

4351 00009117 E962ECFFFF <1> jmp loc\_run\_cmd\_failed

4352 <1>

4353 <1> ;mov esi, Msg\_Permission\_denied

4354 <1> ;call print\_msg

4355 <1> ;jmp loc\_file\_rw\_restore\_retn

4356 <1>

4357 <1> loc\_msg\_not\_same\_device:

4358 0000911C BE[41120100] <1> mov esi, msg\_not\_same\_drv

4359 00009121 E837D2FFFF <1> call print\_msg

4360 00009126 E966F5FFFF <1> jmp loc\_file\_rw\_restore\_retn

4361 <1>

4362 <1> move\_source\_file\_to\_destination\_question:

4363 0000912B A0[E8630100] <1> mov al, [SourceFile\_Drv]

4364 00009130 0441 <1> add al, 'A'

4365 00009132 A2[A3120100] <1> mov [msg\_source\_file\_drv], al

4366 00009137 A0[68640100] <1> mov al, [DestinationFile\_Drv]

4367 0000913C 0441 <1> add al, 'A'

4368 0000913E A2[C2120100] <1> mov [msg\_destination\_file\_drv], al

4369 <1>

4370 00009143 57 <1> push edi ; \*

4371 <1>

4372 00009144 BE[87120100] <1> mov esi, msg\_source\_file

4373 00009149 E80FD2FFFF <1> call print\_msg

4374 0000914E BE[E9630100] <1> mov esi, SourceFile\_Directory

4375 00009153 803E20 <1> cmp byte [esi], 20h

4376 00009156 7605 <1> jna short msftdfq\_sfn

4377 00009158 E800D2FFFF <1> call print\_msg

4378 <1> msftdfq\_sfn:

4379 0000915D BE[2A640100] <1> mov esi, SourceFile\_Name

4380 00009162 E8F6D1FFFF <1> call print\_msg

4381 00009167 BE[A6120100] <1> mov esi, msg\_destination\_file

4382 0000916C E8ECD1FFFF <1> call print\_msg

4383 00009171 BE[69640100] <1> mov esi, DestinationFile\_Directory

4384 00009176 803E20 <1> cmp byte [esi], 20h

4385 00009179 7605 <1> jna short msftdfq\_dfn

4386 0000917B E8DDD1FFFF <1> call print\_msg

4387 <1> msftdfq\_dfn:

4388 00009180 BE[AA640100] <1> mov esi, DestinationFile\_Name

4389 00009185 E8D3D1FFFF <1> call print\_msg

4390 0000918A BE[C5120100] <1> mov esi, msg\_copy\_nextline

4391 0000918F E8C9D1FFFF <1> call print\_msg

4392 00009194 BE[C5120100] <1> mov esi, msg\_copy\_nextline

4393 00009199 E8BFD1FFFF <1> call print\_msg

4394 <1>

4395 <1> loc\_move\_ask\_for\_new\_file\_yes\_no:

4396 0000919E BE[53120100] <1> mov esi, Msg\_DoYouWantMoveFile

4397 000091A3 E8B5D1FFFF <1> call print\_msg

4398 000091A8 BE[57110100] <1> mov esi, Msg\_YesNo

4399 000091AD E8ABD1FFFF <1> call print\_msg

4400 <1> loc\_move\_ask\_for\_new\_file\_again:

4401 000091B2 30E4 <1> xor ah, ah

4402 000091B4 E85D7AFFFF <1> call int16h

4403 000091B9 3C1B <1> cmp al, 1Bh

4404 <1> ;je short loc\_do\_not\_move\_file

4405 000091BB 7441 <1> je short loc\_move\_y\_n\_escape

4406 000091BD 24DF <1> and al, 0DFh

4407 000091BF A2[61110100] <1> mov [Y\_N\_nextline], al

4408 000091C4 3C59 <1> cmp al, 'Y'

4409 000091C6 7404 <1> je short loc\_yes\_move\_file

4410 000091C8 3C4E <1> cmp al, 'N'

4411 000091CA 75E6 <1> jne short loc\_move\_ask\_for\_new\_file\_again

4412 <1>

4413 <1> loc\_do\_not\_move\_file:

4414 <1> loc\_yes\_move\_file:

4415 000091CC E8F9F5FFFF <1> call y\_n\_answer ; 29/12/2017

4416 000091D1 5F <1> pop edi ; \*

4417 <1> ;cmp al, 'Y' ; 'yes'

4418 <1> ;cmc

4419 <1> ;jnc loc\_file\_rw\_restore\_retn

4420 000091D2 3C4E <1> cmp al, 'N' ; 'no'

4421 000091D4 0F84B7F4FFFF <1> je loc\_file\_rw\_restore\_retn

4422 <1>

4423 <1> loc\_move\_yes\_move\_file:

4424 000091DA B002 <1> mov al, 2 ; move procedure Phase 2

4425 000091DC E8171B0000 <1> call move\_source\_file\_to\_destination\_file

4426 <1> ;jc short loc\_move\_cmd\_failed\_2

4427 000091E1 0F83D0F5FFFF <1> jnc move\_source\_file\_to\_destination\_OK

4428 <1>

4429 <1> ;move\_source\_file\_to\_destination\_OK:

4430 <1> ; mov esi, Msg\_OK

4431 <1> ; call print\_msg

4432 <1> ; jmp loc\_file\_rw\_restore\_retn

4433 <1>

4434 <1> loc\_move\_cmd\_failed\_2:

4435 000091E7 3C27 <1> cmp al, 27h

4436 000091E9 0F858FEBFFFF <1> jne loc\_run\_cmd\_failed

4437 <1>

4438 000091EF BE[6C120100] <1> mov esi, msg\_insufficient\_disk\_space

4439 000091F4 E864D1FFFF <1> call print\_msg

4440 <1>

4441 000091F9 E993F4FFFF <1> jmp loc\_file\_rw\_restore\_retn

4442 <1>

4443 <1> loc\_move\_y\_n\_escape:

4444 000091FE B04E <1> mov al, 'N' ; 'no'

4445 00009200 EBCA <1> jmp short loc\_do\_not\_move\_file

4446 <1>

4447 <1> copy\_file:

4448 <1> ; 15/10/2016

4449 <1> ; 24/03/2016

4450 <1> ; 21/03/2016

4451 <1> ; 15/03/2016 (TRDOS 386 = TRDOS v2.0)

4452 <1> ; 21/05/2011 (TRDOS v1, CMD\_INTR.ASM, 'cmp\_cmd\_copy')

4453 <1> ; 01/08/2010

4454 <1>

4455 <1> get\_copy\_source\_fchar:

4456 <1> ; esi = file name

4457 00009202 803E20 <1> cmp byte [esi], 20h

4458 00009205 7614 <1> jna short loc\_copy\_nofilename\_retn

4459 <1>

4460 00009207 8935[E0630100] <1> mov [SourceFilePath], esi

4461 <1>

4462 <1> copy\_scan\_source\_file:

4463 0000920D 46 <1> inc esi

4464 0000920E 803E20 <1> cmp byte [esi], 20h

4465 00009211 7409 <1> je short copy\_scan\_destination\_1

4466 <1> ;jb short loc\_copy\_nofilename\_retn

4467 00009213 0F823AEBFFFF <1> jb loc\_cmd\_failed

4468 00009219 EBF2 <1> jmp short copy\_scan\_source\_file

4469 <1>

4470 <1> loc\_copy\_nofilename\_retn:

4471 0000921B C3 <1> retn

4472 <1>

4473 <1> copy\_scan\_destination\_1:

4474 0000921C C60600 <1> mov byte [esi], 0

4475 <1>

4476 <1> copy\_scan\_destination\_2:

4477 0000921F 46 <1> inc esi

4478 00009220 803E20 <1> cmp byte [esi], 20h

4479 00009223 74FA <1> je short copy\_scan\_destination\_2

4480 <1> ;jb short loc\_copy\_nofilename\_retn

4481 00009225 0F8228EBFFFF <1> jb loc\_cmd\_failed

4482 <1>

4483 0000922B 8935[E4630100] <1> mov [DestinationFilePath], esi

4484 <1>

4485 <1> copy\_scan\_destination\_3:

4486 00009231 46 <1> inc esi

4487 00009232 803E20 <1> cmp byte [esi], 20h

4488 00009235 77FA <1> ja short copy\_scan\_destination\_3

4489 00009237 C60600 <1> mov byte [esi], 0

4490 <1>

4491 <1> loc\_copy\_save\_current\_drive:

4492 0000923A 8A35[FE580100] <1> mov dh, [Current\_Drv]

4493 00009240 8835[5E610100] <1> mov [RUN\_CDRV], dh

4494 <1>

4495 <1> copy\_source\_file\_to\_destination\_phase\_1:

4496 00009246 8B35[E0630100] <1> mov esi, [SourceFilePath]

4497 0000924C 8B3D[E4630100] <1> mov edi, [DestinationFilePath]

4498 <1>

4499 00009252 B001 <1> mov al, 1 ; copy procedure Phase 1

4500 00009254 E83C1D0000 <1> call copy\_source\_file\_to\_destination\_file

4501 00009259 732B <1> jnc short copy\_source\_file\_to\_destination\_question

4502 <1>

4503 <1> loc\_copy\_cmd\_failed\_1:

4504 <1> ; 18/03/2016 (restore current drive and directory)

4505 0000925B 08C0 <1> or al, al

4506 0000925D 7507 <1> jnz short loc\_copy\_cmd\_failed\_2

4507 <1>

4508 0000925F FEC0 <1> inc al ; mov al, 1 ; Bad command or file name !

4509 00009261 E918EBFFFF <1> jmp loc\_run\_cmd\_failed

4510 <1>

4511 <1> loc\_copy\_cmd\_failed\_2:

4512 00009266 3C27 <1> cmp al, 27h ; Insufficient disk space

4513 00009268 740D <1> je short loc\_file\_write\_insuff\_disk\_space\_msg

4514 <1>

4515 <1> ; 29/12/2017

4516 <1> ;cmp al, 05h

4517 0000926A 3C0B <1> cmp al, ERR\_PERM\_DENIED

4518 0000926C 0F850CEBFFFF <1> jne loc\_run\_cmd\_failed

4519 <1>

4520 00009272 E924F4FFFF <1> jmp loc\_permission\_denied

4521 <1>

4522 <1> loc\_file\_write\_insuff\_disk\_space\_msg:

4523 00009277 BE[6C120100] <1> mov esi, msg\_insufficient\_disk\_space

4524 0000927C E8DCD0FFFF <1> call print\_msg

4525 00009281 E90BF4FFFF <1> jmp loc\_file\_rw\_restore\_retn

4526 <1>

4527 <1> copy\_source\_file\_to\_destination\_question:

4528 00009286 57 <1> push edi ; \*

4529 <1>

4530 <1> ; dh = source file attributes

4531 <1> ; dl > 0 -> destination file found

4532 00009287 20D2 <1> and dl, dl

4533 00009289 7449 <1> jz short copy\_source\_file\_to\_destination\_pass\_owrq

4534 <1>

4535 <1> loc\_copy\_ask\_for\_owr\_yes\_no:

4536 0000928B BE[C8120100] <1> mov esi, Msg\_DoYouWantOverWriteFile

4537 00009290 E8C8D0FFFF <1> call print\_msg

4538 00009295 BE[AA640100] <1> mov esi, DestinationFile\_Name

4539 0000929A E8BED0FFFF <1> call print\_msg

4540 0000929F BE[57110100] <1> mov esi, Msg\_YesNo

4541 000092A4 E8B4D0FFFF <1> call print\_msg

4542 <1>

4543 <1> loc\_copy\_ask\_for\_owr\_again:

4544 000092A9 30E4 <1> xor ah, ah

4545 000092AB E86679FFFF <1> call int16h

4546 000092B0 3C1B <1> cmp al, 1Bh

4547 <1> ;je loc\_do\_not\_copy\_file

4548 000092B2 7419 <1> je short loc\_copy\_y\_n\_escape

4549 000092B4 24DF <1> and al, 0DFh

4550 000092B6 A2[61110100] <1> mov [Y\_N\_nextline], al

4551 000092BB 3C59 <1> cmp al, 'Y'

4552 000092BD 0F84B1000000 <1> je loc\_yes\_copy\_file

4553 000092C3 3C4E <1> cmp al, 'N'

4554 000092C5 0F84A9000000 <1> je loc\_do\_not\_copy\_file

4555 000092CB EBDC <1> jmp short loc\_copy\_ask\_for\_owr\_again

4556 <1>

4557 <1> loc\_copy\_y\_n\_escape:

4558 000092CD B04E <1> mov al, 'N' ; 'no'

4559 000092CF E9A0000000 <1> jmp loc\_do\_not\_copy\_file

4560 <1>

4561 <1> copy\_source\_file\_to\_destination\_pass\_owrq:

4562 000092D4 A0[E8630100] <1> mov al, [SourceFile\_Drv]

4563 000092D9 0441 <1> add al, 'A'

4564 000092DB A2[A3120100] <1> mov [msg\_source\_file\_drv], al

4565 000092E0 A0[68640100] <1> mov al, [DestinationFile\_Drv]

4566 000092E5 0441 <1> add al, 'A'

4567 000092E7 A2[C2120100] <1> mov [msg\_destination\_file\_drv], al

4568 <1>

4569 000092EC BE[87120100] <1> mov esi, msg\_source\_file

4570 000092F1 E867D0FFFF <1> call print\_msg

4571 000092F6 BE[E9630100] <1> mov esi, SourceFile\_Directory

4572 000092FB 803E20 <1> cmp byte [esi], 20h

4573 000092FE 7605 <1> jna short csftdfq\_sfn

4574 00009300 E858D0FFFF <1> call print\_msg

4575 <1> csftdfq\_sfn:

4576 00009305 BE[2A640100] <1> mov esi, SourceFile\_Name

4577 0000930A E84ED0FFFF <1> call print\_msg

4578 0000930F BE[A6120100] <1> mov esi, msg\_destination\_file

4579 00009314 E844D0FFFF <1> call print\_msg

4580 00009319 BE[69640100] <1> mov esi, DestinationFile\_Directory

4581 0000931E 803E20 <1> cmp byte [esi], 20h

4582 00009321 7605 <1> jna short csftdfq\_dfn

4583 00009323 E835D0FFFF <1> call print\_msg

4584 <1> csftdfq\_dfn:

4585 00009328 BE[AA640100] <1> mov esi, DestinationFile\_Name

4586 0000932D E82BD0FFFF <1> call print\_msg

4587 00009332 BE[C5120100] <1> mov esi, msg\_copy\_nextline

4588 00009337 E821D0FFFF <1> call print\_msg

4589 0000933C BE[C5120100] <1> mov esi, msg\_copy\_nextline

4590 00009341 E817D0FFFF <1> call print\_msg

4591 <1>

4592 <1> loc\_copy\_ask\_for\_new\_file\_yes\_no:

4593 00009346 BE[E7120100] <1> mov esi, Msg\_DoYouWantCopyFile

4594 0000934B E80DD0FFFF <1> call print\_msg

4595 00009350 BE[57110100] <1> mov esi, Msg\_YesNo

4596 00009355 E803D0FFFF <1> call print\_msg

4597 <1>

4598 <1> loc\_copy\_ask\_for\_new\_file\_again:

4599 0000935A 30E4 <1> xor ah, ah

4600 0000935C E8B578FFFF <1> call int16h

4601 00009361 3C1B <1> cmp al, 1Bh

4602 00009363 740F <1> je short loc\_do\_not\_copy\_file

4603 00009365 24DF <1> and al, 0DFh

4604 00009367 A2[61110100] <1> mov [Y\_N\_nextline], al

4605 0000936C 3C59 <1> cmp al, 'Y'

4606 0000936E 7404 <1> je short loc\_yes\_copy\_file

4607 00009370 3C4E <1> cmp al, 'N'

4608 00009372 75E6 <1> jne short loc\_copy\_ask\_for\_new\_file\_again

4609 <1>

4610 <1> loc\_do\_not\_copy\_file:

4611 <1> loc\_yes\_copy\_file:

4612 00009374 E851F4FFFF <1> call y\_n\_answer ; 29/12/2017

4613 00009379 5F <1> pop edi ; \*

4614 <1> ;cmp al, 'Y' ; 'yes'

4615 <1> ;cmc

4616 <1> ;jnc loc\_file\_rw\_restore\_retn

4617 0000937A 3C4E <1> cmp al, 'N' ; 'no'

4618 0000937C 0F840FF3FFFF <1> je loc\_file\_rw\_restore\_retn

4619 <1>

4620 <1> copy\_source\_file\_to\_destination\_pass\_q:

4621 00009382 B002 <1> mov al, 2 ; copy procedure Phase 2

4622 00009384 E80C1C0000 <1> call copy\_source\_file\_to\_destination\_file

4623 <1> ;jc short loc\_file\_write\_check\_disk\_space\_err

4624 <1>

4625 <1> ; 24/03/2016

4626 <1> ;push cx

4627 00009389 51 <1> push ecx ; 29/12/2017

4628 0000938A BE[C5120100] <1> mov esi, msg\_copy\_nextline

4629 0000938F E8C9CFFFFF <1> call print\_msg

4630 00009394 58 <1> pop eax ; 29/12/2017

4631 <1> ;;pop cx

4632 <1> ;pop ax

4633 <1>

4634 <1> ;or cl, cl

4635 00009395 08C0 <1> or al, al

4636 00009397 7419 <1> jz short copy\_source\_file\_to\_destination\_OK

4637 <1>

4638 <1> ; 15/10/2016 (1Dh -> 18)

4639 <1> ; 18/03/2016 (1Dh)

4640 <1> ;cmp cl, 18 ; write error

4641 00009399 3C12 <1> cmp al, 18

4642 0000939B 7506 <1> jne short copy\_source\_file\_to\_destination\_not\_OK

4643 <1> ;

4644 <1> ;mov al, cl ; error number (write fault!)

4645 0000939D F9 <1> stc

4646 0000939E E9EEF2FFFF <1> jmp loc\_file\_rw\_cmd\_failed

4647 <1>

4648 <1> copy\_source\_file\_to\_destination\_not\_OK:

4649 000093A3 BE[00130100] <1> mov esi, Msg\_read\_file\_error\_before\_EOF

4650 000093A8 E8B0CFFFFF <1> call print\_msg

4651 000093AD E9DFF2FFFF <1> jmp loc\_file\_rw\_restore\_retn

4652 <1>

4653 <1> copy\_source\_file\_to\_destination\_OK:

4654 000093B2 BE[65110100] <1> mov esi, Msg\_OK

4655 000093B7 E8A1CFFFFF <1> call print\_msg

4656 <1>

4657 000093BC E9D0F2FFFF <1> jmp loc\_file\_rw\_restore\_retn

4658 <1>

4659 <1> ;loc\_file\_write\_check\_disk\_space\_err:

4660 <1> ;cmp al, 27h ; Insufficient disk space

4661 <1> ;je loc\_file\_write\_insuff\_disk\_space\_msg

4662 <1> ;jb loc\_file\_rw\_cmd\_failed

4663 <1>

4664 <1> ;call print\_misc\_error\_msg ; 15/03/2016

4665 <1> ;jmp loc\_file\_rw\_restore\_retn

4666 <1>

4667 <1> change\_fs\_file\_attributes:

4668 <1> ; 04/03/2016 ; Temporary

4669 <1> ; AL = File or directory attributes

4670 <1> ; AH = 0 -> Attributes are in MS-DOS format

4671 <1> ; AH > 0 -> Attributes are in SINGLIX format

4672 <1> ;push ebx

4673 <1> ; ... do somethings here ...

4674 <1> ;pop ebx

4675 <1> ; BL = File or directory attributes

4676 000093C1 C3 <1> retn

4677 <1>

4678 <1> set\_get\_env:

4679 <1> ; 11/04/2016 (TRDOS 386 = TRDOS v2.0)

4680 <1> ; 02/09/2011 (TRDOS v1, CMD\_INTR.ASM, 'cmp\_cmd\_set')

4681 <1> ; 2005 - 28/08/2011

4682 <1> get\_setenv\_fchar:

4683 <1> ; esi = environment variable/string

4684 000093C2 8A06 <1> mov al, [esi]

4685 000093C4 3C20 <1> cmp al, 20h

4686 000093C6 771E <1> ja short loc\_find\_env

4687 <1>

4688 000093C8 BE00300900 <1> mov esi, Env\_Page

4689 <1> loc\_print\_setline:

4690 000093CD 803E00 <1> cmp byte [esi], 0

4691 000093D0 7613 <1> jna short loc\_setenv\_retn

4692 000093D2 E886CFFFFF <1> call print\_msg

4693 000093D7 56 <1> push esi

4694 000093D8 BE[6F190100] <1> mov esi, nextline

4695 000093DD E87BCFFFFF <1> call print\_msg

4696 000093E2 5E <1> pop esi

4697 000093E3 EBE8 <1> jmp short loc\_print\_setline

4698 <1>

4699 <1> loc\_setenv\_retn:

4700 000093E5 C3 <1> retn

4701 <1>

4702 <1> loc\_find\_env:

4703 000093E6 3C3D <1> cmp al, '='

4704 000093E8 0F8465E9FFFF <1> je loc\_cmd\_failed

4705 <1>

4706 000093EE 56 <1> push esi

4707 <1> loc\_repeat\_env\_equal\_check:

4708 000093EF 46 <1> inc esi

4709 000093F0 803E3D <1> cmp byte [esi], '='

4710 000093F3 7431 <1> je short pass\_env\_equal\_check

4711 000093F5 803E20 <1> cmp byte [esi], 20h

4712 000093F8 73F5 <1> jnb short loc\_repeat\_env\_equal\_check

4713 000093FA C60600 <1> mov byte [esi], 0

4714 000093FD 5E <1> pop esi

4715 000093FE BF[FE590100] <1> mov edi, TextBuffer ; out buffer

4716 00009403 B9FF000000 <1> mov ecx, 255 ; maximum size (limit)

4717 00009408 30C0 <1> xor al, al ; 0 -> use [ESI]

4718 0000940A E89E000000 <1> call get\_environment\_string

4719 0000940F 72D4 <1> jc short loc\_setenv\_retn

4720 <1>

4721 00009411 BE[FE590100] <1> mov esi, TextBuffer

4722 00009416 E842CFFFFF <1> call print\_msg

4723 0000941B BE[6F190100] <1> mov esi, nextline

4724 00009420 E838CFFFFF <1> call print\_msg

4725 <1>

4726 00009425 C3 <1> retn

4727 <1>

4728 <1> pass\_env\_equal\_check:

4729 00009426 46 <1> inc esi

4730 00009427 803E20 <1> cmp byte [esi], 20h

4731 0000942A 73FA <1> jnb short pass\_env\_equal\_check

4732 0000942C C60600 <1> mov byte [esi], 0

4733 <1>

4734 <1> loc\_call\_set\_env\_string:

4735 0000942F 5E <1> pop esi

4736 00009430 E83B010000 <1> call set\_environment\_string

4737 00009435 73AE <1> jnc short loc\_setenv\_retn

4738 <1>

4739 <1> loc\_set\_cmd\_failed:

4740 00009437 3C08 <1> cmp al, 08h

4741 00009439 0F8514E9FFFF <1> jne loc\_cmd\_failed

4742 <1>

4743 0000943F BE[40130100] <1> mov esi, Msg\_No\_Set\_Space

4744 00009444 E814CFFFFF <1> call print\_msg

4745 <1>

4746 00009449 C3 <1> retn

4747 <1>

4748 <1> set\_get\_path:

4749 <1> ; 11/04/2016 (TRDOS 386 = TRDOS v2.0)

4750 <1> ; 03/09/2011 (TRDOS v1, CMD\_INTR.ASM, 'cmp\_cmd\_path')

4751 <1> ; 2005

4752 <1> get\_path\_fchar:

4753 <1> ; esi = path

4754 0000944A 803E20 <1> cmp byte [esi], 20h

4755 0000944D 7737 <1> ja short loc\_set\_path

4756 <1>

4757 0000944F BE00300900 <1> mov esi, Env\_Page

4758 <1> loc\_print\_path:

4759 00009454 803E00 <1> cmp byte [esi], 0

4760 00009457 762C <1> jna short loc\_path\_retn

4761 <1>

4762 00009459 BE[9F0D0100] <1> mov esi, Cmd\_Path ; 'PATH' address

4763 0000945E BF[FE590100] <1> mov edi, TextBuffer ; out buffer

4764 00009463 30C0 <1> xor al, al ; use [ESI]

4765 00009465 B9FF000000 <1> mov ecx, 255 ; maximum size (limit)

4766 0000946A E83E000000 <1> call get\_environment\_string

4767 0000946F 7214 <1> jc short loc\_path\_retn

4768 <1>

4769 00009471 BE[FE590100] <1> mov esi, TextBuffer

4770 00009476 E8E2CEFFFF <1> call print\_msg

4771 0000947B BE[6F190100] <1> mov esi, nextline

4772 00009480 E8D8CEFFFF <1> call print\_msg

4773 <1>

4774 <1> loc\_path\_retn:

4775 00009485 C3 <1> retn

4776 <1>

4777 <1> loc\_set\_path:

4778 00009486 56 <1> push esi

4779 <1> loc\_set\_path\_find\_end:

4780 00009487 46 <1> inc esi

4781 00009488 803E20 <1> cmp byte [esi], 20h

4782 0000948B 73FA <1> jnb short loc\_set\_path\_find\_end

4783 0000948D C60600 <1> mov byte [esi], 0

4784 <1> loc\_set\_path\_header:

4785 00009490 5E <1> pop esi

4786 <1> set\_path\_x: ; 31/12/2017 ('syspath')

4787 00009491 4E <1> dec esi

4788 00009492 C6063D <1> mov byte [esi], '='

4789 00009495 4E <1> dec esi

4790 00009496 C60648 <1> mov byte [esi], 'H'

4791 00009499 4E <1> dec esi

4792 0000949A C60654 <1> mov byte [esi], 'T'

4793 0000949D 4E <1> dec esi

4794 0000949E C60641 <1> mov byte [esi], 'A'

4795 000094A1 4E <1> dec esi

4796 000094A2 C60650 <1> mov byte [esi], 'P'

4797 <1>

4798 <1> loc\_path\_call\_set\_env\_string:

4799 000094A5 E8C6000000 <1> call set\_environment\_string

4800 000094AA 728B <1> jc short loc\_set\_cmd\_failed

4801 <1>

4802 000094AC C3 <1> retn

4803 <1>

4804 <1> get\_environment\_string:

4805 <1> ; 12/04/2016

4806 <1> ; 11/04/2016

4807 <1> ; 05/04/2016 (TRDOS 386 = TRDOS v2.0)

4808 <1> ; 02/09/2011 (TRDOS v1, MAINPROG.ASM)

4809 <1> ; 28/08/2011

4810 <1> ; INPUT->

4811 <1> ; EDI = Output buffer

4812 <1> ; CX = Buffer length (<= ENV\_PAGE\_SIZE)

4813 <1> ;

4814 <1> ; AL > 0 = AL = String sequence number

4815 <1> ; AL = 0 -> ESI = ASCIIZ Set word

4816 <1> ; (environment variable)

4817 <1> ; OUTPUT ->

4818 <1> ; ESI is not changed

4819 <1> ; EDI is not changed

4820 <1> ; EAX = String length (with zero tail)

4821 <1> ; EDX = Environment variables page address

4822 <1> ; CF = 1 -> Not found (EAX not valid)

4823 <1> ;

4824 <1> ; (Modified registers: EAX, EDX)

4825 <1>

4826 000094AD BA00300900 <1> mov edx, Env\_Page

4827 000094B2 803A00 <1> cmp byte [edx], 0

4828 000094B5 7474 <1> jz short get\_env\_string\_with\_word\_stc\_retn

4829 <1>

4830 000094B7 66890D[6C650100] <1> mov [env\_var\_length], cx

4831 <1>

4832 000094BE 51 <1> push ecx ; \*

4833 000094BF 56 <1> push esi ; \*\*

4834 <1>

4835 000094C0 08C0 <1> or al, al

4836 000094C2 7449 <1> jz short get\_env\_string\_with\_word

4837 <1>

4838 <1> get\_env\_string\_with\_seq\_number:

4839 000094C4 B101 <1> mov cl, 1

4840 000094C6 88C5 <1> mov ch, al

4841 000094C8 31C0 <1> xor eax, eax

4842 000094CA 89D6 <1> mov esi, edx ; Env\_Page

4843 <1>

4844 <1> get\_env\_string\_seq\_number\_check:

4845 000094CC 38CD <1> cmp ch, cl

4846 000094CE 7726 <1> ja short get\_env\_string\_seq\_number\_next

4847 <1>

4848 <1> get\_env\_string\_move\_to\_buff:

4849 000094D0 57 <1> push edi ; \*\*\*

4850 <1>

4851 000094D1 29D2 <1> sub edx, edx

4852 <1>

4853 <1> get\_env\_string\_seq\_number\_repeat1:

4854 000094D3 42 <1> inc edx

4855 000094D4 AC <1> lodsb

4856 000094D5 AA <1> stosb

4857 <1>

4858 000094D6 66FF0D[6C650100] <1> dec word [env\_var\_length]

4859 000094DD 7508 <1> jnz short get\_env\_string\_seq\_number\_repeat3

4860 <1>

4861 <1> get\_env\_string\_seq\_number\_repeat2:

4862 000094DF 20C0 <1> and al, al

4863 000094E1 7408 <1> jz short get\_env\_string\_seq\_number\_ok

4864 000094E3 42 <1> inc edx

4865 000094E4 AC <1> lodsb

4866 000094E5 EBF8 <1> jmp short get\_env\_string\_seq\_number\_repeat2

4867 <1>

4868 <1> get\_env\_string\_seq\_number\_repeat3:

4869 000094E7 08C0 <1> or al, al

4870 000094E9 75E8 <1> jnz short get\_env\_string\_seq\_number\_repeat1

4871 <1>

4872 <1> get\_env\_string\_seq\_number\_ok:

4873 000094EB 5F <1> pop edi ; \*\*\*

4874 000094EC 89D0 <1> mov eax, edx ; Length of the environment string

4875 <1> ; (ASCIIZ, includes ZERO tail)

4876 000094EE BA00300900 <1> mov edx, Env\_Page

4877 <1>

4878 <1> get\_env\_string\_stc\_retn:

4879 000094F3 5E <1> pop esi ; \*\*

4880 000094F4 59 <1> pop ecx ; \*

4881 000094F5 C3 <1> retn

4882 <1>

4883 <1> get\_env\_string\_seq\_number\_next:

4884 000094F6 AC <1> lodsb

4885 000094F7 08C0 <1> or al, al

4886 000094F9 75FB <1> jnz short get\_env\_string\_seq\_number\_next

4887 <1>

4888 000094FB 81FE00320900 <1> cmp esi, Env\_Page + Env\_Page\_Size ; +512 (+4096)

4889 00009501 F5 <1> cmc

4890 00009502 72EF <1> jc short get\_env\_string\_stc\_retn

4891 <1>

4892 00009504 AC <1> lodsb

4893 00009505 3C01 <1> cmp al, 1

4894 00009507 72EA <1> jb short get\_env\_string\_stc\_retn

4895 00009509 FEC1 <1> inc cl

4896 0000950B EBBF <1> jmp short get\_env\_string\_seq\_number\_check

4897 <1>

4898 <1> get\_env\_string\_with\_word:

4899 0000950D 31C9 <1> xor ecx, ecx

4900 <1>

4901 <1> get\_env\_string\_calc\_word\_length:

4902 0000950F AC <1> lodsb

4903 00009510 3C20 <1> cmp al, 20h

4904 00009512 7211 <1> jb short get\_env\_string\_calc\_word\_length\_ok

4905 <1> ;inc cx

4906 00009514 FEC1 <1> inc cl

4907 <1>

4908 00009516 3C61 <1> cmp al, 'a'

4909 00009518 72F5 <1> jb short get\_env\_string\_calc\_word\_length

4910 0000951A 3C7A <1> cmp al, 'z'

4911 0000951C 77F1 <1> ja short get\_env\_string\_calc\_word\_length

4912 0000951E 24DF <1> and al, 0DFh

4913 00009520 8846FF <1> mov [esi-1], al

4914 00009523 EBEA <1> jmp short get\_env\_string\_calc\_word\_length

4915 <1>

4916 <1> get\_env\_string\_calc\_word\_length\_ok:

4917 00009525 08C9 <1> or cl, cl

4918 00009527 7506 <1> jnz short get\_env\_string\_calc\_word\_length\_save

4919 <1>

4920 00009529 5E <1> pop esi ; \*\*

4921 <1>

4922 <1> get\_env\_string\_stc\_retn1:

4923 0000952A 59 <1> pop ecx ; \*

4924 <1>

4925 <1> get\_env\_string\_with\_word\_stc\_retn:

4926 0000952B 31C0 <1> xor eax, eax

4927 0000952D F9 <1> stc

4928 0000952E C3 <1> retn

4929 <1>

4930 <1> get\_env\_string\_calc\_word\_length\_save:

4931 0000952F 871C24 <1> xchg ebx, [esp] ; \*\*

4932 00009532 89DE <1> mov esi, ebx

4933 <1> ; Start of the env string (to be searched)

4934 <1>

4935 00009534 57 <1> push edi ; \*\*\*

4936 00009535 89D7 <1> mov edi, edx ; Env\_Page

4937 <1>

4938 <1> get\_env\_string\_compare:

4939 00009537 57 <1> push edi ; \*\*\*\*

4940 00009538 51 <1> push ecx ; \*\*\*\*\* ; Variable name length

4941 <1>

4942 <1> get\_env\_string\_compare\_rep:

4943 00009539 AC <1> lodsb

4944 0000953A AE <1> scasb

4945 0000953B 7511 <1> jne short get\_env\_string\_compare\_next1

4946 0000953D E2FA <1> loop get\_env\_string\_compare\_rep

4947 <1>

4948 0000953F 803F3D <1> cmp byte [edi], '='

4949 00009542 750A <1> jne short get\_env\_string\_compare\_next1

4950 <1>

4951 00009544 59 <1> pop ecx ; \*\*\*\*\*

4952 00009545 5F <1> pop edi ; \*\*\*\*

4953 00009546 89FE <1> mov esi, edi

4954 00009548 5F <1> pop edi ; \*\*\*

4955 00009549 871C24 <1> xchg ebx, [esp] ; \*\*

4956 0000954C EB82 <1> jmp short get\_env\_string\_move\_to\_buff

4957 <1>

4958 <1> get\_env\_string\_compare\_next1:

4959 0000954E 89FE <1> mov esi, edi

4960 00009550 59 <1> pop ecx ; \*\*\*\*\*

4961 00009551 5F <1> pop edi ; \*\*\*\*

4962 <1> get\_env\_string\_compare\_next2:

4963 00009552 81FEFF310900 <1> cmp esi, Env\_Page + Env\_Page\_Size - 1 ; +511 (+4095)

4964 00009558 7310 <1> jnb short get\_env\_string\_compare\_not\_ok

4965 0000955A 20C0 <1> and al, al

4966 0000955C AC <1> lodsb

4967 0000955D 75F3 <1> jnz short get\_env\_string\_compare\_next2

4968 0000955F 08C0 <1> or al, al

4969 00009561 7407 <1> jz short get\_env\_string\_compare\_not\_ok

4970 00009563 4E <1> dec esi ; 12/04/2016

4971 00009564 89F7 <1> mov edi, esi

4972 00009566 89DE <1> mov esi, ebx

4973 00009568 EBCD <1> jmp short get\_env\_string\_compare

4974 <1>

4975 <1> get\_env\_string\_compare\_not\_ok:

4976 0000956A 5F <1> pop edi ; \*\*\*

4977 0000956B 89DE <1> mov esi, ebx

4978 0000956D 5B <1> pop ebx ; \*\*

4979 0000956E EBBA <1> jmp short get\_env\_string\_stc\_retn1

4980 <1>

4981 <1> set\_environment\_string:

4982 <1> ; 13/04/2016

4983 <1> ; 12/04/2016

4984 <1> ; 11/04/2016

4985 <1> ; 06/04/2016

4986 <1> ; 05/04/2016 (TRDOS 386 = TRDOS v2.0)

4987 <1> ; 02/09/2011 (TRDOS v1, MAINPROG.ASM)

4988 <1> ; 29/08/2011

4989 <1> ; 29/08/2011

4990 <1> ; INPUT->

4991 <1> ; ESI = ASCIIZ environment string

4992 <1> ; OUTPUT ->

4993 <1> ; ESI is not changed

4994 <1> ; CF = 1 -> Could not set,

4995 <1> ; insufficient environment space

4996 <1> ;

4997 <1> ; (EAX, EDX will be changed)

4998 <1> ;

4999 <1> ; (EAX = Start address of the env string if > 0)

5000 <1> ; (EDX = Environment string length)

5001 <1>

5002 00009570 56 <1> push esi ; \*

5003 <1>

5004 00009571 31C0 <1> xor eax, eax

5005 <1>

5006 <1> set\_env\_chk\_validation1:

5007 00009573 FEC4 <1> inc ah ; variable (string) length

5008 00009575 AC <1> lodsb

5009 00009576 3C3D <1> cmp al, '='

5010 00009578 7415 <1> je short set\_env\_chk\_validation2

5011 0000957A 3C20 <1> cmp al, 20h

5012 0000957C 720F <1> jb short set\_env\_string\_stc

5013 <1>

5014 <1> ; 06/04/2016

5015 0000957E 3C61 <1> cmp al, 'a'

5016 00009580 72F1 <1> jb short set\_env\_chk\_validation1

5017 00009582 3C7A <1> cmp al, 'z'

5018 00009584 77ED <1> ja short set\_env\_chk\_validation1

5019 00009586 2C20 <1> sub al, 'a'-'A'

5020 00009588 8846FF <1> mov [esi-1], al

5021 0000958B EBE6 <1> jmp short set\_env\_chk\_validation1

5022 <1>

5023 <1> set\_env\_string\_stc:

5024 0000958D 5E <1> pop esi ; \*

5025 <1> ;stc

5026 0000958E C3 <1> retn

5027 <1>

5028 <1> set\_env\_chk\_validation2:

5029 0000958F 51 <1> push ecx ; \*\*

5030 00009590 53 <1> push ebx ; \*\*\*

5031 00009591 57 <1> push edi ; \*\*\*\*

5032 <1>

5033 <1> ; 12/04/2016

5034 00009592 8B5C240C <1> mov ebx, [esp+12]

5035 <1>

5036 <1> set\_env\_chk\_validation2w:

5037 00009596 89F7 <1> mov edi, esi

5038 00009598 4F <1> dec edi

5039 <1>

5040 00009599 807FFF20 <1> cmp byte [edi-1], 20h

5041 0000959D 771A <1> ja short set\_env\_chk\_validation2z

5042 <1>

5043 0000959F 56 <1> push esi

5044 000095A0 89FE <1> mov esi, edi

5045 000095A2 4E <1> dec esi

5046 <1>

5047 <1> set\_env\_chk\_validation2x:

5048 000095A3 4E <1> dec esi

5049 <1>

5050 000095A4 39DE <1> cmp esi, ebx

5051 000095A6 7207 <1> jb short set\_env\_chk\_validation2y

5052 <1>

5053 000095A8 4F <1> dec edi

5054 <1>

5055 000095A9 8A06 <1> mov al, [esi]

5056 000095AB 8807 <1> mov [edi], al

5057 <1>

5058 000095AD EBF4 <1> jmp short set\_env\_chk\_validation2x

5059 <1>

5060 <1> set\_env\_chk\_validation2y:

5061 000095AF 5E <1> pop esi

5062 <1>

5063 <1> ;mov byte [ebx], 20h

5064 <1>

5065 000095B0 43 <1> inc ebx

5066 000095B1 895C240C <1> mov [esp+12], ebx

5067 <1>

5068 000095B5 FECC <1> dec ah ; 13/04/2016

5069 <1>

5070 000095B7 EBDD <1> jmp short set\_env\_chk\_validation2w

5071 <1>

5072 <1> set\_env\_chk\_validation2z:

5073 000095B9 BA00300900 <1> mov edx, Env\_Page

5074 000095BE 89D7 <1> mov edi, edx

5075 <1>

5076 <1> set\_env\_chk\_validation3:

5077 000095C0 AC <1> lodsb

5078 000095C1 3C20 <1> cmp al, 20h

5079 000095C3 74FB <1> je short set\_env\_chk\_validation3

5080 <1>

5081 000095C5 9C <1> pushf

5082 <1>

5083 <1> ; 12/04/2016

5084 <1> set\_env\_chk\_validation3n:

5085 000095C6 3C61 <1> cmp al, 'a'

5086 000095C8 720C <1> jb short set\_env\_chk\_validation3c

5087 000095CA 3C7A <1> cmp al, 'z'

5088 000095CC 7705 <1> ja short set\_env\_chk\_validation3x

5089 000095CE 2C20 <1> sub al, 'a'-'A'

5090 000095D0 8846FF <1> mov [esi-1], al

5091 <1>

5092 <1> set\_env\_chk\_validation3x:

5093 000095D3 AC <1> lodsb

5094 000095D4 EBF0 <1> jmp short set\_env\_chk\_validation3n

5095 <1>

5096 <1> set\_env\_chk\_validation3c:

5097 000095D6 3C20 <1> cmp al, 20h

5098 000095D8 73F9 <1> jnb short set\_env\_chk\_validation3x

5099 <1>

5100 000095DA 803F00 <1> cmp byte [edi], 0

5101 000095DD 7731 <1> ja short set\_env\_chk\_validation4

5102 <1>

5103 000095DF 9D <1> popf

5104 000095E0 7228 <1> jb short set\_env\_string\_nothing

5105 <1>

5106 000095E2 B900020000 <1> mov ecx, Env\_Page\_Size ; 512 (4096)

5107 <1>

5108 000095E7 89DE <1> mov esi, ebx ; 12/04/2016

5109 <1>

5110 <1> set\_env\_string\_copy\_to\_envb:

5111 000095E9 AC <1> lodsb

5112 000095EA 3C20 <1> cmp al, 20h

5113 000095EC 720A <1> jb short set\_env\_string\_copy\_to\_envb\_z

5114 000095EE AA <1> stosb

5115 000095EF E2F8 <1> loop set\_env\_string\_copy\_to\_envb

5116 <1>

5117 <1> ; 11/04/2016

5118 000095F1 89D7 <1> mov edi, edx ; Env\_Page

5119 000095F3 B900020000 <1> mov ecx, Env\_Page\_Size

5120 <1>

5121 <1> set\_env\_string\_copy\_to\_envb\_z:

5122 000095F8 52 <1> push edx ; Start address of the variable

5123 000095F9 BA00020000 <1> mov edx, Env\_Page\_Size

5124 000095FE 29CA <1> sub edx, ecx ; variable (string) length

5125 <1>

5126 00009600 28C0 <1> sub al, al ; 0

5127 00009602 F3AA <1> rep stosb ; clear remain bytes of the env page

5128 <1>

5129 00009604 58 <1> pop eax ; Start address of the variable

5130 <1>

5131 <1> set\_env\_string\_allocate\_envb\_retn: ; stc or clc return

5132 00009605 5F <1> pop edi ; \*\*\*\*

5133 00009606 5B <1> pop ebx ; \*\*\*

5134 00009607 59 <1> pop ecx ; \*\*

5135 00009608 5E <1> pop esi ; \*

5136 00009609 C3 <1> retn

5137 <1>

5138 <1> set\_env\_string\_nothing:

5139 0000960A 31C0 <1> xor eax, eax

5140 0000960C 31D2 <1> xor edx, edx ; 11/04/2016

5141 0000960E EBF5 <1> jmp short set\_env\_string\_allocate\_envb\_retn

5142 <1>

5143 <1> set\_env\_chk\_validation4:

5144 <1> ; 11/04/2016

5145 00009610 9D <1> popf

5146 <1>

5147 00009611 89D6 <1> mov esi, edx ; Env\_Page

5148 <1>

5149 <1> set\_env\_chk\_validation5:

5150 00009613 89DF <1> mov edi, ebx ; ASCIIZ environment string address

5151 00009615 0FB6CC <1> movzx ecx, ah ; Variable (string) length (with '=')

5152 <1>

5153 <1> set\_env\_chk\_validation5\_loop:

5154 00009618 AC <1> lodsb

5155 00009619 AE <1> scasb

5156 0000961A 750A <1> jne short set\_env\_chk\_validation6

5157 0000961C E2FA <1> loop set\_env\_chk\_validation5\_loop

5158 <1>

5159 0000961E 3C3D <1> cmp al, '='

5160 00009620 0F8483000000 <1> je set\_env\_change\_variable

5161 <1>

5162 <1> set\_env\_chk\_validation6:

5163 00009626 08C0 <1> or al, al ; 0

5164 00009628 7403 <1> jz short set\_env\_chk\_validation7

5165 <1>

5166 0000962A AC <1> lodsb

5167 0000962B EBF9 <1> jmp short set\_env\_chk\_validation6

5168 <1>

5169 <1> set\_env\_chk\_validation7:

5170 0000962D 88E1 <1> mov cl, ah

5171 0000962F 01F1 <1> add ecx, esi

5172 00009631 81F9FF310900 <1> cmp ecx, Env\_Page + Env\_Page\_Size - 1

5173 <1> ; 511 (4095)

5174 <1> ; strlen + '=' + 0

5175 00009637 72DA <1> jb short set\_env\_chk\_validation5

5176 <1>

5177 <1> set\_env\_chk\_validation8: ; variable not found

5178 00009639 0FB6F4 <1> movzx esi, ah ; variable name length (with '=')

5179 0000963C 01DE <1> add esi, ebx ; position just after of the '='

5180 <1>

5181 <1> set\_env\_chk\_validation8\_loop:

5182 0000963E AC <1> lodsb

5183 0000963F 3C20 <1> cmp al, 20h

5184 00009641 74FB <1> je short set\_env\_chk\_validation8\_loop

5185 00009643 72C5 <1> jb short set\_env\_string\_nothing

5186 <1>

5187 <1> set\_env\_chk\_validation9:

5188 00009645 AC <1> lodsb

5189 00009646 3C20 <1> cmp al, 20h

5190 00009648 73FB <1> jnb short set\_env\_chk\_validation9

5191 <1>

5192 <1> ; End of ASCIIZ environment string

5193 <1>

5194 <1> set\_env\_add\_variable:

5195 0000964A 29DE <1> sub esi, ebx ; variable+definition length

5196 <1>

5197 0000964C 56 <1> push esi ; \*\*\*\*\*

5198 <1>

5199 0000964D 89D6 <1> mov esi, edx ; Environment page address

5200 <1>

5201 0000964F B900020000 <1> mov ecx, Env\_Page\_Size ; 512 (4096)

5202 <1>

5203 <1> set\_env\_add\_variable\_loop:

5204 00009654 AC <1> lodsb

5205 00009655 20C0 <1> and al, al

5206 00009657 7406 <1> jz short set\_env\_add\_variable\_chk1 ; 0

5207 00009659 E2F9 <1> loop set\_env\_add\_variable\_loop

5208 <1>

5209 <1> ; 11/04/2016

5210 0000965B 884EFF <1> mov [esi-1], cl ; 0

5211 0000965E 41 <1> inc ecx

5212 <1>

5213 <1> set\_env\_add\_variable\_chk1:

5214 0000965F 49 <1> dec ecx

5215 00009660 7408 <1> jz short set\_env\_add\_variable\_nspc

5216 00009662 AC <1> lodsb

5217 00009663 08C0 <1> or al, al

5218 00009665 740C <1> jz short set\_env\_add\_variable\_chk2 ; 00

5219 00009667 49 <1> dec ecx

5220 00009668 75EA <1> jnz short set\_env\_add\_variable\_loop

5221 <1>

5222 <1> set\_env\_add\_variable\_nspc: ; no space on environment page

5223 0000966A 58 <1> pop eax ; \*\*\*\*\*

5224 0000966B B808000000 <1> mov eax, 8 ; No space for new environment string

5225 00009670 F9 <1> stc

5226 00009671 EB92 <1> jmp short set\_env\_string\_allocate\_envb\_retn

5227 <1>

5228 <1> set\_env\_add\_variable\_chk2:

5229 00009673 8B0C24 <1> mov ecx, [esp] ; \*\*\*\*\*

5230 00009676 4E <1> dec esi ; beginning address of the new variable

5231 00009677 89F0 <1> mov eax, esi

5232 00009679 01C8 <1> add eax, ecx ; string length (with CR)

5233 0000967B 81C200020000 <1> add edx, Env\_Page\_Size ; 512 (4096)

5234 00009681 39D0 <1> cmp eax, edx

5235 00009683 77E5 <1> ja short set\_env\_add\_variable\_nspc

5236 00009685 49 <1> dec ecx ; except CR at the end

5237 00009686 89CA <1> mov edx, ecx ; 12/04/2016

5238 00009688 89F7 <1> mov edi, esi

5239 0000968A 893C24 <1> mov [esp], edi ; \*\*\*\*\* ; Start address of new variable

5240 0000968D 89DE <1> mov esi, ebx ; ASCIIZ environment string address

5241 0000968F F3A4 <1> rep movsb

5242 00009691 28C0 <1> sub al, al

5243 00009693 AA <1> stosb

5244 00009694 58 <1> pop eax ; \*\*\*\*\* ; Beginning address of new variable

5245 00009695 81FF00320900 <1> cmp edi, Env\_Page + Env\_Page\_Size ; 12/04/2016

5246 0000969B 0F8364FFFFFF <1> jnb set\_env\_string\_allocate\_envb\_retn ; OK !

5247 000096A1 880F <1> mov [edi], cl ; 0

5248 000096A3 F8 <1> clc ; 13/04/2016

5249 000096A4 E95CFFFFFF <1> jmp set\_env\_string\_allocate\_envb\_retn ; OK !

5250 <1>

5251 <1> set\_env\_change\_variable:

5252 <1> ; 06/04/2016

5253 <1> ; esi = Variable's address in environment page (after '=')

5254 <1> ; edi = ASCIIZ environment string address (after '=')

5255 <1>

5256 <1> ; ah = variable length from start to the '='

5257 000096A9 8825[6C650100] <1> mov [env\_var\_length], ah

5258 <1>

5259 000096AF 28C9 <1> sub cl, cl ; ecx = 0

5260 <1>

5261 000096B1 57 <1> push edi ; \*\*\*\*\*

5262 <1>

5263 000096B2 89F7 <1> mov edi, esi ; 11/04/2016

5264 <1>

5265 <1> set\_env\_change\_variable\_calc1:

5266 000096B4 AC <1> lodsb

5267 000096B5 08C0 <1> or al, al

5268 000096B7 7403 <1> jz short set\_env\_change\_variable\_calc2

5269 <1>

5270 000096B9 41 <1> inc ecx ; length of environment string (after the '=')

5271 <1>

5272 000096BA EBF8 <1> jmp short set\_env\_change\_variable\_calc1

5273 <1>

5274 <1> set\_env\_change\_variable\_calc2:

5275 000096BC 8B3424 <1> mov esi, [esp] ; ASCIIZ environment string address

5276 <1>

5277 000096BF 29D2 <1> sub edx, edx

5278 <1>

5279 <1> set\_env\_change\_variable\_calc3:

5280 000096C1 AC <1> lodsb

5281 000096C2 3C20 <1> cmp al, 20h

5282 000096C4 7203 <1> jb short set\_env\_change\_variable\_calc4

5283 <1>

5284 000096C6 42 <1> inc edx ; length of ASCIIZ string (after the '=')

5285 <1>

5286 000096C7 EBF8 <1> jmp short set\_env\_change\_variable\_calc3

5287 <1>

5288 <1> set\_env\_change\_variable\_calc4:

5289 000096C9 C646FF00 <1> mov byte [esi-1], 0 ; put ZERO instead of CR

5290 <1>

5291 000096CD 5E <1> pop esi ; \*\*\*\*\* ; ASCIIZ string address (after '=')

5292 <1>

5293 <1> ; EDI = Old variable's address (after '=')

5294 <1>

5295 <1> ; compare the new string with the old string

5296 000096CE 39CA <1> cmp edx, ecx

5297 000096D0 7717 <1> ja short set\_env\_change\_variable\_calc5 ; longer

5298 000096D2 0F828F000000 <1> jb set\_env\_change\_variable\_calc9 ; shorter

5299 <1>

5300 <1> ;same length (simple copy)

5301 000096D8 0FB6C4 <1> movzx eax, ah

5302 000096DB 01C2 <1> add edx, eax

5303 000096DD F7D8 <1> neg eax

5304 000096DF 01F8 <1> add eax, edi

5305 <1> ; EAX = Start address of the variable

5306 <1> ; EDX = Variable length (without ZERO at the end of variable)

5307 <1>

5308 000096E1 F3A4 <1> rep movsb

5309 000096E3 F8 <1> clc ; 13/04/2016

5310 000096E4 E91CFFFFFF <1> jmp set\_env\_string\_allocate\_envb\_retn ; OK !

5311 <1>

5312 <1> set\_env\_change\_variable\_calc5:

5313 <1> ; 11/04/2016

5314 000096E9 52 <1> push edx ; \*\*\*\*\*

5315 000096EA 29CA <1> sub edx, ecx ; difference ; (the new string is longer)

5316 000096EC 89F3 <1> mov ebx, esi

5317 000096EE 89FE <1> mov esi, edi

5318 <1>

5319 <1> set\_env\_change\_variable\_calc6:

5320 000096F0 AC <1> lodsb

5321 000096F1 20C0 <1> and al, al

5322 000096F3 75FB <1> jnz short set\_env\_change\_variable\_calc6

5323 <1>

5324 000096F5 81FE00320900 <1> cmp esi, Env\_Page + Env\_Page\_Size ; 512 (4096)

5325 000096FB 0F8369FFFFFF <1> jnb set\_env\_add\_variable\_nspc

5326 <1>

5327 00009701 89F9 <1> mov ecx, edi ; current (old) variable's address

5328 00009703 89F7 <1> mov edi, esi ; next variable's address

5329 <1>

5330 00009705 AC <1> lodsb

5331 00009706 08C0 <1> or al, al

5332 00009708 7416 <1> jz short set\_env\_change\_variable\_calc8 ; 00

5333 <1>

5334 <1> set\_env\_change\_variable\_calc7:

5335 0000970A AC <1> lodsb

5336 0000970B 20C0 <1> and al, al

5337 0000970D 75FB <1> jnz short set\_env\_change\_variable\_calc7

5338 <1>

5339 0000970F 81FE00320900 <1> cmp esi, Env\_Page + Env\_Page\_Size ; 512 (4096)

5340 00009715 0F834FFFFFFF <1> jnb set\_env\_add\_variable\_nspc

5341 <1>

5342 0000971B AC <1> lodsb

5343 0000971C 08C0 <1> or al, al

5344 0000971E 75EA <1> jnz short set\_env\_change\_variable\_calc7

5345 <1>

5346 <1> set\_env\_change\_variable\_calc8:

5347 00009720 4E <1> dec esi ; address of the second (last) 0 of the 00

5348 <1>

5349 00009721 01F2 <1> add edx, esi ; final position of the last 0

5350 <1>

5351 00009723 81FA00320900 <1> cmp edx, Env\_Page + Env\_Page\_Size ; 512 (4096)

5352 00009729 0F833BFFFFFF <1> jnb set\_env\_add\_variable\_nspc

5353 <1>

5354 0000972F 89C8 <1> mov eax, ecx ; old variable's address (after '=')

5355 <1>

5356 00009731 89F1 <1> mov ecx, esi

5357 00009733 29F9 <1> sub ecx, edi ; count of bytes to move forward

5358 <1>

5359 <1> ; 13/04/2016

5360 00009735 C60200 <1> mov byte [edx], 0

5361 00009738 89D7 <1> mov edi, edx

5362 0000973A 29F2 <1> sub edx, esi ; difference (additional byte count)

5363 0000973C 4F <1> dec edi ; the last zero address (first byte of the 00)

5364 0000973D 89FE <1> mov esi, edi

5365 0000973F 29D6 <1> sub esi, edx ; - displacement

5366 <1>

5367 00009741 FA <1> cli ; disable interrupts

5368 00009742 FD <1> std ; backward

5369 <1>

5370 00009743 F3A4 <1> rep movsb ; move ECX bytes from DS:ESI to ES:EDI

5371 <1>

5372 00009745 FC <1> cld ; forward (default)

5373 00009746 FB <1> sti ; enable interrupts

5374 <1>

5375 00009747 89C7 <1> mov edi, eax

5376 00009749 59 <1> pop ecx ; \*\*\*\*\* ; byte count (after '=')

5377 0000974A 89CA <1> mov edx, ecx

5378 0000974C 89DE <1> mov esi, ebx ; ASCIIZ string address (after '=')

5379 0000974E 89FB <1> mov ebx, edi

5380 <1>

5381 00009750 F3A4 <1> rep movsb

5382 <1>

5383 00009752 880F <1> mov [edi], cl ; 0 ; end of variable

5384 <1>

5385 00009754 0FB605[6C650100] <1> movzx eax, byte [env\_var\_length]

5386 0000975B 01C2 <1> add edx, eax ; variable length (total)

5387 0000975D F7D8 <1> neg eax

5388 0000975F 01D8 <1> add eax, ebx ; start address of the variable

5389 00009761 F8 <1> clc ; 13/04/2016

5390 00009762 E99EFEFFFF <1> jmp set\_env\_string\_allocate\_envb\_retn ; OK !

5391 <1>

5392 <1> set\_env\_change\_variable\_calc9:

5393 <1> ; 11/04/2016

5394 00009767 21D2 <1> and edx, edx ; is empty ?

5395 00009769 753B <1> jnz short set\_env\_change\_variable\_calc15

5396 <1>

5397 0000976B 0FB6DC <1> movzx ebx, ah

5398 0000976E F7DB <1> neg ebx

5399 00009770 01FB <1> add ebx, edi

5400 <1>

5401 <1> ; EBX = Start address of the variable (in env page)

5402 <1> ; EDX = Variable length = 0

5403 <1>

5404 00009772 89FE <1> mov esi, edi

5405 <1>

5406 <1> set\_env\_change\_variable\_calc10:

5407 00009774 AC <1> lodsb

5408 00009775 08C0 <1> or al, al

5409 00009777 75FB <1> jnz short set\_env\_change\_variable\_calc10

5410 <1>

5411 00009779 B9FF310900 <1> mov ecx, Env\_Page + Env\_Page\_Size - 1

5412 <1>

5413 0000977E 39CE <1> cmp esi, ecx ; +511 (+4095)

5414 00009780 7604 <1> jna short set\_env\_change\_variable\_calc11

5415 <1>

5416 00009782 89CE <1> mov esi, ecx

5417 00009784 8806 <1> mov [esi], al ; 0

5418 <1>

5419 <1> set\_env\_change\_variable\_calc11:

5420 00009786 89DF <1> mov edi, ebx ; old variable's start address

5421 <1>

5422 <1> set\_env\_change\_variable\_calc12:

5423 00009788 AC <1> lodsb

5424 00009789 AA <1> stosb

5425 0000978A 20C0 <1> and al, al

5426 0000978C 75FA <1> jnz short set\_env\_change\_variable\_calc12

5427 0000978E 39CE <1> cmp esi, ecx

5428 00009790 7706 <1> ja short set\_env\_change\_variable\_calc13

5429 00009792 AC <1> lodsb

5430 00009793 AA <1> stosb

5431 00009794 20C0 <1> and al, al

5432 00009796 75F0 <1> jnz short set\_env\_change\_variable\_calc12

5433 <1>

5434 <1> set\_env\_change\_variable\_calc13:

5435 00009798 29F9 <1> sub ecx, edi

5436 0000979A 7203 <1> jb short set\_env\_change\_variable\_calc14

5437 0000979C 41 <1> inc ecx ; 1-512 (1-4096)

5438 0000979D F3AA <1> rep stosb ; al = 0

5439 <1>

5440 <1> set\_env\_change\_variable\_calc14:

5441 0000979F 29C0 <1> sub eax, eax ; Start address of the variable

5442 <1> ; EAX = 0 -> Variable is removed

5443 <1> ; EDX = Variable length = 0

5444 <1>

5445 000097A1 E95FFEFFFF <1> jmp set\_env\_string\_allocate\_envb\_retn ; OK !

5446 <1>

5447 <1> set\_env\_change\_variable\_calc15:

5448 000097A6 52 <1> push edx ; \*\*\*\*\*

5449 000097A7 F7DA <1> neg edx

5450 000097A9 01CA <1> add edx, ecx ; difference (the old string is longer)

5451 000097AB 89F3 <1> mov ebx, esi

5452 000097AD 89FE <1> mov esi, edi

5453 <1>

5454 <1> set\_env\_change\_variable\_calc16:

5455 000097AF AC <1> lodsb

5456 000097B0 20C0 <1> and al, al

5457 000097B2 75FB <1> jnz short set\_env\_change\_variable\_calc16

5458 <1>

5459 000097B4 B900320900 <1> mov ecx, Env\_Page + Env\_Page\_Size

5460 <1>

5461 000097B9 39CE <1> cmp esi, ecx ; +512 (+4096)

5462 000097BB 7605 <1> jna short set\_env\_change\_variable\_calc17

5463 <1>

5464 000097BD 89CE <1> mov esi, ecx

5465 000097BF 8846FF <1> mov [esi-1], al ; 0

5466 <1>

5467 <1> set\_env\_change\_variable\_calc17:

5468 000097C2 89F9 <1> mov ecx, edi ; current (old) variable's address

5469 000097C4 89F7 <1> mov edi, esi ; next variable's address

5470 <1>

5471 000097C6 AC <1> lodsb

5472 000097C7 08C0 <1> or al, al

5473 000097C9 741D <1> jz short set\_env\_change\_variable\_calc20

5474 <1>

5475 <1> set\_env\_change\_variable\_calc18:

5476 000097CB AC <1> lodsb

5477 000097CC 20C0 <1> and al, al

5478 000097CE 75FB <1> jnz short set\_env\_change\_variable\_calc18

5479 <1>

5480 000097D0 81FE00320900 <1> cmp esi, Env\_Page + Env\_Page\_Size

5481 000097D6 720B <1> jb short set\_env\_change\_variable\_calc19

5482 000097D8 740E <1> je short set\_env\_change\_variable\_calc20

5483 <1>

5484 000097DA BEFF310900 <1> mov esi, Env\_Page + Env\_Page\_Size - 1

5485 000097DF 8806 <1> mov [esi], al ; 0

5486 000097E1 EB06 <1> jmp short set\_env\_change\_variable\_calc21

5487 <1>

5488 <1> set\_env\_change\_variable\_calc19:

5489 000097E3 AC <1> lodsb

5490 000097E4 08C0 <1> or al, al

5491 000097E6 75E3 <1> jnz short set\_env\_change\_variable\_calc18

5492 <1>

5493 <1> set\_env\_change\_variable\_calc20:

5494 000097E8 4E <1> dec esi ; address of the second (last) 0 of the 00

5495 <1>

5496 <1> set\_env\_change\_variable\_calc21:

5497 <1> ; edx = difference (byte count)

5498 <1>

5499 000097E9 89C8 <1> mov eax, ecx ; old variable's address (after '=')

5500 <1>

5501 000097EB 89F1 <1> mov ecx, esi

5502 000097ED 29F9 <1> sub ecx, edi ; count of bytes to move backward

5503 <1>

5504 000097EF 89FE <1> mov esi, edi ; next variable's address

5505 000097F1 29D7 <1> sub edi, edx ; (displacement)

5506 <1>

5507 000097F3 F3A4 <1> rep movsb

5508 <1>

5509 000097F5 880F <1> mov [edi], cl ; 0 ; 00 ; end of environment variables

5510 <1>

5511 000097F7 89C7 <1> mov edi, eax

5512 000097F9 5A <1> pop edx ; \*\*\*\*\* ; byte count (after '=')

5513 000097FA 89D1 <1> mov ecx, edx

5514 000097FC 89DE <1> mov esi, ebx ; ASCIIZ string address (after '=')

5515 000097FE 89FB <1> mov ebx, edi

5516 <1>

5517 00009800 F3A4 <1> rep movsb

5518 <1>

5519 00009802 880F <1> mov [edi], cl ; 0 ; end of variable

5520 <1>

5521 00009804 0FB605[6C650100] <1> movzx eax, byte [env\_var\_length]

5522 0000980B 01C2 <1> add edx, eax ; variable length (total)

5523 0000980D F7D8 <1> neg eax

5524 0000980F 01D8 <1> add eax, ebx ; start address of the variable

5525 00009811 F8 <1> clc ; 13/04/2016

5526 00009812 E9EEFDFFFF <1> jmp set\_env\_string\_allocate\_envb\_retn ; OK !

5527 <1>

5528 <1> mainprog\_startup\_configuration:

5529 <1> ; 22/11/2017

5530 <1> ; 06/05/2016

5531 <1> ; 14/04/2016 (TRDOS 386 = TRDOS v2.0)

5532 <1> ; 17/09/2011 (TRDOS v1, MAINPROG.ASM)

5533 <1> ;

5534 <1> loc\_load\_mainprog\_cfg\_file:

5535 00009817 BE[190D0100] <1> mov esi, MainProgCfgFile

5536 0000981C 66B80018 <1> mov ax, 1800h ; Except volume label and dirs

5537 00009820 E83EEAFFFF <1> call find\_first\_file

5538 00009825 7256 <1> jc short loc\_load\_mainprog\_cfg\_exit

5539 <1>

5540 <1> ;or eax, eax

5541 <1> ;jz short loc\_load\_mainprog\_cfg\_exit

5542 <1>

5543 <1> loc\_start\_mainprog\_configuration:

5544 <1> ; ESI = FindFile\_DirEntry Location

5545 <1> ; EAX = File Size

5546 <1>

5547 00009827 A3[EC580100] <1> mov [MainProgCfg\_FileSize], eax

5548 <1>

5549 0000982C 668B5614 <1> mov dx, [esi+DirEntry\_FstClusHI]

5550 00009830 C1E210 <1> shl edx, 16

5551 00009833 668B561A <1> mov dx, [esi+DirEntry\_FstClusLO]

5552 00009837 8915[20650100] <1> mov [csftdf\_sf\_cluster], edx

5553 <1>

5554 0000983D 89C1 <1> mov ecx, eax

5555 0000983F 29C0 <1> sub eax, eax

5556 <1>

5557 <1> ; TRDOS 386 (TRDOS v2.0)

5558 <1> ; Allocate contiguous memory block for loading the file

5559 <1>

5560 <1> ; eax = 0 (Allocate memory from the beginning)

5561 <1> ; ecx = File (Allocation) size in bytes

5562 <1>

5563 00009841 E8DEBBFFFF <1> call allocate\_memory\_block

5564 00009846 7235 <1> jc short loc\_load\_mainprog\_cfg\_exit

5565 <1>

5566 00009848 A3[18650100] <1> mov [csftdf\_sf\_mem\_addr], eax ; loading address

5567 0000984D 890D[1C650100] <1> mov [csftdf\_sf\_mem\_bsize], ecx ; block size

5568 <1>

5569 00009853 31DB <1> xor ebx, ebx

5570 <1> ;mov [csftdf\_sf\_rbytes], ebx ; 0, reset

5571 <1>

5572 00009855 8A3D[FE580100] <1> mov bh, [Current\_Drv] ; [FindFile\_Drv]

5573 0000985B BE00010900 <1> mov esi, Logical\_DOSDisks

5574 00009860 01DE <1> add esi, ebx

5575 <1>

5576 00009862 8B1D[18650100] <1> mov ebx, [csftdf\_sf\_mem\_addr] ; memory block address

5577 <1>

5578 00009868 807E0300 <1> cmp byte [esi+LD\_FATType], 0

5579 0000986C 7710 <1> ja short loc\_mcfg\_load\_fat\_file

5580 <1>

5581 0000986E C705[28650100]0000- <1> mov dword [csftdf\_r\_size], 65536

5581 00009876 0100 <1>

5582 00009878 E9A1010000 <1> jmp loc\_mcfg\_load\_fs\_file

5583 <1>

5584 <1> loc\_load\_mainprog\_cfg\_exit:

5585 0000987D C3 <1> retn

5586 <1>

5587 <1> loc\_mcfg\_load\_fat\_file:

5588 0000987E 0FB74611 <1> movzx eax, word [esi+LD\_BPB+BytesPerSec]

5589 00009882 0FB64E13 <1> movzx ecx, byte [esi+LD\_BPB+SecPerClust]

5590 00009886 F7E1 <1> mul ecx

5591 00009888 A3[28650100] <1> mov [csftdf\_r\_size], eax

5592 <1>

5593 <1> loc\_mcfg\_load\_fat\_file\_next:

5594 0000988D E822010000 <1> call mcfg\_read\_fat\_file\_sectors

5595 00009892 0F8206010000 <1> jc mcfg\_deallocate\_mem

5596 <1>

5597 00009898 09D2 <1> or edx, edx ; edx > 0 -> EOF

5598 0000989A 74F1 <1> jz short loc\_mcfg\_load\_fat\_file\_next

5599 <1>

5600 <1> loc\_mcfg\_load\_fat\_file\_ok:

5601 <1> ; 06/05/2016

5602 0000989C C705[BC650100]- <1> mov dword [mainprog\_return\_addr], loc\_mcfg\_ci\_return\_addr

5602 000098A2 [5F990000] <1>

5603 <1> ;

5604 000098A6 8B35[18650100] <1> mov esi, [csftdf\_sf\_mem\_addr]

5605 000098AC 8935[F0580100] <1> mov [MainProgCfg\_LineOffset], esi

5606 <1>

5607 000098B2 A1[EC580100] <1> mov eax, [MainProgCfg\_FileSize]

5608 000098B7 89C2 <1> mov edx, eax

5609 000098B9 01F2 <1> add edx, esi

5610 <1>

5611 <1> loc\_mcfg\_process\_next\_line\_check:

5612 000098BB 89C1 <1> mov ecx, eax

5613 <1>

5614 000098BD 803E2A <1> cmp byte [esi], "\*" ; Remark sign

5615 000098C0 7503 <1> jne short loc\_mcfg\_process\_next\_line

5616 000098C2 46 <1> inc esi

5617 000098C3 EB17 <1> jmp short loc\_move\_mainprog\_cfg\_nl1

5618 <1>

5619 <1> loc\_mcfg\_process\_next\_line:

5620 000098C5 83F94F <1> cmp ecx, 79

5621 000098C8 7605 <1> jna short loc\_start\_mainprog\_cfg\_process

5622 <1>

5623 000098CA B94F000000 <1> mov ecx, 79

5624 <1>

5625 <1> loc\_start\_mainprog\_cfg\_process:

5626 000098CF BF[AE590100] <1> mov edi, CommandBuffer

5627 <1>

5628 <1> loc\_move\_mainprog\_cfg\_line:

5629 000098D4 AC <1> lodsb

5630 000098D5 3C20 <1> cmp al, 20h

5631 000098D7 720C <1> jb short loc\_move\_mainprog\_cfg\_nl2

5632 000098D9 AA <1> stosb

5633 000098DA E2F8 <1> loop loc\_move\_mainprog\_cfg\_line

5634 <1>

5635 <1> loc\_move\_mainprog\_cfg\_nl1:

5636 000098DC 39D6 <1> cmp esi, edx ; + configuration file size

5637 000098DE 7312 <1> jnb short loc\_end\_of\_mainprog\_cfg\_line

5638 000098E0 AC <1> lodsb

5639 000098E1 3C20 <1> cmp al, 20h

5640 000098E3 73F7 <1> jnb short loc\_move\_mainprog\_cfg\_nl1

5641 <1>

5642 <1> loc\_move\_mainprog\_cfg\_nl2:

5643 000098E5 39D6 <1> cmp esi, edx

5644 000098E7 7309 <1> jnb short loc\_end\_of\_mainprog\_cfg\_line

5645 000098E9 8A06 <1> mov al, [esi]

5646 000098EB 3C20 <1> cmp al, 20h

5647 000098ED 7703 <1> ja short loc\_end\_of\_mainprog\_cfg\_line

5648 000098EF 46 <1> inc esi

5649 000098F0 EBF3 <1> jmp short loc\_move\_mainprog\_cfg\_nl2

5650 <1>

5651 <1> loc\_end\_of\_mainprog\_cfg\_line:

5652 000098F2 C60700 <1> mov byte [edi], 0

5653 <1>

5654 000098F5 8935[F0580100] <1> mov [MainProgCfg\_LineOffset], esi

5655 <1>

5656 <1> ; 22/11/2017

5657 000098FB BE[B6590100] <1> mov esi, CommandBuffer + 8

5658 00009900 29FE <1> sub esi, edi

5659 00009902 7606 <1> jna short loc\_move\_mainprog\_cfg\_command

5660 00009904 30C0 <1> xor al, al

5661 <1> loc\_mainprog\_cfg\_clear\_chrs:

5662 00009906 AA <1> stosb

5663 00009907 4E <1> dec esi

5664 00009908 75FC <1> jnz short loc\_mainprog\_cfg\_clear\_chrs

5665 <1>

5666 <1> loc\_move\_mainprog\_cfg\_command:

5667 0000990A BE[AE590100] <1> mov esi, CommandBuffer

5668 0000990F 89F7 <1> mov edi, esi

5669 00009911 31DB <1> xor ebx, ebx

5670 <1> ;xor ecx, ecx

5671 00009913 30C9 <1> xor cl, cl

5672 <1>

5673 <1> loc\_move\_mcfg\_first\_cmd\_char:

5674 00009915 8A041E <1> mov al, [esi+ebx]

5675 00009918 FEC3 <1> inc bl

5676 0000991A 3C20 <1> cmp al, 20h

5677 0000991C 7712 <1> ja short loc\_move\_mcfg\_cmd\_capitalizing

5678 0000991E 7237 <1> jb short loc\_move\_mcfg\_cmd\_arguments\_ok

5679 00009920 80FB4F <1> cmp bl, 79

5680 00009923 72F0 <1> jb short loc\_move\_mcfg\_first\_cmd\_char

5681 00009925 EB30 <1> jmp short loc\_move\_mcfg\_cmd\_arguments\_ok

5682 <1>

5683 <1> loc\_move\_mcfg\_next\_cmd\_char:

5684 00009927 8A041E <1> mov al, [esi+ebx]

5685 0000992A FEC3 <1> inc bl

5686 0000992C 3C20 <1> cmp al, 20h

5687 0000992E 7614 <1> jna short loc\_move\_mcfg\_cmd\_ok

5688 <1>

5689 <1> loc\_move\_mcfg\_cmd\_capitalizing:

5690 00009930 3C61 <1> cmp al, 61h ; 'a'

5691 00009932 7206 <1> jb short loc\_move\_mcfg\_cmd\_caps\_ok

5692 00009934 3C7A <1> cmp al, 7Ah ; 'z'

5693 00009936 7702 <1> ja short loc\_move\_mcfg\_cmd\_caps\_ok

5694 00009938 24DF <1> and al, 0DFh ; sub al, 'a'-'A'

5695 <1>

5696 <1> loc\_move\_mcfg\_cmd\_caps\_ok:

5697 0000993A AA <1> stosb

5698 0000993B FEC1 <1> inc cl

5699 0000993D 80FB4F <1> cmp bl, 79

5700 00009940 72E5 <1> jb short loc\_move\_mcfg\_next\_cmd\_char

5701 00009942 EB13 <1> jmp short loc\_move\_mcfg\_cmd\_arguments\_ok

5702 <1>

5703 <1> loc\_move\_mcfg\_cmd\_ok:

5704 00009944 30C0 <1> xor al, al ; 0

5705 <1>

5706 <1> loc\_move\_mcfg\_cmd\_arguments:

5707 00009946 8807 <1> mov [edi], al

5708 00009948 47 <1> inc edi

5709 00009949 80FB4F <1> cmp bl, 79

5710 0000994C 7309 <1> jnb short loc\_move\_mcfg\_cmd\_arguments\_ok

5711 0000994E 8A041E <1> mov al, [esi+ebx]

5712 00009951 FEC3 <1> inc bl

5713 00009953 3C20 <1> cmp al, 20h

5714 00009955 73EF <1> jnb short loc\_move\_mcfg\_cmd\_arguments

5715 <1>

5716 <1> loc\_move\_mcfg\_cmd\_arguments\_ok:

5717 00009957 C60700 <1> mov byte [edi], 0

5718 <1>

5719 <1> loc\_mcfg\_process\_cmd\_interpreter:

5720 0000995A E825E0FFFF <1> call command\_interpreter

5721 <1>

5722 <1> loc\_mcfg\_ci\_return\_addr:

5723 0000995F A1[EC580100] <1> mov eax, [MainProgCfg\_FileSize]

5724 00009964 89C2 <1> mov edx, eax

5725 00009966 8B35[F0580100] <1> mov esi, [MainProgCfg\_LineOffset]

5726 0000996C 01F2 <1> add edx, esi

5727 0000996E 0305[18650100] <1> add eax, [csftdf\_sf\_mem\_addr]

5728 00009974 29F0 <1> sub eax, esi

5729 00009976 0F873FFFFFFF <1> ja loc\_mcfg\_process\_next\_line\_check

5730 <1>

5731 0000997C E81D000000 <1> call mcfg\_deallocate\_mem

5732 <1>

5733 00009981 B94F000000 <1> mov ecx, 79 ; 80 ?

5734 00009986 BF[AE590100] <1> mov edi, CommandBuffer

5735 0000998B 30C0 <1> xor al, al

5736 0000998D F3AA <1> rep stosb

5737 <1>

5738 <1> ; 06/05/2016

5739 0000998F BE[6F190100] <1> mov esi, nextline

5740 00009994 E8C4C9FFFF <1> call print\_msg

5741 00009999 E963D6FFFF <1> jmp dos\_prompt

5742 <1>

5743 <1> mcfg\_deallocate\_mem:

5744 0000999E A1[18650100] <1> mov eax, [csftdf\_sf\_mem\_addr] ; start address

5745 000099A3 8B0D[1C650100] <1> mov ecx, [csftdf\_sf\_mem\_bsize] ; block size

5746 <1> ;call deallocate\_memory\_block

5747 <1> ;retn

5748 000099A9 E983BCFFFF <1> jmp deallocate\_memory\_block

5749 <1>

5750 <1> mcfg\_read\_file\_sectors:

5751 <1> ; 14/04/2016

5752 000099AE 807E0300 <1> cmp byte [esi+LD\_FATType], 0

5753 000099B2 7669 <1> jna short mcfg\_read\_fs\_file\_sectors

5754 <1>

5755 <1> mcfg\_read\_fat\_file\_sectors:

5756 <1> ; return:

5757 <1> ; CF = 0 & EDX > 0 -> END OF FILE

5758 <1> ; CF = 0 & EDX = 0 -> not EOF

5759 <1> ; CF = 1 -> read error (error code in AL)

5760 <1>

5761 <1> mcfg\_read\_fat\_file\_secs\_0:

5762 000099B4 8B15[EC580100] <1> mov edx, [MainProgCfg\_FileSize]

5763 000099BA 2B15[30650100] <1> sub edx, [csftdf\_sf\_rbytes]

5764 000099C0 3B15[28650100] <1> cmp edx, [csftdf\_r\_size]

5765 000099C6 7306 <1> jnb short mcfg\_read\_fat\_file\_secs\_1

5766 000099C8 8915[28650100] <1> mov [csftdf\_r\_size], edx

5767 <1>

5768 <1> mcfg\_read\_fat\_file\_secs\_1:

5769 000099CE A1[28650100] <1> mov eax, [csftdf\_r\_size]

5770 000099D3 29D2 <1> sub edx, edx

5771 000099D5 0FB74E11 <1> movzx ecx, word [esi+LD\_BPB+BytesPerSec]

5772 000099D9 01C8 <1> add eax, ecx

5773 000099DB 48 <1> dec eax

5774 000099DC F7F1 <1> div ecx

5775 000099DE 89C1 <1> mov ecx, eax ; sector count

5776 000099E0 A1[20650100] <1> mov eax, [csftdf\_sf\_cluster]

5777 <1>

5778 <1> ; EBX = memory block address (current)

5779 <1>

5780 000099E5 E88C230000 <1> call read\_fat\_file\_sectors

5781 000099EA 7230 <1> jc short mcfg\_read\_fat\_file\_secs\_3

5782 <1>

5783 <1> ; EBX = next memory address

5784 <1>

5785 000099EC A1[30650100] <1> mov eax, [csftdf\_sf\_rbytes]

5786 000099F1 0305[28650100] <1> add eax, [csftdf\_r\_size]

5787 000099F7 8B15[EC580100] <1> mov edx, [MainProgCfg\_FileSize]

5788 000099FD 39D0 <1> cmp eax, edx

5789 000099FF 731B <1> jnb short mcfg\_read\_fat\_file\_secs\_3 ; edx > 0

5790 00009A01 A3[30650100] <1> mov [csftdf\_sf\_rbytes], eax

5791 <1>

5792 00009A06 53 <1> push ebx ; \*

5793 <1> ; get next cluster (csftdf\_r\_size! bytes)

5794 00009A07 A1[20650100] <1> mov eax, [csftdf\_sf\_cluster]

5795 00009A0C E837210000 <1> call get\_next\_cluster

5796 00009A11 5B <1> pop ebx ; \*

5797 00009A12 7301 <1> jnc short mcfg\_read\_fat\_file\_secs\_2

5798 <1>

5799 <1> ;mov eax, 17; Read error !

5800 00009A14 C3 <1> retn

5801 <1>

5802 <1> mcfg\_read\_fat\_file\_secs\_2:

5803 00009A15 29D2 <1> sub edx, edx ; 0

5804 00009A17 A3[20650100] <1> mov [csftdf\_sf\_cluster], eax ; next cluster

5805 <1>

5806 <1> mcfg\_read\_fat\_file\_secs\_3:

5807 00009A1C C3 <1> retn

5808 <1>

5809 <1> mcfg\_read\_fs\_file\_sectors:

5810 00009A1D C3 <1> retn

5811 <1>

5812 <1> loc\_mcfg\_load\_fs\_file:

5813 00009A1E C3 <1> retn

5814 <1>

5815 <1> load\_and\_execute\_file:

5816 <1> ; 04/01/2017

5817 <1> ; 06/05/2016, 07/05/2016, 11/05/2016

5818 <1> ; 23/04/2016, 24/04/2016

5819 <1> ; 22/04/2016 (TRDOS 386 = TRDOS v2.0)

5820 <1> ; 05/11/2011

5821 <1> ; (TRDOS v1, CMDINTR.ASM, 'cmp\_cmd\_run', 'cmp\_cmd\_external')

5822 <1> ; ('loc\_run\_check\_filename')

5823 <1> ; 29/08/2011

5824 <1> ; 10/09/2011

5825 <1> ; INPUT->

5826 <1> ; ESI = Path Name address (CommandBuffer address)

5827 <1> ; OUTPUT ->

5828 <1> ; none (error message will be shown if an error will occur)

5829 <1> ;

5830 <1> ; (EAX, EBX, ECX, EDX, ESI, EDI, EBP will be changed)

5831 <1> ;

5832 <1> loc\_run\_check\_filename:

5833 00009A1F 803E20 <1> cmp byte [esi], 20h

5834 00009A22 0F822BE3FFFF <1> jb loc\_cmd\_failed

5835 00009A28 7703 <1> ja short loc\_run\_check\_filename\_ok

5836 00009A2A 46 <1> inc esi

5837 00009A2B EBF2 <1> jmp short loc\_run\_check\_filename

5838 <1>

5839 <1> loc\_run\_check\_filename\_ok:

5840 00009A2D C605[5F590100]00 <1> mov byte [CmdArgStart], 0 ; reset

5841 00009A34 56 <1> push esi ; \*

5842 <1> loc\_run\_get\_first\_arg\_pos:

5843 00009A35 46 <1> inc esi

5844 00009A36 8A06 <1> mov al, [esi]

5845 00009A38 3C20 <1> cmp al, 20h

5846 00009A3A 77F9 <1> ja short loc\_run\_get\_first\_arg\_pos

5847 00009A3C C60600 <1> mov byte [esi], 0

5848 <1> loc\_run\_get\_external\_arg\_pos:

5849 <1> ; 11/05/2016

5850 00009A3F 46 <1> inc esi

5851 00009A40 8A06 <1> mov al, [esi]

5852 00009A42 3C20 <1> cmp al, 20h

5853 00009A44 760C <1> jna short loc\_run\_parse\_path\_name

5854 00009A46 89F0 <1> mov eax, esi

5855 00009A48 2D[AE590100] <1> sub eax, CommandBuffer

5856 00009A4D A2[5F590100] <1> mov byte [CmdArgStart], al

5857 <1> loc\_run\_parse\_path\_name:

5858 00009A52 5E <1> pop esi ; \*

5859 00009A53 BF[A2620100] <1> mov edi, FindFile\_Drv

5860 00009A58 E8D7090000 <1> call parse\_path\_name

5861 00009A5D 0F82F0E2FFFF <1> jc loc\_cmd\_failed

5862 <1>

5863 <1> loc\_run\_check\_filename\_exists:

5864 00009A63 BE[E4620100] <1> mov esi, FindFile\_Name

5865 00009A68 803E20 <1> cmp byte [esi], 20h

5866 00009A6B 0F86E2E2FFFF <1> jna loc\_cmd\_failed

5867 <1>

5868 <1> loc\_run\_check\_exe\_filename\_ext:

5869 00009A71 E890020000 <1> call check\_prg\_filename\_ext

5870 00009A76 0F82D7E2FFFF <1> jc loc\_cmd\_failed

5871 <1>

5872 <1> loc\_run\_check\_exe\_filename\_ext\_ok:

5873 00009A7C 66A3[BA650100] <1> mov word [EXE\_ID], ax

5874 <1>

5875 <1> loc\_run\_drv:

5876 00009A82 C605[B9650100]00 <1> mov byte [Run\_Manual\_Path], 0

5877 00009A89 A1[F8580100] <1> mov eax, [Current\_Dir\_FCluster]

5878 00009A8E A3[B4650100] <1> mov [Run\_CDirFC], eax

5879 <1> ;

5880 00009A93 8A35[FE580100] <1> mov dh, [Current\_Drv]

5881 00009A99 8835[5E610100] <1> mov [RUN\_CDRV], dh

5882 <1>

5883 00009A9F 8A15[A2620100] <1> mov dl, [FindFile\_Drv]

5884 00009AA5 38F2 <1> cmp dl, dh

5885 00009AA7 7412 <1> je short loc\_run\_change\_directory

5886 <1>

5887 00009AA9 8005[B9650100]02 <1> add byte [Run\_Manual\_Path], 2

5888 <1>

5889 00009AB0 E80BD4FFFF <1> call change\_current\_drive

5890 00009AB5 0F82C3E2FFFF <1> jc loc\_run\_cmd\_failed

5891 <1>

5892 <1> loc\_run\_change\_directory:

5893 00009ABB 803D[A3620100]20 <1> cmp byte [FindFile\_Directory], 20h

5894 00009AC2 7623 <1> jna short loc\_run\_find\_executable\_file

5895 <1>

5896 00009AC4 FE05[B9650100] <1> inc byte [Run\_Manual\_Path]

5897 <1>

5898 00009ACA FE05[D30C0100] <1> inc byte [Restore\_CDIR]

5899 <1>

5900 00009AD0 BE[A3620100] <1> mov esi, FindFile\_Directory

5901 00009AD5 30E4 <1> xor ah, ah ; CD\_COMMAND sign -> 0

5902 00009AD7 E842030000 <1> call change\_current\_directory

5903 00009ADC 0F829CE2FFFF <1> jc loc\_run\_cmd\_failed

5904 <1>

5905 <1> loc\_run\_change\_prompt\_dir\_string:

5906 00009AE2 E857020000 <1> call change\_prompt\_dir\_string

5907 <1>

5908 <1> loc\_run\_find\_executable\_file:

5909 00009AE7 66C705[B8650100]00- <1> mov word [Run\_Auto\_Path], 0

5909 00009AEF 00 <1>

5910 <1>

5911 <1> loc\_run\_find\_executable\_file\_next:

5912 00009AF0 BE[E4620100] <1> mov esi, FindFile\_Name

5913 <1> loc\_run\_find\_program\_file\_next:

5914 00009AF5 66B80018 <1> mov ax, 1800h ; Except volume label and dirs

5915 00009AF9 E865E7FFFF <1> call find\_first\_file

5916 <1> ; ESI = Directory Entry (FindFile\_DirEntry) Location

5917 <1> ; EDI = Directory Buffer Directory Entry Location

5918 <1> ; EAX = File size

5919 00009AFE 0F835C010000 <1> jnc loc\_load\_and\_run\_file

5920 <1>

5921 00009B04 3C02 <1> cmp al, 2 ; file not found

5922 00009B06 0F8572E2FFFF <1> jne loc\_run\_cmd\_failed

5923 <1>

5924 00009B0C 66A1[BA650100] <1> mov ax, word [EXE\_ID]

5925 00009B12 80FC2E <1> cmp ah, '.' ; File name has extension sign

5926 00009B15 7424 <1> je short loc\_run\_check\_auto\_path

5927 <1>

5928 00009B17 08C0 <1> or al, al

5929 00009B19 7520 <1> jnz short loc\_run\_check\_auto\_path

5930 <1>

5931 00009B1B 80FC08 <1> cmp ah, 8 ; count of file name chars

5932 00009B1E 771B <1> ja short loc\_run\_check\_auto\_path

5933 <1>

5934 <1> loc\_run\_change\_file\_ext\_to\_prg:

5935 00009B20 0FB6DC <1> movzx ebx, ah ; count of file name chars

5936 00009B23 BE[E4620100] <1> mov esi, FindFile\_Name

5937 00009B28 01F3 <1> add ebx, esi

5938 <1> ; 07/05/2016

5939 00009B2A C7032E505247 <1> mov dword [ebx], '.PRG'

5940 00009B30 66C705[BA650100]50- <1> mov word [EXE\_ID], 'P.'

5940 00009B38 2E <1>

5941 00009B39 EBBA <1> jmp short loc\_run\_find\_program\_file\_next

5942 <1>

5943 <1> loc\_run\_check\_auto\_path:

5944 <1> ; NOTE: /// 07/05/2016 ///

5945 <1> ; If the path is given, value of byte [Run\_Manual\_Path]

5946 <1> ; will not be ZERO. If so, file searching by using

5947 <1> ; Automatic Path (via 'PATH' environment variable)

5948 <1> ; will not be applicable, because the program file

5949 <1> ; is already/absolutely not found.

5950 <1>

5951 00009B3B A0[B9650100] <1> mov al, [Run\_Manual\_Path]

5952 00009B40 08C0 <1> or al, al

5953 00009B42 0F850BE2FFFF <1> jnz loc\_cmd\_failed

5954 <1>

5955 <1> loc\_run\_check\_auto\_path\_again:

5956 00009B48 66833D[B8650100]FF <1> cmp word [Run\_Auto\_Path], 0FFFFh

5957 <1> ; 0FFFFh = Not a valid run path (in ENV block)

5958 00009B50 0F83FDE1FFFF <1> jnb loc\_cmd\_failed

5959 <1> ; xor al, al

5960 00009B56 BE[9F0D0100] <1> mov esi, Cmd\_Path ; 'PATH'

5961 00009B5B BF[FE590100] <1> mov edi, TextBuffer

5962 00009B60 E848F9FFFF <1> call get\_environment\_string

5963 00009B65 730E <1> jnc short loc\_run\_chk\_filename\_ext\_again

5964 00009B67 66C705[B8650100]FF- <1> mov word [Run\_Auto\_Path], 0FFFFh ; invalid

5964 00009B6F FF <1>

5965 00009B70 E9DEE1FFFF <1> jmp loc\_cmd\_failed

5966 <1>

5967 <1> loc\_run\_chk\_filename\_ext\_again:

5968 00009B75 89C1 <1> mov ecx, eax ; string length (with zero tail)

5969 00009B77 49 <1> dec ecx ; without zero tail

5970 00009B78 66A1[BA650100] <1> mov ax, [EXE\_ID]

5971 00009B7E 80FC2E <1> cmp ah, '.'

5972 00009B81 740E <1> je short loc\_run\_chk\_auto\_path\_pos

5973 <1>

5974 <1> loc\_run\_change\_file\_ext\_to\_noext\_again:

5975 00009B83 0FB6DC <1> movzx ebx, ah

5976 00009B86 BE[E4620100] <1> mov esi, FindFile\_Name

5977 00009B8B 01F3 <1> add ebx, esi

5978 00009B8D 29C0 <1> sub eax, eax

5979 00009B8F 8903 <1> mov [ebx], eax ; 0 ; erase extension (.PRG)

5980 <1>

5981 <1> loc\_run\_chk\_auto\_path\_pos:

5982 <1> ;movzx eax, word [Run\_Auto\_Path]

5983 00009B91 66A1[B8650100] <1> mov ax, [Run\_Auto\_Path]

5984 00009B97 39C8 <1> cmp eax, ecx ; ecx = string length (except zero tail)

5985 00009B99 0F83B4E1FFFF <1> jnb loc\_cmd\_failed

5986 <1> ;or eax, eax

5987 00009B9F 6609C0 <1> or ax, ax

5988 00009BA2 7502 <1> jnz short loc\_run\_auto\_path\_pos\_move

5989 00009BA4 B005 <1> mov al, 5

5990 <1>

5991 <1> loc\_run\_auto\_path\_pos\_move:

5992 00009BA6 89FE <1> mov esi, edi ; offset TextBuffer

5993 00009BA8 01C6 <1> add esi, eax

5994 <1>

5995 <1> loc\_run\_auto\_path\_pos\_space\_loop:

5996 00009BAA AC <1> lodsb

5997 00009BAB 3C20 <1> cmp al, 20h

5998 00009BAD 74FB <1> je short loc\_run\_auto\_path\_pos\_space\_loop

5999 00009BAF 0F829EE1FFFF <1> jb loc\_cmd\_failed

6000 00009BB5 AA <1> stosb

6001 <1> loc\_run\_auto\_path\_pos\_move\_next:

6002 00009BB6 AC <1> lodsb

6003 00009BB7 3C3B <1> cmp al, ';'

6004 00009BB9 7414 <1> je short loc\_run\_auto\_path\_pos\_move\_last\_byte

6005 00009BBB 3C20 <1> cmp al, 20h

6006 00009BBD 74F7 <1> je short loc\_run\_auto\_path\_pos\_move\_next

6007 00009BBF 7203 <1> jb short loc\_byte\_ptr\_end\_of\_path

6008 00009BC1 AA <1> stosb

6009 00009BC2 EBF2 <1> jmp short loc\_run\_auto\_path\_pos\_move\_next

6010 <1>

6011 <1> loc\_byte\_ptr\_end\_of\_path:

6012 00009BC4 66C705[B8650100]FF- <1> mov word [Run\_Auto\_Path], 0FFFFh ; end of path

6012 00009BCC FF <1>

6013 00009BCD EB0D <1> jmp short loc\_run\_auto\_path\_move\_ok

6014 <1>

6015 <1> loc\_run\_auto\_path\_pos\_move\_last\_byte:

6016 00009BCF 89F0 <1> mov eax, esi

6017 00009BD1 2D[FE590100] <1> sub eax, TextBuffer

6018 00009BD6 66A3[B8650100] <1> mov [Run\_Auto\_Path], ax ; next path position

6019 <1>

6020 <1> loc\_run\_auto\_path\_move\_ok:

6021 00009BDC 4F <1> dec edi

6022 00009BDD B02F <1> mov al, '/'

6023 00009BDF 3807 <1> cmp [edi], al

6024 00009BE1 7403 <1> je short loc\_run\_auto\_path\_move\_file\_name

6025 00009BE3 47 <1> inc edi

6026 00009BE4 8807 <1> mov [edi], al

6027 <1>

6028 <1> loc\_run\_auto\_path\_move\_file\_name:

6029 00009BE6 47 <1> inc edi

6030 00009BE7 BE[E4620100] <1> mov esi, FindFile\_Name

6031 <1>

6032 <1> loc\_run\_auto\_path\_move\_fn\_loop:

6033 00009BEC AC <1> lodsb

6034 00009BED AA <1> stosb

6035 00009BEE 08C0 <1> or al, al

6036 00009BF0 75FA <1> jnz short loc\_run\_auto\_path\_move\_fn\_loop

6037 <1>

6038 00009BF2 BE[FE590100] <1> mov esi, TextBuffer

6039 00009BF7 BF[A2620100] <1> mov edi, FindFile\_Drv

6040 00009BFC E833080000 <1> call parse\_path\_name

6041 00009C01 0F824CE1FFFF <1> jc loc\_cmd\_failed

6042 <1>

6043 00009C07 8A35[FE580100] <1> mov dh, [Current\_Drv]

6044 00009C0D 8A15[A2620100] <1> mov dl, [FindFile\_Drv]

6045 00009C13 38F2 <1> cmp dl, dh

6046 00009C15 740B <1> je short loc\_run\_change\_directory\_again

6047 <1>

6048 00009C17 E8A4D2FFFF <1> call change\_current\_drive

6049 00009C1C 0F825CE1FFFF <1> jc loc\_run\_cmd\_failed

6050 <1>

6051 <1> loc\_run\_change\_directory\_again:

6052 00009C22 803D[A3620100]20 <1> cmp byte [FindFile\_Directory], 20h

6053 00009C29 761D <1> jna short loc\_load\_executable\_cdir\_chk\_again

6054 <1>

6055 00009C2B FE05[D30C0100] <1> inc byte [Restore\_CDIR]

6056 00009C31 BE[A3620100] <1> mov esi, FindFile\_Directory

6057 00009C36 30E4 <1> xor ah, ah ; CD\_COMMAND sign -> 0

6058 00009C38 E8E1010000 <1> call change\_current\_directory

6059 00009C3D 0F823BE1FFFF <1> jc loc\_run\_cmd\_failed

6060 <1>

6061 <1> loc\_run\_chg\_prompt\_dir\_str\_again:

6062 00009C43 E8F6000000 <1> call change\_prompt\_dir\_string

6063 <1>

6064 <1> loc\_load\_executable\_cdir\_chk\_again:

6065 00009C48 A1[F8580100] <1> mov eax, [Current\_Dir\_FCluster]

6066 00009C4D 3B05[B4650100] <1> cmp eax, [Run\_CDirFC]

6067 00009C53 0F8597FEFFFF <1> jne loc\_run\_find\_executable\_file\_next

6068 00009C59 30C0 <1> xor al, al ; 0

6069 00009C5B E9E8FEFFFF <1> jmp loc\_run\_check\_auto\_path\_again

6070 <1>

6071 <1> loc\_load\_and\_run\_file:

6072 <1> ; 13/11/2017

6073 <1> ; 04/01/2017

6074 <1> ; 23/04/2016

6075 00009C60 BE[E4620100] <1> mov esi, FindFile\_Name

6076 00009C65 BF[FE590100] <1> mov edi, TextBuffer

6077 <1>

6078 <1> ; 24/04/2016

6079 00009C6A 31D2 <1> xor edx, edx

6080 00009C6C 668915[4A040300] <1> mov word [argc], dx ; 0

6081 00009C73 8915[8C030300] <1> mov dword [u.nread], edx ; 0

6082 <1>

6083 <1> loc\_load\_and\_run\_file\_1:

6084 00009C79 AC <1> lodsb

6085 00009C7A AA <1> stosb

6086 00009C7B FF05[8C030300] <1> inc dword [u.nread]

6087 00009C81 20C0 <1> and al, al

6088 00009C83 75F4 <1> jnz short loc\_load\_and\_run\_file\_1

6089 <1>

6090 00009C85 A0[5F590100] <1> mov al, [CmdArgStart]

6091 00009C8A 20C0 <1> and al, al

6092 00009C8C 7445 <1> jz short loc\_load\_and\_run\_file\_7

6093 <1>

6094 00009C8E 0FB6F0 <1> movzx esi, al ; 11/05/2016

6095 00009C91 B950000000 <1> mov ecx, 80

6096 00009C96 29F1 <1> sub ecx, esi

6097 00009C98 81C6[AE590100] <1> add esi, CommandBuffer

6098 <1>

6099 00009C9E 66FF05[4A040300] <1> inc word [argc] ; 11/05/2016

6100 <1>

6101 <1> loc\_load\_and\_run\_file\_2:

6102 00009CA5 AC <1> lodsb

6103 00009CA6 3C20 <1> cmp al, 20h

6104 00009CA8 7717 <1> ja short loc\_load\_and\_run\_file\_5

6105 00009CAA 721E <1> jb short loc\_load\_and\_run\_file\_6

6106 <1>

6107 <1> loc\_load\_and\_run\_file\_3:

6108 00009CAC 803E20 <1> cmp byte [esi], 20h

6109 00009CAF 7707 <1> ja short loc\_load\_and\_run\_file\_4

6110 00009CB1 7217 <1> jb short loc\_load\_and\_run\_file\_6

6111 00009CB3 46 <1> inc esi

6112 00009CB4 E2F6 <1> loop loc\_load\_and\_run\_file\_3

6113 00009CB6 EB12 <1> jmp short loc\_load\_and\_run\_file\_6

6114 <1>

6115 <1> loc\_load\_and\_run\_file\_4:

6116 00009CB8 28C0 <1> sub al, al ; 0

6117 00009CBA 66FF05[4A040300] <1> inc word [argc]

6118 <1> loc\_load\_and\_run\_file\_5:

6119 00009CC1 AA <1> stosb

6120 00009CC2 FF05[8C030300] <1> inc dword [u.nread]

6121 00009CC8 E2DB <1> loop loc\_load\_and\_run\_file\_2

6122 <1>

6123 <1> loc\_load\_and\_run\_file\_6:

6124 00009CCA 30C0 <1> xor al, al ; 0

6125 00009CCC AA <1> stosb

6126 00009CCD FF05[8C030300] <1> inc dword [u.nread]

6127 <1> loc\_load\_and\_run\_file\_7:

6128 00009CD3 8807 <1> mov [edi], al ; 0

6129 00009CD5 66FF05[4A040300] <1> inc word [argc] ; 24/04/2016

6130 00009CDC FF05[8C030300] <1> inc dword [u.nread] ; 24/04/2016

6131 00009CE2 BE[FE590100] <1> mov esi, TextBuffer

6132 00009CE7 8B15[10630100] <1> mov edx, [FindFile\_DirEntry+DirEntry\_FileSize]

6133 00009CED 66A1[08630100] <1> mov ax, [FindFile\_DirEntry+DirEntry\_FstClusHI]

6134 00009CF3 C1E010 <1> shl eax, 16 ; 13/11/2017

6135 00009CF6 66A1[0E630100] <1> mov ax, [FindFile\_DirEntry+DirEntry\_FstClusLO]

6136 <1> ; EAX = First Cluster number

6137 <1> ; EDX = File Size

6138 <1> ; ESI = Argument list address

6139 <1> ; [argc] = argument count

6140 <1> ; [u.nread] = argument list length

6141 00009CFC E89D420000 <1> call load\_and\_run\_file ; trdosk6.s

6142 <1> ;jc loc\_run\_cmd\_failed ; 04/01/2017

6143 <1> loc\_load\_and\_run\_file\_8: ; 06/05/2016

6144 00009D01 E98BE9FFFF <1> jmp loc\_file\_rw\_restore\_retn

6145 <1>

6146 <1> check\_prg\_filename\_ext:

6147 <1> ; 23/04/2016 (TRDOS 386 = TRDOS v2.0)

6148 <1> ; 10/09/2011

6149 <1> ; (TRDOS v1, CMDINTR.ASM, 'proc\_check\_exe\_filename\_ext')

6150 <1> ; 14/11/2009

6151 <1> ; INPUT ->

6152 <1> ; ESI = Dot File Name

6153 <1> ; OUTPUT ->

6154 <1> ; cf = 0 -> EXE\_ID in AL

6155 <1> ; ESI = Last char + 1 position

6156 <1> ; cf = 1 -> Invalid executable file name

6157 <1> ; or no file name extension if AH<=8

6158 <1> ; AL = Last file name char

6159 <1> ; cf = 0 -> AL='P' (PRG), AL=0 (no extension)

6160 <1> ;

6161 <1> ; (Modified registers: EAX, ESI)

6162 <1>

6163 00009D06 30E4 <1> xor ah, ah

6164 <1> loc\_run\_check\_filename\_ext:

6165 00009D08 AC <1> lodsb

6166 00009D09 3C21 <1> cmp al, 21h

6167 00009D0B 7229 <1> jb short loc\_check\_exe\_fn\_retn

6168 00009D0D FEC4 <1> inc ah

6169 00009D0F 3C2E <1> cmp al, '.'

6170 00009D11 75F5 <1> jne short loc\_run\_check\_filename\_ext

6171 <1>

6172 <1> loc\_run\_check\_filename\_ext\_dot:

6173 00009D13 80FC02 <1> cmp ah, 2 ; .??? is not valid

6174 00009D16 88C4 <1> mov ah, al ; '.'

6175 00009D18 7219 <1> jb short loc\_check\_prg\_fn\_retn

6176 <1>

6177 <1> loc\_run\_check\_filename\_ext\_dot\_ok:

6178 00009D1A AC <1> lodsb

6179 00009D1B 24DF <1> and al, 0DFh

6180 <1>

6181 <1> loc\_run\_check\_filename\_ext\_prg:

6182 00009D1D 3C50 <1> cmp al, 'P'

6183 00009D1F 7212 <1> jb short loc\_check\_prg\_fn\_retn

6184 00009D21 7711 <1> ja short loc\_check\_prg\_fn\_stc

6185 00009D23 AC <1> lodsb

6186 00009D24 24DF <1> and al, 0DFh

6187 00009D26 3C52 <1> cmp al, 'R'

6188 00009D28 750A <1> jne short loc\_check\_prg\_fn\_stc

6189 00009D2A AC <1> lodsb

6190 00009D2B 24DF <1> and al, 0DFh

6191 00009D2D 3C47 <1> cmp al, 'G'

6192 00009D2F 7503 <1> jne short loc\_check\_prg\_fn\_stc

6193 <1>

6194 00009D31 B050 <1> mov al, 'P'

6195 <1> loc\_check\_prg\_fn\_retn:

6196 00009D33 C3 <1> retn

6197 <1>

6198 <1> loc\_check\_prg\_fn\_stc:

6199 00009D34 F9 <1> stc

6200 00009D35 C3 <1> retn

6201 <1>

6202 <1> loc\_check\_exe\_fn\_retn:

6203 00009D36 28C0 <1> sub al, al ; 0

6204 00009D38 C3 <1> retn

6205 <1>

6206 <1> find\_and\_list\_files:

6207 00009D39 C3 <1> retn

6208 <1> set\_exec\_arguments:

6209 00009D3A C3 <1> retn

6210 <1> delete\_fs\_directory:

6211 00009D3B 31C0 <1> xor eax, eax

6212 00009D3D C3 <1> retn

2308 %include 'trdosk4.s' ; 24/01/2016

1 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2 <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.0) - Directory Functions : trdosk4.s

3 <1> ; ----------------------------------------------------------------------------

4 <1> ; Last Update: 29/12/2017

5 <1> ; ----------------------------------------------------------------------------

6 <1> ; Beginning: 24/01/2016

7 <1> ; ----------------------------------------------------------------------------

8 <1> ; Assembler: NASM version 2.11 (trdos386.s)

9 <1> ; ----------------------------------------------------------------------------

10 <1> ; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan

11 <1> ; DIR.ASM (09/10/2011)

12 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

13 <1>

14 <1> ; DIR.ASM [ TRDOS KERNEL - COMMAND EXECUTER SECTION - DIRECTORY FUNCTIONS ]

15 <1> ; (c) 2004-2010 Erdogan TAN [ 17/01/2004 ] Last Update: 09/10/2011

16 <1> ; FILE.ASM [ FILE FUNCTIONS ] Last Update: 09/10/2011

17 <1>

18 <1> change\_prompt\_dir\_string:

19 <1> ; 05/10/2016

20 <1> ; 24/01/2016 (TRDOS 386 = TRDOS v2.0)

21 <1> ; 27/03/2011

22 <1> ; 09/10/2009

23 <1> ; INPUT/OUTPUT => none

24 <1> ; this procedure changes current directory string/text

25 <1> ; 2005

26 <1>

27 00009D3E BE[5F610100] <1> mov esi, PATH\_Array

28 <1> change\_prompt\_dir\_str: ; 05/10/2016 (call from 'set\_working\_path')

29 00009D43 BF[02590100] <1> mov edi, Current\_Directory

30 00009D48 8A25[FC580100] <1> mov ah, [Current\_Dir\_Level]

31 00009D4E E807000000 <1> call set\_current\_directory\_string

32 00009D53 880D[5D590100] <1> mov [Current\_Dir\_StrLen], cl

33 <1>

34 00009D59 C3 <1> retn

35 <1>

36 <1> set\_current\_directory\_string:

37 <1> ; 24/01/2016 (TRDOS 386 = TRDOS v2.0)

38 <1> ; 27/03/2011

39 <1> ; 09/10/2009

40 <1> ; INPUT:

41 <1> ; ESI = Path Array Address

42 <1> ; EDI = Current Directory String Buffer

43 <1> ; AH = Current Directory Level

44 <1> ; OUTPUT => EAX, EBX, ESI will be changed

45 <1> ; EDI will be same with input

46 <1> ; ECX = Current Directory String Length

47 <1>

48 00009D5A 57 <1> push edi

49 00009D5B 80FC00 <1> cmp ah, 0

50 00009D5E 7652 <1> jna short pass\_write\_path

51 00009D60 83C610 <1> add esi, 16

52 00009D63 89F3 <1> mov ebx, esi

53 <1> loc\_write\_path:

54 00009D65 B908000000 <1> mov ecx, 8

55 <1> path\_write\_dirname1:

56 00009D6A AC <1> lodsb

57 00009D6B 3C20 <1> cmp al, 20h

58 00009D6D 7612 <1> jna short pass\_write\_dirname1

59 00009D6F AA <1> stosb

60 00009D70 81FF[5C590100] <1> cmp edi, End\_Of\_Current\_Dir\_Str

61 00009D76 733A <1> jnb short pass\_write\_path

62 00009D78 E2F0 <1> loop path\_write\_dirname1

63 00009D7A 803E20 <1> cmp byte [esi], 20h

64 00009D7D 7624 <1> jna short pass\_write\_dirname2

65 00009D7F EB0A <1> jmp short loc\_put\_dot\_cont\_ext

66 <1> pass\_write\_dirname1:

67 00009D81 89DE <1> mov esi, ebx

68 00009D83 83C608 <1> add esi, 8

69 00009D86 803E20 <1> cmp byte [esi], 20h

70 00009D89 7618 <1> jna short pass\_write\_dirname2

71 <1> loc\_put\_dot\_cont\_ext:

72 00009D8B C6072E <1> mov byte [edi], "."

73 <1> ;mov ecx, 3

74 00009D8E B103 <1> mov cl, 3

75 <1> loc\_check\_dir\_name\_ext:

76 00009D90 AC <1> lodsb

77 00009D91 47 <1> inc edi

78 00009D92 3C20 <1> cmp al, 20h

79 00009D94 760D <1> jna short pass\_write\_dirname2

80 00009D96 8807 <1> mov [edi], al

81 00009D98 81FF[5C590100] <1> cmp edi, End\_Of\_Current\_Dir\_Str

82 00009D9E 7312 <1> jnb short pass\_write\_path

83 00009DA0 E2EE <1> loop loc\_check\_dir\_name\_ext

84 00009DA2 47 <1> inc edi

85 <1> pass\_write\_dirname2:

86 00009DA3 FECC <1> dec ah

87 00009DA5 740B <1> jz short pass\_write\_path

88 00009DA7 83C310 <1> add ebx, 16

89 00009DAA 89DE <1> mov esi, ebx

90 00009DAC C6072F <1> mov byte [edi],"/"

91 00009DAF 47 <1> inc edi

92 00009DB0 EBB3 <1> jmp short loc\_write\_path

93 <1> pass\_write\_path:

94 00009DB2 C60700 <1> mov byte [edi], 0

95 00009DB5 47 <1> inc edi

96 00009DB6 89F9 <1> mov ecx, edi

97 00009DB8 5F <1> pop edi

98 00009DB9 29F9 <1> sub ecx, edi

99 <1> ; ECX = Current Directory String Length

100 00009DBB C3 <1> retn

101 <1>

102 <1> get\_current\_directory:

103 <1> ; 15/10/2016

104 <1> ; 14/02/2016

105 <1> ; 24/01/2016 (TRDOS 386 = TRDOS v2.0)

106 <1> ; 27/03/2011

107 <1> ;

108 <1> ; INPUT-> ESI = Current Directory Buffer

109 <1> ; DL = TRDOS Logical Dos Drive Number + 1

110 <1> ; (0= Default/Current Drive)

111 <1> ;

112 <1> ; Note: Required dir buffer length may be <= 92 bytes

113 <1> ; for TRDOS (7\*12 name chars + 7 slash + 0)

114 <1> ; OUTPUT -> ESI = Current Directory Buffer

115 <1> ; EAX, EBX, ECX, EDX, EDI will be changed

116 <1> ; CX/CL = Current Directory String Length

117 <1> ; DL = Drive Number (0 based)

118 <1> ; (If input is 0, output is current drv number)

119 <1> ; DH = same with input

120 <1> ; cf = 0 -> AL = 0

121 <1> ; cf = 1 -> error code in AL

122 <1>

123 <1> loc\_get\_current\_drive\_0:

124 00009DBC 80FA00 <1> cmp dl, 0

125 00009DBF 7708 <1> ja short loc\_get\_current\_drive\_1

126 00009DC1 8A15[FE580100] <1> mov dl, [Current\_Drv]

127 00009DC7 EB17 <1> jmp short loc\_get\_current\_drive\_2

128 <1> loc\_get\_current\_drive\_1:

129 00009DC9 FECA <1> dec dl

130 00009DCB 3A15[D20C0100] <1> cmp dl, [Last\_DOS\_DiskNo]

131 00009DD1 760D <1> jna short loc\_get\_current\_drive\_2

132 00009DD3 B80F000000 <1> mov eax, 0Fh ; Invalid drive (Drive not ready!)

133 00009DD8 F5 <1> cmc ; stc

134 00009DD9 C3 <1> retn

135 <1>

136 <1> loc\_get\_current\_drive\_not\_ready\_retn:

137 00009DDA 5E <1> pop esi

138 <1> ;mov eax, 15

139 00009DDB 66B80F00 <1> mov ax, 15 ; Drive not ready

140 00009DDF C3 <1> retn

141 <1>

142 <1> loc\_get\_current\_drive\_2:

143 00009DE0 31C0 <1> xor eax, eax

144 00009DE2 88D4 <1> mov ah, dl

145 00009DE4 56 <1> push esi

146 00009DE5 BE00010900 <1> mov esi, Logical\_DOSDisks

147 00009DEA 01C6 <1> add esi, eax

148 00009DEC 8A06 <1> mov al, [esi+LD\_Name]

149 00009DEE 3C41 <1> cmp al, 'A'

150 00009DF0 72E8 <1> jb short loc\_get\_current\_drive\_not\_ready\_retn

151 <1>

152 00009DF2 8A667F <1> mov ah, [esi+LD\_CDirLevel]

153 00009DF5 08E4 <1> or ah, ah

154 00009DF7 7506 <1> jnz short loc\_get\_current\_drive\_3

155 <1>

156 <1> ;xor ah, ah ; mov ah, 0

157 00009DF9 8826 <1> mov [esi], ah

158 00009DFB 31C9 <1> xor ecx, ecx

159 00009DFD EB1C <1> jmp short loc\_get\_current\_drive\_4

160 <1>

161 <1> loc\_get\_current\_drive\_3:

162 00009DFF BF[5F610100] <1> mov edi, PATH\_Array

163 00009E04 57 <1> push edi

164 00009E05 81C680000000 <1> add esi, LD\_CurrentDirectory

165 00009E0B B920000000 <1> mov ecx, 32

166 00009E10 F3A5 <1> rep movsd

167 00009E12 5E <1> pop esi ; Path Array Address

168 00009E13 5F <1> pop edi ; pushed esi (current dir buffer offset)

169 <1> ;

170 00009E14 E841FFFFFF <1> call set\_current\_directory\_string

171 00009E19 89FE <1> mov esi, edi

172 <1>

173 <1> loc\_get\_current\_drive\_4:

174 00009E1B 30C0 <1> xor al, al

175 00009E1D C3 <1> retn

176 <1>

177 <1> change\_current\_directory:

178 <1> ; 19/02/2016

179 <1> ; 11/02/2016

180 <1> ; 10/02/2016

181 <1> ; 08/02/2016

182 <1> ; 06/02/2016 (TRDOS 386 = TRDOS v2.0)

183 <1> ; 18/09/2011 (DIR.ASM, 09/10/2011)

184 <1> ; 04/10/2009

185 <1> ; 2005

186 <1> ; INPUT ->

187 <1> ; ESI = Directory string

188 <1> ; ah = CD command (CDh = save current dir string)

189 <1> ; OUTPUT ->

190 <1> ; EDI = DOS Drive Description Table

191 <1> ; cf = 1 -> error

192 <1> ; EAX = Error code

193 <1> ; cf = 0 -> successful

194 <1> ; ESI = PATH\_Array

195 <1> ; EAX = Current Directory First Cluster

196 <1> ;

197 <1> ; (EAX, EBX, ECX, EDX, ESI, EDI will be changed)

198 <1>

199 00009E1E 8825[ED610100] <1> mov [CD\_COMMAND], ah

200 00009E24 803E2F <1> cmp byte [esi], '/'

201 00009E27 7505 <1> jne short loc\_ccd\_cdir\_level

202 00009E29 46 <1> inc esi

203 00009E2A 30C0 <1> xor al, al

204 00009E2C EB05 <1> jmp short loc\_ccd\_parse\_path\_name

205 <1> loc\_ccd\_cdir\_level:

206 00009E2E A0[FC580100] <1> mov al, [Current\_Dir\_Level]

207 <1> loc\_ccd\_parse\_path\_name:

208 00009E33 88C4 <1> mov ah, al

209 00009E35 BF[5F610100] <1> mov edi, PATH\_Array

210 <1>

211 <1> ; Reset directory levels > cdir level

212 <1> ; is this required !?

213 <1> ;

214 <1> ; Relations:

215 <1> ; MAINPROG.ASM (pass\_ccdrv\_reset\_cdir\_FAT\_fcluster)

216 <1> ; proc\_parse\_dir\_name,

217 <1> ; proc\_change\_current\_directory (this procedure)

218 <1> ; proc\_change\_prompt\_dir\_string

219 <1>

220 00009E3A 0FB6C8 <1> movzx ecx, al

221 00009E3D FEC1 <1> inc cl

222 00009E3F C0E104 <1> shl cl, 4

223 00009E42 01CF <1> add edi, ecx

224 00009E44 B107 <1> mov cl, 7

225 00009E46 28C1 <1> sub cl, al

226 00009E48 C0E102 <1> shl cl, 2

227 00009E4B 89C3 <1> mov ebx, eax

228 00009E4D 31C0 <1> xor eax, eax ; 0

229 00009E4F F3AB <1> rep stosd

230 00009E51 89D8 <1> mov eax, ebx

231 <1>

232 00009E53 BF[5F610100] <1> mov edi, PATH\_Array

233 <1>

234 00009E58 803E20 <1> cmp byte [esi], 20h

235 00009E5B F5 <1> cmc

236 00009E5C 7305 <1> jnc short pass\_ccd\_parse\_dir\_name

237 <1>

238 <1> ; ESI = Path name

239 <1> ; AL = CCD\_Level

240 00009E5E E872010000 <1> call parse\_dir\_name

241 <1> ; AL = CCD\_Level

242 <1> ; AH = Last\_Dir\_Level

243 <1> ; (EDI = PATH\_Array)

244 <1>

245 <1> pass\_ccd\_parse\_dir\_name:

246 00009E63 9C <1> pushf

247 <1>

248 <1> ;mov [CCD\_Level], al

249 <1> ;mov [Last\_Dir\_Level], ah

250 00009E64 66A3[E3610100] <1> mov [CCD\_Level], ax

251 <1>

252 00009E6A 31DB <1> xor ebx, ebx

253 00009E6C 8A3D[FE580100] <1> mov bh, [Current\_Drv]

254 00009E72 BE00010900 <1> mov esi, Logical\_DOSDisks

255 00009E77 01DE <1> add esi, ebx

256 <1>

257 00009E79 9D <1> popf

258 00009E7A 720A <1> jc short loc\_ccd\_bad\_path\_name\_retn

259 <1>

260 00009E7C 8935[DF610100] <1> mov [CCD\_DriveDT], esi

261 <1>

262 00009E82 3C07 <1> cmp al, 7

263 00009E84 7209 <1> jb short loc\_ccd\_load\_child\_dir

264 <1>

265 <1> loc\_ccd\_bad\_path\_name\_retn:

266 00009E86 87F7 <1> xchg esi, edi

267 00009E88 B813000000 <1> mov eax, 19 ; Bad directory/path name

268 00009E8D F9 <1> stc

269 <1> loc\_ccd\_retn\_p:

270 00009E8E C3 <1> retn

271 <1>

272 <1> loc\_ccd\_load\_child\_dir:

273 <1> ; AL = CCD\_Level

274 00009E8F 08C0 <1> or al, al

275 00009E91 7468 <1> jz short loc\_ccd\_load\_root\_dir

276 <1>

277 00009E93 6689C1 <1> mov cx, ax

278 00009E96 C0E004 <1> shl al, 4

279 00009E99 0FB6F0 <1> movzx esi, al

280 00009E9C 01FE <1> add esi, edi ; offset PATH\_Array

281 <1>

282 00009E9E 8B460C <1> mov eax, [esi+12]

283 00009EA1 38E9 <1> cmp cl, ch

284 00009EA3 0F84FA000000 <1> je loc\_ccd\_load\_sub\_directory

285 00009EA9 A3[F8580100] <1> mov [Current\_Dir\_FCluster], eax

286 <1>

287 <1> loc\_ccd\_load\_child\_dir\_next:

288 00009EAE 83C610 <1> add esi, 16 ; DOS DirEntry Format FileName Address

289 <1>

290 <1> ; Directory attribute : 10h

291 00009EB1 B010 <1> mov al, 00010000b ; 10h (Attrib AND mask)

292 <1> ;mov ah, 11001000b ; C8h

293 <1> ; Volume name attribute: 8h

294 00009EB3 B408 <1> mov ah, 00001000b ; 08h (Attrib NAND, AND --> zero mask)

295 <1>

296 00009EB5 6631C9 <1> xor cx, cx

297 00009EB8 E8B5010000 <1> call locate\_current\_dir\_file

298 00009EBD 7353 <1> jnc short loc\_ccd\_set\_dir\_cluster\_ptr

299 <1>

300 <1> ; 19/02/2016

301 <1> ;mov edi, [CCD\_DriveDT]

302 00009EBF 8A25[E3610100] <1> mov ah, [CCD\_Level]

303 00009EC5 803D[ED610100]CD <1> cmp byte [CD\_COMMAND], 0CDh ;'CD' command or another

304 00009ECC 7509 <1> jne short loc\_ccd\_load\_child\_dir\_err

305 <1> ; It is better to save recent successful part

306 <1> ; of the (requested) path as current directory.

307 <1> ; (Otherwise the path would be reset to back

308 <1> ; on the next 'CD' command.)

309 00009ECE 88E1 <1> mov cl, ah

310 00009ED0 50 <1> push eax

311 00009ED1 E8E3000000 <1> call loc\_ccd\_save\_current\_dir

312 00009ED6 58 <1> pop eax

313 <1> loc\_ccd\_load\_child\_dir\_err:

314 00009ED7 3C03 <1> cmp al, 3 ; AL = 2 => File not found error

315 00009ED9 7202 <1> jb short loc\_ccd\_path\_not\_found\_retn

316 00009EDB F9 <1> stc

317 00009EDC C3 <1> retn

318 <1>

319 <1> loc\_ccd\_path\_not\_found\_retn:

320 00009EDD B003 <1> mov al, 3 ; Path not found

321 00009EDF C3 <1> retn

322 <1>

323 <1> loc\_ccd\_load\_FAT\_root\_dir:

324 00009EE0 803D[FD580100]02 <1> cmp byte [Current\_FATType], 2

325 00009EE7 776B <1> ja short loc\_ccd\_load\_FAT32\_root\_dir

326 <1>

327 <1> ;mov esi, [CCD\_DriveDT]

328 <1> ;push esi

329 00009EE9 E8B51D0000 <1> call load\_FAT\_root\_directory

330 <1> ;pop edi ; Dos Drv Description Table

331 <1>

332 00009EEE 89F7 <1> mov edi, esi

333 00009EF0 BE[5F610100] <1> mov esi, PATH\_Array

334 00009EF5 7297 <1> jc short loc\_ccd\_retn\_p

335 <1>

336 00009EF7 31C0 <1> xor eax, eax

337 00009EF9 EB78 <1> jmp short loc\_ccd\_set\_cdfc

338 <1>

339 <1> loc\_ccd\_load\_root\_dir:

340 00009EFB 803D[FD580100]01 <1> cmp byte [Current\_FATType], 1

341 00009F02 73DC <1> jnb short loc\_ccd\_load\_FAT\_root\_dir

342 <1>

343 <1> loc\_ccd\_load\_FS\_root\_dir:

344 00009F04 E8611E0000 <1> call load\_FS\_root\_directory

345 00009F09 EB5C <1> jmp short pass\_ccd\_load\_FAT\_sub\_directory

346 <1>

347 <1> loc\_ccd\_load\_FS\_sub\_directory\_next:

348 00009F0B E85B1E0000 <1> call load\_FS\_sub\_directory

349 00009F10 EB1F <1> jmp short pass\_ccd\_set\_dir\_cluster\_ptr

350 <1>

351 <1> loc\_ccd\_set\_dir\_cluster\_ptr:

352 <1> ; EDI = Directory Entry

353 00009F12 668B4714 <1> mov ax, [edi+20] ; First Cluster High Word

354 00009F16 C1E010 <1> shl eax, 16

355 00009F19 668B471A <1> mov ax, [edi+26] ; First Cluster Low Word

356 <1>

357 00009F1D 8B35[DF610100] <1> mov esi, [CCD\_DriveDT]

358 00009F23 803D[FD580100]01 <1> cmp byte [Current\_FATType], 1

359 00009F2A 72DF <1> jb short loc\_ccd\_load\_FS\_sub\_directory\_next

360 <1> ;push esi

361 00009F2C E8FD1D0000 <1> call load\_FAT\_sub\_directory

362 <1> ;pop edi ; Dos Drv Description Table

363 <1>

364 <1> pass\_ccd\_set\_dir\_cluster\_ptr:

365 <1> ;mov edi, esi

366 00009F31 BE[5F610100] <1> mov esi, PATH\_Array

367 00009F36 7264 <1> jc short loc\_ccd\_retn\_c

368 <1>

369 00009F38 A1[2D610100] <1> mov eax, [DirBuff\_Cluster]

370 <1>

371 00009F3D FE05[E3610100] <1> inc byte [CCD\_Level]

372 00009F43 0FB61D[E3610100] <1> movzx ebx, byte [CCD\_Level]

373 00009F4A C0E304 <1> shl bl, 4 ; \* 16 (<= 128)

374 00009F4D 01DE <1> add esi, ebx ; 19/02/2016

375 00009F4F 89460C <1> mov [esi+12], eax

376 00009F52 EB1F <1> jmp short loc\_ccd\_set\_cdfc

377 <1>

378 <1> loc\_ccd\_load\_FAT32\_root\_dir:

379 00009F54 BE[5F610100] <1> mov esi, PATH\_Array

380 00009F59 8B460C <1> mov eax, [esi+12]

381 00009F5C 8B35[DF610100] <1> mov esi, [CCD\_DriveDT]

382 <1>

383 <1> loc\_ccd\_load\_FAT\_sub\_directory:

384 <1> ;push esi

385 00009F62 E8C71D0000 <1> call load\_FAT\_sub\_directory

386 <1> ;pop edi ; Dos Drv Description Table

387 <1>

388 <1> pass\_ccd\_load\_FAT\_sub\_directory:

389 <1> ;mov edi, esi

390 00009F67 BE[5F610100] <1> mov esi, PATH\_Array

391 00009F6C 722E <1> jc short loc\_ccd\_retn\_c

392 <1>

393 00009F6E A1[2D610100] <1> mov eax, [DirBuff\_Cluster]

394 <1>

395 <1> loc\_ccd\_set\_cdfc:

396 00009F73 8A0D[E3610100] <1> mov cl, [CCD\_Level]

397 00009F79 880D[FC580100] <1> mov [Current\_Dir\_Level], cl

398 00009F7F A3[F8580100] <1> mov [Current\_Dir\_FCluster], eax

399 <1>

400 00009F84 8A2D[E4610100] <1> mov ch, [Last\_Dir\_Level]

401 00009F8A 38E9 <1> cmp cl, ch

402 00009F8C 0F821CFFFFFF <1> jb loc\_ccd\_load\_child\_dir\_next

403 <1>

404 00009F92 803D[ED610100]CD <1> cmp byte [CD\_COMMAND], 0CDh ;'CD' command or another

405 00009F99 741E <1> je short loc\_ccd\_save\_current\_dir

406 <1>

407 <1> ; jne -> don't save, restore (the previous cdir) later !

408 <1> ; (saving the cdir would prevent previous cdir restoration!)

409 <1>

410 00009F9B F8 <1> clc

411 <1>

412 <1> loc\_ccd\_retn\_c:

413 00009F9C 8B3D[DF610100] <1> mov edi, [CCD\_DriveDT]

414 00009FA2 C3 <1> retn

415 <1>

416 <1> loc\_ccd\_load\_sub\_directory:

417 00009FA3 8B35[DF610100] <1> mov esi, [CCD\_DriveDT]

418 00009FA9 803D[FD580100]01 <1> cmp byte [Current\_FATType], 1

419 00009FB0 73B0 <1> jnb short loc\_ccd\_load\_FAT\_sub\_directory

420 00009FB2 E8B41D0000 <1> call load\_FS\_sub\_directory

421 00009FB7 EBAE <1> jmp short pass\_ccd\_load\_FAT\_sub\_directory

422 <1>

423 <1> loc\_ccd\_save\_current\_dir:

424 00009FB9 BE[5F610100] <1> mov esi, PATH\_Array ; 19/02/2016

425 00009FBE 8B3D[DF610100] <1> mov edi, [CCD\_DriveDT]

426 00009FC4 57 <1> push edi

427 00009FC5 83C77F <1> add edi, LD\_CDirLevel

428 00009FC8 880F <1> mov [edi], cl

429 00009FCA 47 <1> inc edi ; LD\_CurrentDirectory

430 00009FCB 56 <1> push esi

431 <1> ;mov ecx, 32 ; always < 65536 (in this procedure)

432 00009FCC 66B92000 <1> mov cx, 32

433 00009FD0 F3A5 <1> rep movsd

434 <1> ; Current directory has been saved to

435 <1> ; the DOS drive description table, cdir area !

436 00009FD2 5E <1> pop esi ; PATH\_Array

437 00009FD3 5F <1> pop edi ; Dos Drv Description Table

438 <1>

439 00009FD4 C3 <1> retn

440 <1>

441 <1> parse\_dir\_name:

442 <1> ; 11/02/2016

443 <1> ; 10/02/2016

444 <1> ; 07/02/2016 (TRDOS 386 = TRDOS v2.0)

445 <1> ; 18/09/2011

446 <1> ; 17/10/2009

447 <1> ; INPUT ->

448 <1> ; ESI = ASCIIZ Directory String Address

449 <1> ; AL = Current Directory Level

450 <1> ; EDI = Destination Adress

451 <1> ; (8 levels, each one 12+4 byte)

452 <1> ; OUTPUT ->

453 <1> ; EDI = Dir Entry Formatted Array

454 <1> ; with zero cluster pointer at the last level

455 <1> ; AH = Last Dir Level

456 <1> ; AL = Current Dir Level

457 <1> ;

458 <1> ; (esi, ebx, ecx will be changed)

459 <1>

460 <1> ;mov [PATH\_Array\_Ptr], edi

461 00009FD5 88C4 <1> mov ah, al

462 00009FD7 66A3[84620100] <1> mov [PATH\_CDLevel], ax

463 <1> repeat\_ppdn\_check\_slash:

464 00009FDD AC <1> lodsb

465 00009FDE 3C2F <1> cmp al, '/'

466 00009FE0 74FB <1> je short repeat\_ppdn\_check\_slash

467 00009FE2 3C21 <1> cmp al, 21h

468 00009FE4 7219 <1> jb short loc\_ppdn\_retn

469 00009FE6 57 <1> push edi

470 <1> loc\_ppdn\_get\_dir\_name:

471 00009FE7 B90C000000 <1> mov ecx, 12

472 00009FEC BF[86620100] <1> mov edi, Dir\_File\_Name

473 <1> repeat\_ppdn\_get\_dir\_name:

474 00009FF1 AA <1> stosb

475 00009FF2 AC <1> lodsb

476 00009FF3 3C2F <1> cmp al, '/'

477 00009FF5 740A <1> je short loc\_check\_level\_dot\_conv\_dir\_name

478 00009FF7 3C20 <1> cmp al, 20h

479 00009FF9 7605 <1> jna short loc\_ppdn\_end\_of\_path\_scan

480 00009FFB E2F4 <1> loop repeat\_ppdn\_get\_dir\_name

481 00009FFD 5F <1> pop edi

482 00009FFE F9 <1> stc

483 <1> loc\_ppdn\_retn:

484 00009FFF C3 <1> retn

485 <1>

486 <1> loc\_ppdn\_end\_of\_path\_scan:

487 0000A000 4E <1> dec esi

488 <1> loc\_check\_level\_dot\_conv\_dir\_name:

489 0000A001 31C0 <1> xor eax, eax

490 0000A003 AA <1> stosb

491 0000A004 89F3 <1> mov ebx, esi

492 0000A006 BE[86620100] <1> mov esi, Dir\_File\_Name

493 0000A00B AC <1> lodsb

494 <1> repeat\_ppdn\_name\_check\_dot:

495 0000A00C 3C2E <1> cmp al, '.'

496 0000A00E 7509 <1> jne short loc\_ppdn\_convert\_sub\_dir\_name

497 <1> repeat\_ppdn\_name\_dot\_dot:

498 0000A010 AC <1> lodsb

499 0000A011 3C2E <1> cmp al, '.'

500 0000A013 743E <1> je short loc\_ppdn\_dot\_dot

501 0000A015 3C21 <1> cmp al, 21h

502 0000A017 7226 <1> jb short pass\_ppdn\_convert\_sub\_dir\_name

503 <1> loc\_ppdn\_convert\_sub\_dir\_name:

504 0000A019 8A25[85620100] <1> mov ah, [PATH\_Level]

505 0000A01F 80FC07 <1> cmp ah, 7

506 0000A022 731B <1> jnb short pass\_ppdn\_convert\_sub\_dir\_name

507 0000A024 FEC4 <1> inc ah

508 0000A026 8825[85620100] <1> mov [PATH\_Level], ah

509 0000A02C BE[86620100] <1> mov esi, Dir\_File\_Name

510 <1> ;mov edi, [PATH\_Array\_Ptr]

511 0000A031 B010 <1> mov al, 16

512 0000A033 F6E4 <1> mul ah

513 0000A035 8B3C24 <1> mov edi, [esp]

514 <1> ;push edi

515 0000A038 01C7 <1> add edi, eax

516 0000A03A E82A030000 <1> call convert\_file\_name

517 <1> ;pop edi

518 <1> pass\_ppdn\_convert\_sub\_dir\_name:

519 0000A03F 89DE <1> mov esi, ebx

520 <1> repeat\_ppdn\_check\_last\_slash:

521 0000A041 AC <1> lodsb

522 0000A042 3C2F <1> cmp al, '/'

523 0000A044 74FB <1> je short repeat\_ppdn\_check\_last\_slash

524 0000A046 3C21 <1> cmp al, 21h

525 0000A048 739D <1> jnb short loc\_ppdn\_get\_dir\_name

526 <1> end\_of\_parse\_dir\_name:

527 0000A04A 5F <1> pop edi

528 0000A04B F5 <1> cmc

529 <1> ;mov al, [PATH\_CDLevel]

530 <1> ;mov ah, [PATH\_Level]

531 0000A04C 66A1[84620100] <1> mov ax, [PATH\_CDLevel]

532 0000A052 C3 <1> retn

533 <1>

534 <1> loc\_ppdn\_dot\_dot:

535 0000A053 AC <1> lodsb

536 0000A054 3C21 <1> cmp al, 21h

537 0000A056 73F2 <1> jnb short end\_of\_parse\_dir\_name

538 <1> loc\_ppdn\_dot\_dot\_prev\_level:

539 0000A058 66A1[84620100] <1> mov ax, [PATH\_CDLevel]

540 0000A05E 80EC01 <1> sub ah, 1

541 0000A061 80D400 <1> adc ah, 0

542 0000A064 38E0 <1> cmp al, ah

543 0000A066 7602 <1> jna short pass\_ppdn\_set\_al\_to\_ah

544 0000A068 88E0 <1> mov al, ah

545 <1> pass\_ppdn\_set\_al\_to\_ah:

546 0000A06A 66A3[84620100] <1> mov [PATH\_CDLevel], ax

547 0000A070 EBCD <1> jmp short pass\_ppdn\_convert\_sub\_dir\_name

548 <1>

549 <1> locate\_current\_dir\_file:

550 <1> ; 20/11/2017

551 <1> ; 14/02/2016

552 <1> ; 13/02/2016

553 <1> ; 10/02/2016

554 <1> ; 06/02/2016 (TRDOS 386 = TRDOS v2.0)

555 <1> ; 14/08/2010

556 <1> ; 19/09/2009

557 <1> ; 2005

558 <1> ; INPUT ->

559 <1> ; ESI = DOS DirEntry Format FileName Address

560 <1> ; AL = Attributes Mask

561 <1> ; (<AL AND EntryAttrib> must be equal to AL)

562 <1> ; AH = Negative Attributes Mask (If AH>0)

563 <1> ; (<AH AND EntryAttrib> must be ZERO)

564 <1> ; CH > 0 Find First Free Dir Entry or Deleted Entry

565 <1> ; CL = 0 -> Return the First Free Dir Entry

566 <1> ; CL = E5h -> Return the 1st deleted entry

567 <1> ; CL = FFh -> Return the 1st deleted or free entry

568 <1> ; CL > 0 and CL <> E5h and CL <> FFh -> Return the first

569 <1> ; proper entry (which fits with Atributes Masks)

570 <1> ; CX = 0 Find Valid File/Directory/VolumeName

571 <1> ; ? = Any One Char

572 <1> ; \* = Every Chars

573 <1> ; OUTPUT ->

574 <1> ; EDI = Directory Entry Address (in Directory Buffer)

575 <1> ; ESI = DOS DirEntry Format FileName Address

576 <1> ; CF = 0 -> No Error, Proper Entry,

577 <1> ; DL = Attributes

578 <1> ; DH = Previous Entry Attr (LongName Check)

579 <1> ; AL > 0 -> Ambiguous filename wildcard "?" used

580 <1> ; AH > 0 -> Ambiguous filename wildcard "\*" used

581 <1> ; AX = 0 -> Filename full fits with directory entry

582 <1> ; CH = The 1st Name Char of Current Dir Entry

583 <1> ; CF = 1 -> Proper entry not found, Error Code in EAX/AL

584 <1> ; CL = 0 and CH = 0 -> Free Entry (End Of Dir)

585 <1> ; CL = 0 and CH = E5h -> Deleted Entry fits with filters

586 <1> ; CL > 0 -> Entry not found, CH invalid

587 <1> ; CF = 0 ->

588 <1> ; EBX = Current Directory Entry Index/Number (BX)

589 <1>

590 <1> ;mov word [DirBuff\_EntryCounter], 0 ; Zero Based

591 <1>

592 0000A072 8935[E7610100] <1> mov [CDLF\_FNAddress], esi

593 0000A078 66A3[E5610100] <1> mov [CDLF\_AttributesMask], ax

594 0000A07E 66890D[EB610100] <1> mov [CDLF\_DEType], cx

595 <1>

596 0000A085 31DB <1> xor ebx, ebx

597 0000A087 881D[FC610100] <1> mov [PreviousAttr], bl ; 0 ; 13/02/2016

598 <1>

599 0000A08D 8A3D[FE580100] <1> mov bh, [Current\_Drv]

600 0000A093 381D[28610100] <1> cmp byte [DirBuff\_ValidData], bl ; 0

601 0000A099 761D <1> jna short loc\_lcdf\_reload\_current\_dir2

602 0000A09B 8A1D[26610100] <1> mov bl, [DirBuff\_DRV]

603 0000A0A1 80EB41 <1> sub bl, 'A'

604 0000A0A4 38DF <1> cmp bh, bl

605 0000A0A6 750E <1> jne short loc\_lcdf\_reload\_current\_dir1

606 0000A0A8 8B15[2D610100] <1> mov edx, [DirBuff\_Cluster]

607 0000A0AE 3B15[F8580100] <1> cmp edx, [Current\_Dir\_FCluster]

608 0000A0B4 7412 <1> je short loc\_cdir\_locatefile\_search

609 <1>

610 <1> loc\_lcdf\_reload\_current\_dir1:

611 0000A0B6 30DB <1> xor bl, bl

612 <1> loc\_lcdf\_reload\_current\_dir2:

613 0000A0B8 89DE <1> mov esi, ebx

614 0000A0BA 81C600010900 <1> add esi, Logical\_DOSDisks

615 0000A0C0 E874000000 <1> call reload\_current\_directory

616 0000A0C5 735D <1> jnc short loc\_locatefile\_search\_again

617 0000A0C7 C3 <1> retn

618 <1>

619 <1> loc\_cdir\_locatefile\_search:

620 0000A0C8 31DB <1> xor ebx, ebx

621 0000A0CA 55 <1> push ebp ; 20/11/2017

622 0000A0CB E8A6000000 <1> call find\_directory\_entry

623 0000A0D0 5D <1> pop ebp ; 20/11/2017

624 0000A0D1 7349 <1> jnc short loc\_cdir\_locate\_file\_retn

625 <1>

626 <1> loc\_locatefile\_check\_stc\_reason:

627 0000A0D3 08ED <1> or ch, ch

628 0000A0D5 7444 <1> jz short loc\_cdir\_locate\_file\_stc\_retn

629 <1>

630 <1> loc\_locatefile\_check\_next\_entryblock:

631 0000A0D7 8A3D[FE580100] <1> mov bh, [Current\_Drv]

632 0000A0DD 28DB <1> sub bl, bl

633 0000A0DF 0FB7F3 <1> movzx esi, bx

634 0000A0E2 81C600010900 <1> add esi, Logical\_DOSDisks

635 <1>

636 0000A0E8 803D[FC580100]00 <1> cmp byte [Current\_Dir\_Level], 0

637 0000A0EF 760A <1> jna short loc\_locatefile\_check\_FAT\_type

638 <1>

639 0000A0F1 803D[FD580100]01 <1> cmp byte [Current\_FATType], 1

640 0000A0F8 730A <1> jnb short loc\_locatefile\_load\_subdir\_cluster

641 0000A0FA C3 <1> retn

642 <1>

643 <1> loc\_locatefile\_check\_FAT\_type:

644 0000A0FB 803D[FD580100]03 <1> cmp byte [Current\_FATType], 3

645 0000A102 7218 <1> jb short loc\_cdir\_locate\_file\_retn

646 <1>

647 <1> loc\_locatefile\_load\_subdir\_cluster:

648 0000A104 A1[2D610100] <1> mov eax, [DirBuff\_Cluster]

649 0000A109 E83A1A0000 <1> call get\_next\_cluster

650 0000A10E 730D <1> jnc short loc\_locatefile\_next\_cluster

651 0000A110 09C0 <1> or eax, eax

652 0000A112 7507 <1> jnz short loc\_locatefile\_drive\_not\_ready\_read\_err

653 0000A114 F9 <1> stc

654 <1> loc\_locatefile\_file\_notfound:

655 0000A115 B802000000 <1> mov eax, 2 ; File/Directory/VolName not found

656 0000A11A C3 <1> retn

657 <1>

658 <1> loc\_locatefile\_drive\_not\_ready\_read\_err:

659 <1> ;mov eax, 17 ;Drive not ready or read error

660 <1> loc\_cdir\_locate\_file\_stc\_retn:

661 0000A11B F5 <1> cmc ;stc

662 <1> loc\_cdir\_locate\_file\_retn:

663 0000A11C C3 <1> retn

664 <1>

665 <1> loc\_locatefile\_next\_cluster:

666 0000A11D E80C1C0000 <1> call load\_FAT\_sub\_directory

667 <1> ;jc short loc\_locatefile\_drive\_not\_ready\_read\_err

668 0000A122 72F8 <1> jc short loc\_cdir\_locate\_file\_retn

669 <1>

670 <1> loc\_locatefile\_search\_again:

671 0000A124 8B35[E7610100] <1> mov esi, [CDLF\_FNAddress]

672 0000A12A 66A1[E5610100] <1> mov ax, [CDLF\_AttributesMask]

673 0000A130 668B0D[EB610100] <1> mov cx, [CDLF\_DEType]

674 0000A137 EB8F <1> jmp short loc\_cdir\_locatefile\_search

675 <1>

676 <1> reload\_current\_directory:

677 <1> ; 06/02/2016 (TRDOS 386 = TRDOS v2.0)

678 <1> ; 13/06/2010

679 <1> ; 22/09/2009

680 <1> ;

681 <1> ; INPUT ->

682 <1> ; ESI = Dos drive description table address

683 <1>

684 <1> ;mov al, [esi+LD\_FATType]

685 0000A139 A0[FD580100] <1> mov al, [Current\_FATType]

686 0000A13E 3C02 <1> cmp al, 2

687 0000A140 7729 <1> ja short loc\_reload\_FAT\_sub\_directory

688 0000A142 8A25[FC580100] <1> mov ah, [Current\_Dir\_Level]

689 0000A148 08C0 <1> or al, al

690 0000A14A 740A <1> jz short loc\_reload\_FS\_directory

691 0000A14C 08E4 <1> or ah, ah

692 0000A14E 751B <1> jnz short loc\_reload\_FAT\_sub\_directory

693 <1> loc\_reload\_FAT\_12\_16\_root\_directory:

694 0000A150 E84E1B0000 <1> call load\_FAT\_root\_directory

695 0000A155 C3 <1> retn

696 <1> loc\_reload\_FS\_directory:

697 0000A156 20E4 <1> and ah, ah

698 0000A158 7506 <1> jnz short loc\_reload\_FS\_sub\_directory

699 <1> loc\_reload\_FS\_root\_directory:

700 0000A15A E80B1C0000 <1> call load\_FS\_root\_directory

701 0000A15F C3 <1> retn

702 <1> loc\_reload\_FS\_sub\_directory:

703 0000A160 A1[F8580100] <1> mov eax, [Current\_Dir\_FCluster]

704 0000A165 E8011C0000 <1> call load\_FS\_sub\_directory

705 0000A16A C3 <1> retn

706 <1> loc\_reload\_FAT\_sub\_directory:

707 0000A16B A1[F8580100] <1> mov eax, [Current\_Dir\_FCluster]

708 0000A170 E8B91B0000 <1> call load\_FAT\_sub\_directory

709 0000A175 C3 <1> retn

710 <1>

711 <1> find\_directory\_entry:

712 <1> ; 14/02/2016

713 <1> ; 13/02/2016

714 <1> ; 10/02/2016

715 <1> ; 06/02/2016 (TRDOS 386 = TRDOS v2.0)

716 <1> ; 14/08/2010 (DIR.ASM, "proc\_find\_direntry")

717 <1> ; 19/09/2009

718 <1> ; 2005

719 <1> ; INPUT ->

720 <1> ; ESI = Sub Dir or File Name Address

721 <1> ; AL = Attributes Mask

722 <1> ; (<AL AND EntryAttrib> must be equal to AL)

723 <1> ; AH = Negative Attributes Mask (If AH>0)

724 <1> ; (<AH AND EntryAttrib> must be ZERO)

725 <1> ; CH > 0 Find First Free Dir Entry or Deleted Entry

726 <1> ; CL = 0 -> Return the First Free Dir Entry

727 <1> ; CL = E5h -> Return the 1st deleted entry

728 <1> ; CL = FFh -> Return the 1st deleted or free entry

729 <1> ; CL > 0 and CL <> E5h and CL <> FFh -> Return the first

730 <1> ; proper entry (which fits with Atributes Masks)

731 <1> ; CX = 0 -> Find Valid File/Directory/VolumeName

732 <1> ; ? = Any One Char

733 <1> ; \* = Every Chars

734 <1> ; EBX = Current Dir Entry (BX)

735 <1> ;

736 <1> ; OUTPUT ->

737 <1> ; EDI = Directory Entry Address (in DirectoryBuffer)

738 <1> ; ESI = Sub Dir or File Name Address

739 <1> ; CF = 0 -> No Error, Proper Entry,

740 <1> ; DL = Attributes

741 <1> ; DH = Previous Entry Attr (LongName Check)

742 <1> ; AL > 0 -> Ambiguous filename wildcard "?" used

743 <1> ; AH > 0 -> Ambiguous filename wildcard "\*" used

744 <1> ; AX = 0 -> Filename full fits with directory entry

745 <1> ; EBX = CurrentDirEntry (BX)

746 <1> ; CH = The 1st Name Char of Current Dir Entry

747 <1> ; CF = 1 -> Proper entry not found, Error Code in AX/AL

748 <1> ; CL = 0 and CH = 0 -> Free Entry (End Of Dir)

749 <1> ; CL = 0 and CH = E5h -> Deleted Entry fits with filters

750 <1> ; CL > 0 -> Entry not found, CH invalid

751 <1> ;

752 <1> ; (EAX, EBX, ECX, EDX, EDI, EBP will be changed)

753 <1>

754 0000A176 663B1D[2B610100] <1> cmp bx, [DirBuff\_LastEntry]

755 0000A17D 0F8739010000 <1> ja loc\_ffde\_stc\_retn\_255

756 <1>

757 <1> ;mov [DirBuff\_CurrentEntry], bx

758 <1>

759 0000A183 BF00000800 <1> mov edi, Directory\_Buffer

760 0000A188 66A3[F8610100] <1> mov [FDE\_AttrMask], ax

761 <1>

762 0000A18E 29C0 <1> sub eax, eax

763 <1>

764 <1> ;;mov [PreviousAttr], al ; 0 ;; 13/02/2016

765 0000A190 66A3[FA610100] <1> mov [AmbiguousFileName], ax ; 0

766 <1>

767 0000A196 6689D8 <1> mov ax, bx

768 0000A199 66C1E005 <1> shl ax, 5 ; ; \* 32 ; Directory entry size

769 0000A19D 01C7 <1> add edi, eax

770 <1>

771 0000A19F 08ED <1> or ch, ch

772 0000A1A1 0F852C010000 <1> jnz loc\_find\_free\_deleted\_entry\_0

773 <1>

774 0000A1A7 08C9 <1> or cl, cl

775 0000A1A9 0F850D010000 <1> jnz loc\_ffde\_stc\_retn\_255

776 <1>

777 <1> check\_find\_dir\_entry:

778 0000A1AF 66A1[F8610100] <1> mov ax, [FDE\_AttrMask]

779 0000A1B5 8A2F <1> mov ch, [edi]

780 0000A1B7 80FD00 <1> cmp ch, 0 ; Is it never used entry?

781 0000A1BA 0F86FF000000 <1> jna loc\_find\_direntry\_stc\_retn

782 0000A1C0 56 <1> push esi

783 0000A1C1 8A570B <1> mov dl, [edi+0Bh] ; File attributes

784 0000A1C4 80FDE5 <1> cmp ch, 0E5h ; Is it a deleted file?

785 0000A1C7 746D <1> je short loc\_find\_dir\_next\_entry\_prevdeleted

786 <1>

787 0000A1C9 80FA0F <1> cmp dl, 0Fh ; longname sub component check

788 0000A1CC 7505 <1> jne short loc\_check\_attributes\_mask

789 0000A1CE E8ED010000 <1> call save\_longname\_sub\_component

790 <1>

791 <1> loc\_check\_attributes\_mask:

792 0000A1D3 88C6 <1> mov dh, al

793 0000A1D5 20D6 <1> and dh, dl

794 0000A1D7 38F0 <1> cmp al, dh

795 0000A1D9 0F85BA000000 <1> jne loc\_find\_dir\_next\_entry

796 0000A1DF 20D4 <1> and ah, dl

797 0000A1E1 0F85B2000000 <1> jnz loc\_find\_dir\_next\_entry

798 0000A1E7 80FA0F <1> cmp dl, 0Fh

799 0000A1EA 751A <1> jne short pass\_direntry\_attr\_check

800 <1>

801 0000A1EC 3C0F <1> cmp al, 0Fh ; AL = 0Fh -> find long name

802 0000A1EE 0F85A5000000 <1> jne loc\_find\_dir\_next\_entry

803 <1>

804 0000A1F4 5E <1> pop esi

805 0000A1F5 6631C0 <1> xor ax, ax

806 0000A1F8 8A35[FC610100] <1> mov dh, [PreviousAttr]

807 0000A1FE 66891D[29610100] <1> mov [DirBuff\_CurrentEntry], bx

808 0000A205 C3 <1> retn

809 <1>

810 <1> pass\_direntry\_attr\_check:

811 0000A206 89FD <1> mov ebp, edi ; 14/02/2016

812 0000A208 B908000000 <1> mov ecx, 8

813 <1> loc\_lodsb\_find\_dir:

814 0000A20D AC <1> lodsb

815 0000A20E 3C2A <1> cmp al, '\*'

816 0000A210 7508 <1> jne short pass\_fde\_ambiguous1\_check

817 0000A212 FE05[FB610100] <1> inc byte [AmbiguousFileName+1]

818 0000A218 EB28 <1> jmp short loc\_check\_direntry\_extension

819 <1>

820 <1> pass\_fde\_ambiguous1\_check:

821 0000A21A 3C3F <1> cmp al, '?'

822 0000A21C 750D <1> jne short pass\_fde\_ambiguous2\_check

823 0000A21E FE05[FA610100] <1> inc byte [AmbiguousFileName]

824 0000A224 803F20 <1> cmp byte [edi], 20h

825 0000A227 764E <1> jna short loc\_find\_dir\_next\_entry\_ebp

826 0000A229 EB14 <1> jmp short loc\_scasb\_find\_dir\_inc\_di

827 <1>

828 <1> pass\_fde\_ambiguous2\_check:

829 0000A22B 3C20 <1> cmp al, 20h

830 0000A22D 750C <1> jne short loc\_scasb\_find\_dir

831 0000A22F 803F20 <1> cmp byte [edi], 20h

832 0000A232 7543 <1> jne short loc\_find\_dir\_next\_entry\_ebp

833 0000A234 EB0C <1> jmp short loc\_check\_direntry\_extension

834 <1>

835 <1> loc\_find\_dir\_next\_entry\_prevdeleted:

836 0000A236 80CA80 <1> or dl, 80h ; Bit 7 -> deleted entry sign

837 0000A239 EB5E <1> jmp short loc\_find\_dir\_next\_entry

838 <1>

839 <1> loc\_scasb\_find\_dir:

840 0000A23B 3A07 <1> cmp al, [edi]

841 0000A23D 7538 <1> jne short loc\_find\_dir\_next\_entry\_ebp

842 <1> loc\_scasb\_find\_dir\_inc\_di:

843 0000A23F 47 <1> inc edi

844 0000A240 E2CB <1> loop loc\_lodsb\_find\_dir

845 <1>

846 <1> loc\_check\_direntry\_extension:

847 0000A242 BE08000000 <1> mov esi, 8

848 0000A247 89F7 <1> mov edi, esi ; 8

849 0000A249 033424 <1> add esi, [esp] ; Sub Dir or File Name Address

850 0000A24C 01EF <1> add edi, ebp

851 0000A24E B103 <1> mov cl, 3

852 <1> loc\_lodsb\_find\_dir\_ext:

853 0000A250 AC <1> lodsb

854 0000A251 3C2A <1> cmp al, '\*'

855 0000A253 7508 <1> jne short pass\_fde\_ambiguous3\_check

856 0000A255 FE05[FB610100] <1> inc byte [AmbiguousFileName+1]

857 0000A25B EB1E <1> jmp short loc\_find\_dir\_proper\_direntry

858 <1>

859 <1> pass\_fde\_ambiguous3\_check:

860 0000A25D 3C3F <1> cmp al, '?'

861 0000A25F 750D <1> jne short pass\_fde\_ambiguous4\_check

862 0000A261 FE05[FA610100] <1> inc byte [AmbiguousFileName]

863 0000A267 803F20 <1> cmp byte [edi], 20h

864 0000A26A 760B <1> jna short loc\_find\_dir\_next\_entry\_ebp

865 0000A26C EB49 <1> jmp short loc\_scasb\_find\_dir\_ext\_inc\_di

866 <1>

867 <1> pass\_fde\_ambiguous4\_check:

868 0000A26E 3C20 <1> cmp al, 20h

869 0000A270 7541 <1> jne short loc\_scasb\_find\_dir\_ext

870 0000A272 803F20 <1> cmp byte [edi], 20h

871 0000A275 7404 <1> je short loc\_find\_dir\_proper\_direntry

872 <1>

873 <1> loc\_find\_dir\_next\_entry\_ebp:

874 0000A277 89EF <1> mov edi, ebp ; 14/02/2016

875 0000A279 EB1E <1> jmp short loc\_find\_dir\_next\_entry

876 <1>

877 <1> loc\_find\_dir\_proper\_direntry:

878 0000A27B 30C9 <1> xor cl, cl

879 <1> loc\_find\_dir\_proper\_direntry\_1:

880 0000A27D 5E <1> pop esi

881 0000A27E 89EF <1> mov edi, ebp

882 0000A280 8A2F <1> mov ch, [edi]

883 0000A282 8A570B <1> mov dl, [edi+0Bh] ; Dir entry attributes

884 0000A285 66A1[FA610100] <1> mov ax, [AmbiguousFileName]

885 <1> loc\_find\_dir\_proper\_direntry\_2:

886 0000A28B 8A35[FC610100] <1> mov dh, [PreviousAttr]

887 0000A291 66891D[29610100] <1> mov [DirBuff\_CurrentEntry], bx

888 0000A298 C3 <1> retn

889 <1>

890 <1> loc\_find\_dir\_next\_entry:

891 0000A299 8815[FC610100] <1> mov byte [PreviousAttr], dl ; LongName check

892 <1> loc\_find\_dir\_next\_entry\_1:

893 0000A29F 5E <1> pop esi

894 0000A2A0 83C720 <1> add edi, 32

895 <1> ;inc word [DirBuff\_EntryCounter]

896 0000A2A3 6643 <1> inc bx

897 0000A2A5 663B1D[2B610100] <1> cmp bx, [DirBuff\_LastEntry]

898 0000A2AC 770E <1> ja short loc\_ffde\_stc\_retn\_255

899 0000A2AE E9FCFEFFFF <1> jmp check\_find\_dir\_entry

900 <1>

901 <1> loc\_scasb\_find\_dir\_ext:

902 0000A2B3 3A07 <1> cmp al, [edi]

903 0000A2B5 75C0 <1> jne short loc\_find\_dir\_next\_entry\_ebp

904 <1> loc\_scasb\_find\_dir\_ext\_inc\_di:

905 0000A2B7 47 <1> inc edi

906 0000A2B8 E296 <1> loop loc\_lodsb\_find\_dir\_ext

907 0000A2BA EBC1 <1> jmp short loc\_find\_dir\_proper\_direntry\_1

908 <1>

909 <1> loc\_ffde\_stc\_retn\_255:

910 <1> ;mov cx, 0FFFFh

911 0000A2BC 31C9 <1> xor ecx, ecx

912 0000A2BE 49 <1> dec ecx ; 0FFFFFFFFh

913 <1> ;xor eax, eax

914 <1> loc\_find\_direntry\_stc\_retn:

915 <1> loc\_check\_ffde\_retn\_1:

916 <1> ;mov ax, 2

917 0000A2BF B802000000 <1> mov eax, 2 ; File Not Found

918 0000A2C4 8A35[FC610100] <1> mov dh, [PreviousAttr]

919 0000A2CA 66891D[29610100] <1> mov [DirBuff\_CurrentEntry], bx

920 0000A2D1 F9 <1> stc

921 0000A2D2 C3 <1> retn

922 <1>

923 <1> loc\_find\_free\_deleted\_entry\_0:

924 0000A2D3 66A1[F8610100] <1> mov ax, [FDE\_AttrMask]

925 0000A2D9 8A2F <1> mov ch, [edi]

926 0000A2DB 8A570B <1> mov dl, [edi+0Bh] ; File attributes

927 0000A2DE 08C9 <1> or cl, cl

928 0000A2E0 7407 <1> jz short loc\_check\_ffde\_0\_repeat

929 <1> ;cmp cl, 0E5h

930 <1> ;je short pass\_loc\_check\_ffde\_0\_err

931 0000A2E2 80F9FF <1> cmp cl, 0FFh

932 0000A2E5 7432 <1> je short loc\_find\_free\_deleted\_entry\_1

933 0000A2E7 EB4D <1> jmp short pass\_loc\_check\_ffde\_0\_err

934 <1>

935 <1> loc\_check\_ffde\_0\_repeat:

936 0000A2E9 08ED <1> or ch, ch

937 0000A2EB 7511 <1> jnz short loc\_check\_ffde\_0\_next

938 <1>

939 <1> loc\_check\_ffde\_retn\_2:

940 0000A2ED 6629C0 <1> sub ax, ax

941 0000A2F0 8A35[FC610100] <1> mov dh, [PreviousAttr]

942 0000A2F6 66891D[29610100] <1> mov [DirBuff\_CurrentEntry], bx

943 0000A2FD C3 <1> retn

944 <1>

945 <1> loc\_check\_ffde\_0\_next:

946 0000A2FE 6643 <1> inc bx

947 0000A300 83C720 <1> add edi, 32

948 <1> ;inc word [DirBuff\_EntryCounter]

949 <1>

950 0000A303 663B1D[2B610100] <1> cmp bx, [DirBuff\_LastEntry]

951 0000A30A 77B0 <1> ja short loc\_ffde\_stc\_retn\_255

952 0000A30C 8815[FC610100] <1> mov [PreviousAttr], dl

953 0000A312 8A2F <1> mov ch, [edi]

954 0000A314 8A570B <1> mov dl, [edi+0Bh] ; file attributes

955 0000A317 EBD0 <1> jmp short loc\_check\_ffde\_0\_repeat

956 <1>

957 <1> loc\_find\_free\_deleted\_entry\_1:

958 0000A319 28D2 <1> sub dl, dl

959 <1> loc\_find\_free\_deleted\_entry\_2:

960 0000A31B 20ED <1> and ch, ch

961 0000A31D 74CE <1> jz short loc\_check\_ffde\_retn\_2

962 0000A31F 80FDE5 <1> cmp ch, 0E5h

963 0000A322 74C9 <1> je short loc\_check\_ffde\_retn\_2

964 0000A324 6643 <1> inc bx

965 0000A326 83C720 <1> add edi, 32

966 0000A329 663B1D[2B610100] <1> cmp bx, [DirBuff\_LastEntry]

967 0000A330 778A <1> ja short loc\_ffde\_stc\_retn\_255

968 0000A332 8A2F <1> mov ch, [edi]

969 0000A334 EBE5 <1> jmp short loc\_find\_free\_deleted\_entry\_2

970 <1>

971 <1> pass\_loc\_check\_ffde\_0\_err:

972 0000A336 38CD <1> cmp ch, cl

973 0000A338 741F <1> je short loc\_check\_ffde\_attrib

974 <1>

975 0000A33A 6643 <1> inc bx

976 0000A33C 83C720 <1> add edi, 32

977 0000A33F 663B1D[2B610100] <1> cmp bx, [DirBuff\_LastEntry]

978 0000A346 0F8770FFFFFF <1> ja loc\_ffde\_stc\_retn\_255

979 0000A34C 8815[FC610100] <1> mov [PreviousAttr], dl

980 0000A352 8A2F <1> mov ch, [edi]

981 0000A354 8A570B <1> mov dl, [edi+0Bh]

982 0000A357 EBDD <1> jmp short pass\_loc\_check\_ffde\_0\_err

983 <1>

984 <1> loc\_check\_ffde\_attrib:

985 0000A359 88C6 <1> mov dh, al

986 0000A35B 20D6 <1> and dh, dl

987 0000A35D 38F0 <1> cmp al, dh

988 0000A35F 759D <1> jne short loc\_check\_ffde\_0\_next

989 0000A361 20D4 <1> and ah, dl

990 0000A363 7599 <1> jnz short loc\_check\_ffde\_0\_next

991 0000A365 30C9 <1> xor cl, cl

992 0000A367 EB84 <1> jmp loc\_check\_ffde\_retn\_2

993 <1>

994 <1> convert\_file\_name:

995 <1> ; 06/03/2016

996 <1> ; 11/02/2016

997 <1> ; 07/02/2016 (TRDOS 386 = TRDOS v2.0)

998 <1> ; 06/10/2009

999 <1> ; 2005

1000 <1> ;

1001 <1> ; INPUT ->

1002 <1> ; ESI = Dot File Name Location

1003 <1> ; EDI = Dir Entry Format File Name Location

1004 <1> ; OUTPUT ->

1005 <1> ; EDI = Dir Entry Format File Name Location

1006 <1> ; ESI = Dot File Name Location (capitalized)

1007 <1> ;

1008 <1> ; (ECX, AL will be changed)

1009 <1>

1010 0000A369 56 <1> push esi

1011 0000A36A 57 <1> push edi

1012 <1>

1013 0000A36B B90B000000 <1> mov ecx, 11

1014 0000A370 B020 <1> mov al, 20h

1015 0000A372 F3AA <1> rep stosb

1016 <1>

1017 0000A374 8B3C24 <1> mov edi, [esp]

1018 <1>

1019 0000A377 B10C <1> mov cl, 12 ; file name length (max.)

1020 <1> ; 06/03/2016

1021 <1> ; Directory entry name limit (11 bytes) check for

1022 <1> ; 'rename\_directory\_entry' procedure.

1023 <1> ; (EDI points to Directory Entry)

1024 <1> ; (If the file name would not contain a dot

1025 <1> ; and file name length would be 12, this would cause to

1026 <1> ; overwrite the attributes byte of the directory entry.)

1027 <1> ;

1028 0000A379 B50B <1> mov ch, 11 ; directory entry's name length

1029 <1> loc\_check\_first\_dot:

1030 0000A37B 8A06 <1> mov al, [esi]

1031 0000A37D 3C2E <1> cmp al, 2Eh

1032 0000A37F 750C <1> jne short pass\_check\_first\_dot

1033 0000A381 8807 <1> mov [edi], al

1034 0000A383 47 <1> inc edi

1035 0000A384 46 <1> inc esi

1036 0000A385 FEC9 <1> dec cl

1037 0000A387 75F2 <1> jnz short loc\_check\_first\_dot

1038 <1> ;;(ecx <= 12)

1039 <1> ;;loop loc\_check\_first\_dot

1040 0000A389 EB30 <1> jmp short stop\_convert\_file

1041 <1>

1042 <1> loc\_get\_fchar:

1043 0000A38B 8A06 <1> mov al, [esi]

1044 <1> pass\_check\_first\_dot:

1045 0000A38D 3C61 <1> cmp al, 61h ; 'a'

1046 0000A38F 7208 <1> jb short pass\_name\_capitalize

1047 0000A391 3C7A <1> cmp al, 7Ah ; 'z'

1048 0000A393 7704 <1> ja short pass\_name\_capitalize

1049 0000A395 24DF <1> and al, 0DFh

1050 0000A397 8806 <1> mov [esi], al

1051 <1> pass\_name\_capitalize:

1052 0000A399 3C21 <1> cmp al, 21h

1053 0000A39B 721E <1> jb short stop\_convert\_file

1054 0000A39D 3C2E <1> cmp al, 2Eh ; '.'

1055 0000A39F 750C <1> jne short pass\_dot\_space

1056 <1> add\_dot\_space:

1057 0000A3A1 80F904 <1> cmp cl, 4

1058 0000A3A4 760E <1> jna short inc\_and\_loop

1059 0000A3A6 47 <1> inc edi

1060 0000A3A7 FECD <1> dec ch ; 06/03/2016

1061 0000A3A9 FEC9 <1> dec cl

1062 0000A3AB EBF4 <1> jmp short add\_dot\_space

1063 <1>

1064 <1> ;mov al, 4

1065 <1> ;cmp cl, al

1066 <1> ;jna short inc\_and\_loop

1067 <1> ;sub cl, al

1068 <1> ;add edi, ecx

1069 <1> ;mov cl, al

1070 <1> ;jmp short inc\_and\_loop

1071 <1>

1072 <1> pass\_dot\_space:

1073 0000A3AD 8807 <1> mov [edi], al

1074 <1> loc\_after\_double\_dot:

1075 <1> ; 06/03/2016

1076 0000A3AF FECD <1> dec ch ; count down for 11 bytes dir entry limit

1077 0000A3B1 740A <1> jz short stop\_convert\_file\_x

1078 0000A3B3 47 <1> inc edi

1079 <1> inc\_and\_loop:

1080 0000A3B4 FEC9 <1> dec cl ; count down for 12 bytes filename limit

1081 0000A3B6 7403 <1> jz short stop\_convert\_file

1082 0000A3B8 46 <1> inc esi

1083 <1> ;;(ecx <= 12)

1084 <1> ;;loop loc\_get\_fchar

1085 0000A3B9 EBD0 <1> jmp short loc\_get\_fchar

1086 <1>

1087 <1> stop\_convert\_file:

1088 <1> ; 06/03/2016

1089 0000A3BB 30ED <1> xor ch, ch

1090 <1> ; ECX < 256 ; 'find\_first\_file' -> xor cl, cl

1091 <1> stop\_convert\_file\_x:

1092 0000A3BD 5F <1> pop edi

1093 0000A3BE 5E <1> pop esi

1094 0000A3BF C3 <1> retn

1095 <1>

1096 <1> save\_longname\_sub\_component:

1097 <1> ; 13/02/2016

1098 <1> ; 06/02/2016 (TRDOS 386 = TRDOS v2.0)

1099 <1> ; 28/02/2010

1100 <1> ; 17/10/2009

1101 <1> ; INPUT ->

1102 <1> ; EDI = Directory Entry

1103 <1> ; // This procedure is called

1104 <1> ; // from 'find\_directory\_entry' procedure.

1105 <1> ; // If the last entry returns with

1106 <1> ; // a non-zero LongnameFound value and

1107 <1> ; // if LFN\_CheckSum value is equal to

1108 <1> ; // the next shortname checksum,

1109 <1> ; // long name is valid.

1110 <1> ; // If a longname is longer than 65 bytes,

1111 <1> ; // it is invalid for trdos. (>45h)

1112 <1>

1113 0000A3C0 57 <1> push edi

1114 0000A3C1 56 <1> push esi

1115 <1> ;push ebx

1116 <1> ;push ecx

1117 <1> ;push edx

1118 0000A3C2 50 <1> push eax

1119 <1>

1120 0000A3C3 29C9 <1> sub ecx, ecx

1121 <1> ;sub eax, eax

1122 0000A3C5 B11A <1> mov cl, 26

1123 <1>

1124 0000A3C7 0FB607 <1> movzx eax, byte [edi] ; LDIR\_Order

1125 0000A3CA 3C41 <1> cmp al, 41h ; 40h (last long entry sign) + 1

1126 0000A3CC 722B <1> jb short pass\_pslnsc\_last\_long\_entry

1127 <1>

1128 0000A3CE 88C4 <1> mov ah, al

1129 0000A3D0 80EC40 <1> sub ah, 40h

1130 0000A3D3 8825[FE610100] <1> mov [LFN\_EntryLength], ah

1131 <1>

1132 0000A3D9 3C45 <1> cmp al, 45h ; 40h (last long entry sign) + 5

1133 <1> ; Max 130 byte length is usable in TRDOS

1134 <1> ; 26\*5 = 130

1135 0000A3DB 7753 <1> ja short loc\_pslnsc\_retn

1136 <1>

1137 0000A3DD 2407 <1> and al, 07h ; 0Fh

1138 0000A3DF A2[FD610100] <1> mov [LongNameFound], al

1139 <1>

1140 0000A3E4 FEC8 <1> dec al

1141 <1> ;mov cl, 26

1142 0000A3E6 F6E1 <1> mul cl

1143 <1>

1144 0000A3E8 89C6 <1> mov esi, eax

1145 0000A3EA 01CE <1> add esi, ecx

1146 <1> ; to make is an ASCIIZ string

1147 <1> ; with ax+26 bytes length

1148 0000A3EC 81C6[00620100] <1> add esi, LongFileName

1149 0000A3F2 66C7060000 <1> mov word [esi], 0

1150 0000A3F7 EB16 <1> jmp short loc\_pslsc\_move\_ldir\_name2

1151 <1>

1152 <1> pass\_pslnsc\_last\_long\_entry:

1153 0000A3F9 3C04 <1> cmp al, 04h

1154 0000A3FB 7733 <1> ja short loc\_pslnsc\_retn

1155 0000A3FD FE0D[FD610100] <1> dec byte [LongNameFound]

1156 0000A403 3A05[FD610100] <1> cmp al, [LongNameFound]

1157 0000A409 7525 <1> jne short loc\_pslnsc\_retn

1158 <1>

1159 <1> loc\_pslsc\_move\_ldir\_name1:

1160 0000A40B FEC8 <1> dec al

1161 <1> ;mov cl, 26

1162 0000A40D F6E1 <1> mul cl

1163 <1>

1164 <1> loc\_pslsc\_move\_ldir\_name2:

1165 0000A40F 8A4F0D <1> mov cl, [edi+0Dh] ; long name checksum

1166 0000A412 880D[FF610100] <1> mov [LFN\_CheckSum], cl

1167 0000A418 89FE <1> mov esi, edi ; LDIR\_Order

1168 0000A41A BF[00620100] <1> mov edi, LongFileName

1169 0000A41F 01C7 <1> add edi, eax

1170 0000A421 46 <1> inc esi

1171 0000A422 B105 <1> mov cl, 5 ; chars 1 to 5

1172 0000A424 F366A5 <1> rep movsw

1173 0000A427 83C603 <1> add esi, 3

1174 0000A42A A5 <1> movsd ; char 6 & 7

1175 0000A42B A5 <1> movsd ; char 8 & 9

1176 0000A42C A5 <1> movsd ; char 10 & 11

1177 0000A42D 46 <1> inc esi

1178 0000A42E 46 <1> inc esi

1179 0000A42F A5 <1> movsd ; char 12 & 13

1180 <1>

1181 <1> loc\_pslnsc\_retn:

1182 0000A430 58 <1> pop eax

1183 <1> ;pop edx

1184 <1> ;pop ecx

1185 <1> ;pop ebx

1186 0000A431 5E <1> pop esi

1187 0000A432 5F <1> pop edi

1188 <1>

1189 0000A433 C3 <1> retn

1190 <1>

1191 <1> parse\_path\_name:

1192 <1> ; 10/02/2016

1193 <1> ; 08/02/2016 (TRDOS 386 = TRDOS v2.0)

1194 <1> ; 10/009/2011 ('proc\_parse\_pathname')

1195 <1> ; 27/11/2009

1196 <1> ; 05/12/2004

1197 <1> ;

1198 <1> ; INPUT ->

1199 <1> ; ESI = Beginning of ASCIIZ pathname string

1200 <1> ; EDI = Destination Address

1201 <1> ; (which is TR-DOS FindFile data buffer)

1202 <1> ; OUTPUT ->

1203 <1> ; CF = 1 -> Error

1204 <1> ; EAX = Error Code (AL)

1205 <1> ;

1206 <1> ; (Modified registers: eax, ecx, esi, edi)

1207 <1>

1208 <1> ; Clear the pathname bytes in TR-DOS Findfile data buffer

1209 0000A434 57 <1> push edi

1210 0000A435 B914000000 <1> mov ecx, 20 ; 80 bytes

1211 0000A43A 31C0 <1> xor eax, eax

1212 0000A43C F3AB <1> rep stosd

1213 0000A43E 5F <1> pop edi

1214 <1>

1215 0000A43F 668B06 <1> mov ax, [esi]

1216 0000A442 80FC3A <1> cmp ah, ':'

1217 0000A445 741C <1> je short loc\_ppn\_change\_drive

1218 0000A447 A0[FE580100] <1> mov al, [Current\_Drv]

1219 0000A44C EB33 <1> jmp short pass\_ppn\_change\_drive

1220 <1>

1221 <1> pass\_ppn\_cdir:

1222 0000A44E 8B35[22630100] <1> mov esi, [First\_Path\_Pos]

1223 0000A454 AC <1> lodsb

1224 <1> loc\_ppn\_get\_filename:

1225 0000A455 83C741 <1> add edi, 65 ; FindFile\_Name location

1226 <1> ; TRDOS Filename length must not be more than 12 bytes

1227 <1> ;mov ecx, 12

1228 0000A458 B10C <1> mov cl, 12

1229 <1> loc\_ppn\_get\_fnchar\_next:

1230 0000A45A AA <1> stosb

1231 0000A45B AC <1> lodsb

1232 0000A45C 3C21 <1> cmp al, 21h

1233 0000A45E 7274 <1> jb short loc\_ppn\_clc\_return

1234 0000A460 E2F8 <1> loop loc\_ppn\_get\_fnchar\_next

1235 <1> loc\_ppn\_return:

1236 0000A462 C3 <1> retn

1237 <1>

1238 <1> loc\_ppn\_change\_drive:

1239 0000A463 24DF <1> and al, 0DFh

1240 0000A465 2C41 <1> sub al, 'A'; A:

1241 0000A467 726F <1> jc short loc\_ppn\_invalid\_drive

1242 0000A469 3805[D20C0100] <1> cmp [Last\_DOS\_DiskNo], al

1243 0000A46F 7267 <1> jb short loc\_ppn\_invalid\_drive

1244 <1>

1245 0000A471 46 <1> inc esi

1246 0000A472 46 <1> inc esi

1247 0000A473 8A26 <1> mov ah, [esi]

1248 0000A475 80FC21 <1> cmp ah, 21h

1249 0000A478 7307 <1> jnb short pass\_ppn\_change\_drive

1250 <1>

1251 <1> loc\_ppn\_cmd\_failed:

1252 <1> ; File or directory name is not existing

1253 0000A47A 8807 <1> mov [edi], al ; Drv

1254 0000A47C 66B80100 <1> mov ax, 1 ; eax = 1

1255 <1> ; TR-DOS Error Code 01h = Bad Command Argument

1256 <1> ; MS-DOS Error Code 01h : Invalid Function Number

1257 <1> ;stc

1258 <1> ; (MainProg ErrMsg: "Bad command or file name!")

1259 0000A480 C3 <1> retn

1260 <1>

1261 <1> pass\_ppn\_change\_drive:

1262 0000A481 8935[22630100] <1> mov [First\_Path\_Pos], esi

1263 0000A487 C705[26630100]0000- <1> mov dword [Last\_Slash\_Pos], 0

1263 0000A48F 0000 <1>

1264 0000A491 AA <1> stosb

1265 0000A492 8A06 <1> mov al, [esi]

1266 <1> loc\_scan\_ppn\_dslash:

1267 0000A494 3C2F <1> cmp al, '/'

1268 0000A496 7506 <1> jne short loc\_scan\_next\_slash\_pos

1269 0000A498 8935[26630100] <1> mov [Last\_Slash\_Pos], esi

1270 <1> loc\_scan\_next\_slash\_pos:

1271 0000A49E 46 <1> inc esi

1272 0000A49F 8A06 <1> mov al, [esi]

1273 0000A4A1 3C20 <1> cmp al, 20h

1274 0000A4A3 77EF <1> ja short loc\_scan\_ppn\_dslash

1275 0000A4A5 833D[26630100]00 <1> cmp dword [Last\_Slash\_Pos], 0

1276 0000A4AC 76A0 <1> jna short pass\_ppn\_cdir

1277 <1>

1278 0000A4AE 8B0D[26630100] <1> mov ecx, [Last\_Slash\_Pos]

1279 0000A4B4 8B35[22630100] <1> mov esi, [First\_Path\_Pos]

1280 0000A4BA 29F1 <1> sub ecx, esi

1281 0000A4BC 41 <1> inc ecx

1282 <1> ;cmp ecx, 64

1283 0000A4BD 80F940 <1> cmp cl, 64

1284 0000A4C0 7715 <1> ja short loc\_ppn\_invalid\_drive\_stc

1285 <1>

1286 0000A4C2 89F8 <1> mov eax, edi ; Dest Dir String Location (65 byte)

1287 0000A4C4 F3A4 <1> rep movsb

1288 <1> ;mov [edi], cl ; 0, End of Dir String

1289 0000A4C6 8B35[26630100] <1> mov esi, [Last\_Slash\_Pos]

1290 0000A4CC 46 <1> inc esi

1291 0000A4CD 89C7 <1> mov edi, eax

1292 0000A4CF AC <1> lodsb

1293 0000A4D0 3C21 <1> cmp al, 21h

1294 0000A4D2 7381 <1> jnb short loc\_ppn\_get\_filename

1295 <1> loc\_ppn\_clc\_return:

1296 <1> ;clc

1297 0000A4D4 31C0 <1> xor eax, eax

1298 0000A4D6 C3 <1> retn

1299 <1>

1300 <1> loc\_ppn\_invalid\_drive\_stc:

1301 0000A4D7 F5 <1> cmc ; stc

1302 <1> loc\_ppn\_invalid\_drive:

1303 <1> ; cf = 1

1304 <1> ; The Drive Letter/Char < "A" or > "Z"

1305 0000A4D8 66B80F00 <1> mov ax, 0Fh

1306 <1> ; MS-DOS Error Code 0Fh = Disk Drive Invalid

1307 <1> ; (MainProg ErrMsg: "Drive not ready or read error!")

1308 0000A4DC C3 <1> retn

1309 <1>

1310 <1> find\_longname:

1311 <1> ; 13/02/2016 (TRDOS 386 = TRDOS v2.0)

1312 <1> ; 24/01/2010 (DIR.ASM, 'proc\_find\_longname')

1313 <1> ; 17/10/2009

1314 <1>

1315 <1> ; INPUT ->

1316 <1> ; ESI = DOS short file name address

1317 <1> ; for example: "filename.ext"

1318 <1> ;

1319 <1> ; OUTPUT ->

1320 <1> ; ESI = ASCIIZ longname address (cf = 0)

1321 <1> ; cf = 1 -> error number returns in EAX (AL)

1322 <1> ; AL = 0 & CF=1 -> longname not found

1323 <1> ; the file/directory has no longname

1324 <1> ; cf = 0 -> AL = FAT Type

1325 <1>

1326 <1> ; 17/10/2009

1327 <1> ; ASCIIZ string will be returned

1328 <1> ; as LongFileName

1329 <1> ; clearing/reset is not needed

1330 <1> ;mov ecx, 33

1331 <1> ;mov edi, LongFileName

1332 <1> ;sub ax, ax ; 0

1333 <1> ;rep stosw

1334 <1>

1335 <1> ;mov byte [LongNameFound], 0

1336 <1>

1337 <1> ; ESI = ASCIIZ file/directory name address

1338 <1> ; AL = Attributes AND mask

1339 <1> ; (Result of AND must be equal to AL)

1340 <1> ; AH = Negative attributes mask

1341 <1> ; (Result of AND must be ZERO)

1342 0000A4DD 66B80008 <1> mov ax, 0800h

1343 <1> ; it must not be volume name or longname

1344 0000A4E1 E87DDDFFFF <1> call find\_first\_file

1345 0000A4E6 7216 <1> jc short loc\_fln\_retn

1346 <1>

1347 <1> loc\_fln\_check\_FAT\_Type:

1348 0000A4E8 803D[FD580100]01 <1> cmp byte [Current\_FATType], 1

1349 0000A4EF 7306 <1> jnb short loc\_fln\_check\_longname\_yes\_sign

1350 <1>

1351 0000A4F1 E839000000 <1> call get\_fs\_longname

1352 0000A4F6 C3 <1> retn

1353 <1>

1354 <1> loc\_fln\_check\_longname\_yes\_sign:

1355 0000A4F7 08FF <1> or bh, bh

1356 0000A4F9 7504 <1> jnz short loc\_fln\_check\_longnamefound\_number

1357 <1> loc\_fln\_longname\_not\_found\_retn:

1358 0000A4FB 31C0 <1> xor eax, eax

1359 <1> ; cf = 1 & al = 0 -> longname not found

1360 0000A4FD F9 <1> stc

1361 <1> loc\_fln\_retn:

1362 0000A4FE C3 <1> retn

1363 <1>

1364 <1> loc\_fln\_check\_longnamefound\_number:

1365 <1> ; 'LongNameFound' is set by

1366 <1> ; by 'save\_longname\_sub\_component'

1367 <1> ; which is called from

1368 <1> ; 'find\_directory\_entry'

1369 <1> ; which is called from

1370 <1> ; 'find\_first\_file'

1371 <1> ; It must 1 if the longname is valid

1372 0000A4FF 803D[FD610100]01 <1> cmp byte [LongNameFound], 1

1373 0000A506 75F3 <1> jne short loc\_fln\_longname\_not\_found\_retn

1374 <1>

1375 <1> loc\_fln\_calculate\_checksum:

1376 0000A508 E813000000 <1> call calculate\_checksum

1377 <1> ; AL = shortname checksum

1378 <1>

1379 <1> loc\_fln\_longname\_validation:

1380 <1> ; 'LFN\_CheckSum' has been set already

1381 <1> ; by 'save\_longname\_sub\_component'

1382 <1> ; which is called from

1383 <1> ; 'find\_directory\_entry'

1384 <1> ; which is called from

1385 <1> ; 'find\_first\_file'

1386 0000A50D 3805[FF610100] <1> cmp [LFN\_CheckSum], al

1387 0000A513 75E6 <1> jne short loc\_fln\_longname\_not\_found\_retn

1388 <1>

1389 0000A515 BE[00620100] <1> mov esi, LongFileName

1390 0000A51A A0[FD580100] <1> mov al, [Current\_FATType]

1391 0000A51F C3 <1> retn

1392 <1>

1393 <1> calculate\_checksum:

1394 <1> ; 13/02/2016 (TRDOS 386 = TRDOS v2.0)

1395 <1> ; 17/10/2009 (DIR.ASM, 'proc\_calculate\_checksum')

1396 <1> ;

1397 <1> ; INPUT ->

1398 <1> ; ESI = 11 byte DOS File Name location

1399 <1> ; (in DOS Directory Entry Format)

1400 <1> ; OUTPUT ->

1401 <1> ; AL = 8 bit checksum (CRC) value

1402 <1> ;

1403 <1> ; (Modified registers: EAX, ECX, ESI)

1404 <1>

1405 <1> ; Erdogan Tan [ 17-10-2009 ]

1406 <1> ; 'ror al, 1' instruction

1407 <1>

1408 <1> ; Erdogan Tan [ 20-06-2004 ]

1409 <1> ; This 8086 assembly code is an original code

1410 <1> ; which is adapted from C code in

1411 <1> ; Microsoft FAT32 File System Specification

1412 <1> ; Version 1.03, December 6, 2000

1413 <1> ; Page 28

1414 <1>

1415 0000A520 30C0 <1> xor al, al

1416 0000A522 B90B000000 <1> mov ecx, 11

1417 <1> loc\_next\_sum:

1418 <1> ;xor ah, ah

1419 <1> ;test al, 1

1420 <1> ;jz short pass\_ah\_80h

1421 <1> ;mov ah, 80h

1422 <1> ;pass\_ah\_80h:

1423 <1> ;shr al, 1

1424 0000A527 D0C8 <1> ror al, 1 ; 17/10/2009

1425 0000A529 0206 <1> add al, [esi]

1426 0000A52B 46 <1> inc esi

1427 <1> ;add al, ah

1428 0000A52C E2F9 <1> loop loc\_next\_sum

1429 0000A52E C3 <1> retn

1430 <1>

1431 <1> get\_fs\_longname:

1432 <1> ; temporary (13/02/2016)

1433 0000A52F 31C0 <1> xor eax, eax

1434 0000A531 F9 <1> stc

1435 0000A532 C3 <1> retn

1436 <1>

1437 <1> make\_sub\_directory:

1438 <1> ; 16/10/2016

1439 <1> ; 02/03/2016, 03/03/2016

1440 <1> ; 26/02/2016, 27/02/2016

1441 <1> ; 21/02/2016 (TRDOS 386 = TRDOS v2.0)

1442 <1> ; 01/08/2011 (DIR.ASM, 'proc\_make\_directory')

1443 <1> ; 10/07/2010

1444 <1> ; INPUT ->

1445 <1> ; ESI = ASCIIZ Directory Name

1446 <1> ; CL = Directory Attributes

1447 <1> ; OUTPUT ->

1448 <1> ; EAX = New sub dir's first cluster

1449 <1> ; ESI = Logical Dos Drv Descr. Table Addr.

1450 <1> ; CF = 1 -> error code in AL (EAX)

1451 <1>

1452 <1> ;test cl, 10h ; directory

1453 <1> ;jz short loc\_make\_directory\_access\_denied

1454 <1> ;test cl, 08h ; volume name

1455 <1> ;jnz short loc\_make\_directory\_access\_denied

1456 <1>

1457 0000A533 80E107 <1> and cl, 07h

1458 0000A536 880D[7C630100] <1> mov byte [mkdir\_attrib], cl

1459 <1>

1460 0000A53C 56 <1> push esi

1461 0000A53D 31DB <1> xor ebx, ebx

1462 0000A53F 8A3D[FE580100] <1> mov bh, [Current\_Drv]

1463 0000A545 BE00010900 <1> mov esi, Logical\_DOSDisks

1464 0000A54A 01DE <1> add esi, ebx

1465 0000A54C 5B <1> pop ebx

1466 <1>

1467 <1> ; 10/07/2010 -> 1st writable disk check for trdos

1468 <1> ; LD\_DiskType = 0 for write protection (read only)

1469 0000A54D 807E0101 <1> cmp byte [esi+LD\_DiskType], 1 ; 0 = Invalid

1470 0000A551 730B <1> jnb short loc\_mkdir\_check\_file\_sytem

1471 <1> ; 16/10/2016 (13h -> 30)

1472 0000A553 B81E000000 <1> mov eax, 30 ; 'Disk write-protected' error

1473 0000A558 BA00000000 <1> mov edx, 0

1474 <1> ; err retn: EDX = 0, EBX = Dir name offset

1475 <1> ;ESI = Logical DOS drive description table address

1476 0000A55D C3 <1> retn

1477 <1>

1478 <1> ;loc\_make\_directory\_access\_denied:

1479 <1> ;mov ax, 05h ; access denied (invalid attributes input)

1480 <1> ;stc

1481 <1> ;retn

1482 <1>

1483 <1> loc\_mkdir\_check\_file\_sytem:

1484 0000A55E 807E0301 <1> cmp byte [esi+LD\_FATType], 1

1485 0000A562 730B <1> jnb short loc\_mkdir\_check\_free\_sectors

1486 <1>

1487 <1> loc\_make\_fs\_directory:

1488 0000A564 A1[F8580100] <1> mov eax, [Current\_Dir\_FCluster]

1489 <1> ; EAX = Parent directory DDT Address

1490 <1> ; ESI = Logical DOS Drive DT Address

1491 <1> ; EBX = Directory name offset (as ASCIIZ name)

1492 0000A569 E8D5150000 <1> call make\_fs\_directory

1493 0000A56E C3 <1> retn

1494 <1>

1495 <1> loc\_mkdir\_check\_free\_sectors:

1496 0000A56F 0FB64613 <1> movzx eax, byte [esi+LD\_BPB+SecPerClust]

1497 0000A573 8B4E74 <1> mov ecx, [esi+LD\_FreeSectors]

1498 0000A576 39C1 <1> cmp ecx, eax

1499 0000A578 7255 <1> jb short loc\_mkdir\_insufficient\_disk\_space

1500 <1>

1501 <1> loc\_make\_fat\_directory:

1502 0000A57A 891D[6C630100] <1> mov [mkdir\_DirName\_Offset], ebx

1503 0000A580 890D[78630100] <1> mov [mkdir\_FreeSectors], ecx

1504 <1>

1505 <1> ;mov al, [esi+LD\_BPB+SecPerClust]

1506 0000A586 A2[7E630100] <1> mov byte [mkdir\_SecPerClust], al

1507 <1>

1508 <1> loc\_mkdir\_gffc\_1:

1509 0000A58B E80F180000 <1> call get\_first\_free\_cluster

1510 0000A590 722A <1> jc short loc\_mkdir\_gffc\_retn

1511 <1>

1512 <1> ;loc\_mkdir\_gffc\_1\_cont:

1513 <1> ;cmp eax, 2

1514 <1> ;jb short loc\_mkdir\_gffc\_insufficient\_disk\_space

1515 <1>

1516 <1> ;loc\_mkdir\_gffc\_1\_save\_fcluster:

1517 0000A592 A3[70630100] <1> mov [mkdir\_FFCluster], eax

1518 <1>

1519 <1> loc\_mkdir\_locate\_ffe:

1520 <1> ; Current directory fcluster <> Directory buffer cluster

1521 <1> ; Current directory will be reloaded by

1522 <1> ; 'locate\_current\_dir\_file' procedure

1523 <1> ;

1524 <1> ; ESI = Logical DOS Drive Description Table Address

1525 <1> ;push esi ; 27/02/2016

1526 0000A597 31C0 <1> xor eax, eax

1527 0000A599 89C1 <1> mov ecx, eax

1528 0000A59B 6649 <1> dec cx ; FFFFh

1529 <1> ; CX = FFFFh -> find first deleted or free entry

1530 <1> ; ESI would be ASCIIZ filename address if the call

1531 <1> ; would not be for first free or deleted dir entry

1532 0000A59D E8D0FAFFFF <1> call locate\_current\_dir\_file

1533 0000A5A2 734C <1> jnc short loc\_mkdir\_set\_ff\_dir\_entry\_1

1534 <1> ;pop esi

1535 <1> ; ESI = Logical DOS Drive Description Table Address

1536 0000A5A4 83F802 <1> cmp eax, 2 ; cmp al, 2 ; File/Dir not found !

1537 0000A5A7 752B <1> jne short loc\_mkdir\_stc\_return

1538 <1>

1539 <1> loc\_mkdir\_add\_new\_cluster:

1540 0000A5A9 3805[FD580100] <1> cmp byte [Current\_FATType], al ; 2

1541 <1> ;cmp byte ptr [esi+LD\_FATType], 2

1542 0000A5AF 770C <1> ja short loc\_mkdir\_add\_new\_cluster\_check\_fsc

1543 0000A5B1 803D[FC580100]01 <1> cmp byte [Current\_Dir\_Level], 1

1544 <1> ;cmp byte [esi+LD\_CDirLevel], 1

1545 0000A5B8 7303 <1> jnb short loc\_mkdir\_add\_new\_cluster\_check\_fsc

1546 <1>

1547 0000A5BA B00C <1> mov al, 12 ; No more files

1548 <1> loc\_mkdir\_gffc\_retn:

1549 0000A5BC C3 <1> retn

1550 <1>

1551 <1> loc\_mkdir\_add\_new\_cluster\_check\_fsc:

1552 0000A5BD 8B0D[78630100] <1> mov ecx, [mkdir\_FreeSectors]

1553 <1> ;movzx eax, byte [mkdir\_SecPerClust]

1554 0000A5C3 A0[7E630100] <1> mov al, [mkdir\_SecPerClust]

1555 0000A5C8 66D1E0 <1> shl ax, 1 ; AX = 2 \* AX

1556 0000A5CB 39C1 <1> cmp ecx, eax

1557 0000A5CD 7350 <1> jnb short loc\_mkdir\_add\_new\_subdir\_cluster

1558 <1>

1559 <1> loc\_mkdir\_insufficient\_disk\_space:

1560 <1> ;mov edx, ecx

1561 <1> ;loc\_mkdir\_gffc\_insufficient\_disk\_space:

1562 0000A5CF 66B82700 <1> mov ax, 27h ; MSDOS err => insufficient disk space

1563 <1> ; err retn: EDX = Free sectors, EBX = Dir name offset

1564 <1> ; ESI -> Dos drive description table address

1565 <1> ;; ecx = edx

1566 <1> ;

1567 0000A5D3 C3 <1> retn

1568 <1>

1569 <1> loc\_mkdir\_stc\_return:

1570 0000A5D4 F9 <1> stc

1571 0000A5D5 C3 <1> retn

1572 <1>

1573 <1> loc\_mkdir\_gffc\_2:

1574 0000A5D6 E8C4170000 <1> call get\_first\_free\_cluster

1575 0000A5DB 72DF <1> jc short loc\_mkdir\_gffc\_retn

1576 <1>

1577 <1> ;loc\_mkdir\_gffc\_1\_cont:

1578 <1> ;cmp eax, 2

1579 <1> ;jb short loc\_mkdir\_gffc\_insufficient\_disk\_space

1580 <1>

1581 <1> ;loc\_mkdir\_gffc\_2\_save\_fcluster:

1582 0000A5DD A3[70630100] <1> mov [mkdir\_FFCluster], eax

1583 <1>

1584 0000A5E2 A1[74630100] <1> mov eax, [mkdir\_LastDirCluster]

1585 <1>

1586 0000A5E7 E842170000 <1> call load\_FAT\_sub\_directory

1587 0000A5EC 72CE <1> jc short loc\_mkdir\_gffc\_retn

1588 <1>

1589 0000A5EE 31FF <1> xor edi, edi

1590 <1> loc\_mkdir\_set\_ff\_dir\_entry\_1:

1591 <1> ; 27/02/2016

1592 0000A5F0 56 <1> push esi ; Logical DOS Drv Desc. Tbl. address

1593 <1> ; EDI = Directory Entry Address

1594 0000A5F1 8B35[6C630100] <1> mov esi, [mkdir\_DirName\_Offset]

1595 0000A5F7 A1[70630100] <1> mov eax, [mkdir\_FFCluster]

1596 <1>

1597 0000A5FC 66B91000 <1> mov cx, 10h ; CL = Directory attribute

1598 <1> ; CH = 0 -> File size is 0

1599 0000A600 0A0D[7C630100] <1> or cl, [mkdir\_attrib] ; S, H, R

1600 0000A606 E8B0010000 <1> call make\_directory\_entry

1601 <1>

1602 0000A60B 5E <1> pop esi

1603 <1>

1604 0000A60C C605[28610100]02 <1> mov byte [DirBuff\_ValidData], 2

1605 0000A613 E880020000 <1> call save\_directory\_buffer

1606 0000A618 0F83DA000000 <1> jnc loc\_mkdir\_set\_ff\_dir\_entry\_2

1607 <1>

1608 <1> loc\_mkdir\_return:

1609 0000A61E C3 <1> retn

1610 <1>

1611 <1> loc\_mkdir\_add\_new\_subdir\_cluster:

1612 0000A61F 8B15[2D610100] <1> mov edx, [DirBuff\_Cluster]

1613 0000A625 8915[74630100] <1> mov [mkdir\_LastDirCluster], edx

1614 <1>

1615 0000A62B A1[70630100] <1> mov eax, [mkdir\_FFCluster]

1616 0000A630 E8F9160000 <1> call load\_FAT\_sub\_directory

1617 0000A635 72E7 <1> jc short loc\_mkdir\_return

1618 <1> ; eax = 0

1619 <1> ; ecx = directory buffer sector count (<= 128)

1620 <1>

1621 <1> pass\_mkdir\_add\_new\_subdir\_cluster:

1622 0000A637 29FF <1> sub edi, edi ; 0

1623 <1> ;mov al, 128 ; double word

1624 <1> ;mul ecx ; ecx = directory buffer sector count

1625 <1> ;mov ecx, eax

1626 <1> ;shl cx, 7 ; 128 \* sector count

1627 0000A639 668B4611 <1> mov ax, [esi+LD\_BPB+BytesPerSec] ; 512

1628 0000A63D 66C1E802 <1> shr ax, 2 ; 'byte count / 4' for 'stosd'

1629 0000A641 66F7E1 <1> mul cx ; max = 128\*(512/4) -> 16384 (stosd)

1630 0000A644 6689C1 <1> mov cx, ax

1631 0000A647 6629C0 <1> sub ax, ax ; 0

1632 0000A64A F3AB <1> rep stosd ; clear directory buffer

1633 <1>

1634 0000A64C C605[28610100]02 <1> mov byte [DirBuff\_ValidData], 2

1635 0000A653 E840020000 <1> call save\_directory\_buffer

1636 0000A658 72C4 <1> jc short loc\_mkdir\_return

1637 <1>

1638 <1> loc\_mkdir\_save\_added\_cluster:

1639 0000A65A A1[74630100] <1> mov eax, [mkdir\_LastDirCluster]

1640 0000A65F 8B0D[70630100] <1> mov ecx, [mkdir\_FFCluster]

1641 <1> ; 01/03/2016

1642 0000A665 31D2 <1> xor edx, edx

1643 0000A667 8915[1E610100] <1> mov [FAT\_ClusterCounter], edx ; 0 ; reset

1644 0000A66D E800180000 <1> call update\_cluster

1645 0000A672 7304 <1> jnc short loc\_mkdir\_save\_fat\_buffer\_0

1646 0000A674 09C0 <1> or eax, eax ; EAX = 0 -> cluster value is 0 or eocc

1647 0000A676 7518 <1> jnz short loc\_mkdir\_save\_fat\_buffer\_stc\_retn

1648 <1>

1649 <1> loc\_mkdir\_save\_fat\_buffer\_0:

1650 0000A678 A1[70630100] <1> mov eax, [mkdir\_FFCluster]

1651 0000A67D A3[74630100] <1> mov [mkdir\_LastDirCluster], eax

1652 <1>

1653 0000A682 31C9 <1> xor ecx, ecx

1654 0000A684 49 <1> dec ecx ; FFFFFFFFh

1655 <1> ; ESI = Logical DOS Drive Description Table address

1656 0000A685 E8E8170000 <1> call update\_cluster

1657 0000A68A 731A <1> jnc short loc\_mkdir\_save\_fat\_buffer\_1

1658 0000A68C 09C0 <1> or eax, eax

1659 0000A68E 7416 <1> jz short loc\_mkdir\_save\_fat\_buffer\_1

1660 <1>

1661 <1> loc\_mkdir\_save\_fat\_buffer\_stc\_retn:

1662 <1> ; 01/03/2016

1663 0000A690 803D[1E610100]01 <1> cmp byte [FAT\_ClusterCounter], 1

1664 0000A697 720C <1> jb short loc\_mkdir\_save\_fat\_buffer\_retn

1665 <1>

1666 0000A699 66BB00FF <1> mov bx, 0FF00h ; recalculate free space (BL = 0)

1667 <1> ; (BH = FFh -> Use ESI as Drv Param. Tbl.)

1668 0000A69D 50 <1> push eax

1669 0000A69E E8211B0000 <1> call calculate\_fat\_freespace

1670 0000A6A3 58 <1> pop eax

1671 0000A6A4 F9 <1> stc

1672 <1> loc\_mkdir\_save\_fat\_buffer\_retn:

1673 0000A6A5 C3 <1> retn

1674 <1>

1675 <1> loc\_mkdir\_save\_fat\_buffer\_1:

1676 <1> ; byte [FAT\_BuffValidData] = 2

1677 0000A6A6 E8841A0000 <1> call save\_fat\_buffer

1678 0000A6AB 72E3 <1> jc short loc\_mkdir\_save\_fat\_buffer\_stc\_retn

1679 <1>

1680 <1> ; 01/03/2016

1681 0000A6AD 803D[1E610100]01 <1> cmp byte [FAT\_ClusterCounter], 1

1682 0000A6B4 721B <1> jb short loc\_mkdir\_save\_fat\_buffer\_2

1683 <1>

1684 <1> ; ESI = Logical DOS Drive Description Table address

1685 0000A6B6 A1[1E610100] <1> mov eax, [FAT\_ClusterCounter]

1686 0000A6BB 66BB01FF <1> mov bx, 0FF01h ; add free clusters

1687 0000A6BF E8001B0000 <1> call calculate\_fat\_freespace

1688 <1>

1689 <1> ;inc eax ; 0FFFFFFFFh -> 0 ; recalculation is needed!

1690 <1> ;jnz short loc\_mkdir\_save\_fat\_buffer\_2

1691 <1>

1692 <1> ; ecx > 0 -> Recalculation is needed

1693 0000A6C4 09C9 <1> or ecx, ecx

1694 0000A6C6 7409 <1> jz short loc\_mkdir\_save\_fat\_buffer\_2

1695 <1>

1696 0000A6C8 66BB00FF <1> mov bx, 0FF00h ; ; recalculate free space

1697 0000A6CC E8F31A0000 <1> call calculate\_fat\_freespace

1698 <1>

1699 <1> loc\_mkdir\_save\_fat\_buffer\_2:

1700 0000A6D1 C605[7F630100]01 <1> mov byte [mkdir\_add\_new\_cluster], 1

1701 0000A6D8 E9C4000000 <1> jmp loc\_mkdir\_upd\_parent\_dir\_lmdt

1702 <1>

1703 <1> loc\_mkdir\_update\_sub\_dir\_cluster:

1704 0000A6DD A1[70630100] <1> mov eax, [mkdir\_FFCluster]

1705 0000A6E2 29C9 <1> sub ecx, ecx ; 0

1706 <1> ; 01/03/2016

1707 0000A6E4 890D[1E610100] <1> mov [FAT\_ClusterCounter], ecx ; 0 ; Reset

1708 0000A6EA 49 <1> dec ecx ; 0FFFFFFFFh

1709 <1>

1710 <1> ; ESI = Logical DOS Drive Descisption Table address

1711 0000A6EB E882170000 <1> call update\_cluster

1712 0000A6F0 7379 <1> jnc short loc\_mkdir\_save\_fat\_buffer\_3

1713 0000A6F2 09C0 <1> or eax, eax ; EAX = 0 -> cluster value is 0 or eocc

1714 0000A6F4 7475 <1> jz short loc\_mkdir\_save\_fat\_buffer\_3

1715 <1> ; 01/03/2016

1716 0000A6F6 EB98 <1> jmp short loc\_mkdir\_save\_fat\_buffer\_stc\_retn

1717 <1>

1718 <1> loc\_mkdir\_set\_ff\_dir\_entry\_2:

1719 <1> ; ESI = Logical DOS Drive Description Table address

1720 0000A6F8 A1[70630100] <1> mov eax, [mkdir\_FFCluster]

1721 <1> ; Load disk sectors as a directory cluster

1722 0000A6FD E82C160000 <1> call load\_FAT\_sub\_directory

1723 0000A702 7266 <1> jc short retn\_make\_fat\_directory

1724 <1>

1725 <1> ; eax = 0

1726 <1> ; ecx = directory buffer sector count (<= 128)

1727 <1>

1728 0000A704 BF40000800 <1> mov edi, Directory\_Buffer + 64 ; 26/02/2016

1729 <1>

1730 <1> ; 02/03/2016

1731 0000A709 668B4611 <1> mov ax, [esi+LD\_BPB+BytesPerSec] ; 512

1732 0000A70D 66C1E802 <1> shr ax, 2 ; 'byte count / 4' for 'stosd'

1733 0000A711 F7E1 <1> mul ecx

1734 0000A713 89C1 <1> mov ecx, eax

1735 0000A715 6629C0 <1> sub ax, ax

1736 0000A718 F3AB <1> rep stosd

1737 <1>

1738 <1> ;;mov al, 128 ; double word

1739 <1> ;;mul ecx ; ecx = directory buffer sector count

1740 <1> ;;mov ecx, eax

1741 <1> ;shl cx, 7 ; 128 \* sector count

1742 <1> ;;sub eax, eax

1743 <1> ;;sub al, al ; 0

1744 <1> ;rep stosd ; clear directory buffer

1745 <1>

1746 0000A71A BF00000800 <1> mov edi, Directory\_Buffer ; 26/02/2016

1747 <1>

1748 0000A71F 56 <1> push esi

1749 <1>

1750 0000A720 BE[80630100] <1> mov esi, mkdir\_Name

1751 0000A725 66C7062E00 <1> mov word [esi], 2Eh ; db '.', '0'

1752 <1>

1753 0000A72A A1[70630100] <1> mov eax, [mkdir\_FFCluster]

1754 0000A72F 66B91000 <1> mov cx, 10h ; CL = Directory attribute

1755 <1> ; CH = 0 -> File size is 0

1756 0000A733 E883000000 <1> call make\_directory\_entry

1757 <1>

1758 0000A738 BF20000800 <1> mov edi, Directory\_Buffer + 32 ; 26/02/2016

1759 <1>

1760 <1> ; 03/03/2016

1761 <1> ; Following modification has been done according to

1762 <1> ; 'Microsoft Extensible Firmware Initiative

1763 <1> ; FAT32 File System Specification' document,

1764 <1> ; 'FAT: General Overview of On-Disk Format—Page 25'.

1765 <1> ; "Finally, you set DIR\_FstClusLO and DIR\_FstClusHI

1766 <1> ; for the dotdot entry (the second entry) to the

1767 <1> ; first cluster number of the directory in which you

1768 <1> ; just created the directory (value is 0 if this directory

1769 <1> ; is the root directory even for FAT32 volumes)."

1770 <1> ; (Correctness of this modification has been verified

1771 <1> ; by using Windows 98 'scandisk.exe'.)

1772 <1>

1773 0000A73D 29C0 <1> sub eax, eax

1774 0000A73F 3805[FC580100] <1> cmp byte [Current\_Dir\_Level], al ; 0

1775 0000A745 7605 <1> jna short loc\_mkdir\_set\_ff\_dir\_entry\_3

1776 0000A747 A1[F8580100] <1> mov eax, [Current\_Dir\_FCluster] ; parent dir

1777 <1> loc\_mkdir\_set\_ff\_dir\_entry\_3:

1778 0000A74C 66C746012E00 <1> mov word [esi+1], 2Eh ; db '.', '0'

1779 <1>

1780 <1> ;mov cx, 10h

1781 0000A752 E864000000 <1> call make\_directory\_entry

1782 <1>

1783 0000A757 5E <1> pop esi

1784 <1>

1785 0000A758 C605[28610100]02 <1> mov byte [DirBuff\_ValidData], 2

1786 0000A75F E834010000 <1> call save\_directory\_buffer

1787 0000A764 0F8373FFFFFF <1> jnc loc\_mkdir\_update\_sub\_dir\_cluster

1788 <1>

1789 <1> retn\_make\_fat\_directory:

1790 0000A76A C3 <1> retn

1791 <1>

1792 <1> loc\_mkdir\_save\_fat\_buffer\_3:

1793 <1> ; 01/03/2016

1794 <1> ; byte [FAT\_BuffValidData] = 2

1795 0000A76B E8BF190000 <1> call save\_fat\_buffer

1796 0000A770 0F821AFFFFFF <1> jc loc\_mkdir\_save\_fat\_buffer\_stc\_retn

1797 <1>

1798 0000A776 803D[1E610100]01 <1> cmp byte [FAT\_ClusterCounter], 1

1799 0000A77D 721B <1> jb short loc\_mkdir\_save\_fat\_buffer\_4

1800 <1>

1801 <1> ; ESI = Logical DOS Drive Description Table address

1802 0000A77F A1[1E610100] <1> mov eax, [FAT\_ClusterCounter]

1803 0000A784 66BB01FF <1> mov bx, 0FF01h ; add free clusters

1804 0000A788 E8371A0000 <1> call calculate\_fat\_freespace

1805 <1>

1806 <1> ;inc eax ; 0FFFFFFFFh -> 0 ; recalculation is needed!

1807 <1> ;jnz short loc\_mkdir\_save\_fat\_buffer\_4

1808 <1>

1809 <1> ; ecx > 0 -> Recalculation is needed

1810 0000A78D 09C9 <1> or ecx, ecx

1811 0000A78F 7409 <1> jz short loc\_mkdir\_save\_fat\_buffer\_4

1812 <1>

1813 0000A791 66BB00FF <1> mov bx, 0FF00h ; recalculate free space

1814 0000A795 E82A1A0000 <1> call calculate\_fat\_freespace

1815 <1>

1816 <1> loc\_mkdir\_save\_fat\_buffer\_4:

1817 0000A79A C605[7F630100]00 <1> mov byte [mkdir\_add\_new\_cluster], 0

1818 <1>

1819 <1> loc\_mkdir\_upd\_parent\_dir\_lmdt:

1820 0000A7A1 E88D010000 <1> call update\_parent\_dir\_lmdt

1821 <1>

1822 <1> ; 01/03/2016

1823 0000A7A6 803D[7F630100]00 <1> cmp byte [mkdir\_add\_new\_cluster], 0

1824 0000A7AD 0F8723FEFFFF <1> ja loc\_mkdir\_gffc\_2

1825 <1>

1826 <1> loc\_mkdir\_retn\_new\_dir\_cluster:

1827 0000A7B3 A1[70630100] <1> mov eax, [mkdir\_FFCluster]

1828 0000A7B8 31D2 <1> xor edx, edx

1829 <1> loc\_mkdir\_retn:

1830 0000A7BA C3 <1> retn

1831 <1>

1832 <1> make\_directory\_entry:

1833 <1> ; 02/03/2016

1834 <1> ; 21/02/2016 (TRDOS 386 = TRDOS v2.0)

1835 <1> ; 09/08/2010 (DIR.ASM, 'proc\_make\_directory\_entry')

1836 <1> ; 17/07/2010

1837 <1> ; INPUT ->

1838 <1> ; EDI = Directory Entry Address

1839 <1> ; ESI = Dot File Name Location

1840 <1> ; EAX = First Cluster

1841 <1> ; File Size = 0 (Must be set later)

1842 <1> ; CL = Attributes

1843 <1> ; CH = 0 (File size = 0)

1844 <1> ; (If CH>0, File size is in dword [EBX]) (\*)

1845 <1> ; OUTPUT ->

1846 <1> ; EDI = Directory Entry Address

1847 <1> ; ESI = Dot File Name Location (Capitalized)

1848 <1> ; If CH input = 0, File Size = 0

1849 <1> ; Otherwise file size is as dword [EBX] (\*)

1850 <1> ; DX = Date, AX = Time in DOS Dir Entry format

1851 <1> ; EBX = same

1852 <1> ; ECX = same

1853 <1>

1854 0000A7BB 51 <1> push ecx

1855 <1>

1856 0000A7BC 884F0B <1> mov [edi+11], cl ; Attributes

1857 0000A7BF 6689471A <1> mov [edi+26], ax ; FClusterLw, 26

1858 0000A7C3 C1E810 <1> shr eax, 16

1859 0000A7C6 66894714 <1> mov [edi+20], ax ; FClusterHw, 20

1860 0000A7CA 6631C0 <1> xor ax, ax

1861 0000A7CD 6689470C <1> mov [edi+12], ax ; NTReserved, 12

1862 <1> ; CrtTimeTenth, 13

1863 0000A7D1 08ED <1> or ch, ch

1864 0000A7D3 7402 <1> jz short loc\_make\_direntry\_set\_filesize

1865 <1>

1866 0000A7D5 8B03 <1> mov eax, [ebx]

1867 <1>

1868 <1> loc\_make\_direntry\_set\_filesize:

1869 0000A7D7 89471C <1> mov [edi+28], eax ; FileSize, 28

1870 <1>

1871 0000A7DA E88AFBFFFF <1> call convert\_file\_name

1872 <1> ;EDI = Dir Entry Format File Name Location

1873 <1> ;ESI = Dot File Name Location (capitalized)

1874 <1>

1875 0000A7DF E816000000 <1> call convert\_current\_date\_time

1876 <1> ; OUTPUT -> DX = Date in dos dir entry format

1877 <1> ; AX = Time in dos dir entry format

1878 0000A7E4 6689470E <1> mov [edi+14], ax ; CrtTime, 14

1879 0000A7E8 66895710 <1> mov [edi+16], dx ; CrtDate, 16

1880 0000A7EC 66895712 <1> mov [edi+18], dx ; LastAccDate, 18

1881 0000A7F0 66894716 <1> mov [edi+22], ax ; WrtTime, 14

1882 0000A7F4 66895718 <1> mov [edi+24], dx ; WrtDate, 16

1883 0000A7F8 59 <1> pop ecx

1884 <1>

1885 0000A7F9 C3 <1> retn

1886 <1>

1887 <1> convert\_current\_date\_time:

1888 <1> ; 21/02/2016 (TRDOS 386 = TRDOS v2.0)

1889 <1> ; 13/06/2010 (DIR.ASM, 'proc\_convert\_current\_date\_time')

1890 <1> ; converts date&time to dos dir entry format

1891 <1> ; INPUT -> none

1892 <1> ; OUTPUT -> DX = Date in dos dir entry format

1893 <1> ; AX = Time in dos dir entry format

1894 <1>

1895 0000A7FA B404 <1> mov ah, 04h ; Return Current Date

1896 0000A7FC E879B1FFFF <1> call int1Ah

1897 <1>

1898 0000A801 88E8 <1> mov al, ch ; <- century BCD

1899 0000A803 240F <1> and al, 0Fh

1900 0000A805 88EC <1> mov ah, ch

1901 0000A807 C0EC04 <1> shr ah, 4

1902 0000A80A D50A <1> aad

1903 0000A80C 88C5 <1> mov ch, al ; -> century

1904 <1>

1905 0000A80E 88C8 <1> mov al, cl ; <- year BCD

1906 0000A810 240F <1> and al, 0Fh

1907 0000A812 88CC <1> mov ah, cl

1908 0000A814 C0EC04 <1> shr ah, 4

1909 0000A817 D50A <1> aad

1910 0000A819 88C1 <1> mov cl, al ; -> year

1911 <1>

1912 0000A81B 88E8 <1> mov al, ch

1913 0000A81D B464 <1> mov ah, 100

1914 0000A81F F6E4 <1> mul ah

1915 0000A821 30ED <1> xor ch, ch

1916 0000A823 6601C8 <1> add ax, cx

1917 0000A826 662DBC07 <1> sub ax, 1980 ; ms-dos epoch

1918 0000A82A 6689C1 <1> mov cx, ax

1919 <1>

1920 0000A82D 88F0 <1> mov al, dh ; <- month in bcd

1921 0000A82F 240F <1> and al, 0Fh

1922 0000A831 88F4 <1> mov ah, dh

1923 0000A833 C0EC04 <1> shr ah, 4

1924 0000A836 D50A <1> aad

1925 0000A838 88C6 <1> mov dh, al ; -> month

1926 <1>

1927 0000A83A 88D0 <1> mov al, dl ; <- day BCD

1928 0000A83C 240F <1> and al, 0Fh

1929 0000A83E 88D4 <1> mov ah, dl

1930 0000A840 C0EC04 <1> shr ah, 4

1931 0000A843 D50A <1> aad

1932 0000A845 88C2 <1> mov dl, al ; -> day

1933 <1>

1934 0000A847 88C8 <1> mov al, cl ; count of years from 1980

1935 0000A849 66C1E004 <1> shl ax, 4

1936 0000A84D 08F0 <1> or al, dh ; month of year, 1 to 12

1937 0000A84F 66C1E005 <1> shl ax, 5

1938 0000A853 08D0 <1> or al, dl ; day of year, 1 to 31

1939 <1>

1940 0000A855 6650 <1> push ax ; push date

1941 <1>

1942 0000A857 B402 <1> mov ah, 02h ; Return Current Time

1943 0000A859 E81CB1FFFF <1> call int1Ah

1944 <1>

1945 0000A85E 88E8 <1> mov al, ch ; <- hours BCD

1946 0000A860 240F <1> and al, 0Fh

1947 0000A862 88EC <1> mov ah, ch

1948 0000A864 C0EC04 <1> shr ah, 4

1949 0000A867 D50A <1> aad

1950 0000A869 88C5 <1> mov ch, al ; -> hours

1951 <1>

1952 0000A86B 88C8 <1> mov al, cl ; <- minutes BCD

1953 0000A86D 240F <1> and al, 0Fh

1954 0000A86F 88CC <1> mov ah, cl

1955 0000A871 C0EC04 <1> shr ah, 4

1956 0000A874 D50A <1> aad

1957 0000A876 88C1 <1> mov cl, al ; -> minutes

1958 <1>

1959 0000A878 88F0 <1> mov al, dh ; <- seconds BCD

1960 0000A87A 240F <1> and al, 0Fh

1961 0000A87C 88F4 <1> mov ah, dh

1962 0000A87E C0EC04 <1> shr ah, 4

1963 0000A881 D50A <1> aad

1964 0000A883 88C6 <1> mov dh, al ; -> seconds

1965 <1>

1966 0000A885 88E8 <1> mov al, ch ; hours

1967 0000A887 66C1E006 <1> shl ax, 6

1968 0000A88B 08C8 <1> or al, cl ; minutes

1969 0000A88D 66C1E005 <1> shl ax, 5

1970 0000A891 D0EE <1> shr dh, 1 ; 2 seconds

1971 <1> ; There is a bug in TRDOS v1 here !

1972 <1> ; it was 'or al, dl' !

1973 0000A893 08F0 <1> or al, dh ; seconds

1974 <1>

1975 0000A895 665A <1> pop dx ; pop date

1976 <1>

1977 0000A897 C3 <1> retn

1978 <1>

1979 <1> save\_directory\_buffer:

1980 <1> ; 15/10/2016

1981 <1> ; 23/03/2016

1982 <1> ; 26/02/2016

1983 <1> ; 22/02/2016 (TRDOS 386 = TRDOS v2.0)

1984 <1> ; 01/08/2011

1985 <1> ; 14/03/2010

1986 <1> ; INPUT ->

1987 <1> ; none

1988 <1> ; OUTPUT ->

1989 <1> ; cf = 0 -> write OK...

1990 <1> ; cf = 1 -> error code in AL (EAX)

1991 <1> ; cf = 1 & AL = 0Dh => CH & CL = FS & FAT type

1992 <1> ; EBX = Directory Buffer Address

1993 <1> ;

1994 <1> ; (EAX, ECX, EDX will be modified)

1995 <1>

1996 0000A898 BB00000800 <1> mov ebx, Directory\_Buffer

1997 0000A89D 803D[28610100]02 <1> cmp byte [DirBuff\_ValidData], 2

1998 0000A8A4 7403 <1> je short loc\_save\_dir\_buffer

1999 0000A8A6 31C0 <1> xor eax, eax

2000 0000A8A8 C3 <1> retn

2001 <1>

2002 <1> loc\_save\_dir\_buffer:

2003 0000A8A9 56 <1> push esi

2004 0000A8AA 31DB <1> xor ebx, ebx

2005 0000A8AC 8A3D[26610100] <1> mov bh, [DirBuff\_DRV]

2006 0000A8B2 80EF41 <1> sub bh, 'A'

2007 0000A8B5 BE00010900 <1> mov esi, Logical\_DOSDisks

2008 0000A8BA 01DE <1> add esi, ebx

2009 0000A8BC 668B4E03 <1> mov cx, [esi+LD\_FATType]

2010 <1> ; CH = FS Type (A1h for FS)

2011 <1> ; CL = FAT Type (0 for FS)

2012 0000A8C0 08C9 <1> or cl, cl

2013 0000A8C2 7433 <1> jz short loc\_save\_dir\_buff\_stc\_retn

2014 <1>

2015 <1> loc\_save\_dir\_buffer\_check\_cluster\_no:

2016 0000A8C4 A1[2D610100] <1> mov eax, [DirBuff\_Cluster]

2017 0000A8C9 28FF <1> sub bh, bh ; ebx = 0

2018 0000A8CB 09C0 <1> or eax, eax

2019 0000A8CD 7540 <1> jnz short loc\_save\_sub\_dir\_buffer

2020 0000A8CF 8A25[27610100] <1> mov ah, [DirBuff\_FATType]

2021 0000A8D5 FEC3 <1> inc bl ; bl = 1

2022 0000A8D7 38DC <1> cmp ah, bl

2023 0000A8D9 721D <1> jb short loc\_save\_dir\_buff\_inv\_data\_retn

2024 0000A8DB FEC3 <1> inc bl ; bl = 2

2025 0000A8DD 38E3 <1> cmp bl, ah

2026 0000A8DF 7217 <1> jb short loc\_save\_dir\_buff\_inv\_data\_retn

2027 <1>

2028 <1> loc\_save\_root\_dir\_buffer:

2029 0000A8E1 668B5E17 <1> mov bx, [esi+LD\_BPB+RootDirEnts]

2030 0000A8E5 6683C30F <1> add bx, 15

2031 0000A8E9 66C1EB04 <1> shr bx, 4 ; 16 dir entries per sector

2032 0000A8ED 6609DB <1> or bx, bx

2033 0000A8F0 7405 <1> jz short loc\_save\_dir\_buff\_stc\_retn

2034 <1> ;mov ecx, ebx

2035 0000A8F2 8B4664 <1> mov eax, [esi+LD\_ROOTBegin] ; 26/02/2016

2036 0000A8F5 EB23 <1> jmp short loc\_write\_directory\_to\_disk

2037 <1>

2038 <1> loc\_save\_dir\_buff\_stc\_retn:

2039 0000A8F7 F9 <1> stc

2040 <1> loc\_save\_dir\_buff\_inv\_data\_retn:

2041 <1> ; 15/10/2016 (0Dh -> 29)

2042 0000A8F8 B01D <1> mov al, 29 ; Invalid data !

2043 0000A8FA C605[28610100]00 <1> mov byte [DirBuff\_ValidData], 0

2044 0000A901 EB05 <1> jmp short loc\_save\_dir\_buff\_retn

2045 <1>

2046 <1> loc\_write\_directory\_to\_disk\_err:

2047 <1> ; 15/10/2016 (disk write error code, 1Dh -> 18)

2048 0000A903 B812000000 <1> mov eax, 18 ; Drive not ready or write error

2049 <1>

2050 <1> loc\_save\_dir\_buff\_retn:

2051 0000A908 BB00000800 <1> mov ebx, Directory\_Buffer

2052 0000A90D 5E <1> pop esi

2053 0000A90E C3 <1> retn

2054 <1>

2055 <1> loc\_save\_sub\_dir\_buffer:

2056 <1> ; ebx = 0

2057 0000A90F 83E802 <1> sub eax, 2

2058 0000A912 8A5E13 <1> mov bl, [esi+LD\_BPB+SecPerClust]

2059 0000A915 F7E3 <1> mul ebx

2060 0000A917 034668 <1> add eax, [esi+LD\_DATABegin]

2061 <1> ;mov ecx, ebx

2062 <1>

2063 <1> loc\_write\_directory\_to\_disk:

2064 0000A91A 89D9 <1> mov ecx, ebx

2065 0000A91C BB00000800 <1> mov ebx, Directory\_Buffer

2066 0000A921 E8A34E0000 <1> call disk\_write

2067 0000A926 72DB <1> jc short loc\_write\_directory\_to\_disk\_err

2068 <1>

2069 <1> loc\_save\_dir\_buff\_validate\_retn:

2070 0000A928 C605[28610100]01 <1> mov byte [DirBuff\_ValidData], 1

2071 0000A92F 31C0 <1> xor eax, eax

2072 <1> ; 26/02/2016

2073 0000A931 EBD5 <1> jmp short loc\_save\_dir\_buff\_retn

2074 <1>

2075 <1> update\_parent\_dir\_lmdt:

2076 <1> ; 29/12/2017

2077 <1> ; 22/02/2016 (TRDOS 386 = TRDOS v2.0)

2078 <1> ; 01/08/2011

2079 <1> ; 16/10/2010

2080 <1> ;

2081 <1> ; INPUT ->

2082 <1> ; none

2083 <1> ; OUTPUT ->

2084 <1> ; (last modification date & time of the parent dir

2085 <1> ; will be changed/updated)

2086 <1> ;

2087 <1> ; (EAX, EBX, ECX, EDX, EDI will be changed)

2088 <1>

2089 0000A933 29C0 <1> sub eax, eax

2090 0000A935 8A25[FC580100] <1> mov ah, [Current\_Dir\_Level]

2091 0000A93B A0[FD580100] <1> mov al, [Current\_FATType]

2092 0000A940 3C01 <1> cmp al, 1

2093 0000A942 723A <1> jb short loc\_UPDLMDT\_proc\_retn

2094 <1>

2095 <1> loc\_update\_parent\_dir\_lm\_date\_time:

2096 0000A944 08E4 <1> or ah, ah

2097 0000A946 7436 <1> jz short loc\_UPDLMDT\_proc\_retn

2098 <1>

2099 0000A948 56 <1> push esi ; \*

2100 0000A949 8825[A0630100] <1> mov [UPDLMDT\_CDirLevel], ah

2101 0000A94F 8B15[F8580100] <1> mov edx, [Current\_Dir\_FCluster]

2102 0000A955 8915[A1630100] <1> mov [UPDLMDT\_CDirFCluster], edx

2103 <1>

2104 0000A95B FECC <1> dec ah

2105 0000A95D B90C000000 <1> mov ecx, 12

2106 0000A962 BE[5F610100] <1> mov esi, PATH\_Array

2107 <1>

2108 0000A967 8825[FC580100] <1> mov [Current\_Dir\_Level], ah

2109 0000A96D 08E4 <1> or ah, ah

2110 0000A96F 750E <1> jnz short loc\_update\_parent\_dir\_lmdt\_load\_sub\_dir\_1

2111 0000A971 803D[FD580100]02 <1> cmp byte [Current\_FATType], 2

2112 0000A978 770B <1> ja short loc\_update\_parent\_dir\_lmdt\_load\_sub\_dir\_2

2113 0000A97A 28C0 <1> sub al, al ; eax = 0

2114 0000A97C EB0A <1> jmp short loc\_update\_parent\_dir\_lmdt\_load\_sub\_dir\_3

2115 <1>

2116 <1> loc\_UPDLMDT\_proc\_retn:

2117 0000A97E C3 <1> retn

2118 <1>

2119 <1> loc\_update\_parent\_dir\_lmdt\_load\_sub\_dir\_1:

2120 0000A97F B010 <1> mov al, 16

2121 0000A981 F6E4 <1> mul ah

2122 0000A983 01C6 <1> add esi, eax

2123 <1>

2124 <1> loc\_update\_parent\_dir\_lmdt\_load\_sub\_dir\_2:

2125 0000A985 8B460C <1> mov eax, [esi+12] ; Parent Dir First Cluster

2126 <1>

2127 <1> loc\_update\_parent\_dir\_lmdt\_load\_sub\_dir\_3:

2128 0000A988 A3[F8580100] <1> mov [Current\_Dir\_FCluster], eax

2129 <1>

2130 0000A98D 83C610 <1> add esi, 16

2131 0000A990 66BF[8662] <1> mov di, Dir\_File\_Name

2132 0000A994 F3A4 <1> rep movsb

2133 <1>

2134 0000A996 BE00010900 <1> mov esi, Logical\_DOSDisks

2135 0000A99B 29DB <1> sub ebx, ebx

2136 0000A99D 8A3D[FE580100] <1> mov bh, [Current\_Drv]

2137 0000A9A3 01DE <1> add esi, ebx

2138 0000A9A5 E88FF7FFFF <1> call reload\_current\_directory

2139 0000A9AA 7230 <1> jc short loc\_update\_parent\_dir\_lmdt\_restore\_cdirlevel

2140 <1>

2141 <1> loc\_update\_parent\_dir\_lmdt\_locate\_dir:

2142 0000A9AC BE[86620100] <1> mov esi, Dir\_File\_Name

2143 0000A9B1 6631C9 <1> xor cx, cx

2144 0000A9B4 66B81008 <1> mov ax, 0810h ; Only directories

2145 0000A9B8 E8B5F6FFFF <1> call locate\_current\_dir\_file

2146 <1> ; EDI = DirBuff Directory Entry Address

2147 0000A9BD 721D <1> jc short loc\_update\_parent\_dir\_lmdt\_restore\_cdirlevel

2148 <1>

2149 0000A9BF E836FEFFFF <1> call convert\_current\_date\_time

2150 0000A9C4 66895712 <1> mov [edi+18], dx ; Last Access Date

2151 0000A9C8 66895718 <1> mov [edi+24], dx ; Last Write Date

2152 0000A9CC 66894716 <1> mov [edi+22], ax ; Last Write Time

2153 <1>

2154 0000A9D0 C605[28610100]02 <1> mov byte [DirBuff\_ValidData], 2

2155 0000A9D7 E8BCFEFFFF <1> call save\_directory\_buffer

2156 <1> ; 29/12/2017

2157 <1> ;jc short loc\_update\_parent\_dir\_lmdt\_restore\_cdirlevel

2158 <1> ;xor al, al

2159 <1> loc\_update\_parent\_dir\_lmdt\_restore\_cdirlevel:

2160 <1> ;current directory level restoration

2161 0000A9DC 8A25[A0630100] <1> mov ah, [UPDLMDT\_CDirLevel]

2162 0000A9E2 8825[FC580100] <1> mov [Current\_Dir\_Level], ah

2163 0000A9E8 8B15[A1630100] <1> mov edx, [UPDLMDT\_CDirFCluster]

2164 0000A9EE 8915[F8580100] <1> mov [Current\_Dir\_FCluster], edx

2165 <1>

2166 0000A9F4 5E <1> pop esi ; \*

2167 0000A9F5 C3 <1> retn

2168 <1>

2169 <1> delete\_longname:

2170 <1> ; 27/02/2016 (TRDOS 386 = TRDOS v2.0)

2171 <1> ; 01/08/2011 (DIR.ASM, 'proc\_delete\_longname')

2172 <1> ; 14/03/2010

2173 <1> ; INPUT ->

2174 <1> ; EAX = Directory Entry (Index) Number (< 65536)

2175 <1> ; OUTPUT ->

2176 <1> ; cf = 0 -> OK (EAX = 0)

2177 <1> ; cf = 1 -> error code in EAX (AL)

2178 <1> ;

2179 <1> ; (Modified registers: EAX, EDX, ECX, EBX, EDI)

2180 <1>

2181 0000A9F6 66A3[D0630100] <1> mov [DLN\_EntryNumber], ax

2182 0000A9FC C605[D2630100]40 <1> mov byte [DLN\_40h], 40h

2183 <1>

2184 0000AA03 E858000000 <1> call locate\_current\_dir\_entry

2185 0000AA08 7308 <1> jnc short loc\_dln\_check\_attributes

2186 0000AA0A C3 <1> retn

2187 <1>

2188 <1> loc\_dln\_longname\_not\_found:

2189 0000AA0B B802000000 <1> mov eax, 2

2190 0000AA10 F9 <1> stc

2191 0000AA11 C3 <1> retn

2192 <1>

2193 <1> loc\_dln\_check\_attributes:

2194 0000AA12 B00F <1> mov al, 0Fh ; long name

2195 0000AA14 8A670B <1> mov ah, [edi+0Bh] ; dir entry attributes

2196 0000AA17 38C4 <1> cmp ah, al

2197 0000AA19 75F0 <1> jne short loc\_dln\_longname\_not\_found

2198 0000AA1B 8A27 <1> mov ah, [edi]

2199 0000AA1D 2A25[D2630100] <1> sub ah, [DLN\_40h]

2200 0000AA23 76E6 <1> jna short loc\_dln\_longname\_not\_found

2201 0000AA25 80FC14 <1> cmp ah, 14h ; 84-64=20 -> 20\*13=260 bytes

2202 0000AA28 77E1 <1> ja short loc\_dln\_longname\_not\_found

2203 <1>

2204 0000AA2A C607E5 <1> mov byte [edi], 0E5h ; deleted sign

2205 0000AA2D C605[28610100]02 <1> mov byte [DirBuff\_ValidData], 2 ; changed/write sign

2206 0000AA34 C605[D2630100]00 <1> mov byte [DLN\_40h], 0 ; 40h -> 0

2207 <1>

2208 <1> loc\_dln\_delete\_next\_ln\_entry:

2209 0000AA3B 80FC01 <1> cmp ah, 1

2210 0000AA3E 7616 <1> jna short loc\_dln\_longname\_retn

2211 <1> loc\_dln\_delete\_next\_ln\_entry\_0:

2212 0000AA40 66FF05[D0630100] <1> inc word [DLN\_EntryNumber]

2213 0000AA47 0FB705[D0630100] <1> movzx eax, word [DLN\_EntryNumber]

2214 0000AA4E E80D000000 <1> call locate\_current\_dir\_entry

2215 0000AA53 73BD <1> jnc short loc\_dln\_check\_attributes

2216 <1>

2217 <1> loc\_dln\_longname\_stc\_retn:

2218 0000AA55 C3 <1> retn

2219 <1>

2220 <1> loc\_dln\_longname\_retn:

2221 <1> ;cmp byte [DirBuff\_ValidData], 2

2222 <1> ;jne short loc\_dln\_longname\_retn\_xor\_eax

2223 0000AA56 E83DFEFFFF <1> call save\_directory\_buffer

2224 0000AA5B 72F8 <1> jc short loc\_dln\_longname\_stc\_retn

2225 <1>

2226 <1> loc\_dln\_longname\_retn\_xor\_eax:

2227 0000AA5D 31C0 <1> xor eax, eax

2228 0000AA5F C3 <1> retn

2229 <1>

2230 <1> locate\_current\_dir\_entry:

2231 <1> ; 16/10/2016

2232 <1> ; 15/10/2016

2233 <1> ; 23/03/2016

2234 <1> ; 27/02/2016 (TRDOS 386 = TRDOS v2.0)

2235 <1> ; 01/08/2011 (DIR.ASM, 'proc\_locate\_current\_dir\_entry')

2236 <1> ; 07/03/2010

2237 <1> ; INPUT ->

2238 <1> ; EAX = Directory Entry (Index) Number (< 65536)

2239 <1> ; OUTPUT ->

2240 <1> ; EDI = Directory Entry Address

2241 <1> ; EAX = Cluster Number of Directory Buffer

2242 <1> ; EBX = Directory Buffer Entry Offset

2243 <1> ; ECX = DirBuff Valid Data identifier (CL)

2244 <1> ; If CF = 0 and CL = 2 then

2245 <1> ; directory buffer modified and

2246 <1> ; must be written to disk.

2247 <1> ; If CF = 0 and CL = 1 then

2248 <1> ; dir buffer has been written to disk, already.

2249 <1> ; CF = 1 -> Error code in EAX (AL)

2250 <1> ;

2251 <1> ; (Modified registers: EAX, EDX, ECX, EBX, EDI)

2252 <1>

2253 <1> loc\_locate\_current\_dir\_entry:

2254 0000AA60 56 <1> push esi

2255 0000AA61 89C1 <1> mov ecx, eax

2256 0000AA63 BA20000000 <1> mov edx, 32

2257 0000AA68 F7E2 <1> mul edx

2258 0000AA6A A3[DC630100] <1> mov [LCDE\_ByteOffset], eax

2259 0000AA6F 31DB <1> xor ebx, ebx

2260 0000AA71 8A3D[FE580100] <1> mov bh, [Current\_Drv]

2261 0000AA77 A0[26610100] <1> mov al, [DirBuff\_DRV]

2262 0000AA7C 2C41 <1> sub al, 'A'

2263 0000AA7E BE00010900 <1> mov esi, Logical\_DOSDisks

2264 0000AA83 01DE <1> add esi, ebx

2265 0000AA85 38C7 <1> cmp bh, al

2266 0000AA87 0F8592000000 <1> jne loc\_lcde\_reload\_current\_directory

2267 <1> loc\_lcde\_cdl\_check:

2268 0000AA8D 803D[FC580100]00 <1> cmp byte [Current\_Dir\_Level], 0

2269 0000AA94 772A <1> ja short loc\_lcde\_calc\_dirbuff\_cluster\_offset

2270 <1> ; 27/02/2016

2271 <1> ; TRDOS v1 has bug here for FAT32 fs !

2272 <1> ; (Root Directory Entries for FAT32 = 0)

2273 0000AA96 807E0303 <1> cmp byte [esi+LD\_FATType], 3 ; FAT32

2274 0000AA9A 7324 <1> jnb short loc\_lcde\_calc\_dirbuff\_cluster\_offset

2275 <1>

2276 <1> loc\_lcde\_cdl\_check\_FAT12\_16:

2277 0000AA9C 668B4617 <1> mov ax, [esi+LD\_BPB+RootDirEnts]

2278 0000AAA0 6648 <1> dec ax

2279 <1> ;xor dx, dx

2280 0000AAA2 6639C8 <1> cmp ax, cx ; cx = Directory Entry (Index) Number

2281 0000AAA5 720E <1> jb short loc\_lcde\_stc\_12h\_retn

2282 0000AAA7 66890D[D4630100] <1> mov [LCDE\_EntryIndex], cx

2283 0000AAAE 31C0 <1> xor eax, eax

2284 0000AAB0 E993000000 <1> jmp loc\_lcde\_check\_dir\_buffer\_cluster

2285 <1>

2286 <1> loc\_lcde\_stc\_12h\_retn:

2287 0000AAB5 5E <1> pop esi

2288 0000AAB6 89CB <1> mov ebx, ecx

2289 0000AAB8 89D1 <1> mov ecx, edx

2290 <1> ; 16/10/2016 (12h -> 12)

2291 0000AABA B80C000000 <1> mov eax, 12 ; No more files

2292 0000AABF C3 <1> retn

2293 <1>

2294 <1> loc\_lcde\_calc\_dirbuff\_cluster\_offset:

2295 0000AAC0 8A5E13 <1> mov bl, [esi+LD\_BPB+SecPerClust]

2296 0000AAC3 30FF <1> xor bh, bh

2297 0000AAC5 668B4611 <1> mov ax, [esi+LD\_BPB+BytesPerSec]

2298 0000AAC9 66F7E3 <1> mul bx

2299 0000AACC 6609D2 <1> or dx, dx ; If bytes per cluster > 32KB it is invalid

2300 0000AACF 755D <1> jnz short loc\_lcde\_invalid\_format

2301 <1> ;mov ecx, eax

2302 0000AAD1 6689C1 <1> mov cx, ax ; BYTES PER CLUSTER

2303 0000AAD4 A1[DC630100] <1> mov eax, [LCDE\_ByteOffset]

2304 <1> ;sub edx, edx

2305 0000AAD9 F7F1 <1> div ecx

2306 0000AADB 3DFFFF0000 <1> cmp eax, 65535

2307 0000AAE0 774C <1> ja short loc\_lcde\_invalid\_format

2308 <1>

2309 <1> ; cluster sequence number of directory (< 65536)

2310 0000AAE2 66A3[D6630100] <1> mov [LCDE\_ClusterSN], ax

2311 <1>

2312 0000AAE8 6689D0 <1> mov ax, dx ; byte offset in cluster (directory buffer)

2313 0000AAEB 66BB2000 <1> mov bx, 32 ; ; 1 dir entry = 32 bytes

2314 0000AAEF 6629D2 <1> sub dx, dx ; 0

2315 0000AAF2 66F7F3 <1> div bx

2316 0000AAF5 66A3[D4630100] <1> mov [LCDE\_EntryIndex], ax ; dir entry index/sequence number

2317 <1> ; (in directory buffer/cluster)

2318 <1> loc\_lcde\_get\_current\_sub\_dir\_fcluster:

2319 0000AAFB A1[F8580100] <1> mov eax, [Current\_Dir\_FCluster]

2320 <1>

2321 <1> loc\_lcde\_get\_next\_cluster:

2322 0000AB00 66833D[D6630100]00 <1> cmp word [LCDE\_ClusterSN], 0

2323 0000AB08 763E <1> jna short loc\_lcde\_check\_dir\_buffer\_cluster

2324 0000AB0A A3[D8630100] <1> mov [LCDE\_Cluster], eax

2325 0000AB0F E834100000 <1> call get\_next\_cluster

2326 0000AB14 7220 <1> jc short loc\_lcde\_check\_gnc\_error

2327 0000AB16 66FF0D[D6630100] <1> dec word [LCDE\_ClusterSN]

2328 0000AB1D EBE1 <1> jmp short loc\_lcde\_get\_next\_cluster

2329 <1>

2330 <1> loc\_lcde\_reload\_current\_directory:

2331 0000AB1F 51 <1> push ecx

2332 0000AB20 E814F6FFFF <1> call reload\_current\_directory

2333 0000AB25 59 <1> pop ecx

2334 0000AB26 0F8361FFFFFF <1> jnc loc\_lcde\_cdl\_check

2335 0000AB2C 5E <1> pop esi

2336 0000AB2D C3 <1> retn

2337 <1>

2338 <1> loc\_lcde\_invalid\_format:

2339 <1> ; 15/10/2016 (0Bh -> 28)

2340 0000AB2E B81C000000 <1> mov eax, 28 ; Invalid Format !

2341 <1> loc\_lcde\_drive\_not\_ready\_read\_err:

2342 0000AB33 F9 <1> stc

2343 0000AB34 5E <1> pop esi

2344 0000AB35 C3 <1> retn

2345 <1>

2346 <1> loc\_lcde\_check\_gnc\_error:

2347 0000AB36 09C0 <1> or eax, eax

2348 0000AB38 75F9 <1> jnz short loc\_lcde\_drive\_not\_ready\_read\_err

2349 0000AB3A 66FF0D[D6630100] <1> dec word [LCDE\_ClusterSN]

2350 0000AB41 75EB <1> jnz short loc\_lcde\_invalid\_format

2351 0000AB43 A1[D8630100] <1> mov eax, [LCDE\_Cluster]

2352 <1>

2353 <1> loc\_lcde\_check\_dir\_buffer\_cluster:

2354 0000AB48 3B05[2D610100] <1> cmp eax, [DirBuff\_Cluster]

2355 0000AB4E 755C <1> jne short loc\_lcde\_load\_dir\_cluster

2356 0000AB50 803D[28610100]00 <1> cmp byte [DirBuff\_ValidData], 0

2357 0000AB57 7727 <1> ja short lcde\_check\_dir\_buffer\_cluster\_next

2358 0000AB59 803D[FC580100]00 <1> cmp byte [Current\_Dir\_Level], 0

2359 0000AB60 775F <1> ja short loc\_lcde\_load\_dir\_cluster\_0

2360 <1> ; 27/02/2016

2361 <1> ; TRDOS v1 has bug here for FAT32 fs !

2362 0000AB62 807E0303 <1> cmp byte [esi+LD\_FATType], 3 ; FAT32

2363 0000AB66 7359 <1> jnb short loc\_lcde\_load\_dir\_cluster\_0

2364 <1> ;

2365 0000AB68 0FB74E17 <1> movzx ecx, word [esi+LD\_BPB+RootDirEnts]

2366 0000AB6C 6683C10F <1> add cx, 15 ; round up (16 entries per sector)

2367 0000AB70 66C1E904 <1> shr cx, 4 ; 1 sector contains 16 dir entries

2368 <1>

2369 0000AB74 8B4664 <1> mov eax, [esi+LD\_ROOTBegin]

2370 0000AB77 EB54 <1> jmp short loc\_lcde\_load\_dir\_cluster\_1

2371 <1>

2372 <1> loc\_lcde\_validate\_dirBuff:

2373 0000AB79 C605[28610100]01 <1> mov byte [DirBuff\_ValidData], 1

2374 <1>

2375 <1> lcde\_check\_dir\_buffer\_cluster\_next:

2376 0000AB80 0FB71D[D4630100] <1> movzx ebx, word [LCDE\_EntryIndex]

2377 0000AB87 663B1D[2B610100] <1> cmp bx, [DirBuff\_LastEntry]

2378 0000AB8E 779E <1> ja short loc\_lcde\_invalid\_format

2379 0000AB90 B820000000 <1> mov eax, 32

2380 0000AB95 F7E3 <1> mul ebx

2381 <1> ;or edx, edx

2382 <1> ;jnz short loc\_lcde\_invalid\_format

2383 <1>

2384 0000AB97 BF00000800 <1> mov edi, Directory\_Buffer

2385 0000AB9C 01C7 <1> add edi, eax ; add entry offset to buffer address

2386 <1>

2387 <1> loc\_lcde\_dir\_buffer\_last\_check:

2388 0000AB9E A1[2D610100] <1> mov eax, [DirBuff\_Cluster]

2389 0000ABA3 0FB60D[28610100] <1> movzx ecx, byte [DirBuff\_ValidData]

2390 <1>

2391 <1> loc\_lcde\_retn:

2392 0000ABAA 5E <1> pop esi

2393 0000ABAB C3 <1> retn

2394 <1>

2395 <1> loc\_lcde\_load\_dir\_cluster:

2396 <1> ;cmp byte [DirBuff\_ValidData], 2

2397 <1> ;jne short loc\_lcde\_load\_dir\_cluster\_n2

2398 0000ABAC 50 <1> push eax

2399 0000ABAD E8E6FCFFFF <1> call save\_directory\_buffer

2400 0000ABB2 58 <1> pop eax

2401 0000ABB3 72F5 <1> jc short loc\_lcde\_retn

2402 <1>

2403 <1> loc\_lcde\_load\_dir\_cluster\_n2:

2404 0000ABB5 C605[28610100]00 <1> mov byte [DirBuff\_ValidData], 0

2405 0000ABBC A3[2D610100] <1> mov [DirBuff\_Cluster], eax

2406 <1>

2407 <1> loc\_lcde\_load\_dir\_cluster\_0:

2408 0000ABC1 83E802 <1> sub eax, 2

2409 0000ABC4 0FB64E13 <1> movzx ecx, byte [esi+LD\_BPB+SecPerClust]

2410 0000ABC8 F7E1 <1> mul ecx

2411 0000ABCA 034668 <1> add eax, [esi+LD\_DATABegin]

2412 <1>

2413 <1> loc\_lcde\_load\_dir\_cluster\_1:

2414 0000ABCD BB00000800 <1> mov ebx, Directory\_Buffer

2415 <1> ; ecx = sector count

2416 0000ABD2 E8014C0000 <1> call disk\_read

2417 0000ABD7 73A0 <1> jnc short loc\_lcde\_validate\_dirBuff

2418 <1>

2419 <1> ; 15/10/2016

2420 <1> ; (Disk read error instead of drv not ready err)

2421 0000ABD9 B811000000 <1> mov eax, 17 ; Drive not ready or read error !

2422 0000ABDE EBCA <1> jmp short loc\_lcde\_retn

2423 <1>

2424 <1>

2425 <1> remove\_file:

2426 <1> ; 15/10/2016

2427 <1> ; 28/02/2016 (TRDOS 386 = TRDOS v2.0)

2428 <1> ; 10/04/2011 (FILE.ASM, 'proc\_delete\_file')

2429 <1> ; 09/08/2010

2430 <1> ; INPUT ->

2431 <1> ; EDI = Directory Buffer Entry Address

2432 <1> ; CX = Directory Buffer Entry Counter/Index

2433 <1> ; BL = Longname Entry Length

2434 <1> ; BH = Logical DOS Drive Number

2435 <1>

2436 0000ABE0 29C0 <1> sub eax, eax

2437 0000ABE2 88FC <1> mov ah, bh

2438 0000ABE4 BE00010900 <1> mov esi, Logical\_DOSDisks

2439 0000ABE9 01C6 <1> add esi, eax

2440 <1>

2441 0000ABEB 807E0301 <1> cmp byte [esi+LD\_FATType], 1

2442 0000ABEF 7312 <1> jnb short loc\_del\_fat\_file

2443 <1>

2444 0000ABF1 807E04A1 <1> cmp byte [esi+LD\_FSType], 0A1h

2445 0000ABF5 7406 <1> je short loc\_del\_fs\_file

2446 <1>

2447 <1> loc\_del\_file\_invalid\_format:

2448 0000ABF7 30E4 <1> xor ah, ah

2449 <1> ; 15/10/2016 (0Bh -> 28)

2450 0000ABF9 B01C <1> mov al, 28 ; Invalid Format

2451 0000ABFB F9 <1> stc

2452 0000ABFC C3 <1> retn

2453 <1>

2454 <1> loc\_del\_fs\_file:

2455 0000ABFD E83F0F0000 <1> call delete\_fs\_file

2456 0000AC02 C3 <1> retn

2457 <1>

2458 <1> loc\_del\_fat\_file:

2459 0000AC03 E808000000 <1> call delete\_directory\_entry

2460 0000AC08 7205 <1> jc short loc\_del\_file\_err\_retn

2461 <1>

2462 <1> loc\_delfile\_unlink\_cluster\_chain:

2463 0000AC0A E863170000 <1> call truncate\_cluster\_chain

2464 <1> ;jc short loc\_del\_file\_err\_retn

2465 <1>

2466 <1> loc\_delfile\_return:

2467 <1> loc\_del\_file\_err\_retn:

2468 0000AC0F C3 <1> retn

2469 <1>

2470 <1> delete\_directory\_entry:

2471 <1> ; 15/10/2016

2472 <1> ; 28/02/2016 (TRDOS 386 = TRDOS v2.0)

2473 <1> ; 01/08/2011 (DIR.ASM, 'proc\_delete\_directory\_entry')

2474 <1> ; 10/04/2011

2475 <1> ; INPUT ->

2476 <1> ; ESI = Logical Dos Drive Descripton Table Address

2477 <1> ; EDI = Directory Buffer Entry Address

2478 <1> ; CX = Directory Buffer Entry Counter/Index

2479 <1> ; BL = Longname Entry Length

2480 <1> ; OUTPUT ->

2481 <1> ; ESI = Logical dos drive descripton table address

2482 <1> ; EAX = First cluster to be truncated/unlinked

2483 <1> ; CF = 1 -> Error code in EAX (AL)

2484 <1> ; CF = 0 & BH <> 0 -> LMDT write error (BH = 1)

2485 <1> ; CF = 0 & BL <> 0 -> Long name delete error (BL = FFh)

2486 <1> ;

2487 <1> ; (EDI, EBX, ECX register contents will be changed)

2488 <1>

2489 0000AC10 881D[6A630100] <1> mov [DelFile\_LNEL], bl

2490 0000AC16 66890D[68630100] <1> mov [DelFile\_EntryCounter], cx

2491 <1>

2492 0000AC1D 668B4714 <1> mov ax, [edi+20] ; First Cluster High Word

2493 0000AC21 C1E010 <1> shl eax, 16

2494 0000AC24 668B471A <1> mov ax, [edi+26] ; First Cluster Low Word

2495 <1>

2496 0000AC28 A3[64630100] <1> mov [DelFile\_FCluster], eax

2497 <1>

2498 <1> loc\_del\_short\_name:

2499 0000AC2D C607E5 <1> mov byte [edi], 0E5h ; Deleted sign

2500 <1>

2501 0000AC30 C605[28610100]02 <1> mov byte [DirBuff\_ValidData], 2

2502 0000AC37 E85CFCFFFF <1> call save\_directory\_buffer

2503 0000AC3C 723D <1> jc short loc\_delete\_direntry\_err\_return

2504 <1>

2505 <1> loc\_del\_long\_name:

2506 0000AC3E 0FB615[6A630100] <1> movzx edx, byte [DelFile\_LNEL]

2507 0000AC45 08D2 <1> or dl, dl

2508 0000AC47 7416 <1> jz short loc\_del\_dir\_entry\_update\_parent\_dir\_lm\_date

2509 <1>

2510 0000AC49 8835[6A630100] <1> mov byte [DelFile\_LNEL], dh ; 0

2511 <1>

2512 0000AC4F 0FB705[68630100] <1> movzx eax, word [DelFile\_EntryCounter]

2513 0000AC56 29D0 <1> sub eax, edx

2514 <1> ;jnc short loc\_del\_long\_name\_continue

2515 0000AC58 7205 <1> jc short loc\_del\_dir\_entry\_update\_parent\_dir\_lm\_date

2516 <1>

2517 <1> ;loc\_del\_direntry\_inv\_data\_return: ; 15/10/2016 (0Dh -> 29)

2518 <1> ; mov eax, 29 ; 0Dh (TRDOS 8086) ; Invalid data

2519 <1> ; retn

2520 <1>

2521 <1> loc\_del\_long\_name\_continue:

2522 <1> ; AX = Directory Entry Number of the long name last entry

2523 0000AC5A E897FDFFFF <1> call delete\_longname

2524 <1> ;jc short loc\_delete\_direntry\_err\_return

2525 <1>

2526 <1> loc\_del\_dir\_entry\_update\_parent\_dir\_lm\_date:

2527 0000AC5F 801D[6A630100]00 <1> sbb byte [DelFile\_LNEL], 0 ; 0FFh if cf = 1

2528 <1>

2529 0000AC66 E8C8FCFFFF <1> call update\_parent\_dir\_lmdt

2530 0000AC6B B700 <1> mov bh, 0

2531 0000AC6D 80D700 <1> adc bh, 0

2532 <1>

2533 0000AC70 8A1D[6A630100] <1> mov bl, byte [DelFile\_LNEL]

2534 <1>

2535 <1> loc\_delete\_direntry\_return:

2536 0000AC76 A1[64630100] <1> mov eax, [DelFile\_FCluster]

2537 <1> loc\_delete\_direntry\_err\_return:

2538 0000AC7B C3 <1> retn

2539 <1>

2540 <1> rename\_directory\_entry:

2541 <1> ; 13/11/2017

2542 <1> ; 15/10/2016

2543 <1> ; 06/03/2016 (TRDOS 386 = TRDOS v2.0)

2544 <1> ; 01/08/2011 (DIR.ASM, 'proc\_rename\_directory\_entry')

2545 <1> ; 19/11/2010

2546 <1> ; INPUT -> (Current Directory)

2547 <1> ; CX = Directory Entry Number

2548 <1> ; EAX = First Cluster number of file or directory

2549 <1> ; EBX = Longname Length (dir entry count) (< 256)

2550 <1> ; ESI = New file (or directory) name (no path).

2551 <1> ; (ASCIIZ string)

2552 <1> ; OUTPUT ->

2553 <1> ; CF = 0 -> successfull

2554 <1> ; CF = 1 -> error code in EAX (AL)

2555 <1> ;

2556 <1> ; (EAX, EBX, ECX, EDX, ESI, EDI will be changed)

2557 <1>

2558 0000AC7C 803D[FD580100]00 <1> cmp byte [Current\_FATType], 0

2559 0000AC83 7706 <1> ja short loc\_rename\_directory\_entry

2560 <1>

2561 0000AC85 E8B80E0000 <1> call rename\_fs\_file\_or\_directory

2562 0000AC8A C3 <1> retn

2563 <1>

2564 <1> loc\_rename\_directory\_entry:

2565 0000AC8B 881D[6A630100] <1> mov [DelFile\_LNEL], bl

2566 0000AC91 66890D[68630100] <1> mov [DelFile\_EntryCounter], cx

2567 0000AC98 A3[64630100] <1> mov [DelFile\_FCluster], eax

2568 <1>

2569 0000AC9D 0FB7C1 <1> movzx eax, cx

2570 0000ACA0 E8BBFDFFFF <1> call locate\_current\_dir\_entry

2571 0000ACA5 7308 <1> jnc short loc\_rename\_direntry\_check\_fcluster

2572 <1>

2573 <1> loc\_rename\_direntry\_pop\_retn:

2574 0000ACA7 C3 <1> retn

2575 <1>

2576 <1> loc\_rename\_direntry\_pop\_invd\_retn:

2577 0000ACA8 F9 <1> stc

2578 <1> loc\_rename\_direntry\_invd\_retn:

2579 <1> ; 15/10/2016 (0Dh -> 29)

2580 0000ACA9 B81D000000 <1> mov eax, 29 ; Invalid data

2581 <1> loc\_rename\_retn:

2582 0000ACAE C3 <1> retn

2583 <1>

2584 <1> loc\_rename\_direntry\_check\_fcluster:

2585 0000ACAF 668B5714 <1> mov dx, [edi+20] ; First Cluster HW

2586 0000ACB3 C1E210 <1> shl edx, 16 ; 13/11/2017

2587 0000ACB6 668B571A <1> mov dx, [edi+26] ; First Cluster LW

2588 0000ACBA 3B15[64630100] <1> cmp edx, [DelFile\_FCluster]

2589 0000ACC0 75E6 <1> jne short loc\_rename\_direntry\_pop\_invd\_retn

2590 <1> ; ESI = New file (or directory) name. (ASCIIZ string)

2591 <1> ; 06/03/2016

2592 <1> ; TRDOS v2 - NOTE: 'convert\_file\_name' procedure

2593 <1> ; has been modified for eliminating following situation.

2594 <1> ;

2595 <1> ; TRDOS v1 - NOTE: If file/dir name is more than 11 bytes

2596 <1> ; without a dot, attributes (edi+11) byte will be overwritten !

2597 <1> ; (Dot file name input must be proper for 11 byte dir entry

2598 <1> ; type file name output.)

2599 0000ACC2 E8A2F6FFFF <1> call convert\_file\_name

2600 <1>

2601 0000ACC7 C605[28610100]02 <1> mov byte [DirBuff\_ValidData], 2

2602 0000ACCE E8C5FBFFFF <1> call save\_directory\_buffer

2603 0000ACD3 72D9 <1> jc short loc\_rename\_retn

2604 <1>

2605 <1> loc\_rename\_direntry\_del\_ln:

2606 0000ACD5 0FB615[6A630100] <1> movzx edx, byte [DelFile\_LNEL]

2607 0000ACDC 08D2 <1> or dl, dl

2608 0000ACDE 7410 <1> jz short loc\_rename\_direntry\_update\_parent\_dir\_lm\_date

2609 <1>

2610 0000ACE0 0FB705[68630100] <1> movzx eax, word [DelFile\_EntryCounter]

2611 0000ACE7 29D0 <1> sub eax, edx

2612 0000ACE9 72BE <1> jc short loc\_rename\_direntry\_invd\_retn

2613 <1>

2614 <1> loc\_rename\_direntry\_del\_ln\_continue:

2615 <1> ; EAX = Directory Entry Number of the long name last entry

2616 0000ACEB E806FDFFFF <1> call delete\_longname

2617 <1>

2618 <1> loc\_rename\_direntry\_update\_parent\_dir\_lm\_date:

2619 0000ACF0 E83EFCFFFF <1> call update\_parent\_dir\_lmdt

2620 0000ACF5 31C0 <1> xor eax, eax

2621 0000ACF7 C3 <1> retn

2622 <1>

2623 <1> move\_source\_file\_to\_destination\_file:

2624 <1> ; 15/10/2016

2625 <1> ; 11/03/2016

2626 <1> ; 10/03/2016 (TRDOS 386 = TRDOS v2.0)

2627 <1> ; 01/08/2011 (FILE.ASM)

2628 <1> ; 04/08/2010

2629 <1> ;

2630 <1> ; Phase 1 -> Check destination file,

2631 <1> ; 'not found' is required

2632 <1> ; Phase 2 -> Check source file

2633 <1> ; 'found' and proper attributes is required

2634 <1> ; Phase 3 -> Make destination directory entry,

2635 <1> ; add new dir cluster or section if it is required

2636 <1> ; Phase 4 -> Delete source directory entry.

2637 <1> ; cf = 1 causes to return before the phase 4.

2638 <1> ; (source file protection against any possible errors)

2639 <1> ;

2640 <1> ; 08/05/2011 major modification

2641 <1> ; -> destination file deleting is removed

2642 <1> ; for msdos move/rename compatibility.

2643 <1> ; (Access denied error will return if

2644 <1> ; the destination file is found...)

2645 <1> ; INPUT ->

2646 <1> ; ESI = Source File Pathname (Asciiz)

2647 <1> ; EDI = Destination File Pathname (Asciiz)

2648 <1> ; AL = 0 --> Interrupt (System call)

2649 <1> ; AL > 0 --> Command Interpreter (Question)

2650 <1> ; AL = 1 --> Question Phase

2651 <1> ; AL = 2 --> Progress Phase

2652 <1> ; OUTPUT ->

2653 <1> ; cf = 0 -> OK

2654 <1> ; EAX = Destination directory first cluster

2655 <1> ; ESI = Logical DOS drive description table

2656 <1> ; EBX = Destination file structure offset

2657 <1> ; CX = 0 (CX > 0 --> calculate free space error)

2658 <1> ; cf = 1 -> Error code in EAX (AL)

2659 <1> ;

2660 <1> ; (EDX, ECX, EBX, ESI, EDI will be changed)

2661 <1>

2662 0000ACF8 3C02 <1> cmp al, 2

2663 0000ACFA 0F847F010000 <1> je msftdf\_df2\_check\_directory

2664 0000AD00 A2[EA640100] <1> mov [move\_cmd\_phase], al

2665 <1>

2666 <1> msftdf\_parse\_sf\_path:

2667 <1> ; ESI = ASCIIZ pathname (Source)

2668 0000AD05 57 <1> push edi

2669 0000AD06 BF[E8630100] <1> mov edi, SourceFile\_Drv

2670 0000AD0B E824F7FFFF <1> call parse\_path\_name

2671 0000AD10 5E <1> pop esi

2672 0000AD11 7211 <1> jc short msftdf\_psf\_retn

2673 <1>

2674 <1> msftdf\_parse\_df\_path:

2675 <1> ; ESI = ASCIIZ pathname (Destination)

2676 0000AD13 BF[68640100] <1> mov edi, DestinationFile\_Drv

2677 0000AD18 E817F7FFFF <1> call parse\_path\_name

2678 0000AD1D 7306 <1> jnc short msftdf\_check\_sf\_drv

2679 <1>

2680 0000AD1F 3C01 <1> cmp al, 1 ; File or directory name is not existing

2681 0000AD21 7602 <1> jna short msftdf\_check\_sf\_drv

2682 <1>

2683 <1> msftdf\_stc\_retn:

2684 0000AD23 F9 <1> stc

2685 <1> msftdf\_psf\_retn:

2686 0000AD24 C3 <1> retn

2687 <1>

2688 <1> msftdf\_check\_sf\_drv:

2689 0000AD25 A0[E8630100] <1> mov al, [SourceFile\_Drv]

2690 <1>

2691 <1> msftdf\_check\_df\_drv:

2692 0000AD2A 8A15[68640100] <1> mov dl, [DestinationFile\_Drv]

2693 <1>

2694 <1> msftdf\_compare\_sf\_df\_drv:

2695 0000AD30 29DB <1> sub ebx, ebx

2696 0000AD32 8A3D[FE580100] <1> mov bh, [Current\_Drv]

2697 0000AD38 38C2 <1> cmp dl, al

2698 0000AD3A 7409 <1> je short msftdf\_check\_sf\_df\_drv\_ok

2699 <1>

2700 <1> msftdf\_not\_same\_drv:

2701 <1> ; DL = source file's drive number

2702 0000AD3C 88C6 <1> mov dh, al ; destination file's drive number

2703 <1> ; 15/10/2016 (11h -> 21)

2704 0000AD3E B815000000 <1> mov eax, 21 ; Not the same drive

2705 0000AD43 F9 <1> stc

2706 0000AD44 C3 <1> retn

2707 <1>

2708 <1> msftdf\_check\_sf\_df\_drv\_ok:

2709 0000AD45 8815[EB640100] <1> mov [msftdf\_sf\_df\_drv], dl

2710 <1>

2711 0000AD4B 29C0 <1> sub eax, eax

2712 0000AD4D 88D4 <1> mov ah, dl

2713 0000AD4F 0500010900 <1> add eax, Logical\_DOSDisks

2714 0000AD54 A3[EC640100] <1> mov [msftdf\_drv\_offset], eax

2715 <1>

2716 0000AD59 38FA <1> cmp dl, bh ; byte [Current\_Drv]

2717 0000AD5B 7407 <1> je short msftdf\_df\_check\_directory

2718 <1>

2719 <1> msftdf\_change\_drv:

2720 0000AD5D E85EC1FFFF <1> call change\_current\_drive

2721 0000AD62 726D <1> jc short msftdf\_df\_error\_retn

2722 <1>

2723 <1> msftdf\_check\_destination\_file:

2724 <1> msftdf\_df\_check\_directory:

2725 0000AD64 BE[69640100] <1> mov esi, DestinationFile\_Directory

2726 0000AD69 803E20 <1> cmp byte [esi], 20h

2727 0000AD6C 760F <1> jna short msftdf\_df\_find\_1

2728 <1>

2729 <1> msftdf\_df\_change\_directory:

2730 0000AD6E FE05[D30C0100] <1> inc byte [Restore\_CDIR]

2731 0000AD74 30E4 <1> xor ah, ah ; CD\_COMMAND sign -> 0

2732 0000AD76 E8A3F0FFFF <1> call change\_current\_directory

2733 0000AD7B 7254 <1> jc short msftdf\_df\_error\_retn

2734 <1>

2735 <1> ;msftdf\_df\_change\_prompt\_dir\_string:

2736 <1> ; call change\_prompt\_dir\_string

2737 <1>

2738 <1> msftdf\_df\_find\_1:

2739 0000AD7D BE[AA640100] <1> mov esi, DestinationFile\_Name

2740 0000AD82 803E20 <1> cmp byte [esi], 20h

2741 0000AD85 7631 <1> jna short msftdf\_df\_copy\_sf\_name

2742 <1>

2743 <1> msftdf\_df\_find\_2:

2744 0000AD87 6631C0 <1> xor ax, ax ; DestinationFile\_AttributesMask -> any/zero

2745 0000AD8A E8D4D4FFFF <1> call find\_first\_file

2746 0000AD8F 0F838D000000 <1> jnc msftdf\_permission\_denied\_retn

2747 <1>

2748 <1> msftdf\_df\_check\_error\_code:

2749 <1> ;cmp eax, 2 ; File not found error

2750 0000AD95 3C02 <1> cmp al, 2

2751 0000AD97 7537 <1> jne short msftdf\_df\_stc\_retn

2752 <1>

2753 <1> msftdf\_df\_check\_fname:

2754 <1> ; 15/10/2016

2755 0000AD99 BE[AA640100] <1> mov esi, DestinationFile\_Name ; \*

2756 0000AD9E E87ED8FFFF <1> call check\_filename

2757 0000ADA3 7307 <1> jnc short msftdf\_convert\_df\_direntry\_name

2758 <1> ; invalid file name chars !

2759 0000ADA5 B81A000000 <1> mov eax, ERR\_INV\_FILE\_NAME ; 26

2760 0000ADAA EB24 <1> jmp short msftdf\_df\_stc\_retn

2761 <1>

2762 <1> msftdf\_convert\_df\_direntry\_name:

2763 <1> ; mov esi, DestinationFile\_Name ; \*

2764 0000ADAC BF[BA640100] <1> mov edi, DestinationFile\_DirEntry

2765 0000ADB1 E8B3F5FFFF <1> call convert\_file\_name

2766 0000ADB6 EB1A <1> jmp short msftdf\_restore\_current\_dir\_1

2767 <1>

2768 <1> msftdf\_df\_copy\_sf\_name:

2769 0000ADB8 89F7 <1> mov edi, esi

2770 0000ADBA 57 <1> push edi

2771 0000ADBB BE[2A640100] <1> mov esi, SourceFile\_Name

2772 0000ADC0 B90C000000 <1> mov ecx, 12

2773 <1> msftdf\_df\_copy\_sf\_name\_loop:

2774 0000ADC5 AC <1> lodsb

2775 0000ADC6 AA <1> stosb

2776 0000ADC7 08C0 <1> or al, al

2777 0000ADC9 7402 <1> jz short msftdf\_df\_copy\_sf\_name\_ok

2778 0000ADCB E2F8 <1> loop msftdf\_df\_copy\_sf\_name\_loop

2779 <1> msftdf\_df\_copy\_sf\_name\_ok:

2780 0000ADCD 5E <1> pop esi

2781 0000ADCE EBB7 <1> jmp short msftdf\_df\_find\_2

2782 <1>

2783 <1> msftdf\_df\_stc\_retn:

2784 0000ADD0 F9 <1> stc

2785 <1> msftdf\_restore\_cdir\_failed:

2786 <1> msftdf\_df\_error\_retn:

2787 0000ADD1 C3 <1> retn

2788 <1>

2789 <1> msftdf\_restore\_current\_dir\_1:

2790 0000ADD2 803D[D30C0100]00 <1> cmp byte [Restore\_CDIR], 0

2791 0000ADD9 760D <1> jna short msftdf\_sf\_check\_directory

2792 0000ADDB 8B35[EC640100] <1> mov esi, [msftdf\_drv\_offset]

2793 0000ADE1 E891C1FFFF <1> call restore\_current\_directory

2794 0000ADE6 72E9 <1> jc short msftdf\_restore\_cdir\_failed

2795 <1>

2796 <1> msftdf\_sf\_check\_directory:

2797 0000ADE8 BE[E9630100] <1> mov esi, SourceFile\_Directory

2798 0000ADED 803E20 <1> cmp byte [esi], 20h

2799 0000ADF0 760F <1> jna short msftdf\_sf\_find

2800 <1> msftdf\_sf\_change\_directory:

2801 0000ADF2 FE05[D30C0100] <1> inc byte [Restore\_CDIR]

2802 0000ADF8 30E4 <1> xor ah, ah ; CD\_COMMAND sign -> 0

2803 0000ADFA E81FF0FFFF <1> call change\_current\_directory

2804 0000ADFF 7227 <1> jc short msftdf\_return

2805 <1>

2806 <1> ;msftdf\_sf\_change\_prompt\_dir\_string:

2807 <1> ; call change\_prompt\_dir\_string

2808 <1>

2809 <1> msftdf\_sf\_find:

2810 0000AE01 BE[2A640100] <1> mov esi, SourceFile\_Name ; Offset 66

2811 0000AE06 66B80018 <1> mov ax, 1800h ; Only files

2812 0000AE0A E854D4FFFF <1> call find\_first\_file

2813 0000AE0F 7217 <1> jc short msftdf\_return

2814 <1>

2815 <1> msftdf\_sf\_ambgfn\_check:

2816 0000AE11 6609D2 <1> or dx, dx ; Ambiguous filename chars used sign (DX>0)

2817 0000AE14 7407 <1> jz short msftdf\_sf\_found

2818 <1>

2819 <1> msftdf\_ambiguous\_file\_name\_error:

2820 0000AE16 B802000000 <1> mov eax, 2 ; File not found error

2821 0000AE1B F9 <1> stc

2822 0000AE1C C3 <1> retn

2823 <1>

2824 <1> msftdf\_sf\_found:

2825 0000AE1D 80E31F <1> and bl, 1Fh ; Attributes, D-V-S-H-R

2826 0000AE20 7416 <1> jz short msftdf\_save\_sf\_structure

2827 <1>

2828 <1> msftdf\_permission\_denied\_retn:

2829 0000AE22 B805000000 <1> mov eax, 05h ; Access (Permission) denied !

2830 0000AE27 F9 <1> stc

2831 <1> msftdf\_rest\_cdir\_err\_retn:

2832 <1> msftdf\_return:

2833 0000AE28 C3 <1> retn

2834 <1>

2835 <1> msftdf\_phase\_1\_return:

2836 0000AE29 31C0 <1> xor eax, eax

2837 0000AE2B A2[EA640100] <1> mov [move\_cmd\_phase], al ; 0

2838 0000AE30 FEC0 <1> inc al ; mov al, 1

2839 0000AE32 BB[7FAE0000] <1> mov ebx, msftdf\_df2\_check\_directory

2840 <1> ;mov edx, 0FFFFFFFFh

2841 0000AE37 C3 <1> retn

2842 <1>

2843 <1> msftdf\_save\_sf\_structure:

2844 0000AE38 BE[F4620100] <1> mov esi, FindFile\_DirEntry

2845 0000AE3D BF[3A640100] <1> mov edi, SourceFile\_DirEntry

2846 0000AE42 B908000000 <1> mov ecx, 8

2847 0000AE47 F3A5 <1> rep movsd

2848 <1>

2849 <1> msftdf\_df\_copy\_sf\_parameters:

2850 0000AE49 BE0B000000 <1> mov esi, 11

2851 0000AE4E 89F7 <1> mov edi, esi

2852 0000AE50 81C6[3A640100] <1> add esi, SourceFile\_DirEntry

2853 0000AE56 81C7[BA640100] <1> add edi, DestinationFile\_DirEntry

2854 <1> ;mov ecx, 21

2855 0000AE5C B115 <1> mov cl, 21

2856 0000AE5E F3A4 <1> rep movsb

2857 <1>

2858 <1> msftdf\_restore\_current\_dir\_2:

2859 0000AE60 803D[D30C0100]00 <1> cmp byte [Restore\_CDIR], 0

2860 0000AE67 760D <1> jna short msftdf\_df2\_check\_move\_cmd\_phase

2861 0000AE69 8B35[EC640100] <1> mov esi, [msftdf\_drv\_offset]

2862 0000AE6F E803C1FFFF <1> call restore\_current\_directory

2863 0000AE74 72B2 <1> jc short msftdf\_rest\_cdir\_err\_retn

2864 <1>

2865 <1> msftdf\_df2\_check\_move\_cmd\_phase:

2866 0000AE76 803D[EA640100]01 <1> cmp byte [move\_cmd\_phase], 1

2867 0000AE7D 74AA <1> je short msftdf\_phase\_1\_return

2868 <1>

2869 <1> msftdf\_df2\_check\_directory:

2870 0000AE7F BE[69640100] <1> mov esi, DestinationFile\_Directory

2871 0000AE84 803E20 <1> cmp byte [esi], 20h

2872 0000AE87 760F <1> jna short msftdf\_make\_dfde\_locate\_ffe\_on\_directory

2873 <1> msftdf\_df2\_change\_directory:

2874 0000AE89 FE05[D30C0100] <1> inc byte [Restore\_CDIR]

2875 0000AE8F 30E4 <1> xor ah, ah ; CD\_COMMAND sign -> 0

2876 0000AE91 E888EFFFFF <1> call change\_current\_directory

2877 0000AE96 7290 <1> jc short msftdf\_return

2878 <1>

2879 <1> ;msftdf\_df2\_change\_prompt\_dir\_string:

2880 <1> ; call change\_prompt\_dir\_string

2881 <1>

2882 <1> msftdf\_make\_dfde\_locate\_ffe\_on\_directory:

2883 <1> ; Current directory fcluster <> Directory buffer cluster

2884 <1> ; Current directory will be reloaded by

2885 <1> ; 'locate\_current\_dir\_file' procedure

2886 <1> ;

2887 <1> ;xor ax, ax

2888 0000AE98 31C0 <1> xor eax, eax

2889 0000AE9A 89C1 <1> mov ecx, eax

2890 0000AE9C 6649 <1> dec cx ; FFFFh

2891 <1> ; CX = FFFFh -> find first deleted or free entry

2892 <1> ; ESI would be ASCIIZ filename address if the call

2893 <1> ; would not be for first free or deleted dir entry

2894 0000AE9E E8CFF1FFFF <1> call locate\_current\_dir\_file

2895 0000AEA3 733F <1> jnc msftdf\_make\_dfde\_set\_ff\_dir\_entry

2896 <1>

2897 <1> ;cmp eax, 2

2898 0000AEA5 3C02 <1> cmp al, 2

2899 0000AEA7 7537 <1> jne short msftdf\_error\_retn

2900 <1>

2901 <1> msftdf\_add\_new\_dir\_entry\_check\_fs:

2902 0000AEA9 8B35[EC640100] <1> mov esi, [msftdf\_drv\_offset]

2903 0000AEAF A1[2D610100] <1> mov eax, [DirBuff\_Cluster]

2904 0000AEB4 807E0300 <1> cmp byte [esi+LD\_FATType], 0

2905 0000AEB8 7711 <1> ja short msftdf\_add\_new\_subdir\_cluster

2906 <1>

2907 <1> msftdf\_add\_new\_fs\_subdir\_section:

2908 <1> ;CL=0, CH=E5h --> deleted entry, CH=0 --> free entry

2909 <1> ;xor cx, cx

2910 0000AEBA 30ED <1> xor ch, ch ; cx = 0 --> add a new subdir section

2911 0000AEBC E8830C0000 <1> call add\_new\_fs\_section

2912 0000AEC1 721E <1> jc short msftdf\_dsfde\_error\_retn

2913 <1> ;mov [createfile\_LastDirCluster], eax

2914 <1>

2915 0000AEC3 E8A30E0000 <1> call load\_FS\_sub\_directory

2916 <1> ;mov ebx, Directory\_Buffer

2917 0000AEC8 7318 <1> jnc short msftdf\_add\_new\_fs\_subdir\_section\_ok

2918 0000AECA C3 <1> retn

2919 <1>

2920 <1> msftdf\_add\_new\_subdir\_cluster:

2921 0000AECB E881150000 <1> call add\_new\_cluster

2922 0000AED0 720F <1> jc short msftdf\_dsfde\_error\_retn

2923 <1>

2924 <1> ;mov [createfile\_LastDirCluster], eax

2925 <1>

2926 0000AED2 E8570E0000 <1> call load\_FAT\_sub\_directory

2927 0000AED7 7309 <1> jnc short msftdf\_add\_new\_subdir\_cluster\_ok

2928 <1> ; EBX = Directory buffer address

2929 <1>

2930 <1> msftdf\_ansdc\_update\_parent\_dir\_lmdt:

2931 <1> msftdf\_make\_dfde\_err\_upd\_pdir\_lmdt:

2932 0000AED9 50 <1> push eax

2933 0000AEDA E854FAFFFF <1> call update\_parent\_dir\_lmdt

2934 0000AEDF 58 <1> pop eax

2935 <1>

2936 <1> msftdf\_error\_retn:

2937 0000AEE0 F9 <1> stc

2938 <1> msftdf\_dsfde\_restore\_cdir\_failed:

2939 <1> msftdf\_dsfde\_error\_retn:

2940 0000AEE1 C3 <1> retn

2941 <1>

2942 <1> msftdf\_add\_new\_fs\_subdir\_section\_ok:

2943 <1> msftdf\_add\_new\_subdir\_cluster\_ok:

2944 0000AEE2 89DF <1> mov edi, ebx ; Directory buffer address

2945 <1>

2946 <1> msftdf\_make\_dfde\_set\_ff\_dir\_entry:

2947 0000AEE4 8B15[F8580100] <1> mov edx, [Current\_Dir\_FCluster]

2948 0000AEEA 8915[50650100] <1> mov [createfile\_FFCluster], edx

2949 <1> ; EDI = Directory entry offset

2950 0000AEF0 BE[BA640100] <1> mov esi, DestinationFile\_DirEntry

2951 0000AEF5 B908000000 <1> mov ecx, 8

2952 0000AEFA F3A5 <1> rep movsd

2953 <1>

2954 0000AEFC C605[28610100]02 <1> mov byte [DirBuff\_ValidData], 2

2955 0000AF03 E890F9FFFF <1> call save\_directory\_buffer

2956 0000AF08 72CF <1> jc short msftdf\_make\_dfde\_err\_upd\_pdir\_lmdt

2957 <1>

2958 <1> msftdf\_make\_dfde\_update\_pdir\_lmdt:

2959 0000AF0A E824FAFFFF <1> call update\_parent\_dir\_lmdt

2960 <1>

2961 <1> msftdf\_dsfde\_restore\_current\_dir\_1:

2962 0000AF0F 803D[D30C0100]00 <1> cmp byte [Restore\_CDIR], 0

2963 0000AF16 760D <1> jna short msftdf\_dsfde\_check\_directory

2964 0000AF18 8B35[EC640100] <1> mov esi, [msftdf\_drv\_offset]

2965 0000AF1E E854C0FFFF <1> call restore\_current\_directory

2966 0000AF23 72BC <1> jc short msftdf\_dsfde\_restore\_cdir\_failed

2967 <1>

2968 <1> msftdf\_dsfde\_check\_directory:

2969 0000AF25 BE[E9630100] <1> mov esi, SourceFile\_Directory

2970 0000AF2A 803E20 <1> cmp byte [esi], 20h

2971 0000AF2D 760F <1> jna short msftdf\_dsfde\_find\_file

2972 <1>

2973 <1> msftdf\_dsfde\_change\_directory:

2974 0000AF2F FE05[D30C0100] <1> inc byte [Restore\_CDIR]

2975 0000AF35 28E4 <1> sub ah, ah ; CD\_COMMAND sign -> 0

2976 0000AF37 E8E2EEFFFF <1> call change\_current\_directory

2977 0000AF3C 72A3 <1> jc short msftdf\_dsfde\_error\_retn

2978 <1>

2979 <1> ;msftdf\_dsfde\_sf\_change\_prompt\_dir\_string:

2980 <1> ; call change\_prompt\_dir\_string

2981 <1>

2982 <1> msftdf\_dsfde\_find\_file:

2983 0000AF3E BE[2A640100] <1> mov esi, SourceFile\_Name ; Offset 66

2984 0000AF43 668B460E <1> mov ax, [esi+14] ; 80 -> SourceFile\_AttributesMask

2985 0000AF47 E817D3FFFF <1> call find\_first\_file

2986 0000AF4C 7293 <1> jc short msftdf\_dsfde\_error\_retn

2987 <1>

2988 <1> msftdf\_dsfde\_delete\_direntry:

2989 0000AF4E 8B35[EC640100] <1> mov esi, [msftdf\_drv\_offset]

2990 <1>

2991 0000AF54 807E0300 <1> cmp byte [esi+LD\_FATType], 0

2992 0000AF58 770A <1> ja short msftdf\_delete\_FAT\_direntry

2993 <1>

2994 0000AF5A 30DB <1> xor bl, bl

2995 <1> ; BL = 0 -> File

2996 <1> ; EDI -> Directory buffer entry offset/address

2997 0000AF5C E8E40B0000 <1> call delete\_fs\_directory\_entry

2998 0000AF61 7315 <1> jnc short msftdf\_dsfde\_restore\_current\_dir\_2

2999 0000AF63 C3 <1> retn

3000 <1>

3001 <1> msftdf\_delete\_FAT\_direntry:

3002 0000AF64 8A1D[F1620100] <1> mov bl, [FindFile\_LongNameEntryLength]

3003 0000AF6A 668B0D[1C630100] <1> mov cx, [FindFile\_DirEntryNumber]

3004 <1> ; ESI = Logical DOS drive description table address

3005 <1> ; EDI = Directory buffer entry offset/address

3006 0000AF71 E89AFCFFFF <1> call delete\_directory\_entry

3007 0000AF76 721C <1> jc short msftdf\_retn

3008 <1>

3009 <1> msftdf\_dsfde\_restore\_current\_dir\_2:

3010 0000AF78 803D[D30C0100]00 <1> cmp byte [Restore\_CDIR], 0

3011 0000AF7F 7607 <1> jna short msftdf\_new\_dir\_fcluster\_retn

3012 <1> ;mov esi, [msftdf\_drv\_offset]

3013 0000AF81 E8F1BFFFFF <1> call restore\_current\_directory

3014 0000AF86 720C <1> jc short msftdf\_retn

3015 <1>

3016 <1> msftdf\_new\_dir\_fcluster\_retn:

3017 0000AF88 31C9 <1> xor ecx, ecx

3018 0000AF8A A1[50650100] <1> mov eax, [createfile\_FFCluster]

3019 0000AF8F BB[68640100] <1> mov ebx, DestinationFile\_Drv

3020 <1>

3021 <1> msftdf\_retn:

3022 0000AF94 C3 <1> retn

3023 <1>

3024 <1>

3025 <1> copy\_source\_file\_to\_destination\_file:

3026 <1> ; 17/10/2016

3027 <1> ; 16/10/2016

3028 <1> ; 15/10/2016

3029 <1> ; 30/03/2016, 31/03/2016

3030 <1> ; 24/03/2016, 25/03/2016, 28/03/2016

3031 <1> ; 21/03/2016, 22/03/2016, 23/03/2016

3032 <1> ; 16/03/2016, 17/03/2016, 18/03/2016

3033 <1> ; 15/03/2016 (TRDOS 386 = TRDOS v2.0)

3034 <1> ; 02/09/2011 (FILE.ASM 'copy\_source\_file\_to\_destination\_file')

3035 <1> ; 01/08/2010 - 18/05/2011

3036 <1> ;

3037 <1> ; Command Interpreter phase 1 enter ->

3038 <1> ; AL = 1 -> Caller is command interpreter

3039 <1> ; AL = 2 -> The second call, re-enter/continue

3040 <1> ; Phase 1 -> Check source file

3041 <1> ; 'found' is required

3042 <1> ; Phase 2 -> Check destination file,

3043 <1> ; save 'found' or 'not found' status

3044 <1> ; 'permission denied' error will be return

3045 <1> ; if attributes have not for ordinary file

3046 <1> ; without readonly attribute

3047 <1> ; Command Interpreter phase 1 return ->

3048 <1> ; DH = Source file attributes

3049 <1> ; DL = Destination file found status

3050 <1> ; EAX = 0

3051 <1> ; Command Interpreter phase 2 enter ->

3052 <1> ; AL = 2 -> Continue from the last position

3053 <1> ; AH =

3054 <1> ; Phase 3 -> Load source file or use read/write cluster method

3055 <1> ; Phase 4 -> Create destination file if it is not found

3056 <1> ; Phase 5 -> Open destination file

3057 <1> ; Phase 6 -> Read from source and write to destination

3058 <1> ; Phase 7 -> Unload source file, if it is loaded at memory

3059 <1> ; cf = 1 causes to return before the phase 7

3060 <1> ; but loaded file will be unloaded

3061 <1> ; (allocated memory block will be deallocated)

3062 <1> ;

3063 <1> ; INPUT ->

3064 <1> ; ESI = Source File Pathname (Asciiz)

3065 <1> ; EDI = Destination File Pathname (Asciiz)

3066 <1> ; AL = 0 --> Interrupt (System call)

3067 <1> ; AL > 0 --> Command Interpreter (Question)

3068 <1> ; AL = 1 --> Question Phase

3069 <1> ; AL = 2 --> Progress Phase

3070 <1> ;

3071 <1> ; OUTPUT ->

3072 <1> ; cf = 0 -> OK

3073 <1> ; EAX = Destination file first cluster

3074 <1> ;

3075 <1> ; CL > 0 if there is file reading error before EOF

3076 <1> ; (incomplete copy)

3077 <1> ; CH > 0 if file is (full) loaded at memory

3078 <1> ;

3079 <1> ; cf = 1 -> Error code in AL (EAX)

3080 <1> ;

3081 <1> ; (EBX, ECX, ESI, EDI register contents will be changed)

3082 <1>

3083 <1>

3084 0000AF95 3C02 <1> cmp al, 2

3085 0000AF97 0F845A020000 <1> je csftdf2\_check\_cdrv

3086 <1>

3087 <1> ; Phase 1

3088 <1>

3089 0000AF9D A2[10650100] <1> mov byte [copy\_cmd\_phase], al

3090 <1>

3091 0000AFA2 57 <1> push edi ; \*

3092 <1>

3093 <1> csftdf\_parse\_sf\_path:

3094 0000AFA3 BF[E8630100] <1> mov edi, SourceFile\_Drv

3095 0000AFA8 E887F4FFFF <1> call parse\_path\_name

3096 0000AFAD 721C <1> jc short csftdf\_parse\_sf\_path\_failed

3097 <1>

3098 <1> csftdf\_parse\_df\_path:

3099 0000AFAF 5E <1> pop esi ; \* (pushed edi)

3100 <1>

3101 <1> csftdf\_sf\_check\_filename\_exists:

3102 0000AFB0 803D[2A640100]21 <1> cmp byte [SourceFile\_Name], 21h

3103 0000AFB7 7215 <1> jb short csftdf\_sf\_file\_not\_found\_error

3104 <1>

3105 0000AFB9 BF[68640100] <1> mov edi, DestinationFile\_Drv

3106 0000AFBE E871F4FFFF <1> call parse\_path\_name

3107 0000AFC3 7310 <1> jnc short csftdf\_check\_sf\_cdrv

3108 <1>

3109 0000AFC5 3C01 <1> cmp al, 1 ; File or directory name is not existing

3110 0000AFC7 760C <1> jna short csftdf\_check\_sf\_cdrv

3111 <1>

3112 <1> csftdf\_parse\_df\_path\_failed:

3113 0000AFC9 F9 <1> stc

3114 <1> csftdf\_sf\_error\_retn:

3115 0000AFCA C3 <1> retn

3116 <1>

3117 <1> csftdf\_parse\_sf\_path\_failed:

3118 0000AFCB 5F <1> pop edi ; \*

3119 0000AFCC EBFC <1> jmp short csftdf\_sf\_error\_retn

3120 <1>

3121 <1> csftdf\_sf\_file\_not\_found\_error:

3122 0000AFCE B802000000 <1> mov eax, 2 ; File not found

3123 0000AFD3 EBF5 <1> jmp short csftdf\_sf\_error\_retn

3124 <1>

3125 <1> csftdf\_check\_sf\_cdrv:

3126 0000AFD5 8A3D[FE580100] <1> mov bh, [Current\_Drv]

3127 <1>

3128 0000AFDB 883D[13650100] <1> mov [csftdf\_cdrv], bh ; 23/03/2016

3129 <1>

3130 0000AFE1 8A15[E8630100] <1> mov dl, [SourceFile\_Drv]

3131 0000AFE7 38FA <1> cmp dl, bh ; byte [Current\_Drv]

3132 0000AFE9 7407 <1> je short csftdf\_sf\_check\_directory

3133 <1>

3134 0000AFEB E8D0BEFFFF <1> call change\_current\_drive

3135 0000AFF0 72D8 <1> jc short csftdf\_sf\_error\_retn

3136 <1>

3137 <1> csftdf\_sf\_check\_directory:

3138 0000AFF2 BE[E9630100] <1> mov esi, SourceFile\_Directory

3139 0000AFF7 803E20 <1> cmp byte [esi], 20h

3140 0000AFFA 760F <1> jna short csftdf\_find\_sf

3141 <1>

3142 <1> csftdf\_sf\_change\_directory:

3143 0000AFFC FE05[D30C0100] <1> inc byte [Restore\_CDIR]

3144 0000B002 30E4 <1> xor ah, ah ; CD\_COMMAND sign -> 0

3145 0000B004 E815EEFFFF <1> call change\_current\_directory

3146 0000B009 72BF <1> jc short csftdf\_sf\_error\_retn

3147 <1>

3148 <1> ;csftdf\_sf\_change\_prompt\_dir\_string:

3149 <1> ; call change\_prompt\_dir\_string

3150 <1>

3151 <1> csftdf\_find\_sf:

3152 0000B00B BE[2A640100] <1> mov esi, SourceFile\_Name

3153 0000B010 66B80018 <1> mov ax, 1800h ; Except volume label and dirs

3154 0000B014 E84AD2FFFF <1> call find\_first\_file

3155 0000B019 72AF <1> jc short csftdf\_sf\_error\_retn

3156 <1>

3157 <1> csftdf\_sf\_ambgfn\_check:

3158 0000B01B 6621D2 <1> and dx, dx ; Ambiguous filename chars used sign (DX>0)

3159 0000B01E 7407 <1> jz short csftdf\_sf\_found

3160 <1>

3161 <1> csftdf\_ambiguous\_file\_name\_error:

3162 0000B020 B802000000 <1> mov eax, 2 ; File not found error

3163 0000B025 F9 <1> stc

3164 0000B026 C3 <1> retn

3165 <1>

3166 <1> csftdf\_sf\_found:

3167 0000B027 A3[14650100] <1> mov [csftdf\_filesize], eax

3168 <1>

3169 0000B02C 09C0 <1> or eax, eax

3170 0000B02E 7507 <1> jnz short csftdf\_set\_source\_file\_direntry

3171 <1>

3172 <1> csftdf\_sf\_file\_size\_zero:

3173 0000B030 B814000000 <1> mov eax, 20 ; TRDOS zero length (file size) error

3174 0000B035 F9 <1> stc

3175 0000B036 C3 <1> retn

3176 <1>

3177 <1> csftdf\_set\_source\_file\_direntry:

3178 0000B037 BE[F4620100] <1> mov esi, FindFile\_DirEntry

3179 0000B03C BF[3A640100] <1> mov edi, SourceFile\_DirEntry

3180 0000B041 B908000000 <1> mov ecx, 8

3181 0000B046 F3A5 <1> rep movsd

3182 <1>

3183 <1> csftdf\_sf\_restore\_cdrv:

3184 <1> ; 22/03/2016

3185 0000B048 8A15[13650100] <1> mov dl, [csftdf\_cdrv]

3186 0000B04E 3A15[FE580100] <1> cmp dl, [Current\_Drv]

3187 0000B054 7407 <1> je short csftdf\_sf\_restore\_cdir

3188 0000B056 E865BEFFFF <1> call change\_current\_drive

3189 0000B05B 724F <1> jc short csftdf\_df\_error\_retn ; 30/03/2016

3190 <1>

3191 <1> csftdf\_sf\_restore\_cdir:

3192 0000B05D 803D[D30C0100]00 <1> cmp byte [Restore\_CDIR], 0

3193 0000B064 7612 <1> jna short csftdf\_df\_check\_filename\_exists

3194 0000B066 29C0 <1> sub eax, eax

3195 0000B068 BE00010900 <1> mov esi, Logical\_DOSDisks

3196 0000B06D 88D4 <1> mov ah, dl ; byte [csftdf\_cdrv]

3197 0000B06F 01C6 <1> add esi, eax

3198 0000B071 E801BFFFFF <1> call restore\_current\_directory

3199 0000B076 7234 <1> jc short csftdf\_df\_error\_retn

3200 <1>

3201 <1> csftdf\_df\_check\_filename\_exists:

3202 0000B078 803D[AA640100]20 <1> cmp byte [DestinationFile\_Name], 20h

3203 0000B07F 7716 <1> ja short csftdf\_check\_df\_cdrv

3204 <1>

3205 <1> csftdf\_copy\_sf\_name:

3206 0000B081 BF[AA640100] <1> mov edi, DestinationFile\_Name

3207 0000B086 BE[2A640100] <1> mov esi, SourceFile\_Name

3208 0000B08B B10C <1> mov cl, 12

3209 <1>

3210 <1> csftdf\_df\_copy\_sf\_name\_loop:

3211 0000B08D AC <1> lodsb

3212 0000B08E AA <1> stosb

3213 0000B08F 08C0 <1> or al, al

3214 0000B091 7404 <1> jz short csftdf\_check\_df\_cdrv

3215 0000B093 FEC9 <1> dec cl

3216 0000B095 75F6 <1> jnz csftdf\_df\_copy\_sf\_name\_loop

3217 <1>

3218 <1> csftdf\_check\_df\_cdrv:

3219 0000B097 8A15[68640100] <1> mov dl, [DestinationFile\_Drv]

3220 0000B09D 3A15[FE580100] <1> cmp dl, [Current\_Drv]

3221 0000B0A3 7408 <1> je short csftdf\_df\_check\_directory

3222 <1>

3223 0000B0A5 E816BEFFFF <1> call change\_current\_drive

3224 0000B0AA 7301 <1> jnc short csftdf\_df\_check\_directory

3225 <1>

3226 <1> csftdf\_df\_error\_retn:

3227 0000B0AC C3 <1> retn

3228 <1>

3229 <1> csftdf\_df\_check\_directory:

3230 0000B0AD BE[69640100] <1> mov esi, DestinationFile\_Directory

3231 0000B0B2 803E20 <1> cmp byte [esi], 20h

3232 0000B0B5 760F <1> jna short csftdf\_find\_df

3233 <1>

3234 <1> csftdf\_df\_change\_directory:

3235 0000B0B7 FE05[D30C0100] <1> inc byte [Restore\_CDIR]

3236 0000B0BD 28E4 <1> sub ah, ah ; CD\_COMMAND sign -> 0

3237 0000B0BF E85AEDFFFF <1> call change\_current\_directory

3238 0000B0C4 72E6 <1> jc short csftdf\_df\_error\_retn

3239 <1>

3240 <1> ;csftdf\_df\_change\_prompt\_dir\_string:

3241 <1> ; call change\_prompt\_dir\_string

3242 <1>

3243 <1> csftdf\_find\_df:

3244 <1> ; 23/03/2016

3245 0000B0C6 29DB <1> sub ebx, ebx

3246 0000B0C8 8A3D[68640100] <1> mov bh, [DestinationFile\_Drv]

3247 0000B0CE 81C300010900 <1> add ebx, Logical\_DOSDisks

3248 0000B0D4 891D[40650100] <1> mov [csftdf\_df\_drv\_dt], ebx

3249 <1>

3250 0000B0DA BE[AA640100] <1> mov esi, DestinationFile\_Name

3251 0000B0DF 6631C0 <1> xor ax, ax

3252 <1> ; DestinationFile\_AttributesMask -> any/zero

3253 0000B0E2 E87CD1FFFF <1> call find\_first\_file

3254 0000B0E7 7218 <1> jc short csftdf\_df\_check\_error\_code

3255 <1>

3256 <1> csftdf\_df\_ambgfn\_check:

3257 0000B0E9 6609D2 <1> or dx, dx ; Ambiguous filename chars used sign (DX>0)

3258 0000B0EC 752A <1> jnz short csftdf\_df\_error\_inv\_fname

3259 <1>

3260 <1> csftdf\_df\_found:

3261 0000B0EE C605[12650100]01 <1> mov byte [DestinationFileFound], 1

3262 <1> ; 17/10/2016 (cl -> bl)

3263 0000B0F5 80E31F <1> and bl, 1Fh ; Attributes, D-V-S-H-R

3264 0000B0F8 745F <1> jz short csftdf\_df\_save\_first\_cluster

3265 <1>

3266 <1> csftdf\_df\_permission\_denied\_retn:

3267 0000B0FA B805000000 <1> mov eax, 05h ; Access/Permission denied.

3268 <1> csftdf\_df\_error\_stc\_retn:

3269 0000B0FF F9 <1> stc

3270 0000B100 C3 <1> retn

3271 <1>

3272 <1> csftdf\_df\_check\_error\_code:

3273 <1> ;cmp eax, 2

3274 0000B101 3C02 <1> cmp al, 2

3275 0000B103 75FA <1> jne short csftdf\_df\_error\_stc\_retn

3276 <1>

3277 0000B105 C605[12650100]00 <1> mov byte [DestinationFileFound], 0

3278 <1>

3279 <1> ; 15/10/2016

3280 0000B10C BE[E4620100] <1> mov esi, FindFile\_Name ; \*

3281 0000B111 E80BD5FFFF <1> call check\_filename

3282 0000B116 7307 <1> jnc short csftdf\_df\_valid\_fname

3283 <1> csftdf\_df\_error\_inv\_fname: ; 'invalid file name !'

3284 0000B118 B81A000000 <1> mov eax, ERR\_INV\_FILE\_NAME ; 26

3285 0000B11D F9 <1> stc

3286 0000B11E C3 <1> retn

3287 <1>

3288 <1> csftdf\_df\_valid\_fname:

3289 <1> ; 21/03/2016

3290 <1> ; (Capitalized file name)

3291 <1> ;mov esi, FindFile\_Name ; \* ; 15/10/2016

3292 0000B11F BF[AA640100] <1> mov edi, DestinationFile\_Name

3293 0000B124 A5 <1> movsd

3294 0000B125 A5 <1> movsd

3295 0000B126 A5 <1> movsd

3296 <1> ;movsb

3297 <1>

3298 <1> csftdf\_check\_disk\_free\_size\_0:

3299 0000B127 A1[56640100] <1> mov eax, [SourceFile\_DirEntry+DirEntry\_FileSize]

3300 <1>

3301 <1> csftdf\_check\_disk\_free\_size\_1:

3302 <1> ;sub ebx, ebx

3303 <1> ;mov esi, Logical\_DOSDisks

3304 <1> ;mov bh, [DestinationFile\_Drv]

3305 <1> ;add esi, ebx

3306 <1>

3307 0000B12C 8B35[40650100] <1> mov esi, [csftdf\_df\_drv\_dt] ; 23/03/2016

3308 <1>

3309 0000B132 0FB74E11 <1> movzx ecx, word [esi+LD\_BPB+BytesPerSec] ; 17, LD\_BPB + 0Bh

3310 0000B136 01C8 <1> add eax, ecx

3311 0000B138 48 <1> dec eax ; file size (additional bytes) + 511 (round up)

3312 <1> csftdf\_check\_disk\_free\_size\_3: ; 16/03/2016

3313 0000B139 29D2 <1> sub edx, edx

3314 0000B13B F7F1 <1> div ecx ; bytes per sector

3315 <1>

3316 <1> csftdf\_check\_disk\_free\_size:

3317 0000B13D 3B4674 <1> cmp eax, [esi+LD\_FreeSectors]

3318 0000B140 0F8294000000 <1> jb csftdf\_check\_disk\_free\_size\_ok

3319 0000B146 770A <1> ja short csftdf\_df\_insufficient\_disk\_space

3320 <1>

3321 0000B148 807E0300 <1> cmp byte [esi+LD\_FATType], 0 ; FS needs FDT sector also.

3322 0000B14C 0F8788000000 <1> ja csftdf\_check\_disk\_free\_size\_ok

3323 <1>

3324 <1> csftdf\_df\_insufficient\_disk\_space:

3325 0000B152 B827000000 <1> mov eax, 27h ; insufficient disk space

3326 0000B157 EBA6 <1> jmp short csftdf\_df\_error\_stc\_retn

3327 <1>

3328 <1> csftdf\_df\_save\_first\_cluster:

3329 <1> ; ESI = FindFile\_DirEntry (for the old destination file)

3330 <1> ; EAX = Old destination file size

3331 <1> ; 24/03/2016

3332 <1> ; EDI = Directory entry address (within Dir Buffer boundaries)

3333 0000B159 81EF00000800 <1> sub edi, Directory\_Buffer ; (<65536)

3334 0000B15F 66C1EF05 <1> shr di, 5 ; Convert entry offset to entry index/number

3335 0000B163 66893D[E2640100] <1> mov [DestinationFile\_DirEntryNumber], di ; (<2048)

3336 <1>

3337 <1> csftdf\_df\_check\_sf\_df\_fcluster:

3338 0000B16A 668B5614 <1> mov dx, [esi+DirEntry\_FstClusHI]

3339 0000B16E C1E210 <1> shl edx, 16

3340 0000B171 668B561A <1> mov dx, [esi+DirEntry\_FstClusLO]

3341 0000B175 8915[24650100] <1> mov [csftdf\_df\_cluster], edx

3342 <1> csftdf\_df\_check\_sf\_df\_fcluster\_1:

3343 0000B17B 668B15[4E640100] <1> mov dx, [SourceFile\_DirEntry+DirEntry\_FstClusHI]

3344 0000B182 C1E210 <1> shl edx, 16

3345 0000B185 668B15[54640100] <1> mov dx, [SourceFile\_DirEntry+DirEntry\_FstClusLO]

3346 0000B18C 3B15[24650100] <1> cmp edx, [csftdf\_df\_cluster]

3347 0000B192 7512 <1> jne short csftdf\_df\_check\_sf\_df\_fcluster\_ok

3348 <1> csftdf\_df\_check\_sf\_df\_drv:

3349 0000B194 8A15[E8630100] <1> mov dl, [SourceFile\_Drv]

3350 0000B19A 3A15[68640100] <1> cmp dl, [DestinationFile\_Drv]

3351 0000B1A0 7504 <1> jne short csftdf\_df\_check\_sf\_df\_fcluster\_ok

3352 <1>

3353 <1> ; source and destination files are same !

3354 <1> ; (they have same first cluster value on same logical disk)

3355 <1>

3356 0000B1A2 31C0 <1> xor eax, eax ; mov eax, 0 -> Bad command or file name !

3357 0000B1A4 F9 <1> stc

3358 0000B1A5 C3 <1> retn

3359 <1>

3360 <1> csftdf\_df\_check\_sf\_df\_fcluster\_ok:

3361 <1> csftdf\_df\_move\_findfile\_struct:

3362 <1> ; mov esi, FindFile\_DirEntry

3363 0000B1A6 BF[BA640100] <1> mov edi, DestinationFile\_DirEntry

3364 0000B1AB B908000000 <1> mov ecx, 8

3365 0000B1B0 F3A5 <1> rep movsd

3366 <1>

3367 <1> csftdf\_check\_disk\_free\_size\_2:

3368 0000B1B2 89C2 <1> mov edx, eax ; Old destination file size

3369 <1>

3370 <1> ;mov eax, [SourceFile\_DirEntry+DirEntry\_FileSize]

3371 0000B1B4 A1[14650100] <1> mov eax, [csftdf\_filesize] ; 23/03/2016

3372 <1>

3373 <1> ;;sub ecx, ecx ; 0

3374 <1> ;mov esi, Logical\_DOSDisks

3375 <1> ;mov ch, [DestinationFile\_Drv]

3376 <1> ;add esi, ecx

3377 <1> ;

3378 <1> ;mov [csftdf\_df\_drv\_dt], esi

3379 <1>

3380 0000B1B9 8B35[40650100] <1> mov esi, [csftdf\_df\_drv\_dt] ; 23/03/2016

3381 <1>

3382 0000B1BF 668B4E11 <1> mov cx, [esi+LD\_BPB+BytesPerSec] ; 17, LD\_BPB + 0Bh

3383 0000B1C3 01CA <1> add edx, ecx ; + 512

3384 0000B1C5 01C8 <1> add eax, ecx ; + 512

3385 0000B1C7 4A <1> dec edx ; old file size + 511 (round up)

3386 0000B1C8 48 <1> dec eax ; new file size + 511 (round up)

3387 0000B1C9 F7D9 <1> neg ecx ; -512 ; 0FFFFFE00h

3388 0000B1CB 21CA <1> and edx, ecx ; = old sector count \* 512

3389 0000B1CD 21C8 <1> and eax, ecx ; = new sector count \* 512

3390 <1>

3391 0000B1CF 29D0 <1> sub eax, edx ; new file size - old file size (on disk)

3392 0000B1D1 7607 <1> jna short csftdf\_check\_disk\_free\_size\_ok

3393 <1>

3394 0000B1D3 F7D9 <1> neg ecx ; 512 (bytes per sector) ; 200h

3395 <1> ; check free space for additional sectors

3396 <1> ; eax = number of additional sectors \* bytes per sector

3397 <1> ; esi = Logical DOS drive number (of destination disk)

3398 0000B1D5 E95FFFFFFF <1> jmp csftdf\_check\_disk\_free\_size\_3

3399 <1>

3400 <1> csftdf\_check\_disk\_free\_size\_ok:

3401 <1> ; 18/03/2016

3402 <1> csftdf\_df\_check\_copy\_cmd\_phase:

3403 0000B1DA A0[10650100] <1> mov al, [copy\_cmd\_phase]

3404 0000B1DF 3C01 <1> cmp al, 1

3405 0000B1E1 7514 <1> jne short csftdf2\_check\_cdrv

3406 <1>

3407 0000B1E3 31C0 <1> xor eax, eax

3408 0000B1E5 A2[10650100] <1> mov [copy\_cmd\_phase], al ; 0

3409 <1>

3410 0000B1EA 8A15[12650100] <1> mov dl, [DestinationFileFound]

3411 0000B1F0 8A35[45640100] <1> mov dh, [SourceFile\_DirEntry+11] ; Attributes

3412 <1>

3413 <1> csftdf\_return:

3414 0000B1F6 C3 <1> retn

3415 <1>

3416 <1> ; Phase 2

3417 <1>

3418 <1> csftdf2\_check\_cdrv:

3419 <1> ; 18/03/2016

3420 <1> ; Here, destination drive and directory are ready !

3421 <1> ; (checking/restoring is not needed)

3422 <1> ; (Since at the end of the phase 1)

3423 <1>

3424 <1> ; mov dl, [DestinationFile\_Drv]

3425 <1> ; cmp dl, [Current\_Drv]

3426 <1> ; je short csftdf2\_df\_check\_directory

3427 <1> ;

3428 <1> ; call change\_current\_drive

3429 <1> ; jc short csftdf2\_read\_error

3430 <1> ;

3431 <1> ;csftdf2\_df\_check\_directory:

3432 <1> ; mov esi, DestinationFile\_Directory

3433 <1> ; cmp byte [esi], 20h

3434 <1> ; jna short csftdf2\_df\_check\_found\_or\_not

3435 <1> ;

3436 <1> ;csftdf2\_df\_change\_directory:

3437 <1> ; inc byte [Restore\_CDIR]

3438 <1> ; xor ah, ah ; CD\_COMMAND sign -> 0

3439 <1> ; call change\_current\_directory

3440 <1> ; jc short csftdf2\_stc\_return

3441 <1> ;

3442 <1> ;;csftdf2\_df\_change\_prompt\_dir\_string:

3443 <1> ;; call change\_prompt\_dir\_string

3444 <1>

3445 <1> csftdf2\_df\_check\_found\_or\_not:

3446 <1> ; 21/03/2016

3447 0000B1F7 803D[12650100]00 <1> cmp byte [DestinationFileFound], 0

3448 0000B1FE 7739 <1> ja short csftdf2\_set\_sf\_percentage

3449 <1>

3450 <1> csftdf2\_create\_file:

3451 0000B200 BE[AA640100] <1> mov esi, DestinationFile\_Name

3452 0000B205 A1[14650100] <1> mov eax, [csftdf\_filesize]

3453 0000B20A 30C9 <1> xor cl, cl ; 0

3454 <1>

3455 0000B20C 31DB <1> xor ebx, ebx ; 0

3456 0000B20E 4B <1> dec ebx ; 0FFFFFFFFh

3457 <1>

3458 <1> ; INPUT ->

3459 <1> ; EAX -> File Size

3460 <1> ; ESI = ASCIIZ File name

3461 <1> ; CL = File attributes

3462 <1> ; EBX = FFFFFFFFh -> empty file sign for FAT fs

3463 <1> ; EBX <> FFFFFFFFh -> use file size for FAT fs

3464 <1> ;

3465 <1> ; OUTPUT ->

3466 <1> ; EAX = New file's first cluster

3467 <1> ; ESI = Logical Dos Drv Descr. Table Addr.

3468 <1> ; EBX = CreateFile\_Size address

3469 <1> ; ECX = Sectors per cluster (<256)

3470 <1> ; EDX = Directory Entry Index/Number (<65536)

3471 <1> ;

3472 <1> ; cf = 1 -> error code in AL (EAX)

3473 <1>

3474 0000B20F E8EC050000 <1> call create\_file

3475 <1> ;pop esi

3476 0000B214 0F82A3050000 <1> jc csftdf2\_rw\_error

3477 <1>

3478 <1> csftdf2\_create\_file\_OK:

3479 0000B21A A3[24650100] <1> mov [csftdf\_df\_cluster], eax

3480 <1>

3481 <1> ; 24/03/2016

3482 0000B21F 668915[E2640100] <1> mov [DestinationFile\_DirEntryNumber], dx

3483 <1>

3484 <1> ; 21/03/2016

3485 0000B226 BE00000800 <1> mov esi, Directory\_Buffer

3486 0000B22B C1E205 <1> shl edx, 5 ; 32 \* index number

3487 0000B22E 01D6 <1> add esi, edx

3488 0000B230 BF[BA640100] <1> mov edi, DestinationFile\_DirEntry

3489 0000B235 B108 <1> mov cl, 8 ; 32 bytes

3490 0000B237 F3A5 <1> rep movsd

3491 <1>

3492 <1> csftdf2\_set\_sf\_percentage:

3493 <1> ; 17/03/2016

3494 0000B239 31C0 <1> xor eax, eax

3495 0000B23B A2[38650100] <1> mov [csftdf\_percentage], al ; 0, reset

3496 <1>

3497 0000B240 A3[30650100] <1> mov [csftdf\_sf\_rbytes], eax ; 0, reset

3498 0000B245 A3[34650100] <1> mov [csftdf\_df\_wbytes], eax ; 0, reset

3499 <1>

3500 0000B24A 8A25[E8630100] <1> mov ah, [SourceFile\_Drv]

3501 0000B250 BE00010900 <1> mov esi, Logical\_DOSDisks

3502 0000B255 01C6 <1> add esi, eax

3503 <1>

3504 0000B257 8935[3C650100] <1> mov [csftdf\_sf\_drv\_dt], esi ; 23/03/2016

3505 <1>

3506 0000B25D 668B15[4E640100] <1> mov dx, [SourceFile\_DirEntry+DirEntry\_FstClusHI]

3507 0000B264 C1E210 <1> shl edx, 16

3508 0000B267 668B15[54640100] <1> mov dx, [SourceFile\_DirEntry+DirEntry\_FstClusLO]

3509 0000B26E 8915[20650100] <1> mov [csftdf\_sf\_cluster], edx

3510 <1>

3511 <1> ; 16/03/2016

3512 <1> ; Note: Singlix FS boot sector parameters (for cluster

3513 <1> ; related calculations) has same offset

3514 <1> ; values from LD\_BPB as in FAT file system.

3515 <1> ; [esi+LD\_BPB+SecPerClust] is 1 for Singlix FS.

3516 <1> ;

3517 0000B274 0FB64E13 <1> movzx ecx, byte [esi+LD\_BPB+SecPerClust]

3518 0000B278 880D[66640100] <1> mov [SourceFile\_SecPerClust], cl

3519 <1>

3520 <1> ; 17/03/2016

3521 0000B27E 386E03 <1> cmp [esi+LD\_FATType], ch ; 0

3522 0000B281 7707 <1> ja short csftdf2\_set\_sf\_percent\_rsize1

3523 <1>

3524 0000B283 B800000100 <1> mov eax, 65536 ; read/write buffer size for Singlix FS

3525 0000B288 EB06 <1> jmp short csftdf2\_set\_sf\_percent\_rsize2

3526 <1>

3527 <1> csftdf2\_set\_sf\_percent\_rsize1:

3528 0000B28A 668B4611 <1> mov ax, [esi+LD\_BPB+BytesPerSec]

3529 0000B28E F7E1 <1> mul ecx

3530 <1> ;sub edx, edx

3531 <1> csftdf2\_set\_sf\_percent\_rsize2:

3532 0000B290 A3[28650100] <1> mov [csftdf\_r\_size], eax

3533 <1>

3534 <1> csftdf2\_set\_df\_percentage:

3535 <1> ;sub eax, eax

3536 <1> ;mov ah, [DestinationFile\_Drv]

3537 <1> ;mov edi, Logical\_DOSDisks

3538 <1> ;add edi, eax

3539 <1> ;mov [csftdf\_df\_drv\_dt], edi ; 17/03/2016

3540 <1>

3541 0000B295 8B3D[40650100] <1> mov edi, [csftdf\_df\_drv\_dt] ; 23/03/2016

3542 <1>

3543 <1> ; 16/03/2016

3544 <1> ; Note: Singlix FS boot sector parameters (for cluster

3545 <1> ; related calculations) has same offset

3546 <1> ; values from LD\_BPB as in FAT file system.

3547 <1> ; [edi+LD\_BPB+SecPerClust] is 1 for Singlix FS.

3548 <1> ;

3549 <1> ;movzx ecx, byte [edi+LD\_BPB+SecPerClust]

3550 0000B29B 8A4F13 <1> mov cl, [edi+LD\_BPB+SecPerClust]

3551 0000B29E 880D[E6640100] <1> mov [DestinationFile\_SecPerClust], cl

3552 <1>

3553 <1> ; 17/03/2016

3554 0000B2A4 386F03 <1> cmp [edi+LD\_FATType], ch ; 0

3555 0000B2A7 7707 <1> ja short csftdf2\_set\_df\_percent\_wsize1

3556 <1>

3557 0000B2A9 B800000100 <1> mov eax, 65536 ; read/write buffer size for Singlix FS

3558 0000B2AE EB06 <1> jmp short csftdf2\_set\_df\_percent\_wsize2

3559 <1>

3560 <1> csftdf2\_set\_df\_percent\_wsize1:

3561 0000B2B0 0FB74711 <1> movzx eax, word [edi+LD\_BPB+BytesPerSec]

3562 0000B2B4 F7E1 <1> mul ecx

3563 <1> ;sub edx, edx

3564 <1> csftdf2\_set\_df\_percent\_wsize2:

3565 0000B2B6 A3[2C650100] <1> mov [csftdf\_w\_size], eax

3566 <1>

3567 0000B2BB A1[14650100] <1> mov eax, [csftdf\_filesize]

3568 <1>

3569 0000B2C0 3D00000100 <1> cmp eax, 65536 ; 64KB ; small file

3570 0000B2C5 721F <1> jb short csftdf2\_load\_file ; do not display percentage

3571 <1>

3572 <1> csftdf2\_reset\_wf\_percent\_ptr\_chk\_64k:

3573 0000B2C7 B201 <1> mov dl, 1 ; 25/03/2016

3574 <1>

3575 0000B2C9 3D00000400 <1> cmp eax, 65536\*4 ; 256KB

3576 0000B2CE 7310 <1> jnb short csftdf2\_enable\_percentage\_display ; big file

3577 <1>

3578 <1> ; 64-128KB file size for floppy disks

3579 0000B2D0 3815[E8630100] <1> cmp byte [SourceFile\_Drv], dl ; 1 ; read from floppy disk ?

3580 0000B2D6 7608 <1> jna short csftdf2\_enable\_percentage\_display

3581 <1>

3582 0000B2D8 3815[68640100] <1> cmp byte [DestinationFile\_Drv], dl ; 1 ; write to floppy disk ?

3583 0000B2DE 7706 <1> ja short csftdf2\_load\_file

3584 <1>

3585 <1> csftdf2\_enable\_percentage\_display:

3586 0000B2E0 8815[38650100] <1> mov [csftdf\_percentage], dl ; 1

3587 <1>

3588 <1> csftdf2\_load\_file:

3589 <1> ; 13/05/2016

3590 <1> ; 19/03/2016

3591 <1> ; 18/03/2016

3592 <1> ; 17/03/2016

3593 0000B2E6 B40F <1> mov ah, 0Fh

3594 0000B2E8 E8AD61FFFF <1> call \_int10h

3595 <1> ; 13/05/2016

3596 0000B2ED 883D[39650100] <1> mov [csftdf\_videopage], bh ; active video page

3597 0000B2F3 B403 <1> mov ah, 03h

3598 0000B2F5 E8A061FFFF <1> call \_int10h

3599 0000B2FA 668915[3A650100] <1> mov [csftdf\_cursorpos], dx

3600 <1>

3601 0000B301 29C0 <1> sub eax, eax

3602 0000B303 A2[11650100] <1> mov [csftdf\_rw\_err], al ; 0

3603 <1>

3604 <1> ; ///

3605 <1> csftdf\_sf\_amb: ; 15/03/2016

3606 0000B308 8B0D[14650100] <1> mov ecx, [csftdf\_filesize] ; 23/03/2016

3607 <1>

3608 <1> ; TRDOS 386 (TRDOS v2.0)

3609 <1> ; Allocate contiguous memory block for loading the file

3610 <1>

3611 <1> ;mov ecx, [SourceFile\_DirEntry+DirEntry\_FileSize]

3612 <1>

3613 <1> ;sub eax, eax ; First free memory aperture

3614 <1>

3615 <1> ; eax = 0 (Allocate memory from the beginning)

3616 <1> ; ecx = File (Allocation) size in bytes

3617 <1>

3618 0000B30E E811A1FFFF <1> call allocate\_memory\_block

3619 0000B313 7304 <1> jnc short loc\_check\_sf\_save\_loading\_parms

3620 <1>

3621 0000B315 29C0 <1> sub eax, eax

3622 0000B317 29C9 <1> sub ecx, ecx

3623 <1>

3624 <1> loc\_check\_sf\_save\_loading\_parms:

3625 0000B319 A3[18650100] <1> mov [csftdf\_sf\_mem\_addr], eax ; loading address

3626 0000B31E 890D[1C650100] <1> mov [csftdf\_sf\_mem\_bsize], ecx ; block size

3627 <1> ; ///

3628 <1> ; 19/03/2016

3629 0000B324 8B35[3C650100] <1> mov esi, [csftdf\_sf\_drv\_dt] ; logical dos drv desc. tbl.

3630 <1>

3631 <1> ; 17/03/2016

3632 0000B32A 09C0 <1> or eax, eax ; contiguous free memory block address

3633 0000B32C 0F845B010000 <1> jz csftdf2\_read\_sf\_cluster

3634 <1>

3635 <1> ; 18/03/2016

3636 0000B332 8B1D[18650100] <1> mov ebx, [csftdf\_sf\_mem\_addr] ; memory block address

3637 <1>

3638 0000B338 807E0300 <1> cmp byte [esi+LD\_FATType], 0

3639 0000B33C 0F8605020000 <1> jna csftdf2\_load\_fs\_file

3640 <1>

3641 <1> csftdf2\_load\_fat\_file:

3642 0000B342 53 <1> push ebx ; \*

3643 <1>

3644 <1> csftdf2\_load\_fat\_file\_next:

3645 0000B343 BE[23130100] <1> mov esi, msg\_reading

3646 0000B348 E810B0FFFF <1> call print\_msg

3647 <1>

3648 0000B34D 803D[38650100]00 <1> cmp byte [csftdf\_percentage], 0

3649 0000B354 7605 <1> jna short csftdf2\_load\_fat\_file\_1

3650 <1>

3651 0000B356 E87C000000 <1> call csftdf2\_print\_percentage ; 19/03/2016

3652 <1>

3653 <1> csftdf2\_load\_fat\_file\_1:

3654 0000B35B 8B35[3C650100] <1> mov esi, [csftdf\_sf\_drv\_dt]

3655 0000B361 5B <1> pop ebx ; \*

3656 <1>

3657 <1> csftdf2\_load\_fat\_file\_2:

3658 0000B362 E8B8000000 <1> call csftdf2\_read\_fat\_file\_sectors ; 19/03/2016

3659 0000B367 0F8250040000 <1> jc csftdf2\_rw\_error ; eocc! or disk error!

3660 <1>

3661 0000B36D 09D2 <1> or edx, edx ; edx > 0 -> EOF

3662 0000B36F 7520 <1> jnz short csftdf2\_load\_fat\_file\_ok

3663 <1>

3664 0000B371 803D[38650100]00 <1> cmp byte [csftdf\_percentage], 0

3665 0000B378 76E8 <1> jna short csftdf2\_load\_fat\_file\_2

3666 <1>

3667 0000B37A 53 <1> push ebx ; \*

3668 <1>

3669 <1> ; Set cursor position

3670 <1> ; AH= 02h, BH= Page Number, DH= Row, DL= Column

3671 0000B37B 8A3D[39650100] <1> mov bh, [csftdf\_videopage]

3672 0000B381 668B15[3A650100] <1> mov dx, [csftdf\_cursorpos]

3673 0000B388 B402 <1> mov ah, 2

3674 0000B38A E80B61FFFF <1> call \_int10h

3675 0000B38F EBB2 <1> jmp short csftdf2\_load\_fat\_file\_next

3676 <1>

3677 <1> csftdf2\_load\_fat\_file\_ok:

3678 0000B391 803D[38650100]00 <1> cmp byte [csftdf\_percentage], 0

3679 0000B398 0F8651020000 <1> jna csftdf2\_save\_file ; 25/03/2016

3680 <1>

3681 <1> ; "Reading... 100%"

3682 0000B39E BF[3B130100] <1> mov edi, percentagestr

3683 0000B3A3 B031 <1> mov al, '1'

3684 0000B3A5 AA <1> stosb

3685 0000B3A6 B030 <1> mov al, '0'

3686 0000B3A8 AA <1> stosb

3687 0000B3A9 AA <1> stosb

3688 <1>

3689 0000B3AA 8A3D[39650100] <1> mov bh, [csftdf\_videopage]

3690 0000B3B0 668B15[3A650100] <1> mov dx, [csftdf\_cursorpos]

3691 0000B3B7 B402 <1> mov ah, 2

3692 0000B3B9 E8DC60FFFF <1> call \_int10h

3693 <1>

3694 0000B3BE BE[23130100] <1> mov esi, msg\_reading

3695 0000B3C3 E895AFFFFF <1> call print\_msg

3696 <1>

3697 0000B3C8 BE[3B130100] <1> mov esi, percentagestr

3698 0000B3CD E88BAFFFFF <1> call print\_msg

3699 <1>

3700 0000B3D2 E918020000 <1> jmp csftdf2\_save\_file ; 25/03/2016

3701 <1>

3702 <1> csftdf2\_print\_percentage:

3703 <1> ; 09/12/2017

3704 <1> ; 19/03/2016

3705 <1> ; 18/03/2016

3706 0000B3D7 B020 <1> mov al, 20h

3707 0000B3D9 BF[3B130100] <1> mov edi, percentagestr

3708 0000B3DE AA <1> stosb

3709 0000B3DF AA <1> stosb

3710 0000B3E0 A1[30650100] <1> mov eax, [csftdf\_sf\_rbytes]

3711 0000B3E5 BA64000000 <1> mov edx, 100

3712 0000B3EA F7E2 <1> mul edx

3713 0000B3EC 8B0D[14650100] <1> mov ecx, [csftdf\_filesize]

3714 0000B3F2 F7F1 <1> div ecx

3715 0000B3F4 B10A <1> mov cl, 10

3716 0000B3F6 F6F1 <1> div cl

3717 0000B3F8 80C430 <1> add ah, '0'

3718 0000B3FB 8827 <1> mov [edi], ah

3719 0000B3FD 20C0 <1> and al, al

3720 0000B3FF 740A <1> jz short csftdf2\_print\_percent\_1

3721 0000B401 4F <1> dec edi

3722 <1> ;cbw

3723 0000B402 28E4 <1> sub ah, ah ; 09/12/2017

3724 0000B404 F6F1 <1> div cl

3725 0000B406 80C430 <1> add ah, '0'

3726 0000B409 8827 <1> mov [edi], ah

3727 <1> ;and al, al

3728 <1> ;jz short csftdf2\_print\_percent\_1

3729 <1> ;dec edi

3730 <1> ;mov [edi], '1' ; 100%

3731 <1>

3732 <1> csftdf2\_print\_percent\_1:

3733 0000B40B BE[3B130100] <1> mov esi, percentagestr

3734 <1> ;call print\_msg

3735 <1> ;retn

3736 0000B410 E948AFFFFF <1> jmp print\_msg

3737 <1>

3738 <1> csftdf2\_read\_file\_sectors:

3739 <1> ; 19/03/2016

3740 0000B415 807E0300 <1> cmp byte [esi+LD\_FATType], 0

3741 0000B419 0F8627070000 <1> jna csftdf2\_read\_fs\_file\_sectors

3742 <1>

3743 <1> csftdf2\_read\_fat\_file\_sectors:

3744 <1> ; 19/03/2016

3745 <1> ; 18/03/2016

3746 <1> ; return:

3747 <1> ; CF = 0 & EDX > 0 -> END OF FILE

3748 <1> ; CF = 0 & EDX = 0 -> not EOF

3749 <1> ; CF = 1 -> read error (error code in AL)

3750 <1>

3751 <1> csftdf2\_read\_fat\_file\_secs\_0:

3752 0000B41F 8B15[14650100] <1> mov edx, [csftdf\_filesize]

3753 0000B425 2B15[30650100] <1> sub edx, [csftdf\_sf\_rbytes]

3754 0000B42B 3B15[28650100] <1> cmp edx, [csftdf\_r\_size]

3755 0000B431 7306 <1> jnb short csftdf2\_read\_fat\_file\_secs\_1

3756 0000B433 8915[28650100] <1> mov [csftdf\_r\_size], edx

3757 <1>

3758 <1> csftdf2\_read\_fat\_file\_secs\_1:

3759 0000B439 A1[28650100] <1> mov eax, [csftdf\_r\_size]

3760 0000B43E 29D2 <1> sub edx, edx

3761 0000B440 0FB74E11 <1> movzx ecx, word [esi+LD\_BPB+BytesPerSec]

3762 0000B444 01C8 <1> add eax, ecx

3763 0000B446 48 <1> dec eax

3764 0000B447 F7F1 <1> div ecx

3765 0000B449 89C1 <1> mov ecx, eax ; sector count

3766 0000B44B A1[20650100] <1> mov eax, [csftdf\_sf\_cluster]

3767 <1>

3768 <1> ; EBX = memory block address (current)

3769 <1>

3770 0000B450 E821090000 <1> call read\_fat\_file\_sectors

3771 0000B455 7235 <1> jc short csftdf2\_read\_fat\_file\_secs\_3

3772 <1>

3773 <1> ; EBX = next memory address

3774 <1>

3775 0000B457 A1[30650100] <1> mov eax, [csftdf\_sf\_rbytes]

3776 0000B45C 0305[28650100] <1> add eax, [csftdf\_r\_size]

3777 0000B462 8B15[14650100] <1> mov edx, [csftdf\_filesize]

3778 0000B468 39D0 <1> cmp eax, edx

3779 0000B46A 7320 <1> jnb short csftdf2\_read\_fat\_file\_secs\_3 ; edx > 0

3780 0000B46C A3[30650100] <1> mov [csftdf\_sf\_rbytes], eax

3781 <1>

3782 0000B471 53 <1> push ebx ; \*

3783 <1> ; get next cluster (csftdf\_r\_size! bytes)

3784 0000B472 A1[20650100] <1> mov eax, [csftdf\_sf\_cluster]

3785 0000B477 E8CC060000 <1> call get\_next\_cluster

3786 0000B47C 5B <1> pop ebx ; \*

3787 0000B47D 7306 <1> jnc short csftdf2\_read\_fat\_file\_secs\_2

3788 <1>

3789 <1> ; 15/10/2016

3790 <1> ;Disk read error instad of drv not ready err

3791 0000B47F B811000000 <1> mov eax, 17 ; Read error !

3792 0000B484 C3 <1> retn

3793 <1>

3794 <1> csftdf2\_read\_fat\_file\_secs\_2:

3795 0000B485 29D2 <1> sub edx, edx ; 0

3796 0000B487 A3[20650100] <1> mov [csftdf\_sf\_cluster], eax ; next cluster

3797 <1>

3798 <1> csftdf2\_read\_fat\_file\_secs\_3:

3799 0000B48C C3 <1> retn

3800 <1>

3801 <1> csftdf2\_read\_sf\_cluster:

3802 <1> ; 19/03/2016

3803 0000B48D BB00000700 <1> mov ebx, Cluster\_Buffer ; buffer address (64KB)

3804 <1>

3805 0000B492 803D[38650100]00 <1> cmp byte [csftdf\_percentage], 0

3806 0000B499 760D <1> jna short csftdf2\_read\_sf\_clust\_2

3807 <1>

3808 0000B49B 53 <1> push ebx ; \*

3809 <1>

3810 <1> csftdf2\_read\_sf\_clust\_next:

3811 0000B49C E836FFFFFF <1> call csftdf2\_print\_percentage

3812 <1>

3813 <1> csftdf2\_read\_sf\_clust\_0:

3814 0000B4A1 8B35[3C650100] <1> mov esi, [csftdf\_sf\_drv\_dt]

3815 <1> csftdf2\_read\_sf\_clust\_1:

3816 0000B4A7 5B <1> pop ebx ; \*

3817 <1>

3818 <1> csftdf2\_read\_sf\_clust\_2:

3819 0000B4A8 89DA <1> mov edx, ebx

3820 0000B4AA 0315[28650100] <1> add edx, [csftdf\_r\_size]

3821 0000B4B0 81FA00000800 <1> cmp edx, Cluster\_Buffer + 65536

3822 0000B4B6 772F <1> ja short csftdf2\_write\_df\_cluster

3823 <1>

3824 0000B4B8 E858FFFFFF <1> call csftdf2\_read\_file\_sectors ; 19/03/2016

3825 0000B4BD 0F8280020000 <1> jc csftdf2\_save\_fat\_file\_err2 ; eocc! or disk error!

3826 <1>

3827 0000B4C3 09D2 <1> or edx, edx ; edx > 0 -> EOF

3828 0000B4C5 7520 <1> jnz short csftdf2\_write\_df\_cluster

3829 <1>

3830 0000B4C7 803D[38650100]00 <1> cmp byte [csftdf\_percentage], 0

3831 0000B4CE 76D8 <1> jna short csftdf2\_read\_sf\_clust\_2

3832 <1>

3833 0000B4D0 53 <1> push ebx ; \*

3834 <1>

3835 <1> ; Set cursor position

3836 <1> ; AH= 02h, BH= Page Number, DH= Row, DL= Column

3837 0000B4D1 8A3D[39650100] <1> mov bh, [csftdf\_videopage]

3838 0000B4D7 668B15[3A650100] <1> mov dx, [csftdf\_cursorpos]

3839 0000B4DE B402 <1> mov ah, 2

3840 0000B4E0 E8B55FFFFF <1> call \_int10h

3841 0000B4E5 EBB5 <1> jmp short csftdf2\_read\_sf\_clust\_next

3842 <1>

3843 <1> csftdf2\_write\_df\_cluster:

3844 <1> ; 19/03/2016

3845 0000B4E7 8B35[40650100] <1> mov esi, [csftdf\_df\_drv\_dt]

3846 0000B4ED BB00000700 <1> mov ebx, Cluster\_Buffer ; buffer address (64KB)

3847 <1>

3848 <1> csftdf2\_write\_df\_clust\_next:

3849 0000B4F2 E855000000 <1> call csftdf2\_write\_file\_sectors ; 19/03/2016

3850 0000B4F7 0F8246020000 <1> jc csftdf2\_save\_fat\_file\_err2 ; eocc! or disk error!

3851 <1>

3852 0000B4FD 09D2 <1> or edx, edx ; edx > 0 -> EOF

3853 0000B4FF 750A <1> jnz short csftdf2\_rw\_f\_clust\_ok

3854 <1>

3855 0000B501 81FB00000800 <1> cmp ebx, Cluster\_Buffer + 65536

3856 0000B507 72E9 <1> jb short csftdf2\_write\_df\_clust\_next

3857 <1>

3858 0000B509 EB82 <1> jmp short csftdf2\_read\_sf\_cluster

3859 <1>

3860 <1> csftdf2\_rw\_f\_clust\_ok:

3861 0000B50B 803D[38650100]00 <1> cmp byte [csftdf\_percentage], 0

3862 0000B512 0F86B2010000 <1> jna csftdf2\_save\_fat\_file\_4 ; 25/03/2016

3863 <1>

3864 <1> ; "100%"

3865 0000B518 BF[3B130100] <1> mov edi, percentagestr

3866 0000B51D B031 <1> mov al, '1'

3867 0000B51F AA <1> stosb

3868 0000B520 B030 <1> mov al, '0'

3869 0000B522 AA <1> stosb

3870 0000B523 AA <1> stosb

3871 <1>

3872 0000B524 8A3D[39650100] <1> mov bh, [csftdf\_videopage]

3873 0000B52A 668B15[3A650100] <1> mov dx, [csftdf\_cursorpos]

3874 0000B531 B402 <1> mov ah, 2

3875 0000B533 E8625FFFFF <1> call \_int10h

3876 <1>

3877 0000B538 BE[3B130100] <1> mov esi, percentagestr

3878 0000B53D E81BAEFFFF <1> call print\_msg

3879 <1>

3880 0000B542 E983010000 <1> jmp csftdf2\_save\_fat\_file\_4

3881 <1>

3882 <1> csftdf2\_load\_fs\_file:

3883 <1> ; temporary - 18/03/2016

3884 0000B547 E96F020000 <1> jmp csftdf2\_read\_error

3885 <1>

3886 <1> csftdf2\_write\_file\_sectors:

3887 <1> ; 19/03/2016

3888 0000B54C 807E0300 <1> cmp byte [esi+LD\_FATType], 0

3889 0000B550 0F86F1050000 <1> jna csftdf2\_write\_fs\_file\_sectors

3890 <1>

3891 <1> csftdf2\_write\_fat\_file\_sectors:

3892 <1> ; 19/03/2016

3893 <1> ; 18/03/2016

3894 <1> ; return:

3895 <1> ; CF = 0 & EDX > 0 -> END OF FILE

3896 <1> ; CF = 0 & EDX = 0 -> not EOF

3897 <1> ; CF = 1 -> write error (error code in AL)

3898 <1>

3899 <1> csftdf2\_write\_fat\_file\_secs\_0:

3900 0000B556 8B15[14650100] <1> mov edx, [csftdf\_filesize]

3901 0000B55C 2B15[34650100] <1> sub edx, [csftdf\_df\_wbytes]

3902 0000B562 3B15[2C650100] <1> cmp edx, [csftdf\_w\_size]

3903 0000B568 7306 <1> jnb short csftdf2\_write\_fat\_file\_secs\_1

3904 0000B56A 8915[2C650100] <1> mov [csftdf\_w\_size], edx

3905 <1>

3906 <1> csftdf2\_write\_fat\_file\_secs\_1:

3907 0000B570 A1[2C650100] <1> mov eax, [csftdf\_w\_size]

3908 0000B575 29D2 <1> sub edx, edx

3909 0000B577 0FB74E11 <1> movzx ecx, word [esi+LD\_BPB+BytesPerSec]

3910 0000B57B 01C8 <1> add eax, ecx

3911 0000B57D 48 <1> dec eax

3912 0000B57E F7F1 <1> div ecx

3913 0000B580 89C1 <1> mov ecx, eax ; sector count

3914 0000B582 A1[24650100] <1> mov eax, [csftdf\_df\_cluster]

3915 <1>

3916 <1> ; EBX = memory block address (current)

3917 <1>

3918 0000B587 E8A20F0000 <1> call write\_fat\_file\_sectors

3919 0000B58C 7259 <1> jc short csftdf2\_write\_fat\_file\_secs\_4

3920 <1>

3921 <1> ; EBX = next memory address

3922 <1>

3923 0000B58E A1[34650100] <1> mov eax, [csftdf\_df\_wbytes]

3924 0000B593 0305[2C650100] <1> add eax, [csftdf\_w\_size]

3925 0000B599 8B15[14650100] <1> mov edx, [csftdf\_filesize]

3926 0000B59F 39D0 <1> cmp eax, edx

3927 0000B5A1 7344 <1> jnb short csftdf2\_write\_fat\_file\_secs\_4

3928 0000B5A3 A3[34650100] <1> mov [csftdf\_df\_wbytes], eax

3929 <1> ;

3930 0000B5A8 A3[D6640100] <1> mov [DestinationFile\_DirEntry+DirEntry\_FileSize], eax

3931 <1>

3932 0000B5AD 53 <1> push ebx ; \*

3933 <1>

3934 0000B5AE 803D[12650100]01 <1> cmp byte [DestinationFileFound], 1

3935 0000B5B5 7210 <1> jb short csftdf2\_write\_fat\_file\_secs\_2

3936 <1>

3937 <1> ; get next cluster (csftdf\_w\_size! bytes)

3938 0000B5B7 A1[24650100] <1> mov eax, [csftdf\_df\_cluster]

3939 0000B5BC E887050000 <1> call get\_next\_cluster

3940 0000B5C1 731C <1> jnc short csftdf2\_write\_fat\_file\_secs\_3

3941 <1>

3942 0000B5C3 21C0 <1> and eax, eax ; end of cluster chain!?

3943 0000B5C5 7521 <1> jnz short csftdf2\_write\_fat\_file\_secs\_5 ; disk error !

3944 <1>

3945 <1> csftdf2\_write\_fat\_file\_secs\_2:

3946 0000B5C7 A1[24650100] <1> mov eax, [csftdf\_df\_cluster] ; last cluster

3947 0000B5CC E8800E0000 <1> call add\_new\_cluster

3948 0000B5D1 7215 <1> jc short csftdf2\_write\_fat\_file\_secs\_5

3949 <1>

3950 <1> ; NOTE: Destination file size may be bigger than

3951 <1> ; source file size when the last reading fails after here.

3952 <1> ; (The last -empty- cluster of destination file must be

3953 <1> ; truncated and LMDT must be current date&time for partial

3954 <1> ; copy result!)

3955 0000B5D3 8B15[2C650100] <1> mov edx, [csftdf\_w\_size] ; bytes per cluster

3956 0000B5D9 0115[D6640100] <1> add [DestinationFile\_DirEntry+DirEntry\_FileSize], edx

3957 <1>

3958 <1> csftdf2\_write\_fat\_file\_secs\_3:

3959 0000B5DF 5B <1> pop ebx ; \*

3960 0000B5E0 29D2 <1> sub edx, edx ; 0

3961 0000B5E2 A3[24650100] <1> mov [csftdf\_df\_cluster], eax ; next cluster

3962 <1>

3963 <1> csftdf2\_write\_fat\_file\_secs\_4:

3964 0000B5E7 C3 <1> retn

3965 <1>

3966 <1> csftdf2\_write\_fat\_file\_secs\_5:

3967 0000B5E8 5B <1> pop ebx ; \*

3968 <1> ; 16/10/2016 (1Dh -> 18)

3969 0000B5E9 B812000000 <1> mov eax, 18 ; Write error !

3970 0000B5EE C3 <1> retn

3971 <1>

3972 <1> csftdf2\_save\_file:

3973 <1> ; 09/12/2017

3974 <1> ; 25/03/2016

3975 <1> ; 19/03/2016

3976 <1> ; 18/03/2016

3977 0000B5EF 8B35[40650100] <1> mov esi, [csftdf\_df\_drv\_dt] ; logical dos drv desc. tbl.

3978 <1>

3979 0000B5F5 8B1D[18650100] <1> mov ebx, [csftdf\_sf\_mem\_addr] ; memory block address

3980 <1>

3981 0000B5FB 807E0300 <1> cmp byte [esi+LD\_FATType], 0

3982 0000B5FF 0F86F4010000 <1> jna csftdf2\_save\_fs\_file

3983 <1>

3984 <1> csftdf2\_save\_fat\_file:

3985 0000B605 53 <1> push ebx; \*

3986 <1>

3987 0000B606 803D[38650100]00 <1> cmp byte [csftdf\_percentage], 0

3988 0000B60D 7724 <1> ja short csftdf2\_save\_fat\_file\_0

3989 <1>

3990 <1> ; Set cursor position

3991 <1> ; AH= 02h, BH= Page Number, DH= Row, DL= Column

3992 0000B60F 8A3D[39650100] <1> mov bh, [csftdf\_videopage]

3993 0000B615 668B15[3A650100] <1> mov dx, [csftdf\_cursorpos]

3994 0000B61C B402 <1> mov ah, 2

3995 0000B61E E8775EFFFF <1> call \_int10h

3996 <1>

3997 0000B623 BE[2F130100] <1> mov esi, msg\_writing

3998 0000B628 E830ADFFFF <1> call print\_msg

3999 <1>

4000 <1> csftdf2\_save\_fat\_file\_next:

4001 0000B62D 8B35[40650100] <1> mov esi, [csftdf\_df\_drv\_dt] ; 25/03/2016

4002 <1>

4003 <1> csftdf2\_save\_fat\_file\_0:

4004 0000B633 5B <1> pop ebx ; \*

4005 <1>

4006 <1> csftdf2\_save\_fat\_file\_1:

4007 0000B634 E813FFFFFF <1> call csftdf2\_write\_file\_sectors ; 19/03/2016

4008 0000B639 0F827E010000 <1> jc csftdf2\_rw\_error ; eocc! or disk error!

4009 <1>

4010 0000B63F 09D2 <1> or edx, edx ; edx > 0 -> EOF

4011 0000B641 756D <1> jnz short csftdf2\_save\_fat\_file\_3 ; 25/03/2016

4012 <1>

4013 0000B643 803D[38650100]00 <1> cmp byte [csftdf\_percentage], 0

4014 0000B64A 76E8 <1> jna short csftdf2\_save\_fat\_file\_1

4015 <1>

4016 0000B64C B020 <1> mov al, 20h

4017 0000B64E BF[3B130100] <1> mov edi, percentagestr

4018 0000B653 AA <1> stosb

4019 0000B654 AA <1> stosb

4020 0000B655 A1[34650100] <1> mov eax, [csftdf\_df\_wbytes]

4021 0000B65A BA64000000 <1> mov edx, 100

4022 0000B65F F7E2 <1> mul edx

4023 0000B661 8B0D[14650100] <1> mov ecx, [csftdf\_filesize]

4024 0000B667 F7F1 <1> div ecx

4025 0000B669 B10A <1> mov cl, 10

4026 0000B66B F6F1 <1> div cl

4027 0000B66D 80C430 <1> add ah, '0'

4028 0000B670 8827 <1> mov [edi], ah

4029 0000B672 20C0 <1> and al, al

4030 0000B674 740A <1> jz short csftdf2\_save\_fat\_file\_2

4031 0000B676 4F <1> dec edi

4032 <1> ;cbw

4033 0000B677 30E4 <1> xor ah, ah ; 09/12/2017

4034 0000B679 F6F1 <1> div cl

4035 0000B67B 80C430 <1> add ah, '0'

4036 0000B67E 8827 <1> mov [edi], ah

4037 <1> ;and al, al

4038 <1> ;jz short csftdf2\_save\_fat\_file\_2

4039 <1> ;dec edi

4040 <1> ;mov [edi], '1' ; 100%

4041 <1>

4042 <1> csftdf2\_save\_fat\_file\_2:

4043 0000B680 53 <1> push ebx ; \*

4044 <1>

4045 0000B681 E802000000 <1> call csftdf2\_print\_wr\_percentage ; 25/03/2016

4046 <1>

4047 0000B686 EBA5 <1> jmp csftdf2\_save\_fat\_file\_next

4048 <1>

4049 <1> csftdf2\_print\_wr\_percentage:

4050 <1> ; Set cursor position

4051 <1> ; AH= 02h, BH= Page Number, DH= Row, DL= Column

4052 0000B688 8A3D[39650100] <1> mov bh, [csftdf\_videopage]

4053 0000B68E 668B15[3A650100] <1> mov dx, [csftdf\_cursorpos]

4054 0000B695 B402 <1> mov ah, 2

4055 0000B697 E8FE5DFFFF <1> call \_int10h

4056 <1>

4057 0000B69C BE[2F130100] <1> mov esi, msg\_writing

4058 0000B6A1 E8B7ACFFFF <1> call print\_msg

4059 <1>

4060 0000B6A6 BE[3B130100] <1> mov esi, percentagestr

4061 <1> ;call print\_msg

4062 <1> ;retn

4063 0000B6AB E9ADACFFFF <1> jmp print\_msg

4064 <1>

4065 <1> csftdf2\_save\_fat\_file\_3:

4066 0000B6B0 803D[38650100]00 <1> cmp byte [csftdf\_percentage], 0

4067 0000B6B7 7611 <1> jna csftdf2\_save\_fat\_file\_4 ; 25/03/2016

4068 <1>

4069 <1> ; "100%"

4070 0000B6B9 BF[3B130100] <1> mov edi, percentagestr

4071 0000B6BE B031 <1> mov al, '1'

4072 0000B6C0 AA <1> stosb

4073 0000B6C1 B030 <1> mov al, '0'

4074 0000B6C3 AA <1> stosb

4075 0000B6C4 AA <1> stosb

4076 <1>

4077 0000B6C5 E8BEFFFFFF <1> call csftdf2\_print\_wr\_percentage

4078 <1>

4079 <1> csftdf2\_save\_fat\_file\_4:

4080 0000B6CA 803D[12650100]00 <1> cmp byte [DestinationFileFound], 0

4081 0000B6D1 7647 <1> jna short csftdf2\_save\_fat\_file\_6

4082 <1>

4083 0000B6D3 8B35[40650100] <1> mov esi, [csftdf\_df\_drv\_dt] ; 31/03/2016

4084 <1>

4085 0000B6D9 A1[24650100] <1> mov eax, [csftdf\_df\_cluster] ; last cluster

4086 0000B6DE E865040000 <1> call get\_next\_cluster

4087 0000B6E3 7235 <1> jc short csftdf2\_save\_fat\_file\_6 ; eocc! or disk error!

4088 <1>

4089 0000B6E5 A1[24650100] <1> mov eax, [csftdf\_df\_cluster] ; last cluster

4090 <1> ;xor ecx, ecx

4091 <1> ;mov [FAT\_ClusterCounter], ecx ; 0 ; reset

4092 <1> ;dec ecx ; 0FFFFFFFFh

4093 <1> ;shr ecx, 4 ; 28 bit ; 0FFFFFFFh

4094 0000B6EA B9FFFFFF0F <1> mov ecx, 0FFFFFFFh

4095 0000B6EF E87E070000 <1> call update\_cluster

4096 0000B6F4 7224 <1> jc short csftdf2\_save\_fat\_file\_6 ; really last cluster!?

4097 <1>

4098 0000B6F6 A3[24650100] <1> mov [csftdf\_df\_cluster], eax ; next cluster

4099 <1>

4100 <1> ; byte [FAT\_BuffValidData] = 2

4101 0000B6FB E82F0A0000 <1> call save\_fat\_buffer

4102 0000B700 730E <1> jnc short csftdf2\_save\_fat\_file\_5

4103 <1>

4104 0000B702 8B15[14650100] <1> mov edx, [csftdf\_filesize]

4105 0000B708 8915[D6640100] <1> mov [DestinationFile\_DirEntry+DirEntry\_FileSize], edx

4106 0000B70E EB58 <1> jmp short csftdf2\_save\_fat\_file\_err3

4107 <1>

4108 <1> csftdf2\_save\_fat\_file\_5:

4109 0000B710 A1[24650100] <1> mov eax, [csftdf\_df\_cluster]

4110 <1>

4111 <1> ; EAX = First cluster to be truncated/unlinked

4112 <1> ; ESI = Logical dos drive description table address

4113 0000B715 E8580C0000 <1> call truncate\_cluster\_chain

4114 <1>

4115 <1> csftdf2\_save\_fat\_file\_6:

4116 <1> ; 28/03/2016

4117 0000B71A BE[45640100] <1> mov esi, SourceFile\_DirEntry+DirEntry\_Attr ; +11 to + 18

4118 0000B71F BF[C5640100] <1> mov edi, DestinationFile\_DirEntry+DirEntry\_Attr ; +11 to + 18

4119 0000B724 A4 <1> movsb ; +11

4120 0000B725 A5 <1> movsd ; +12 .. +15

4121 0000B726 66A5 <1> movsw ; +16 .. +17

4122 <1> ; + 18

4123 0000B728 83C604 <1> add esi, 4

4124 0000B72B 83C704 <1> add edi, 4

4125 0000B72E A5 <1> movsd ; DirEntry\_WrtTime ; +22 .. +25

4126 <1>

4127 0000B72F 8B15[14650100] <1> mov edx, [csftdf\_filesize]

4128 0000B735 8915[D6640100] <1> mov [DestinationFile\_DirEntry+DirEntry\_FileSize], edx

4129 <1>

4130 0000B73B E8BAF0FFFF <1> call convert\_current\_date\_time

4131 <1> ; DX = Date in dos dir entry format

4132 <1> ; AX = Time in dos dir entry format

4133 0000B740 EB4D <1> jmp short csftdf2\_save\_fat\_file\_7

4134 <1>

4135 <1> csftdf2\_save\_fat\_file\_err1:

4136 0000B742 5B <1> pop ebx ; \*

4137 <1> csftdf2\_save\_fat\_file\_err2:

4138 0000B743 A1[34650100] <1> mov eax, [csftdf\_df\_wbytes]

4139 0000B748 8B15[D6640100] <1> mov edx, [DestinationFile\_DirEntry+DirEntry\_FileSize]

4140 0000B74E 39C2 <1> cmp edx, eax

4141 0000B750 7616 <1> jna short csftdf2\_save\_fat\_file\_err3

4142 0000B752 A1[24650100] <1> mov eax, [csftdf\_df\_cluster] ; last (empty) cluster

4143 <1> ; ESI = Logical dos drive description table address

4144 0000B757 E8160C0000 <1> call truncate\_cluster\_chain

4145 0000B75C 720A <1> jc short csftdf2\_save\_fat\_file\_err3

4146 0000B75E A1[34650100] <1> mov eax, [csftdf\_df\_wbytes]

4147 0000B763 A3[D6640100] <1> mov [DestinationFile\_DirEntry+DirEntry\_FileSize], eax

4148 <1> csftdf2\_save\_fat\_file\_err3:

4149 0000B768 E88DF0FFFF <1> call convert\_current\_date\_time

4150 <1> ; DX = Date in dos dir entry format

4151 <1> ; AX = Time in dos dir entry format

4152 0000B76D C605[C7640100]00 <1> mov byte [DestinationFile\_DirEntry+DirEntry\_CrtTimeTenth], 0

4153 0000B774 66A3[C8640100] <1> mov [DestinationFile\_DirEntry+DirEntry\_CrtTime], ax

4154 0000B77A 668915[CA640100] <1> mov [DestinationFile\_DirEntry+DirEntry\_CrtDate], dx

4155 0000B781 66A3[D0640100] <1> mov [DestinationFile\_DirEntry+DirEntry\_WrtTime], ax

4156 0000B787 668915[D2640100] <1> mov [DestinationFile\_DirEntry+DirEntry\_WrtDate], dx

4157 0000B78E F9 <1> stc

4158 <1> csftdf2\_save\_fat\_file\_7:

4159 0000B78F 9C <1> pushf

4160 0000B790 668915[CC640100] <1> mov [DestinationFile\_DirEntry+DirEntry\_LastAccDate], dx

4161 0000B797 BE[BA640100] <1> mov esi, DestinationFile\_DirEntry

4162 0000B79C BF00000800 <1> mov edi, Directory\_Buffer

4163 0000B7A1 0FB70D[E2640100] <1> movzx ecx, word [DestinationFile\_DirEntryNumber] ; (<2048)

4164 0000B7A8 66C1E105 <1> shl cx, 5 ; 32 \* directory entry number

4165 0000B7AC 01CF <1> add edi, ecx

4166 <1> ;mov ecx, 8

4167 0000B7AE 66B90800 <1> mov cx, 8

4168 0000B7B2 F3A5 <1> rep movsd

4169 0000B7B4 9D <1> popf

4170 0000B7B5 730B <1> jnc short csftdf2\_write\_file\_OK

4171 <1>

4172 <1> csftdf2\_write\_error:

4173 <1> ; 18/03/2016

4174 0000B7B7 B01D <1> mov al, 1Dh ; write error

4175 0000B7B9 EB02 <1> jmp short csftdf2\_rw\_error

4176 <1>

4177 <1> ; 16/03/2016

4178 <1> csftdf2\_read\_error:

4179 0000B7BB B011 <1> mov al, 17 ; ; Drive not ready or read error!

4180 <1> csftdf2\_rw\_error:

4181 0000B7BD A2[11650100] <1> mov [csftdf\_rw\_err], al

4182 <1>

4183 <1> csftdf2\_write\_file\_OK:

4184 <1> ; 18/03/2016

4185 0000B7C2 C605[28610100]02 <1> mov byte [DirBuff\_ValidData], 2

4186 0000B7C9 E8CAF0FFFF <1> call save\_directory\_buffer

4187 <1>

4188 <1> ; Update last modification date&time of destination

4189 <1> ; file's (parent) directory

4190 0000B7CE E860F1FFFF <1> call update\_parent\_dir\_lmdt

4191 <1> ;

4192 0000B7D3 A1[18650100] <1> mov eax, [csftdf\_sf\_mem\_addr] ; start address

4193 <1>

4194 0000B7D8 21C0 <1> and eax, eax

4195 0000B7DA 750E <1> jnz short csftdf2\_dealloc\_mblock

4196 <1>

4197 0000B7DC 88C5 <1> mov ch, al ; 0 (Cluster r/w, not full loading)

4198 <1> csftdf2\_dealloc\_retn:

4199 0000B7DE 8A0D[11650100] <1> mov cl, [csftdf\_rw\_err]

4200 0000B7E4 A1[24650100] <1> mov eax, [csftdf\_df\_cluster]

4201 0000B7E9 C3 <1> retn

4202 <1>

4203 <1> csftdf2\_dealloc\_mblock:

4204 0000B7EA 8B0D[1C650100] <1> mov ecx, [csftdf\_sf\_mem\_bsize] ; block size

4205 0000B7F0 E83C9EFFFF <1> call deallocate\_memory\_block

4206 0000B7F5 B5FF <1> mov ch, 0FFh ; (File was full loaded at memory)

4207 0000B7F7 EBE5 <1> jmp short csftdf2\_dealloc\_retn

4208 <1>

4209 <1> csftdf2\_save\_fs\_file:

4210 <1> ; 16/10/2016 (1Dh -> 18)

4211 <1> ; temporary - (21/03/2016)

4212 0000B7F9 B812000000 <1> mov eax, 18 ; write error

4213 0000B7FE F9 <1> stc

4214 0000B7FF C3 <1> retn

4215 <1>

4216 <1> create\_file:

4217 <1> ; 16/10/2016

4218 <1> ; 24/03/2016, 31/03/2016

4219 <1> ; 20/03/2016, 21/03/2016, 23/03/2016

4220 <1> ; 19/03/2016 (TRDOS 396 = TRDOS v2.0)

4221 <1> ; 03/09/2011 (FILE.ASM, 'proc\_create\_file')

4222 <1> ; 09/08/2010

4223 <1> ;

4224 <1> ; INPUT ->

4225 <1> ; EAX = File Size

4226 <1> ; ESI = ASCIIZ File Name

4227 <1> ; CL = File Attributes

4228 <1> ; EBX = FFFFFFFFh -> create empty file

4229 <1> ; (only for FAT fs)

4230 <1> ; OUTPUT ->

4231 <1> ; CF = 0 ->

4232 <1> ; EAX = New file's first cluster

4233 <1> ; ESI = Logical Dos Drv Descr. Table Addr.

4234 <1> ; EBX = offset CreateFile\_Size

4235 <1> ; ECX = Sectors per cluster (<256)

4236 <1> ; EDX = Directory entry index/number (<65536)

4237 <1> ; CF = 1 -> error code in AL

4238 <1>

4239 <1> ; test cl, 18h (directory or volume name)

4240 <1> ; jnz short loc\_createfile\_access\_denied

4241 0000B800 80E107 <1> and cl, 07h ; S, H, R

4242 0000B803 880D[60650100] <1> mov [createfile\_attrib], cl

4243 <1>

4244 0000B809 89D9 <1> mov ecx, ebx

4245 0000B80B 89F3 <1> mov ebx, esi ; ASCIIZ File Name address

4246 0000B80D 29D2 <1> sub edx, edx

4247 0000B80F 8A35[FE580100] <1> mov dh, [Current\_Drv]

4248 0000B815 BE00010900 <1> mov esi, Logical\_DOSDisks

4249 0000B81A 01D6 <1> add esi, edx

4250 <1>

4251 0000B81C 8815[6B650100] <1> mov [createfile\_UpdatePDir], dl ; 0 ; 31/03/2016

4252 <1>

4253 <1> ; LD\_DiskType = 0 for write protection (read only)

4254 0000B822 807E0101 <1> cmp byte [esi+LD\_DiskType], 1 ; 0 = Invalid

4255 0000B826 730A <1> jnb short loc\_createfile\_check\_file\_sytem

4256 <1> ; 16/10/2016 (TRDOS Error code: 30, disk write protected)

4257 0000B828 B81E000000 <1> mov eax, 30 ; 13h, MSDOS err : Disk write-protected

4258 0000B82D 66BA0000 <1> mov dx, 0

4259 <1> ; err retn: EDX = 0, EBX = File name offset

4260 <1> ; ESI -> Dos drive description table address

4261 0000B831 C3 <1> retn

4262 <1>

4263 <1> ;loc\_createfile\_access\_denied:

4264 <1> ; mov eax, 05h ; access denied (invalid attributes input)

4265 <1> ; stc

4266 <1> ; retn

4267 <1>

4268 <1> loc\_createfile\_check\_file\_sytem:

4269 0000B832 807E0301 <1> cmp byte [esi+LD\_FATType], 1

4270 0000B836 730A <1> jnb short loc\_createfile\_chk\_empty\_FAT\_file\_sign1

4271 <1>

4272 0000B838 A3[4C650100] <1> mov [createfile\_size], eax

4273 <1> ; ESI = Logical Dos Drive Description Table address

4274 <1> ; EBX = ASCIIZ File Name address

4275 0000B83D E9FE020000 <1> jmp create\_fs\_file

4276 <1>

4277 <1> loc\_createfile\_chk\_empty\_FAT\_file\_sign1:

4278 <1> ; ECX = FFFFFFFFh -> create empty file if drive has FAT fs

4279 0000B842 41 <1> inc ecx

4280 0000B843 7506 <1> jnz short loc\_createfile\_chk\_empty\_FAT\_file\_sign2

4281 0000B845 890D[4C650100] <1> mov [createfile\_size], ecx ; 0 ; empty file

4282 <1>

4283 <1> loc\_createfile\_chk\_empty\_FAT\_file\_sign2:

4284 <1> ; 23/03/2016

4285 0000B84B 668B4E11 <1> mov cx, [esi+LD\_BPB+BytesPerSec]

4286 0000B84F 66890D[68650100] <1> mov [createfile\_BytesPerSec], cx

4287 <1>

4288 <1> ; EBX = ASCIIZ File Name address

4289 0000B856 0FB65613 <1> movzx edx, byte [esi+LD\_BPB+SecPerClust]

4290 0000B85A 8815[61650100] <1> mov [createfile\_SecPerClust], dl

4291 0000B860 8B4E74 <1> mov ecx, [esi+LD\_FreeSectors]

4292 0000B863 39D1 <1> cmp ecx, edx ; byte [createfile\_SecPerClust]

4293 0000B865 7306 <1> jnb short loc\_create\_fat\_file

4294 <1>

4295 <1> loc\_createfile\_insufficient\_disk\_space:

4296 0000B867 B827000000 <1> mov eax, 27h

4297 <1> loc\_createfile\_gffc\_retn:

4298 0000B86C C3 <1> retn

4299 <1>

4300 <1> loc\_create\_fat\_file:

4301 0000B86D 891D[44650100] <1> mov [createfile\_Name\_Offset], ebx

4302 0000B873 890D[48650100] <1> mov [createfile\_FreeSectors], ecx

4303 <1>

4304 <1> loc\_createfile\_gffc\_1:

4305 0000B879 E821050000 <1> call get\_first\_free\_cluster

4306 0000B87E 72EC <1> jc short loc\_createfile\_gffc\_retn

4307 <1>

4308 0000B880 A3[50650100] <1> mov [createfile\_FFCluster], eax

4309 <1>

4310 <1> loc\_createfile\_locate\_ffe\_on\_directory:

4311 <1> ; Current directory fcluster <> Directory buffer cluster

4312 <1> ; Current directory will be reloaded by

4313 <1> ; 'locate\_current\_dir\_file' procedure

4314 <1> ;

4315 <1> ; ESI = Logical Dos Drv Desc. Table Adress

4316 0000B885 56 <1> push esi ; \*

4317 0000B886 31C0 <1> xor eax, eax

4318 <1>

4319 0000B888 A3[1E610100] <1> mov dword [FAT\_ClusterCounter], eax ; 0

4320 <1> ; 21/03/2016

4321 0000B88D A2[6A650100] <1> mov byte [createfile\_wfc], al ; 0

4322 <1>

4323 0000B892 89C1 <1> mov ecx, eax

4324 0000B894 6649 <1> dec cx ; FFFFh

4325 <1> ; CX = FFFFh -> find first deleted or free entry

4326 <1> ; ESI would be ASCIIZ filename address if the call

4327 <1> ; would not be for first free or deleted dir entry

4328 0000B896 E8D7E7FFFF <1> call locate\_current\_dir\_file

4329 0000B89B 0F83EE000000 <1> jnc loc\_createfile\_set\_ff\_dir\_entry

4330 0000B8A1 5E <1> pop esi ; \*

4331 <1> ; ESI = Logical DOS Drv. Description Table Address

4332 0000B8A2 83F802 <1> cmp eax, 2

4333 0000B8A5 7402 <1> je short loc\_createfile\_add\_new\_cluster

4334 <1> loc\_createfile\_locate\_file\_stc\_retn:

4335 0000B8A7 F9 <1> stc

4336 0000B8A8 C3 <1> retn

4337 <1>

4338 <1> loc\_createfile\_add\_new\_cluster:

4339 0000B8A9 803D[FD580100]02 <1> cmp byte [Current\_FATType], 2

4340 <1> ;cmp byte [esi+LD\_FATType], 2

4341 0000B8B0 770C <1> ja short loc\_createfile\_add\_new\_cluster\_check\_fsc

4342 0000B8B2 803D[FC580100]01 <1> cmp byte [Current\_Dir\_Level], 1

4343 <1> ;cmp byte [esi+LD\_CDirLevel], 1

4344 0000B8B9 7303 <1> jnb short loc\_createfile\_add\_new\_cluster\_check\_fsc

4345 <1>

4346 <1> ;mov eax, 12

4347 0000B8BB B00C <1> mov al, 12 ; No more files

4348 <1>

4349 <1> loc\_createfile\_anc\_retn:

4350 0000B8BD C3 <1> retn

4351 <1>

4352 <1> loc\_createfile\_add\_new\_cluster\_check\_fsc:

4353 0000B8BE 8B0D[48650100] <1> mov ecx, [createfile\_FreeSectors]

4354 0000B8C4 0FB605[61650100] <1> movzx eax, byte [createfile\_SecPerClust]

4355 0000B8CB 66D1E0 <1> shl ax, 1 ; AX = 2 \* AX

4356 0000B8CE 39C1 <1> cmp ecx, eax

4357 0000B8D0 7295 <1> jb short loc\_createfile\_insufficient\_disk\_space

4358 <1>

4359 <1> loc\_createfile\_add\_new\_subdir\_cluster:

4360 0000B8D2 8B15[2D610100] <1> mov edx, [DirBuff\_Cluster]

4361 0000B8D8 8915[54650100] <1> mov [createfile\_LastDirCluster], edx

4362 <1>

4363 0000B8DE A1[50650100] <1> mov eax, [createfile\_FFCluster]

4364 0000B8E3 E846040000 <1> call load\_FAT\_sub\_directory

4365 0000B8E8 72D3 <1> jc short loc\_createfile\_anc\_retn

4366 <1>

4367 <1> pass\_createfile\_add\_new\_subdir\_cluster:

4368 <1> ;movzx eax, word [esi+LD\_BPB+BytesPerSec]

4369 0000B8EA 0FB705[68650100] <1> movzx eax, word [createfile\_BytesPerSec] ; 23/03/2016

4370 0000B8F1 F7E1 <1> mul ecx ; ecx = directory buffer sector count

4371 0000B8F3 89C1 <1> mov ecx, eax

4372 0000B8F5 C1E902 <1> shr ecx, 2 ; dword count

4373 0000B8F8 29C0 <1> sub eax, eax ; 0

4374 0000B8FA F3AB <1> rep stosd

4375 <1> ;

4376 0000B8FC C605[28610100]02 <1> mov byte [DirBuff\_ValidData], 2

4377 0000B903 E890EFFFFF <1> call save\_directory\_buffer

4378 0000B908 72B3 <1> jc short loc\_createfile\_anc\_retn

4379 <1>

4380 <1> loc\_createfile\_save\_added\_subdir\_cluster:

4381 0000B90A A1[54650100] <1> mov eax, [createfile\_LastDirCluster]

4382 0000B90F 8B0D[50650100] <1> mov ecx, [createfile\_FFCluster]

4383 0000B915 E858050000 <1> call update\_cluster

4384 0000B91A 7304 <1> jnc short loc\_createfile\_save\_fat\_buffer\_0

4385 0000B91C 09C0 <1> or eax, eax ; EAX = 0 -> cluster value is 0 or eocc

4386 0000B91E 751A <1> jnz short loc\_createfile\_save\_fat\_buffer\_stc\_retn

4387 <1>

4388 <1> loc\_createfile\_save\_fat\_buffer\_0:

4389 0000B920 A1[50650100] <1> mov eax, [createfile\_FFCluster]

4390 0000B925 A3[54650100] <1> mov [createfile\_LastDirCluster], eax

4391 0000B92A B9FFFFFF0F <1> mov ecx, 0FFFFFFFh ; 28 bit

4392 0000B92F E83E050000 <1> call update\_cluster

4393 0000B934 7306 <1> jnc short loc\_createfile\_save\_fat\_buffer\_1

4394 0000B936 09C0 <1> or eax, eax ; Was it free cluster

4395 0000B938 7402 <1> jz short loc\_createfile\_save\_fat\_buffer\_1

4396 <1>

4397 <1> loc\_createfile\_save\_fat\_buffer\_stc\_retn:

4398 0000B93A F9 <1> stc

4399 <1> loc\_createfile\_save\_fat\_buffer\_retn:

4400 <1> loc\_createfile\_gffc\_2\_stc\_retn:

4401 0000B93B C3 <1> retn

4402 <1>

4403 <1> loc\_createfile\_save\_fat\_buffer\_1:

4404 <1> ; byte [FAT\_BuffValidData] = 2

4405 0000B93C E8EE070000 <1> call save\_fat\_buffer

4406 0000B941 72F8 <1> jc short loc\_createfile\_save\_fat\_buffer\_retn

4407 <1>

4408 0000B943 803D[1E610100]01 <1> cmp byte [FAT\_ClusterCounter], 1

4409 0000B94A 7222 <1> jb short loc\_createfile\_save\_fat\_buffer\_2

4410 <1>

4411 <1> ; ESI = Logical DOS Drive Description Table address

4412 0000B94C A1[1E610100] <1> mov eax, [FAT\_ClusterCounter]

4413 <1>

4414 0000B951 C605[1E610100]00 <1> mov byte [FAT\_ClusterCounter], 0 ; 21/03/2016

4415 <1>

4416 0000B958 66BB01FF <1> mov bx, 0FF01h ; add free clusters

4417 0000B95C E863080000 <1> call calculate\_fat\_freespace

4418 <1>

4419 <1> ;inc eax ; 0FFFFFFFFh -> 0 ; recalculation is needed!

4420 <1> ;jnz short loc\_createfile\_save\_fat\_buffer\_2

4421 <1>

4422 <1> ; ecx > 0 -> Recalculation is needed

4423 0000B961 09C9 <1> or ecx, ecx

4424 0000B963 7409 <1> jz short loc\_createfile\_save\_fat\_buffer\_2

4425 <1>

4426 0000B965 66BB00FF <1> mov bx, 0FF00h ; ; recalculate free space

4427 0000B969 E856080000 <1> call calculate\_fat\_freespace

4428 <1>

4429 <1> loc\_createfile\_save\_fat\_buffer\_2:

4430 <1> ;call update\_parent\_dir\_lmdt

4431 <1>

4432 <1> loc\_createfile\_gffc\_2:

4433 0000B96E E82C040000 <1> call get\_first\_free\_cluster

4434 0000B973 72C6 <1> jc short loc\_createfile\_gffc\_2\_stc\_retn

4435 <1>

4436 0000B975 A3[50650100] <1> mov [createfile\_FFCluster], eax

4437 <1>

4438 0000B97A A1[54650100] <1> mov eax, [createfile\_LastDirCluster]

4439 <1>

4440 0000B97F E8AA030000 <1> call load\_FAT\_sub\_directory

4441 0000B984 72B5 <1> jc short loc\_createfile\_gffc\_2\_stc\_retn

4442 <1>

4443 0000B986 BF00000800 <1> mov edi, Directory\_Buffer

4444 <1>

4445 0000B98B 6629DB <1> sub bx, bx ; directory entry index/number = 0

4446 <1>

4447 0000B98E 56 <1> push esi ; \* ; 23/03/2016

4448 <1>

4449 <1> loc\_createfile\_set\_ff\_dir\_entry:

4450 0000B98F 66891D[62650100] <1> mov [createfile\_DirIndex], bx

4451 <1>

4452 <1> ; EDI = Directory entry address

4453 0000B996 8B35[44650100] <1> mov esi, [createfile\_Name\_Offset]

4454 0000B99C A1[50650100] <1> mov eax, [createfile\_FFCluster]

4455 0000B9A1 A3[58650100] <1> mov [createfile\_Cluster], eax ; 24/03/2016

4456 0000B9A6 B5FF <1> mov ch, 0FFh

4457 0000B9A8 8A0D[60650100] <1> mov cl, [createfile\_attrib] ; file attributes

4458 <1> ; CH > 0 -> File size is in [EBX]

4459 0000B9AE BB[4C650100] <1> mov ebx, createfile\_size

4460 <1>

4461 0000B9B3 E803EEFFFF <1> call make\_directory\_entry

4462 <1>

4463 0000B9B8 5E <1> pop esi ; \* ; ESI = Logical Dos Drv Desc. Table address

4464 <1>

4465 0000B9B9 C605[28610100]02 <1> mov byte [DirBuff\_ValidData], 2

4466 0000B9C0 E8D3EEFFFF <1> call save\_directory\_buffer

4467 0000B9C5 7221 <1> jc short loc\_createfile\_set\_ff\_dir\_entry\_retn

4468 <1>

4469 0000B9C7 C605[6B650100]01 <1> mov byte [createfile\_UpdatePDir], 1 ; 31/03/2016

4470 <1>

4471 <1> loc\_createfile\_get\_set\_write\_file\_cluster:

4472 0000B9CE A1[4C650100] <1> mov eax, [createfile\_size]

4473 0000B9D3 09C0 <1> or eax, eax

4474 0000B9D5 7570 <1> jnz short loc\_createfile\_get\_set\_wfc\_cont

4475 0000B9D7 40 <1> inc eax

4476 <1> ; 23/03/2016

4477 0000B9D8 0FB61D[61650100] <1> movzx ebx, byte [createfile\_SecPerClust]

4478 <1> ;movzx ecx, word [esi+LD\_BPB+BytesPerSec] ; 512

4479 0000B9DF 0FB70D[68650100] <1> movzx ecx, word [createfile\_BytesPerSec] ; 512

4480 0000B9E6 EB7C <1> jmp loc\_createfile\_set\_cluster\_count

4481 <1>

4482 <1> loc\_createfile\_set\_ff\_dir\_entry\_retn:

4483 0000B9E8 C3 <1> retn

4484 <1>

4485 <1> loc\_createfile\_write\_fcluster\_to\_disk:

4486 0000B9E9 034668 <1> add eax, [esi+LD\_DATABegin] ; convert to physical address

4487 0000B9EC BB00000700 <1> mov ebx, Cluster\_Buffer

4488 <1> ; ESI = Logical DOS Drv. Desc. Tbl. address

4489 <1> ; EAX = Disk address

4490 <1> ; EBX = Sector Buffer

4491 <1> ; ECX = sectors per cluster

4492 0000B9F1 E8D33D0000 <1> call disk\_write

4493 0000B9F6 7211 <1> jc short loc\_createfile\_dsk\_wr\_err

4494 <1>

4495 <1> loc\_createfile\_update\_fat\_cluster:

4496 <1> ; 21/03/2016

4497 0000B9F8 803D[6A650100]00 <1> cmp byte [createfile\_wfc], 0

4498 0000B9FF 7712 <1> ja short loc\_createfile\_update\_fat\_cluster\_n1

4499 <1>

4500 0000BA01 FE05[6A650100] <1> inc byte [createfile\_wfc] ; 1

4501 0000BA07 EB24 <1> jmp short loc\_createfile\_update\_fat\_cluster\_n2

4502 <1>

4503 <1> loc\_createfile\_dsk\_wr\_err:

4504 <1> ; 16/10/2016 (1Dh -> 18)

4505 <1> ; 23/03/2016

4506 0000BA09 B812000000 <1> mov eax, 18 ; Drive not ready or write error !

4507 0000BA0E E9BD000000 <1> jmp loc\_createfile\_stc\_retn

4508 <1>

4509 <1> loc\_createfile\_update\_fat\_cluster\_n1:

4510 0000BA13 A1[5C650100] <1> mov eax, [createfile\_PCluster]

4511 0000BA18 8B0D[58650100] <1> mov ecx, [createfile\_Cluster]

4512 0000BA1E E84F040000 <1> call update\_cluster

4513 0000BA23 7308 <1> jnc short loc\_createfile\_update\_fat\_cluster\_n2

4514 0000BA25 09C0 <1> or eax, eax ; EAX = 0 -> cluster value is 0 or eocc

4515 0000BA27 0F85A3000000 <1> jnz loc\_createfile\_stc\_retn

4516 <1>

4517 <1> loc\_createfile\_update\_fat\_cluster\_n2:

4518 0000BA2D A1[58650100] <1> mov eax, [createfile\_Cluster]

4519 0000BA32 B9FFFFFF0F <1> mov ecx, 0FFFFFFFh

4520 0000BA37 E836040000 <1> call update\_cluster

4521 0000BA3C 734E <1> jnc short loc\_createfile\_save\_fat\_buffer\_3

4522 0000BA3E 09C0 <1> or eax, eax ; EAX = 0 -> cluster value is 0 or eocc

4523 0000BA40 744A <1> jz short loc\_createfile\_save\_fat\_buffer\_3

4524 <1>

4525 <1> loc\_createfile\_upd\_fat\_fcluster\_stc\_retn:

4526 0000BA42 E989000000 <1> jmp loc\_createfile\_stc\_retn

4527 <1>

4528 <1> loc\_createfile\_get\_set\_wfc\_cont:

4529 <1> ;movzx ecx, word [esi+LD\_BPB+BytesPerSec] ; 512

4530 0000BA47 0FB70D[68650100] <1> movzx ecx, word [createfile\_BytesPerSec] ; 512

4531 0000BA4E 01C8 <1> add eax, ecx

4532 0000BA50 48 <1> dec eax ; add eax, 511

4533 0000BA51 29D2 <1> sub edx, edx

4534 0000BA53 F7F1 <1> div ecx

4535 0000BA55 0FB61D[61650100] <1> movzx ebx, byte [createfile\_SecPerClust]

4536 0000BA5C 01D8 <1> add eax, ebx

4537 0000BA5E 48 <1> dec eax ; add eax, SecPerClust - 1

4538 0000BA5F 6631D2 <1> xor dx, dx

4539 0000BA62 F7F3 <1> div ebx

4540 <1>

4541 <1> loc\_createfile\_set\_cluster\_count:

4542 0000BA64 A3[64650100] <1> mov [createfile\_CCount], eax

4543 <1>

4544 0000BA69 BF00000700 <1> mov edi, Cluster\_Buffer

4545 0000BA6E 89C8 <1> mov eax, ecx ; Bytes per Sector

4546 0000BA70 F7E3 <1> mul ebx ; Sectors per Cluster

4547 <1> ; EAX = Bytes per Cluster

4548 0000BA72 89C1 <1> mov ecx, eax

4549 0000BA74 C1E902 <1> shr ecx, 2 ; dword count

4550 0000BA77 31C0 <1> xor eax, eax

4551 0000BA79 F3AB <1> rep stosd ; clear cluster buffer

4552 <1>

4553 0000BA7B A1[58650100] <1> mov eax, [createfile\_Cluster] ; 24/03/2016

4554 <1>

4555 0000BA80 89D9 <1> mov ecx, ebx

4556 <1>

4557 <1> loc\_createfile\_get\_set\_wf\_fclust\_cont:

4558 0000BA82 83E802 <1> sub eax, 2

4559 0000BA85 F7E1 <1> mul ecx

4560 <1> ; EAX = Logical DOS disk address (offset)

4561 0000BA87 E95DFFFFFF <1> jmp loc\_createfile\_write\_fcluster\_to\_disk

4562 <1>

4563 <1> loc\_createfile\_save\_fat\_buffer\_3:

4564 <1> ; byte [FAT\_BuffValidData] = 2

4565 0000BA8C E89E060000 <1> call save\_fat\_buffer

4566 0000BA91 723D <1> jc loc\_createfile\_stc\_retn

4567 <1>

4568 <1> ; 21/03/2016

4569 0000BA93 803D[1E610100]01 <1> cmp byte [FAT\_ClusterCounter], 1

4570 0000BA9A 721B <1> jb short loc\_createfile\_save\_fat\_buffer\_4

4571 <1>

4572 <1> ; ESI = Logical DOS Drive Description Table address

4573 0000BA9C A1[1E610100] <1> mov eax, [FAT\_ClusterCounter]

4574 0000BAA1 66BB01FF <1> mov bx, 0FF01h ; add free clusters

4575 0000BAA5 E81A070000 <1> call calculate\_fat\_freespace

4576 <1>

4577 <1> ;inc eax ; 0FFFFFFFFh -> 0 ; recalculation is needed!

4578 <1> ;jnz short loc\_createfile\_save\_fat\_buffer\_4

4579 <1>

4580 <1> ; ecx > 0 -> Recalculation is needed

4581 0000BAAA 09C9 <1> or ecx, ecx

4582 0000BAAC 7409 <1> jz short loc\_createfile\_save\_fat\_buffer\_4

4583 <1>

4584 0000BAAE 66BB00FF <1> mov bx, 0FF00h ; ; recalculate free space

4585 0000BAB2 E80D070000 <1> call calculate\_fat\_freespace

4586 <1>

4587 <1> loc\_createfile\_save\_fat\_buffer\_4:

4588 0000BAB7 FF0D[64650100] <1> dec dword [createfile\_CCount]

4589 <1> ;jz short loc\_createfile\_upd\_dir\_modif\_date\_time

4590 0000BABD 743F <1> jz short loc\_createfile\_stc\_retn\_cc ; 31/03/2016

4591 <1>

4592 <1> loc\_createfile\_get\_set\_write\_next\_cluster:

4593 0000BABF E8DB020000 <1> call get\_first\_free\_cluster

4594 0000BAC4 720A <1> jc short loc\_createfile\_stc\_retn

4595 <1>

4596 <1> loc\_createfile\_get\_set\_write\_next\_cluster\_1:

4597 0000BAC6 83F8FF <1> cmp eax, 0FFFFFFFFh

4598 0000BAC9 7213 <1> jb short loc\_createfile\_get\_set\_write\_next\_cluster\_2

4599 <1>

4600 <1> loc\_createfile\_wnc\_insufficient\_disk\_space:

4601 0000BACB B827000000 <1> mov eax, 27h ; Insufficient disk space

4602 <1>

4603 <1> loc\_createfile\_stc\_retn:

4604 0000BAD0 803D[6A650100]01 <1> cmp byte [createfile\_wfc], 1

4605 0000BAD7 7324 <1> jnb short loc\_createfile\_err\_retn

4606 0000BAD9 C3 <1> retn

4607 <1>

4608 <1> loc\_createfile\_wnc\_inv\_format\_retn:

4609 <1> ;mov eax, 28

4610 0000BADA B01C <1> mov al, 28 ; Invalid format

4611 0000BADC EBF2 <1> jmp short loc\_createfile\_stc\_retn

4612 <1>

4613 <1> loc\_createfile\_get\_set\_write\_next\_cluster\_2:

4614 0000BADE 83F802 <1> cmp eax, 2

4615 0000BAE1 72F7 <1> jb short loc\_createfile\_wnc\_inv\_format\_retn

4616 <1>

4617 <1> loc\_createfile\_get\_set\_write\_next\_cluster\_3:

4618 0000BAE3 8B0D[58650100] <1> mov ecx, [createfile\_Cluster]

4619 0000BAE9 A3[58650100] <1> mov [createfile\_Cluster], eax

4620 0000BAEE 890D[5C650100] <1> mov [createfile\_PCluster], ecx

4621 0000BAF4 0FB60D[61650100] <1> movzx ecx, byte [createfile\_SecPerClust]

4622 0000BAFB EB85 <1> jmp short loc\_createfile\_get\_set\_wf\_fclust\_cont

4623 <1>

4624 <1> loc\_createfile\_err\_retn:

4625 0000BAFD F9 <1> stc

4626 <1>

4627 <1> ;loc\_createfile\_upd\_dir\_modif\_date\_time:

4628 <1> loc\_createfile\_stc\_retn\_cc: ; 31/03/2016

4629 0000BAFE 9C <1> pushf ; cpu is here for an error return or completion

4630 0000BAFF 50 <1> push eax ; error code if cf = 1

4631 <1>

4632 <1> ;call update\_parent\_dir\_lmdt

4633 <1>

4634 <1> ;loc\_createfile\_stc\_retn\_cc:

4635 0000BB00 A1[1E610100] <1> mov eax, [FAT\_ClusterCounter]

4636 0000BB05 09C0 <1> or eax, eax

4637 0000BB07 741A <1> jz short loc\_createfile\_stc\_retn\_pop\_eax

4638 0000BB09 8A3D[FE580100] <1> mov bh, [Current\_Drv]

4639 0000BB0F B301 <1> mov bl, 01h ; BL = 1 -> add clusters

4640 <1> ; NOTE: EAX value will be added to Free Cluster Count

4641 <1> ; (If EAX value is negative, Free Cluster Count will be decreased)

4642 0000BB11 E8AE060000 <1> call calculate\_fat\_freespace

4643 <1> ; ESI = Logical DOS Drive Description Table Address

4644 <1> ;jc short loc\_createfile\_stc\_retn\_pop\_eax\_cf

4645 0000BB16 21C9 <1> and ecx, ecx ; cx = 0 -> valid free sector count

4646 0000BB18 7409 <1> jz short loc\_createfile\_stc\_retn\_pop\_eax

4647 <1>

4648 <1> loc\_createfile\_stc\_retn\_recalc\_FAT\_freespace:

4649 0000BB1A 66BB00FF <1> mov bx, 0FF00h ; bh = 0FFh ->

4650 <1> ; ESI = Logical DOS Drv DT Addr

4651 <1> ; BL = 0 -> Recalculate

4652 0000BB1E E8A1060000 <1> call calculate\_fat\_freespace

4653 <1>

4654 <1> loc\_createfile\_stc\_retn\_pop\_eax:

4655 0000BB23 58 <1> pop eax

4656 0000BB24 9D <1> popf

4657 0000BB25 7218 <1> jc short loc\_createfile\_retn

4658 <1>

4659 <1> loc\_createfile\_retn\_fcluster:

4660 0000BB27 A1[50650100] <1> mov eax, [createfile\_FFCluster]

4661 0000BB2C BB[4C650100] <1> mov ebx, createfile\_size

4662 <1> ;movzx ecx, byte [esi+LD\_BPB+SecPerClust]

4663 0000BB31 0FB60D[61650100] <1> movzx ecx, byte [createfile\_SecPerClust] ; 23/03/2016

4664 0000BB38 0FB715[62650100] <1> movzx edx, word [createfile\_DirIndex]

4665 <1>

4666 <1> loc\_createfile\_retn:

4667 0000BB3F C3 <1> retn

4668 <1>

4669 <1> create\_fs\_file:

4670 <1> ; temporary (21/03/2016)

4671 0000BB40 C3 <1> retn

4672 <1>

4673 <1> delete\_fs\_file:

4674 <1> ; temporary (28/02/2016)

4675 0000BB41 C3 <1> retn

4676 <1>

4677 <1> rename\_fs\_file\_or\_directory:

4678 0000BB42 C3 <1> retn

4679 <1>

4680 <1> make\_fs\_directory:

4681 <1> ; temporary (21/02/2016)

4682 0000BB43 C3 <1> retn

4683 <1>

4684 <1> add\_new\_fs\_section:

4685 <1> ; temporary (11/03/2016)

4686 0000BB44 C3 <1> retn

4687 <1>

4688 <1> delete\_fs\_directory\_entry:

4689 <1> ; temporary (11/03/2016)

4690 0000BB45 C3 <1> retn

4691 <1>

4692 <1> csftdf2\_read\_fs\_file\_sectors:

4693 <1> ; temporary (19/03/2016)

4694 0000BB46 C3 <1> retn

4695 <1>

4696 <1> csftdf2\_write\_fs\_file\_sectors:

4697 <1> ; temporary (19/03/2016)

4698 0000BB47 C3 <1> retn

2309 %include 'trdosk5.s' ; 24/01/2016

1 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2 <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.0) - File System Procedures : trdosk5s

3 <1> ; ----------------------------------------------------------------------------

4 <1> ; Last Update: 23/10/2016

5 <1> ; ----------------------------------------------------------------------------

6 <1> ; Beginning: 24/01/2016

7 <1> ; ----------------------------------------------------------------------------

8 <1> ; Assembler: NASM version 2.11 (trdos386.s)

9 <1> ; ----------------------------------------------------------------------------

10 <1> ; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan

11 <1> ; DRV\_FAT.ASM (21/08/2011)

12 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

13 <1> ; DRV\_FAT.ASM (c) 2005-2011 Erdogan TAN [ 07/07/2009 ] Last Update: 21/08/2011

14 <1>

15 <1> get\_next\_cluster:

16 <1> ; 15/10/2016

17 <1> ; 23/03/2016

18 <1> ; 01/02/2016 (TRDOS 386 = TRDOS v2.0)

19 <1> ; 05/07/2011

20 <1> ; 07/07/2009

21 <1> ; 2005

22 <1> ; INPUT ->

23 <1> ; EAX = Cluster Number (32 bit)

24 <1> ; ESI = Logical DOS Drive Parameters Table

25 <1> ; OUTPUT ->

26 <1> ; cf = 0 -> No Error, EAX valid

27 <1> ; cf = 1 & EAX = 0 -> End Of Cluster Chain

28 <1> ; cf = 1 & EAX > 0 -> Error

29 <1> ; ECX = Current/Previous cluster (if CF = 0)

30 <1> ; EAX = Next Cluster Number (32 bit)

31 <1> ;

32 <1> ; (Modified registers: EAX, ECX, EBX, EDX)

33 <1>

34 0000BB48 A3[12610100] <1> mov [FAT\_CurrentCluster], eax

35 <1> check\_next\_cluster\_fat\_type:

36 0000BB4D 29D2 <1> sub edx, edx ; 0

37 0000BB4F 807E0302 <1> cmp byte [esi+LD\_FATType], 2

38 0000BB53 7250 <1> jb short get\_FAT12\_next\_cluster

39 0000BB55 0F87AF000000 <1> ja get\_FAT32\_next\_cluster

40 <1> get\_FAT16\_next\_cluster:

41 0000BB5B BB00030000 <1> mov ebx, 300h ;768

42 0000BB60 F7F3 <1> div ebx

43 <1> ; EAX = Count of 3 FAT sectors

44 <1> ; EDX = Cluster Offset (< 768)

45 0000BB62 66D1E2 <1> shl dx, 1 ; Multiply by 2

46 0000BB65 89D3 <1> mov ebx, edx ; Byte Offset

47 0000BB67 81C3001C0900 <1> add ebx, FAT\_Buffer

48 0000BB6D 66BA0300 <1> mov dx, 3

49 0000BB71 F7E2 <1> mul edx

50 <1> ; EAX = FAT Sector (<= 256)

51 <1> ; EDX = 0

52 0000BB73 8A0E <1> mov cl, [esi+LD\_Name]

53 0000BB75 803D[16610100]00 <1> cmp byte [FAT\_BuffValidData], 0

54 0000BB7C 0F86CC000000 <1> jna load\_FAT\_sectors0

55 0000BB82 3A0D[17610100] <1> cmp cl, [FAT\_BuffDrvName]

56 0000BB88 0F85C0000000 <1> jne load\_FAT\_sectors0

57 0000BB8E 3B05[1A610100] <1> cmp eax, [FAT\_BuffSector]

58 0000BB94 0F85BA000000 <1> jne load\_FAT\_sectors1

59 <1> ;movzx eax, word [ebx]

60 0000BB9A 668B03 <1> mov ax, [ebx]

61 <1> ; 01/02/2016

62 <1> ; DRV\_FAT.ASM (21/08/2011) had a FATal bug here !

63 <1> ; (cmp ah, 0Fh) ! (ax >= FF7h)

64 <1> ; (how can i do a such mistake!?)

65 <1> ;cmp al, 0F7h

66 <1> ;jb short loc\_pass\_gnc\_FAT16\_eoc\_check

67 <1> ;cmp ah, 0FFh

68 <1> ;jb short loc\_pass\_gnc\_FAT16\_eoc\_check

69 0000BB9D 6683F8F7 <1> cmp ax, 0FFF7h

70 0000BBA1 725A <1> jb short loc\_pass\_gnc\_FAT16\_eoc\_check

71 <1> ; ax >= FFF7h (cluster 0002h to FFF6h is valid, in use)

72 0000BBA3 EB56 <1> jmp short loc\_pass\_gnc\_FAT16\_eoc\_check\_xor\_eax

73 <1>

74 <1> get\_FAT12\_next\_cluster:

75 0000BBA5 BB00040000 <1> mov ebx, 400h ;1024

76 0000BBAA F7F3 <1> div ebx

77 <1> ; EAX = Count of 3 FAT sectors

78 <1> ; EDX = Cluster Offset (< 1024)

79 0000BBAC 6650 <1> push ax

80 0000BBAE 66B80300 <1> mov ax, 3

81 0000BBB2 66F7E2 <1> mul dx ; Multiply by 3

82 0000BBB5 66D1E8 <1> shr ax, 1 ; Divide by 2

83 0000BBB8 6689C3 <1> mov bx, ax ; Byte Offset

84 0000BBBB 81C3001C0900 <1> add ebx, FAT\_Buffer

85 0000BBC1 6658 <1> pop ax

86 0000BBC3 66BA0300 <1> mov dx, 3

87 0000BBC7 F7E2 <1> mul edx

88 <1> ; EAX = FAT Sector (<= 12)

89 <1> ; EDX = 0

90 0000BBC9 8A0E <1> mov cl, [esi+LD\_Name]

91 0000BBCB 803D[16610100]00 <1> cmp byte [FAT\_BuffValidData], 0

92 0000BBD2 767A <1> jna short load\_FAT\_sectors0

93 0000BBD4 3A0D[17610100] <1> cmp cl, [FAT\_BuffDrvName]

94 0000BBDA 7572 <1> jne short load\_FAT\_sectors0

95 0000BBDC 3B05[1A610100] <1> cmp eax, [FAT\_BuffSector]

96 0000BBE2 7570 <1> jne short load\_FAT\_sectors1

97 0000BBE4 A1[12610100] <1> mov eax, [FAT\_CurrentCluster]

98 0000BBE9 66D1E8 <1> shr ax, 1

99 <1> ;movzx eax, word [ebx]

100 0000BBEC 668B03 <1> mov ax, [ebx]

101 0000BBEF 7314 <1> jnc short get\_FAT12\_nc\_even

102 0000BBF1 66C1E804 <1> shr ax, 4

103 <1> loc\_gnc\_fat12\_eoc\_check:

104 <1> ;cmp al, 0F7h

105 <1> ;jb short loc\_pass\_gnc\_FAT16\_eoc\_check

106 <1> ;cmp ah, 0Fh

107 <1> ;jb short loc\_pass\_gnc\_FAT16\_eoc\_check

108 0000BBF5 663DF70F <1> cmp ax, 0FF7h

109 0000BBF9 7202 <1> jb short loc\_pass\_gnc\_FAT16\_eoc\_check

110 <1> ; ax >= FF7h (cluster 0002h to FF6h is valid, in use)

111 <1>

112 <1> loc\_pass\_gnc\_FAT16\_eoc\_check\_xor\_eax:

113 0000BBFB 31C0 <1> xor eax, eax ; 0

114 <1> loc\_pass\_gnc\_FAT16\_eoc\_check:

115 <1> loc\_pass\_gnc\_FAT32\_eoc\_check:

116 0000BBFD 8B0D[12610100] <1> mov ecx, [FAT\_CurrentCluster]

117 0000BC03 F5 <1> cmc

118 0000BC04 C3 <1> retn

119 <1>

120 <1> get\_FAT12\_nc\_even:

121 0000BC05 80E40F <1> and ah, 0Fh

122 0000BC08 EBEB <1> jmp short loc\_gnc\_fat12\_eoc\_check

123 <1>

124 <1> get\_FAT32\_next\_cluster:

125 0000BC0A BB80010000 <1> mov ebx, 180h ;384

126 0000BC0F F7F3 <1> div ebx

127 <1> ; EAX = Count of 3 FAT sectors

128 <1> ; EDX = Cluster Offset (< 384)

129 0000BC11 66C1E202 <1> shl dx, 2 ; Multiply by 4

130 0000BC15 89D3 <1> mov ebx, edx ; Byte Offset

131 0000BC17 81C3001C0900 <1> add ebx, FAT\_Buffer

132 0000BC1D 66BA0300 <1> mov dx, 3

133 0000BC21 F7E2 <1> mul edx

134 <1> ; EAX = FAT Sector (<= 2097152) ; (FFFFFF7h \* 4) / 512

135 <1> ; for 32KB cluster size:

136 <1> ; EAX <= 1024 = (4GB / 32KB) \* 4) / 512

137 <1> ; EDX = 0

138 0000BC23 8A0E <1> mov cl, [esi+LD\_Name]

139 0000BC25 803D[16610100]00 <1> cmp byte [FAT\_BuffValidData], 0

140 0000BC2C 7620 <1> jna short load\_FAT\_sectors0

141 0000BC2E 3A0D[17610100] <1> cmp cl, [FAT\_BuffDrvName]

142 0000BC34 7518 <1> jne short load\_FAT\_sectors0

143 0000BC36 3B05[1A610100] <1> cmp eax, [FAT\_BuffSector] ; 0, 3, 6, 9 ...

144 0000BC3C 7516 <1> jne short load\_FAT\_sectors1

145 0000BC3E 8B03 <1> mov eax, [ebx]

146 0000BC40 25FFFFFF0F <1> and eax, 0FFFFFFFh ; 28 bit Cluster

147 0000BC45 3DF7FFFF0F <1> cmp eax, 0FFFFFF7h

148 0000BC4A 72B1 <1> jb short loc\_pass\_gnc\_FAT32\_eoc\_check

149 <1> ; eax >= FFFFFF7h (cluster 0002h to FFFFFF6h is valid)

150 0000BC4C EBAD <1> jmp short loc\_pass\_gnc\_FAT16\_eoc\_check\_xor\_eax

151 <1>

152 <1> load\_FAT\_sectors0:

153 0000BC4E 880D[17610100] <1> mov [FAT\_BuffDrvName], cl

154 <1> load\_FAT\_sectors1:

155 0000BC54 A3[1A610100] <1> mov [FAT\_BuffSector], eax

156 0000BC59 89C3 <1> mov ebx, eax

157 0000BC5B 034660 <1> add eax, [esi+LD\_FATBegin]

158 0000BC5E 807E0302 <1> cmp byte [esi+LD\_FATType], 2

159 0000BC62 7706 <1> ja short load\_FAT\_sectors3

160 0000BC64 0FB74E1C <1> movzx ecx, word [esi+LD\_BPB+BPB\_FATSz16]

161 0000BC68 EB03 <1> jmp short load\_FAT\_sectors4

162 <1> load\_FAT\_sectors3:

163 0000BC6A 8B4E2A <1> mov ecx, [esi+LD\_BPB+BPB\_FATSz32]

164 <1> load\_FAT\_sectors4:

165 0000BC6D 29D9 <1> sub ecx, ebx ; [FAT\_BuffSector]

166 0000BC6F 83F903 <1> cmp ecx, 3

167 0000BC72 7605 <1> jna short load\_FAT\_sectors5

168 0000BC74 B903000000 <1> mov ecx, 3

169 <1> load\_FAT\_sectors5:

170 0000BC79 BB001C0900 <1> mov ebx, FAT\_Buffer

171 0000BC7E E8553B0000 <1> call disk\_read

172 0000BC83 730D <1> jnc short load\_FAT\_sectors\_ok

173 <1> ; 15/10/2016 (15h -> 17)

174 <1> ; 23/03/2016 (15h)

175 0000BC85 B811000000 <1> mov eax, 17 ; Drive not ready or read error

176 0000BC8A C605[16610100]00 <1> mov byte [FAT\_BuffValidData], 0

177 0000BC91 C3 <1> retn

178 <1> load\_FAT\_sectors\_ok:

179 0000BC92 C605[16610100]01 <1> mov byte [FAT\_BuffValidData], 1

180 0000BC99 A1[12610100] <1> mov eax, [FAT\_CurrentCluster]

181 0000BC9E E9AAFEFFFF <1> jmp check\_next\_cluster\_fat\_type

182 <1>

183 <1> load\_FAT\_root\_directory:

184 <1> ; 23/10/2016

185 <1> ; 15/10/2016

186 <1> ; 07/02/2016

187 <1> ; 02/02/2016

188 <1> ; 01/02/2016 (TRDOS 386 = TRDOS v2.0)

189 <1> ; 21/05/2011

190 <1> ; 22/08/2009

191 <1> ;

192 <1> ; INPUT ->

193 <1> ; ESI = Logical DOS Drive Description Table

194 <1> ; OUTPUT ->

195 <1> ; cf = 1 -> Root directory could not be loaded

196 <1> ; EAX > 0 -> Error number

197 <1> ; cf = 0 -> EAX = 0

198 <1> ; ECX = Directory buffer size in sectors (CL)

199 <1> ; EBX = Directory buffer address

200 <1> ; NOTE: DirBuffer\_Size is in bytes ! (word)

201 <1> ;

202 <1> ; (Modified registers: EAX, ECX, EBX, EDX)

203 <1>

204 <1> ; NOTE: Only for FAT12 and FAT16 file systems !

205 <1> ; (FAT32 fs root dir must be loaded as sub directory)

206 <1>

207 0000BCA3 8A1E <1> mov bl, [esi+LD\_Name]

208 0000BCA5 8A7E03 <1> mov bh, [esi+LD\_FATType]

209 <1>

210 <1> ;mov [DirBuff\_DRV], bl

211 <1> ;mov [DirBuff\_FATType], bh

212 0000BCA8 66891D[26610100] <1> mov [DirBuff\_DRV], bx

213 <1>

214 <1> ;cmp bh, 2

215 <1> ;ja short load\_FAT32\_root\_dir0 ; FAT32 root dir

216 <1>

217 <1> load\_FAT\_root\_dir0: ; 23/10/2016

218 0000BCAF 0FB75617 <1> movzx edx, word [esi+LD\_BPB+RootDirEnts]

219 <1>

220 <1> ;or dx, dx ; 0 for FAT32 file systems

221 <1> ;jz short load\_FAT32\_root\_dir0 ; FAT32 root dir

222 <1>

223 0000BCB3 6681FA0002 <1> cmp dx, 512 ; Number of Root Dir Entries

224 0000BCB8 7414 <1> je short lrd\_mov\_ecx\_32

225 0000BCBA 89D0 <1> mov eax, edx

226 <1> ; 23/10/2016

227 0000BCBC 89C1 <1> mov ecx, eax

228 0000BCBE 6683C10F <1> add cx, 15 ; round up

229 0000BCC2 66C1E904 <1> shr cx, 4 ; 16 entries per sector (512/32)

230 <1> ; ecx = Root directory size in sectors

231 0000BCC6 66C1E005 <1> shl ax, 5 ; Root directory size in bytes

232 0000BCCA 664A <1> dec dx ; Last entry number of root dir

233 <1> ; cx = Dir Buffer sector count

234 0000BCCC EB0B <1> jmp short lrd\_check\_dir\_buffer

235 <1>

236 <1> lrd\_mov\_ecx\_32:

237 0000BCCE B920000000 <1> mov ecx, 32

238 0000BCD3 664A <1> dec dx ; 511

239 0000BCD5 66B80040 <1> mov ax, 32\*512

240 <1>

241 <1> lrd\_check\_dir\_buffer:

242 0000BCD9 29DB <1> sub ebx, ebx ; 0

243 0000BCDB 881D[28610100] <1> mov [DirBuff\_ValidData], bl ; 0

244 0000BCE1 668915[2B610100] <1> mov [DirBuff\_LastEntry], dx

245 0000BCE8 891D[2D610100] <1> mov [DirBuff\_Cluster], ebx ; 0

246 0000BCEE 66A3[31610100] <1> mov [DirBuffer\_Size], ax

247 <1>

248 0000BCF4 8B4664 <1> mov eax, [esi+LD\_ROOTBegin]

249 <1> read\_directory:

250 0000BCF7 BB00000800 <1> mov ebx, Directory\_Buffer

251 0000BCFC 51 <1> push ecx ; Directory buffer sector count

252 0000BCFD 53 <1> push ebx

253 0000BCFE E8D53A0000 <1> call disk\_read

254 0000BD03 5B <1> pop ebx

255 0000BD04 720B <1> jc short load\_DirBuff\_error

256 <1>

257 <1> validate\_DirBuff\_and\_return:

258 0000BD06 59 <1> pop ecx ; Number of loaded sectors

259 0000BD07 C605[28610100]01 <1> mov byte [DirBuff\_ValidData], 1

260 0000BD0E 31C0 <1> xor eax, eax ; 0 = no error

261 0000BD10 C3 <1> retn

262 <1>

263 <1> load\_DirBuff\_error:

264 0000BD11 89C8 <1> mov eax, ecx ; remaining sectors

265 0000BD13 59 <1> pop ecx ; sector count

266 0000BD14 29C1 <1> sub ecx, eax ; Number of loaded sectors

267 <1> ; 15/10/2016 (15h -> 17)

268 0000BD16 B811000000 <1> mov eax, 17 ; DRV NOT READY OR READ ERROR !

269 0000BD1B F9 <1> stc

270 0000BD1C C3 <1> retn

271 <1>

272 <1> load\_FAT32\_root\_directory:

273 <1> ; 02/02/2016 (TRDOS 386 = TRDOS v2.0)

274 <1> ;

275 <1> ; INPUT ->

276 <1> ; ESI = Logical DOS Drive Description Table

277 <1> ; OUTPUT ->

278 <1> ; cf = 1 -> Root directory could not be loaded

279 <1> ; EAX > 0 -> Error number

280 <1> ; cf = 0 -> EAX = 0

281 <1> ; ECX = Directory buffer size in sectors (CL)

282 <1> ; EBX = Directory buffer address

283 <1> ; NOTE: DirBuffer\_Size is in bytes ! (word)

284 <1> ;

285 <1> ; (Modified registers: EAX, ECX, EBX, EDX)

286 <1>

287 <1>

288 0000BD1D 8A1E <1> mov bl, [esi+LD\_Name]

289 0000BD1F 8A7E03 <1> mov bh, [esi+LD\_FATType]

290 <1>

291 <1> ;mov [DirBuff\_DRV], bl

292 <1> ;mov [DirBuff\_FATType], bh

293 0000BD22 66891D[26610100] <1> mov [DirBuff\_DRV], bx

294 <1>

295 <1> load\_FAT32\_root\_dir0:

296 0000BD29 8B4632 <1> mov eax, [esi+LD\_BPB+FAT32\_RootFClust]

297 0000BD2C EB0C <1> jmp short load\_FAT\_sub\_dir0

298 <1>

299 <1> load\_FAT\_sub\_directory:

300 <1> ; 01/02/2016 (TRDOS 386 = TRDOS v2.0)

301 <1> ; 05/07/2011

302 <1> ; 23/08/2009

303 <1> ;

304 <1> ; INPUT ->

305 <1> ; ESI = Logical DOS Drive Description Table

306 <1> ; EAX = Cluster Number

307 <1> ; OUTPUT ->

308 <1> ; cf = 1 -> Sub directory could not be loaded

309 <1> ; EAX > 0 -> Error number

310 <1> ; cf = 0 -> EAX = 0

311 <1> ; ECX = Directory buffer size in sectors (CL)

312 <1> ; EBX = Directory buffer address

313 <1> ;

314 <1> ; NOTE: DirBuffer\_Size is in bytes ! (word)

315 <1> ;

316 <1> ; (Modified registers: EAX, ECX, EBX, EDX)

317 <1>

318 0000BD2E 8A1E <1> mov bl, [esi+LD\_Name]

319 0000BD30 8A7E03 <1> mov bh, [esi+LD\_FATType]

320 <1>

321 <1> ;mov [DirBuff\_DRV], bl

322 <1> ;mov [DirBuff\_FATType], bh

323 0000BD33 66891D[26610100] <1> mov [DirBuff\_DRV], bx

324 <1>

325 <1> load\_FAT\_sub\_dir0:

326 0000BD3A 0FB64E13 <1> movzx ecx, byte [esi+LD\_BPB+SecPerClust]

327 <1>

328 0000BD3E 882D[28610100] <1> mov [DirBuff\_ValidData], ch ; 0

329 0000BD44 A3[2D610100] <1> mov [DirBuff\_Cluster], eax

330 <1>

331 0000BD49 0FB74611 <1> movzx eax, word [esi+LD\_BPB+BytesPerSec]

332 0000BD4D F7E1 <1> mul ecx

333 0000BD4F C1E805 <1> shr eax, 5 ; directory entry count (dir size / 32)

334 0000BD52 6648 <1> dec ax ; last entry

335 0000BD54 66A3[2B610100] <1> mov [DirBuff\_LastEntry], ax

336 <1>

337 0000BD5A A1[2D610100] <1> mov eax, [DirBuff\_Cluster]

338 0000BD5F 83E802 <1> sub eax, 2

339 0000BD62 F7E1 <1> mul ecx

340 0000BD64 034668 <1> add eax, [esi+LD\_DATABegin]

341 <1> ; ecx = sector per cluster (dir buffer size = 32 sectors)

342 0000BD67 EB8E <1> jmp short read\_directory

343 <1>

344 <1> ; DRV\_FS.ASM

345 <1>

346 <1> load\_current\_FS\_directory:

347 0000BD69 C3 <1> retn

348 <1> load\_FS\_root\_directory:

349 0000BD6A C3 <1> retn

350 <1> load\_FS\_sub\_directory:

351 0000BD6B C3 <1> retn

352 <1>

353 <1> read\_cluster:

354 <1> ; 15/10/2016

355 <1> ; 18/03/2016

356 <1> ; 16/03/2016

357 <1> ; 17/02/2016

358 <1> ; 15/02/2016 (TRDOS 386 = TRDOS v2.0)

359 <1> ;

360 <1> ; INPUT ->

361 <1> ; EAX = Cluster Number (Sector index for SINGLIX FS)

362 <1> ; ESI = Logical DOS Drive Description Table address

363 <1> ; EBX = Cluster (File R/W) Buffer address (max. 64KB)

364 <1> ; Only for SINGLIX FS:

365 <1> ; EDX = File Number (The 1st FDT address)

366 <1> ; OUTPUT ->

367 <1> ; cf = 1 -> Cluster can not be loaded at the buffer

368 <1> ; EAX > 0 -> Error number

369 <1> ; cf = 0 -> Cluster has been loaded at the buffer

370 <1> ;

371 <1> ; (Modified registers: EAX, ECX, EBX, EDX)

372 <1>

373 0000BD6C 0FB64E13 <1> movzx ecx, byte [esi+LD\_BPB+BPB\_SecPerClust]

374 <1> ; CL = 1 = [esi+LD\_FS\_Reserved2] ; SectPerClust for Singlix FS

375 <1>

376 <1> read\_file\_sectors: ; 16/03/2016

377 0000BD70 807E0300 <1> cmp byte [esi+LD\_FATType], 0

378 0000BD74 761C <1> jna short read\_fs\_cluster

379 <1>

380 <1> read\_fat\_file\_sectors: ; 18/03/2016

381 0000BD76 83E802 <1> sub eax, 2 ; Beginning cluster number is always 2

382 0000BD79 0FB65613 <1> movzx edx, byte [esi+LD\_BPB+BPB\_SecPerClust] ; 18/03/2016

383 0000BD7D F7E2 <1> mul edx

384 0000BD7F 034668 <1> add eax, [esi+LD\_DATABegin] ; absolute address of the cluster

385 <1>

386 <1> ; EAX = Disk sector address

387 <1> ; ECX = Sector count

388 <1> ; EBX = Buffer address

389 <1> ; (EDX = 0)

390 <1> ; ESI = Logical DOS drive description table address

391 <1>

392 0000BD82 E8513A0000 <1> call disk\_read

393 0000BD87 7306 <1> jnc short rclust\_retn

394 <1>

395 <1> ; 15/10/2016 (15h -> 17)

396 0000BD89 B811000000 <1> mov eax, 17 ; Drive not ready or read error !

397 0000BD8E C3 <1> retn

398 <1>

399 <1> rclust\_retn:

400 0000BD8F 29C0 <1> sub eax, eax ; 0

401 0000BD91 C3 <1> retn

402 <1>

403 <1> read\_fs\_cluster:

404 <1> ; 15/02/2016 (TRDOS 386 = TRDOS v2.0)

405 <1> ; Singlix FS

406 <1>

407 <1> ; EAX = Cluster number is sector index number of the file (eax)

408 <1>

409 <1> ; EDX = File number is the first File Descriptor Table address

410 <1> ; of the file. (Absolute address of the FDT).

411 <1>

412 <1> ; eax = sector index (0 for the first sector)

413 <1> ; edx = FDT0 address

414 <1> ; 64 KB buffer = 128 sectors (limit)

415 0000BD92 B980000000 <1> mov ecx, 128 ; maximum count of sectors (before eof)

416 0000BD97 E801000000 <1> call read\_fs\_sectors

417 0000BD9C C3 <1> retn

418 <1>

419 <1> read\_fs\_sectors:

420 <1> ; 15/02/2016 (TRDOS 386 = TRDOS v2.0)

421 0000BD9D F9 <1> stc

422 0000BD9E C3 <1> retn

423 <1>

424 <1> get\_first\_free\_cluster:

425 <1> ; 02/03/2016

426 <1> ; 21/02/2016 (TRDOS 386 = TRDOS v2.0)

427 <1> ; 26/10/2010 (DRV\_FAT.ASM, 'proc\_get\_first\_free\_cluster')

428 <1> ; 10/07/2010

429 <1> ; INPUT ->

430 <1> ; ESI = Logical DOS Drive Description Table address

431 <1> ; OUTPUT ->

432 <1> ; cf = 1 -> Error code in AL (EAX)

433 <1> ; cf = 0 ->

434 <1> ; EAX = Cluster number

435 <1> ; If EAX = FFFFFFFFh -> no free space

436 <1> ; If the drive has FAT32 fs:

437 <1> ; EBX = FAT32 FSI sector buffer address (if > 0)

438 <1>

439 0000BD9F 8B4678 <1> mov eax, [esi+LD\_Clusters]

440 0000BDA2 40 <1> inc eax ; add eax, 1

441 0000BDA3 A3[B0630100] <1> mov [gffc\_last\_free\_cluster], eax

442 <1>

443 0000BDA8 31DB <1> xor ebx, ebx ; 0 ; 02/03/2016

444 <1>

445 0000BDAA 807E0302 <1> cmp byte [esi+LD\_FATType], 2

446 0000BDAE 760E <1> jna short loc\_gffc\_get\_first\_fat\_free\_cluster0

447 <1>

448 <1> loc\_gffc\_get\_first\_fat32\_free\_cluster:

449 <1> ; 02/03/2016

450 0000BDB0 E844060000 <1> call get\_fat32\_fsinfo\_sector\_parms

451 0000BDB5 7207 <1> jc short loc\_gffc\_get\_first\_fat\_free\_cluster0

452 <1>

453 <1> loc\_gffc\_check\_fsinfo\_parms:

454 <1> ;;mov ebx, DOSBootSectorBuff

455 <1> ;cmp dword [ebx], 41615252h

456 <1> ;jne short loc\_gffc\_fat32\_fsinfo\_err

457 <1> ;cmp dword [ebx+484], 61417272h

458 <1> ;jne short loc\_gffc\_fat32\_fsinfo\_err

459 <1> ;mov eax, [ebx+492] ; FSI\_Next\_Free

460 <1> ;EAX = First free cluster

461 <1> ;(from FAT32 FSInfo sector)

462 0000BDB7 89D0 <1> mov eax, edx ; FSI\_Next\_Free (First Free Cluster)

463 0000BDB9 83F8FF <1> cmp eax, 0FFFFFFFFh ; invalid (unknown) !

464 0000BDBC 7205 <1> jb short loc\_gffc\_get\_first\_fat\_free\_cluster1

465 <1>

466 <1> ; Start from the 1st cluster of the FAT(32) file system

467 <1> loc\_gffc\_get\_first\_fat\_free\_cluster0:

468 0000BDBE B802000000 <1> mov eax, 2

469 <1> ;xor edx, edx

470 <1>

471 <1> loc\_gffc\_get\_first\_fat\_free\_cluster1:

472 0000BDC3 53 <1> push ebx ; 02/03/2016

473 <1>

474 <1> loc\_gffc\_get\_first\_fat\_free\_cluster2:

475 0000BDC4 A3[AC630100] <1> mov [gffc\_first\_free\_cluster], eax

476 0000BDC9 A3[A8630100] <1> mov [gffc\_next\_free\_cluster], eax

477 <1>

478 <1> ; EBX = FAT32 FSINFO sector buffer address

479 <1> ; (EBX = 0, if the drive has not got FAT32 fs or

480 <1> ; FAT32 FSINFO sector buffer is invalid.)

481 <1>

482 <1> loc\_gffc\_get\_first\_fat\_free\_cluster3:

483 0000BDCE E875FDFFFF <1> call get\_next\_cluster

484 0000BDD3 7307 <1> jnc short loc\_gffc\_get\_first\_fat\_free\_cluster4

485 0000BDD5 09C0 <1> or eax, eax

486 0000BDD7 740B <1> jz short loc\_gffc\_first\_free\_fat\_cluster\_next

487 0000BDD9 5B <1> pop ebx ; 02/03/2016

488 0000BDDA F5 <1> cmc ; stc

489 0000BDDB C3 <1> retn

490 <1>

491 <1> loc\_gffc\_get\_first\_fat\_free\_cluster4:

492 0000BDDC 21C0 <1> and eax, eax ; next cluster value

493 0000BDDE 7504 <1> jnz short loc\_gffc\_first\_free\_fat\_cluster\_next

494 0000BDE0 89C8 <1> mov eax, ecx ; current (previous cluster) value

495 0000BDE2 EB22 <1> jmp short loc\_gffc\_check\_for\_set

496 <1>

497 <1> loc\_gffc\_first\_free\_fat\_cluster\_next:

498 0000BDE4 A1[A8630100] <1> mov eax, [gffc\_next\_free\_cluster]

499 0000BDE9 3B05[B0630100] <1> cmp eax, [gffc\_last\_free\_cluster]

500 0000BDEF 7308 <1> jnb short retn\_stc\_from\_get\_first\_free\_cluster

501 <1> pass\_gffc\_last\_cluster\_eax\_check:

502 0000BDF1 40 <1> inc eax ; add eax, 1

503 0000BDF2 A3[A8630100] <1> mov [gffc\_next\_free\_cluster], eax

504 0000BDF7 EBD5 <1> jmp short loc\_gffc\_get\_first\_fat\_free\_cluster3

505 <1>

506 <1> retn\_stc\_from\_get\_first\_free\_cluster:

507 0000BDF9 A1[AC630100] <1> mov eax, [gffc\_first\_free\_cluster]

508 0000BDFE 83F802 <1> cmp eax, 2

509 0000BE01 7709 <1> ja short loc\_gffc\_check\_previous\_clusters

510 0000BE03 29C0 <1> sub eax, eax

511 0000BE05 48 <1> dec eax ; FFFFFFFFh

512 <1>

513 <1> loc\_gffc\_check\_for\_set:

514 <1> ; 02/03/2016

515 0000BE06 5B <1> pop ebx

516 <1>

517 <1> ; EBX = FAT32 FSINFO sector buffer address

518 <1> ; (EBX = 0, if the drive has not got FAT32 fs or

519 <1> ; FAT32 FSINFO sector buffer is invalid.)

520 <1>

521 0000BE07 09DB <1> or ebx, ebx

522 0000BE09 750E <1> jnz short loc\_gffc\_set\_ffree\_fat32\_cluster

523 <1>

524 <1> ;cmp byte [esi+LD\_FATType], 3

525 <1> ;jnb short loc\_gffc\_set\_ffree\_fat32\_cluster

526 <1>

527 <1> ;xor ebx, ebx ; 0

528 <1>

529 <1> loc\_gffc\_retn:

530 0000BE0B C3 <1> retn

531 <1>

532 <1> loc\_gffc\_check\_previous\_clusters:

533 0000BE0C 48 <1> dec eax ; sub eax, 1

534 0000BE0D A3[B0630100] <1> mov [gffc\_last\_free\_cluster], eax

535 0000BE12 B802000000 <1> mov eax, 2

536 <1> ;xor edx, edx

537 0000BE17 EBAB <1> jmp short loc\_gffc\_get\_first\_fat\_free\_cluster2

538 <1>

539 <1> loc\_gffc\_set\_ffree\_fat32\_cluster:

540 <1> ;call set\_first\_free\_cluster

541 <1> ;retn

542 <1> ;jmp short set\_first\_free\_cluster

543 <1>

544 <1> set\_first\_free\_cluster:

545 <1> ; 15/10/2016

546 <1> ; 23/03/2016

547 <1> ; 02/03/2016

548 <1> ; 29/02/2016

549 <1> ; 26/02/2016

550 <1> ; 21/02/2016 (TRDOS 386 = TRDOS v2.0)

551 <1> ; 21/08/2011 (DRV\_FAT.ASM, 'proc\_set\_first\_free\_cluster')

552 <1> ; 11/07/2010

553 <1> ; INPUT ->

554 <1> ; ESI = Logical DOS Drive Description Table address

555 <1> ; EAX = First free cluster

556 <1> ; EBX = FSINFO sector buffer address

557 <1> ; ;;If EBX > 0, it is FSINFO sector buffer address

558 <1> ; ;;EBX = 0, if FSINFO sector is not loaded

559 <1> ; OUTPUT->

560 <1> ; ESI = Logical DOS Drive Description Table address

561 <1> ; If EBX > 0, it is FSINFO sector buffer address

562 <1> ; EBX = 0, if FSINFO sector could not be loaded

563 <1> ; CF = 1 -> Error code in AL (EAX)

564 <1> ; CF = 0 -> first free cluster is successfully updated

565 <1>

566 <1> ;cmp byte [esi+LD\_FATType], 3

567 <1> ;jb short loc\_sffc\_invalid\_drive

568 <1>

569 <1> ; Save First Free Cluster value for 'update\_cluster'

570 0000BE19 89463E <1> mov [esi+LD\_BPB+BPB\_Reserved+4], eax ; First free Cluster

571 <1>

572 <1> ;or ebx, ebx

573 <1> ;jnz short loc\_sffc\_read\_fsinfo\_sector

574 <1>

575 0000BE1C 813B52526141 <1> cmp dword [ebx], 41615252h

576 0000BE22 7540 <1> jne short loc\_sffc\_read\_fsinfo\_sector

577 0000BE24 81BBE4010000727241- <1> cmp dword [ebx+484], 61417272h

577 0000BE2D 61 <1>

578 0000BE2E 7534 <1> jne short loc\_sffc\_read\_fsinfo\_sector

579 <1>

580 0000BE30 3B83EC010000 <1> cmp eax, [ebx+492] ; FSI\_Next\_Free

581 0000BE36 741F <1> je short loc\_sffc\_retn

582 <1>

583 <1> loc\_sffc\_write\_fsinfo\_sector:

584 <1> ; EBX = FSINFO sector buffer

585 <1> ; [CFS\_FAT32FSINFOSEC] is set in 'get\_fat32\_fsinfo\_sector\_parms'

586 0000BE38 8983EC010000 <1> mov [ebx+492], eax

587 0000BE3E A1[C0630100] <1> mov eax, [CFS\_FAT32FSINFOSEC]

588 0000BE43 B901000000 <1> mov ecx, 1

589 0000BE48 53 <1> push ebx

590 0000BE49 E87B390000 <1> call disk\_write

591 0000BE4E 7208 <1> jc short loc\_sffc\_read\_fsinfo\_sector\_err1

592 0000BE50 5B <1> pop ebx

593 <1>

594 0000BE51 8B83EC010000 <1> mov eax, [ebx+492] ; First (Next) Free Cluster

595 <1>

596 <1> loc\_sffc\_retn:

597 0000BE57 C3 <1> retn

598 <1>

599 <1> ;loc\_sffc\_invalid\_drive:

600 <1> ; mov eax, 0Fh ; MSDOS Error : Invalid drive

601 <1> ; push edx

602 <1>

603 <1> loc\_sffc\_read\_fsinfo\_sector\_err1:

604 0000BE58 BB00000000 <1> mov ebx, 0

605 <1> ; 15/10/2016 (1Dh -> 18)

606 <1> ; 23/03/2016 (1Dh)

607 0000BE5D B812000000 <1> mov eax, 18 ; Drive not ready or write error

608 <1>

609 <1> loc\_sffc\_read\_fsinfo\_sector\_err2:

610 0000BE62 5A <1> pop edx

611 0000BE63 C3 <1> retn

612 <1>

613 <1> loc\_sffc\_read\_fsinfo\_sector:

614 0000BE64 50 <1> push eax

615 <1>

616 0000BE65 E88F050000 <1> call get\_fat32\_fsinfo\_sector\_parms

617 0000BE6A 72F6 <1> jc short loc\_sffc\_read\_fsinfo\_sector\_err2

618 <1>

619 0000BE6C 58 <1> pop eax

620 <1> ; EDX = First (Next) Free Cluster value from FSINFO sector

621 <1> ; EAX = First Free Cluster value from 'get\_next\_cluster'

622 <1> ; (edx = old value)

623 0000BE6D 39D0 <1> cmp eax, edx ; First free Cluster (eax = new value)

624 0000BE6F 75C7 <1> jne short loc\_sffc\_write\_fsinfo\_sector

625 <1>

626 0000BE71 C3 <1> retn

627 <1>

628 <1> update\_cluster:

629 <1> ; 23/10/2016

630 <1> ; 23/03/2016

631 <1> ; 02/03/2016

632 <1> ; 01/03/2016

633 <1> ; 29/02/2016

634 <1> ; 27/02/2016

635 <1> ; 26/02/2016

636 <1> ; 22/02/2016 (TRDOS 386 = TRDOS v2.0)

637 <1> ; 11/08/2011

638 <1> ; 09/02/2005

639 <1> ; INPUT ->

640 <1> ; EAX = Cluster Number

641 <1> ; ECX = New Cluster Value

642 <1> ; ESI = Logical Dos Drive Parameters Table

643 <1> ;

644 <1> ; /// dword [FAT\_ClusterCounter] ///

645 <1> ;

646 <1> ; OUTPUT ->

647 <1> ; cf = 0 -> No Error, EAX is valid

648 <1> ; cf = 1 & EAX = 0 -> End Of Cluster Chain

649 <1> ; cf = 1 & EAX > 0 -> Error

650 <1> ; (ECX -> any value)

651 <1> ; EAX = Next Cluster

652 <1> ; ECX = New Cluster Value

653 <1> ;

654 <1> ; /// [FAT\_ClusterCounter] is updated,

655 <1> ; /// decreased when a free cluster is assigned,

656 <1> ; /// increased if an assigned cluster is freed.

657 <1> ;

658 <1> ;

659 <1> ; (Modified registers: EAX, EBX, -ECX-, EDX)

660 <1>

661 0000BE72 A3[12610100] <1> mov [FAT\_CurrentCluster], eax

662 0000BE77 890D[B4630100] <1> mov [ClusterValue], ecx

663 <1>

664 <1> loc\_update\_cluster\_check\_fat\_buffer:

665 0000BE7D 8A1E <1> mov bl, [esi+LD\_Name]

666 0000BE7F 381D[17610100] <1> cmp [FAT\_BuffDrvName], bl

667 0000BE85 741A <1> je short loc\_update\_cluster\_check\_fat\_type

668 0000BE87 803D[16610100]02 <1> cmp byte [FAT\_BuffValidData], 2

669 0000BE8E 0F84C2000000 <1> je loc\_uc\_save\_fat\_buffer

670 <1>

671 <1> loc\_uc\_reset\_fat\_buffer\_validation:

672 0000BE94 C605[16610100]00 <1> mov byte [FAT\_BuffValidData], 0

673 <1>

674 <1> loc\_uc\_check\_fat\_type\_reset\_drvname:

675 0000BE9B 881D[17610100] <1> mov [FAT\_BuffDrvName], bl

676 <1>

677 <1> loc\_update\_cluster\_check\_fat\_type:

678 0000BEA1 29D2 <1> sub edx, edx ; 26/02/2016

679 0000BEA3 8A5E03 <1> mov bl, [esi+LD\_FATType]

680 0000BEA6 83F802 <1> cmp eax, 2

681 0000BEA9 0F82BE000000 <1> jb update\_cluster\_inv\_data

682 0000BEAF 80FB02 <1> cmp bl, 2

683 0000BEB2 0F877A010000 <1> ja update\_fat32\_cluster

684 <1> ;cmp bl, 1

685 <1> ;jb short update\_cluster\_inv\_data

686 0000BEB8 8B4E78 <1> mov ecx, [esi+LD\_Clusters]

687 0000BEBB 41 <1> inc ecx

688 0000BEBC 890D[22610100] <1> mov [LastCluster], ecx

689 0000BEC2 39C8 <1> cmp eax, ecx ; dword [LastCluster]

690 0000BEC4 0F87A6000000 <1> ja return\_uc\_fat\_stc

691 <1> ; TRDOS v1 has a FATal bug here !

692 <1> ; or bl, bl ; cmp bl, 0

693 <1> ; jz short update\_fat12\_cluster

694 <1> ; !! It would destroy FAT12 floppy disk fs here !!

695 <1> ; ('A:' disks of TRDOS v1 operating system project

696 <1> ; had 'singlix fs', so, I could not differ this mistake

697 <1> ; on a drive 'A:')

698 0000BECA 80FB01 <1> cmp bl, 1 ; correct comparison is this !

699 0000BECD 0F86A2000000 <1> jna update\_fat12\_cluster

700 <1>

701 <1> update\_fat16\_cluster:

702 <1> pass\_uc\_fat16\_errc:

703 <1> ;sub edx, edx

704 0000BED3 BB00030000 <1> mov ebx, 300h ;768

705 0000BED8 F7F3 <1> div ebx

706 <1> ; EAX = Count of 3 FAT sectors

707 <1> ; DX = Cluster offset in FAT buffer

708 0000BEDA 6689D3 <1> mov bx, dx

709 0000BEDD 66D1E3 <1> shl bx, 1 ; Multiply by 2

710 0000BEE0 66BA0300 <1> mov dx, 3

711 0000BEE4 F7E2 <1> mul edx

712 <1> ; EAX = FAT Sector

713 <1> ; EDX = 0

714 <1> ; EBX = Byte offset in FAT buffer

715 0000BEE6 8A0D[16610100] <1> mov cl, [FAT\_BuffValidData]

716 0000BEEC 80F902 <1> cmp cl, 2

717 0000BEEF 750A <1> jne short loc\_uc\_check\_fat16\_buff\_sector\_load

718 <1>

719 <1> loc\_uc\_check\_fat16\_buff\_sector\_save:

720 0000BEF1 3B05[1A610100] <1> cmp eax, [FAT\_BuffSector]

721 0000BEF7 755D <1> jne short loc\_uc\_save\_fat\_buffer

722 0000BEF9 EB15 <1> jmp short loc\_update\_fat16\_cell

723 <1>

724 <1> loc\_uc\_check\_fat16\_buff\_sector\_load:

725 0000BEFB 80F901 <1> cmp cl, 1 ; byte [FAT\_BuffValidData]

726 0000BEFE 0F85FB010000 <1> jne loc\_uc\_load\_fat\_sectors

727 0000BF04 3B05[1A610100] <1> cmp eax, [FAT\_BuffSector]

728 0000BF0A 0F85EF010000 <1> jne loc\_uc\_load\_fat\_sectors

729 <1>

730 <1> loc\_update\_fat16\_cell:

731 <1> loc\_update\_fat16\_buffer:

732 0000BF10 81C3001C0900 <1> add ebx, FAT\_Buffer ; 26/02/2016

733 <1> ;movzx eax, word [ebx]

734 0000BF16 668B03 <1> mov ax, [ebx]

735 <1> ; 01/03/2016

736 0000BF19 89C2 <1> mov edx, eax ; old value of the cluster

737 0000BF1B A3[12610100] <1> mov [FAT\_CurrentCluster], eax

738 0000BF20 8B0D[B4630100] <1> mov ecx, [ClusterValue] ; 32 bits

739 0000BF26 66890B <1> mov [ebx], cx ; 16 bits !

740 <1>

741 0000BF29 C605[16610100]02 <1> mov byte [FAT\_BuffValidData], 2

742 <1>

743 0000BF30 6683F802 <1> cmp ax, 2

744 0000BF34 723A <1> jb short return\_uc\_fat\_stc

745 0000BF36 3B05[22610100] <1> cmp eax, [LastCluster]

746 0000BF3C 7732 <1> ja short return\_uc\_fat\_stc

747 <1>

748 <1> loc\_fat\_buffer\_updated:

749 <1> ; 01/03/2016

750 0000BF3E F8 <1> clc

751 <1> loc\_fat\_buffer\_stc\_1:

752 0000BF3F 9C <1> pushf

753 0000BF40 21C9 <1> and ecx, ecx

754 0000BF42 7506 <1> jnz short loc\_fat\_buffer\_updated\_1

755 <1>

756 <1> ; 01/03/2016

757 <1> ; new value of the cluster = 0 (free)

758 <1> ; increase free(d) cluster count

759 0000BF44 FF05[1E610100] <1> inc dword [FAT\_ClusterCounter]

760 <1>

761 <1> loc\_fat\_buffer\_updated\_1: ; new value of the cluster > 0

762 0000BF4A 09D2 <1> or edx, edx ; 02/03/2016

763 0000BF4C 7506 <1> jnz short loc\_fat\_buffer\_updated\_2

764 <1> ; old value of the cluster = 0 (it was free cluster)

765 <1> ; decrease free(d) cluster count

766 0000BF4E FF0D[1E610100] <1> dec dword [FAT\_ClusterCounter] ; it may be negative number

767 <1>

768 <1> loc\_fat\_buffer\_updated\_2:

769 0000BF54 9D <1> popf

770 0000BF55 C3 <1> retn

771 <1>

772 <1> loc\_uc\_save\_fat\_buffer:

773 <1> ; byte [FAT\_BuffValidData] = 2

774 0000BF56 E8D4010000 <1> call save\_fat\_buffer

775 0000BF5B 0F8297010000 <1> jc loc\_fat\_sectors\_rw\_error2

776 <1> ;mov byte [FAT\_BuffValidData], 1

777 0000BF61 A1[12610100] <1> mov eax, [FAT\_CurrentCluster]

778 <1> ;mov ecx, [ClusterValue]

779 <1> ;jmp short loc\_update\_cluster\_check\_fat\_buffer

780 0000BF66 8A1E <1> mov bl, [esi+LD\_Name] ; 01/03/2016

781 0000BF68 E927FFFFFF <1> jmp loc\_uc\_reset\_fat\_buffer\_validation

782 <1>

783 <1> update\_cluster\_inv\_data:

784 <1> ;mov eax, 0Dh

785 0000BF6D B00D <1> mov al, 0Dh ; Invalid Data

786 0000BF6F C3 <1> retn

787 <1>

788 <1> return\_uc\_fat\_stc:

789 <1> ; 01/03/2016

790 0000BF70 31C0 <1> xor eax, eax

791 0000BF72 F9 <1> stc

792 0000BF73 EBCA <1> jmp short loc\_fat\_buffer\_stc\_1

793 <1>

794 <1> update\_fat12\_cluster:

795 <1> pass\_uc\_fat12\_errc:

796 <1> ;sub edx, edx

797 0000BF75 BB00040000 <1> mov ebx, 400h ;1024

798 0000BF7A F7F3 <1> div ebx

799 <1> ; EAX = Count of 3 FAT sectors

800 <1> ; DX = Cluster offset in FAT buffer

801 0000BF7C 66B90300 <1> mov cx, 3

802 0000BF80 6689C3 <1> mov bx, ax

803 0000BF83 6689C8 <1> mov ax, cx ; 3

804 0000BF86 66F7E2 <1> mul dx ; Multiply by 3

805 0000BF89 66D1E8 <1> shr ax, 1 ; Divide by 2

806 0000BF8C 6693 <1> xchg bx, ax

807 <1> ; EAX = Count of 3 FAT sectors

808 <1> ; EBX = Byte Offset in FAT buffer

809 0000BF8E 66F7E1 <1> mul cx ; 3 \* AX

810 <1> ; EAX = FAT Beginning Sector

811 <1> ; EDX = 0

812 0000BF91 8A0D[16610100] <1> mov cl, [FAT\_BuffValidData]

813 <1> ; TRDOS v1 has a FATal bug here !

814 <1> ; (it does not have 'cmp cl, 2' instruction here !

815 <1> ; while 'jne' is existing !)

816 0000BF97 80F902 <1> cmp cl, 2 ; 2 = dirty buffer (must be written to disk)

817 0000BF9A 750A <1> jne short loc\_uc\_check\_fat12\_buff\_sector\_load

818 <1>

819 <1> loc\_uc\_check\_fat12\_buff\_sector\_save:

820 0000BF9C 3B05[1A610100] <1> cmp eax, [FAT\_BuffSector]

821 0000BFA2 75B2 <1> jne short loc\_uc\_save\_fat\_buffer

822 0000BFA4 EB15 <1> jmp short loc\_update\_fat12\_cell

823 <1>

824 <1> loc\_uc\_check\_fat12\_buff\_sector\_load:

825 0000BFA6 80F901 <1> cmp cl, 1 ; byte ptr [FAT\_BuffValidData]

826 0000BFA9 0F8550010000 <1> jne loc\_uc\_load\_fat\_sectors

827 0000BFAF 3B05[1A610100] <1> cmp eax, [FAT\_BuffSector]

828 0000BFB5 0F8544010000 <1> jne loc\_uc\_load\_fat\_sectors

829 <1>

830 <1> loc\_update\_fat12\_cell:

831 0000BFBB 81C3001C0900 <1> add ebx, FAT\_Buffer ; 26/02/2016

832 0000BFC1 668B0D[12610100] <1> mov cx, [FAT\_CurrentCluster]

833 0000BFC8 66D1E9 <1> shr cx, 1

834 0000BFCB 668B03 <1> mov ax, [ebx]

835 0000BFCE 6689C2 <1> mov dx, ax

836 0000BFD1 7344 <1> jnc short uc\_fat12\_nc\_even

837 <1>

838 0000BFD3 6683E00F <1> and ax, 0Fh

839 0000BFD7 8B0D[B4630100] <1> mov ecx, [ClusterValue] ; 32 bits

840 0000BFDD 66C1E104 <1> shl cx, 4

841 0000BFE1 6609C1 <1> or cx, ax

842 0000BFE4 6689D0 <1> mov ax, dx

843 0000BFE7 66890B <1> mov [ebx], cx ; 16 bits !

844 0000BFEA 66C1E804 <1> shr ax, 4 ; al(bit4..7)+ah(bit0..7)

845 <1>

846 <1> update\_fat12\_buffer:

847 0000BFEE A3[12610100] <1> mov [FAT\_CurrentCluster], eax

848 0000BFF3 89C2 <1> mov edx, eax ; 01/03/2016

849 0000BFF5 C605[16610100]02 <1> mov byte [FAT\_BuffValidData], 2

850 0000BFFC 6683F802 <1> cmp ax, 2

851 0000C000 0F826AFFFFFF <1> jb return\_uc\_fat\_stc

852 0000C006 3B05[22610100] <1> cmp eax, [LastCluster]

853 0000C00C 0F875EFFFFFF <1> ja return\_uc\_fat\_stc

854 0000C012 E927FFFFFF <1> jmp loc\_fat\_buffer\_updated

855 <1>

856 <1> uc\_fat12\_nc\_even:

857 0000C017 662500F0 <1> and ax, 0F000h

858 0000C01B 8B0D[B4630100] <1> mov ecx, [ClusterValue] ; 32 bits

859 0000C021 80E50F <1> and ch, 0Fh

860 0000C024 6609C1 <1> or cx, ax

861 0000C027 6689D0 <1> mov ax, dx

862 0000C02A 66890B <1> mov [ebx], cx ; 16 bits !

863 0000C02D 80E40F <1> and ah, 0Fh ; al(bit0..7)+ah(bit0..3)

864 0000C030 EBBC <1> jmp short update\_fat12\_buffer

865 <1>

866 <1> update\_fat32\_cluster:

867 0000C032 8B4E78 <1> mov ecx, [esi+LD\_Clusters]

868 0000C035 41 <1> inc ecx

869 0000C036 890D[22610100] <1> mov [LastCluster], ecx

870 <1>

871 0000C03C 39C8 <1> cmp eax, ecx

872 0000C03E 0F872CFFFFFF <1> ja return\_uc\_fat\_stc

873 <1>

874 <1> pass\_uc\_fat32\_errc:

875 <1> ;sub edx, edx

876 0000C044 BB80010000 <1> mov ebx, 180h ;384

877 0000C049 F7F3 <1> div ebx

878 <1> ; EAX = Count of 3 FAT sectors

879 <1> ; DX = Cluster offset in FAT buffer

880 0000C04B 89D3 <1> mov ebx, edx

881 0000C04D C1E302 <1> shl ebx, 2 ; Multiply by 4

882 0000C050 BA03000000 <1> mov edx, 3

883 0000C055 F7E2 <1> mul edx

884 <1> ; EBX = Cluster Offset in FAT buffer

885 <1> ; EAX = FAT Sector

886 <1> ; EDX = 0

887 0000C057 8A0D[16610100] <1> mov cl, [FAT\_BuffValidData]

888 0000C05D 80F902 <1> cmp cl, 2

889 0000C060 750E <1> jne short loc\_uc\_check\_fat32\_buff\_sector\_load

890 <1>

891 <1> loc\_uc\_check\_fat32\_buff\_sector\_save:

892 0000C062 3B05[1A610100] <1> cmp eax, [FAT\_BuffSector]

893 0000C068 0F85E8FEFFFF <1> jne loc\_uc\_save\_fat\_buffer

894 0000C06E EB11 <1> jmp short loc\_update\_fat32\_cell

895 <1>

896 <1> loc\_uc\_check\_fat32\_buff\_sector\_load:

897 0000C070 80F901 <1> cmp cl, 1 ; byte [FAT\_BuffValidData]

898 0000C073 0F8586000000 <1> jne loc\_uc\_load\_fat\_sectors

899 0000C079 3B05[1A610100] <1> cmp eax, [FAT\_BuffSector]

900 0000C07F 757E <1> jne loc\_uc\_load\_fat\_sectors

901 <1>

902 <1> loc\_update\_fat32\_cell:

903 <1> loc\_update\_fat32\_buffer:

904 0000C081 81C3001C0900 <1> add ebx, FAT\_Buffer ; 26/02/2016

905 0000C087 8B03 <1> mov eax, [ebx]

906 0000C089 25FFFFFF0F <1> and eax, 0FFFFFFFh ; 28 bit cluster value

907 <1>

908 0000C08E 8B15[12610100] <1> mov edx, [FAT\_CurrentCluster] ; 01/03/2016

909 <1>

910 0000C094 A3[12610100] <1> mov [FAT\_CurrentCluster], eax

911 0000C099 8B0D[B4630100] <1> mov ecx, [ClusterValue]

912 0000C09F 890B <1> mov [ebx], ecx ; 29/02/2016

913 <1>

914 0000C0A1 C605[16610100]02 <1> mov byte [FAT\_BuffValidData], 2

915 <1>

916 <1> ; 01/03/2016

917 0000C0A8 21C0 <1> and eax, eax ; was it free cluster ?

918 0000C0AA 7514 <1> jnz short loc\_upd\_fat32\_c0

919 <1>

920 <1> ;or ecx, ecx ; it will be left free ?!

921 <1> ;jz short loc\_upd\_fat32\_c3

922 <1>

923 0000C0AC 3B563E <1> cmp edx, [esi+LD\_BPB+BPB\_Reserved+4] ; First free cluster

924 0000C0AF 7520 <1> jne short loc\_upd\_fat32\_c3

925 <1>

926 0000C0B1 3B15[22610100] <1> cmp edx, [LastCluster]

927 0000C0B7 7207 <1> jb short loc\_upd\_fat32\_c0

928 <1>

929 0000C0B9 BA02000000 <1> mov edx, 2 ; rewind !

930 0000C0BE EB0E <1> jmp short loc\_upd\_fat32\_c2

931 <1>

932 <1> loc\_upd\_fat32\_c0:

933 0000C0C0 FF463E <1> inc dword [esi+LD\_BPB+BPB\_Reserved+4] ; set it to next cluster

934 0000C0C3 EB0C <1> jmp short loc\_upd\_fat32\_c3

935 <1>

936 <1> loc\_upd\_fat32\_c1:

937 0000C0C5 09C9 <1> or ecx, ecx ; will it be free cluster ?

938 0000C0C7 7508 <1> jnz short loc\_upd\_fat32\_c3

939 <1>

940 0000C0C9 3B563E <1> cmp edx, [esi+LD\_BPB+BPB\_Reserved+4] ; First free cluster

941 0000C0CC 7303 <1> jnb short loc\_upd\_fat32\_c3

942 <1>

943 <1> loc\_upd\_fat32\_c2:

944 0000C0CE 89563E <1> mov [esi+LD\_BPB+BPB\_Reserved+4], edx

945 <1>

946 <1> loc\_upd\_fat32\_c3:

947 0000C0D1 89C2 <1> mov edx, eax

948 <1>

949 <1> loc\_upd\_fat32\_c4:

950 0000C0D3 83F802 <1> cmp eax, 2

951 0000C0D6 0F8294FEFFFF <1> jb return\_uc\_fat\_stc

952 <1>

953 <1> pass\_uc\_fat32\_c\_zero\_check\_2:

954 0000C0DC 3B05[22610100] <1> cmp eax, [LastCluster]

955 0000C0E2 0F8788FEFFFF <1> ja return\_uc\_fat\_stc

956 <1>

957 0000C0E8 E951FEFFFF <1> jmp loc\_fat\_buffer\_updated

958 <1>

959 <1> loc\_fat\_sectors\_rw\_error1:

960 <1> ;mov byte [FAT\_BuffValidData], 0

961 <1> ; 23/10/2016 (15h -> 17)

962 <1> ; 23/03/2016

963 0000C0ED B811000000 <1> mov eax, 17 ; Drive not ready or read error

964 0000C0F2 8825[16610100] <1> mov [FAT\_BuffValidData], ah ; 0

965 <1>

966 <1> loc\_fat\_sectors\_rw\_error2:

967 <1> ;mov eax, error code

968 <1> ;mov edx, 0

969 0000C0F8 8B0D[B4630100] <1> mov ecx, [ClusterValue]

970 0000C0FE C3 <1> retn

971 <1>

972 <1> loc\_uc\_load\_fat\_sectors:

973 0000C0FF A3[1A610100] <1> mov [FAT\_BuffSector], eax

974 <1>

975 <1> load\_uc\_fat\_sectors\_zero:

976 0000C104 034660 <1> add eax, [esi+LD\_FATBegin]

977 0000C107 BB001C0900 <1> mov ebx, FAT\_Buffer

978 0000C10C B903000000 <1> mov ecx, 3

979 0000C111 E8C2360000 <1> call disk\_read

980 0000C116 72D5 <1> jc short loc\_fat\_sectors\_rw\_error1

981 <1>

982 0000C118 C605[16610100]01 <1> mov byte [FAT\_BuffValidData], 1

983 0000C11F A1[12610100] <1> mov eax, [FAT\_CurrentCluster]

984 0000C124 8B0D[B4630100] <1> mov ecx, [ClusterValue]

985 0000C12A E972FDFFFF <1> jmp loc\_update\_cluster\_check\_fat\_type

986 <1>

987 <1> save\_fat\_buffer:

988 <1> ; 15/10/2016

989 <1> ; 01/03/2016

990 <1> ; 22/02/2016 (TRDOS 386 = TRDOS v2.0)

991 <1> ; 11/08/2011

992 <1> ; 09/02/2005

993 <1> ; INPUT ->

994 <1> ; None

995 <1> ; OUTPUT ->

996 <1> ; cf = 0 -> OK.

997 <1> ; cf = 1 -> error code in AL (EAX)

998 <1> ;

999 <1> ; EBX = FAT\_Buffer address

1000 <1> ;

1001 <1> ; (EAX, EDX, ECX will be modified)

1002 <1>

1003 <1> ;cmp byte [FAT\_BuffValidData], 2

1004 <1> ;je short loc\_save\_fat\_buff

1005 <1>

1006 <1> ;loc\_save\_fat\_buffer\_retn:

1007 <1> ; xor eax, eax

1008 <1> ; retn

1009 <1>

1010 <1> loc\_save\_fat\_buff:

1011 0000C12F 31D2 <1> xor edx, edx

1012 0000C131 8A35[17610100] <1> mov dh, [FAT\_BuffDrvName]

1013 0000C137 80FE41 <1> cmp dh, 'A'

1014 0000C13A 722E <1> jb short loc\_save\_fat\_buffer\_inv\_data\_retn

1015 0000C13C 80EE41 <1> sub dh, 'A'

1016 0000C13F 56 <1> push esi ; \*

1017 0000C140 BE00010900 <1> mov esi, Logical\_DOSDisks

1018 0000C145 01D6 <1> add esi, edx

1019 <1>

1020 0000C147 8A5603 <1> mov dl, [esi+LD\_FATType]

1021 0000C14A 20D2 <1> and dl, dl

1022 0000C14C 741B <1> jz short loc\_save\_fat\_buffer\_inv\_data\_pop\_retn

1023 <1>

1024 0000C14E A1[1A610100] <1> mov eax, [FAT\_BuffSector]

1025 0000C153 80FA02 <1> cmp dl, 2

1026 0000C156 770A <1> ja short loc\_save\_fat32\_buff

1027 <1>

1028 <1> loc\_save\_fat\_12\_16\_buff:

1029 <1> ; 01/03/2016

1030 <1> ; TRDOS v1 has a FATal bug here!

1031 <1> ; Correct code: mov dx, word ptr [FAT\_BuffSector]+2

1032 <1> ; (DX:AX in TRDOS v1 -> EAX in TRDOS v2)

1033 <1> ;

1034 0000C158 0FB74E1C <1> movzx ecx, word [esi+LD\_BPB+FATSecs]

1035 0000C15C 29C1 <1> sub ecx, eax

1036 <1> ; TRDOS v1 has a bug here... ('pop esi' was forgotten!)

1037 <1> ;jna short loc\_save\_fat\_buffer\_inv\_data\_retn ; wrong addr!

1038 0000C15E 7609 <1> jna short loc\_save\_fat\_buffer\_inv\_data\_pop\_retn ; correct addr.

1039 0000C160 EB15 <1> jmp short loc\_save\_fat\_buffer\_check\_rs3

1040 <1>

1041 <1> loc\_save\_fat32\_buff:

1042 0000C162 8B4E2A <1> mov ecx, [esi+LD\_BPB+FAT32\_FAT\_Size]

1043 0000C165 29C1 <1> sub ecx, eax

1044 0000C167 770E <1> ja short loc\_save\_fat\_buffer\_check\_rs3

1045 <1>

1046 <1> loc\_save\_fat\_buffer\_inv\_data\_pop\_retn:

1047 0000C169 5E <1> pop esi ; \*

1048 <1> loc\_save\_fat\_buffer\_inv\_data\_retn:

1049 0000C16A B80D000000 <1> mov eax, 0Dh ; Invalid DATA

1050 0000C16F C3 <1> retn

1051 <1>

1052 <1> loc\_save\_fat\_buff\_remain\_sectors\_3:

1053 0000C170 B903000000 <1> mov ecx, 3

1054 0000C175 EB05 <1> jmp short loc\_save\_fat\_buff\_continue

1055 <1>

1056 <1> loc\_save\_fat\_buffer\_check\_rs3:

1057 0000C177 83F903 <1> cmp ecx, 3

1058 0000C17A 77F4 <1> ja short loc\_save\_fat\_buff\_remain\_sectors\_3

1059 <1>

1060 <1> loc\_save\_fat\_buff\_continue:

1061 0000C17C BB001C0900 <1> mov ebx, FAT\_Buffer

1062 0000C181 034660 <1> add eax, [esi+LD\_FATBegin]

1063 0000C184 51 <1> push ecx

1064 0000C185 E83F360000 <1> call disk\_write

1065 0000C18A 59 <1> pop ecx

1066 0000C18B 722B <1> jc short loc\_save\_FAT\_buff\_write\_err

1067 <1>

1068 0000C18D 807E0302 <1> cmp byte [esi+LD\_FATType], 2

1069 0000C191 7605 <1> jna short loc\_calc\_2nd\_fat12\_16\_addr

1070 <1>

1071 <1> loc\_calc\_2nd\_fat32\_addr:

1072 0000C193 8B462A <1> mov eax, [esi+LD\_BPB+FAT32\_FAT\_Size]

1073 0000C196 EB04 <1> jmp short loc\_calc\_2nd\_fat\_addr

1074 <1>

1075 <1> loc\_calc\_2nd\_fat12\_16\_addr:

1076 0000C198 0FB7461C <1> movzx eax, word [esi+LD\_BPB+FATSecs]

1077 <1>

1078 <1> loc\_calc\_2nd\_fat\_addr:

1079 0000C19C 034660 <1> add eax, [esi+LD\_FATBegin]

1080 0000C19F 0305[1A610100] <1> add eax, [FAT\_BuffSector]

1081 0000C1A5 BB001C0900 <1> mov ebx, FAT\_Buffer

1082 <1> ; ecx = 1 to 3

1083 0000C1AA E81A360000 <1> call disk\_write

1084 0000C1AF 7207 <1> jc short loc\_save\_FAT\_buff\_write\_err

1085 <1> ; Valid buffer (1 = valid but do not save)

1086 0000C1B1 C605[16610100]01 <1> mov byte [FAT\_BuffValidData], 1

1087 <1>

1088 <1> loc\_save\_FAT\_buff\_write\_err:

1089 0000C1B8 5E <1> pop esi ; \*

1090 0000C1B9 BB001C0900 <1> mov ebx, FAT\_Buffer

1091 <1> ; 15/10/2016 (1Dh -> 18)

1092 <1> ; 23/03/2016 (1Dh)

1093 0000C1BE B812000000 <1> mov eax, 18 ; Drive not ready or write error

1094 0000C1C3 C3 <1> retn

1095 <1>

1096 <1> calculate\_fat\_freespace:

1097 <1> ; 23/03/2016

1098 <1> ; 02/03/2016

1099 <1> ; 01/03/2016

1100 <1> ; 29/02/2016

1101 <1> ; 22/02/2016 (TRDOS 386 = TRDOS v2.0)

1102 <1> ; 30/04/2011

1103 <1> ; 03/04/2010

1104 <1> ; 2005

1105 <1> ; INPUT ->

1106 <1> ; EAX = Cluster count to be added or subtracted

1107 <1> ; If BH = FFh, ESI = TR-DOS Logical Drive Description Table

1108 <1> ; If BH < FFh, BH = TR-DOS Logical Drive Number

1109 <1> ; BL:

1110 <1> ; 0 = Calculate, 1 = Add, 2 = Subtract, 3 = Get (Not Set/Calc)

1111 <1> ; OUTPUT ->

1112 <1> ; EAX = Free Space in sectors

1113 <1> ; ESI = Logical Dos Drive Description Table address

1114 <1> ; BH = Logical Dos Drive Number (same with input value of BH)

1115 <1> ; BL = Type of operation (same with input value of BL)

1116 <1> ; ECX = 0 -> valid

1117 <1> ; ECX > 0 -> error or invalid

1118 <1> ; If EAX = FFFFFFFFh, it is 're-calculation needed'

1119 <1> ; sign due to r/w error

1120 <1>

1121 0000C1C4 66891D[BA630100] <1> mov [CFS\_OPType], bx

1122 0000C1CB A3[BC630100] <1> mov [CFS\_CC], eax

1123 <1>

1124 0000C1D0 80FFFF <1> cmp bh, 0FFh

1125 0000C1D3 740B <1> je short pass\_calculate\_freespace\_get\_drive\_dt\_offset

1126 <1>

1127 <1> loc\_calculate\_freespace\_get\_drive\_dt\_offset:

1128 0000C1D5 31C0 <1> xor eax, eax

1129 0000C1D7 88FC <1> mov ah, bh

1130 0000C1D9 BE00010900 <1> mov esi, Logical\_DOSDisks

1131 0000C1DE 01C6 <1> add esi, eax

1132 <1>

1133 <1> pass\_calculate\_freespace\_get\_drive\_dt\_offset:

1134 0000C1E0 08DB <1> or bl, bl

1135 0000C1E2 7435 <1> jz short loc\_reset\_fcc

1136 <1>

1137 <1> loc\_get\_free\_sectors:

1138 0000C1E4 8B4674 <1> mov eax, [esi+LD\_FreeSectors]

1139 <1>

1140 <1> ;xor ecx, ecx

1141 <1> ;dec ecx ; 0FFFFFFFFh

1142 <1> ;cmp eax, ecx ; 29/02/2016

1143 <1> ;je short loc\_get\_free\_sectors\_retn ; recalculation is needed!

1144 <1>

1145 <1> ; 23/03/2016

1146 0000C1E7 8B4E70 <1> mov ecx, [esi+LD\_TotalSectors]

1147 0000C1EA 39C1 <1> cmp ecx, eax ; Total sectors must be greater than Free sectors !

1148 0000C1EC 7707 <1> ja short loc\_get\_free\_sectors\_check\_optype

1149 <1>

1150 0000C1EE 31C0 <1> xor eax, eax

1151 0000C1F0 48 <1> dec eax ; 0FFFFFFFFh ; recalculation is needed!

1152 0000C1F1 894674 <1> mov [esi+LD\_FreeSectors], eax ; reset (for recalculation)

1153 <1>

1154 <1> loc\_get\_free\_sectors\_retn:

1155 0000C1F4 C3 <1> retn

1156 <1>

1157 <1> loc\_get\_free\_sectors\_check\_optype:

1158 0000C1F5 80FB03 <1> cmp bl, 3

1159 0000C1F8 7203 <1> jb short loc\_set\_fcc

1160 <1>

1161 0000C1FA 29C9 <1> sub ecx, ecx ; 0

1162 <1>

1163 0000C1FC C3 <1> retn

1164 <1>

1165 <1> loc\_set\_fcc:

1166 0000C1FD 807E0302 <1> cmp byte [esi+LD\_FATType], 2

1167 0000C201 0F87DF000000 <1> ja loc\_update\_FAT32\_fs\_info\_fcc

1168 <1>

1169 <1> ;mov eax, [esi+LD\_FreeSectors]

1170 0000C207 0FB64E13 <1> movzx ecx, byte [esi+LD\_BPB+SecPerClust]

1171 0000C20B 29D2 <1> sub edx, edx

1172 0000C20D F7F1 <1> div ecx

1173 <1> ;or dx, dx

1174 <1> ; ; DX -> Remain sectors < SecPerClust

1175 <1> ; ; DX > 0 -> invalid free sector count

1176 <1> ;jnz short loc\_reset\_fcc

1177 <1>

1178 <1> ;pass\_set\_fcc\_div32:

1179 0000C20F A3[33610100] <1> mov [FreeClusterCount], eax

1180 0000C214 E988000000 <1> jmp loc\_set\_free\_sectors\_FAT12\_FAT16

1181 <1>

1182 <1> loc\_reset\_fcc:

1183 0000C219 31C0 <1> xor eax, eax

1184 0000C21B A3[33610100] <1> mov [FreeClusterCount], eax ; 0

1185 0000C220 8B5678 <1> mov edx, [esi+LD\_Clusters]

1186 0000C223 42 <1> inc edx

1187 0000C224 8915[22610100] <1> mov [LastCluster], edx

1188 <1>

1189 0000C22A 807E0302 <1> cmp byte [esi+LD\_FATType], 2

1190 0000C22E 7647 <1> jna short loc\_count\_free\_fat\_clusters\_0

1191 <1>

1192 0000C230 48 <1> dec eax ; FFFFFFFFh

1193 0000C231 A3[C4630100] <1> mov [CFS\_FAT32FC], eax

1194 <1>

1195 <1> ; 29/02/2016

1196 0000C236 89463A <1> mov [esi+LD\_BPB+BPB\_Reserved], eax ; reset

1197 0000C239 89463E <1> mov [esi+LD\_BPB+BPB\_Reserved+4], eax ; reset

1198 <1>

1199 0000C23C B802000000 <1> mov eax, 2

1200 <1>

1201 <1> loc\_count\_fc\_next\_cluster\_0:

1202 0000C241 50 <1> push eax

1203 0000C242 E801F9FFFF <1> call get\_next\_cluster

1204 0000C247 7310 <1> jnc short loc\_check\_fat32\_ff\_cluster

1205 0000C249 09C0 <1> or eax, eax

1206 0000C24B 741E <1> jz short pass\_inc\_cfs\_fcc\_0

1207 <1>

1208 <1> loc\_put\_fcc\_unknown\_sign:

1209 0000C24D 58 <1> pop eax

1210 <1> ; "Free count is Unknown" sign

1211 <1> ;mov dword [FreeClusterCount], 0FFFFFFFFh

1212 <1>

1213 <1> ; 29/02/2016

1214 <1> ; Save Free Cluster Count value in FAT32 'BPB\_Reserved' area

1215 <1> ;mov [esi+LD\_BPB+BPB\_Reserved], 0FFFFFFFFh ; unknown!

1216 0000C24E 8B15[C4630100] <1> mov edx, [CFS\_FAT32FC] ; First Free Cluster

1217 <1> ; Save First Free Cluster value in FAT32 'BPB\_Reserved+4' area

1218 0000C254 89563E <1> mov [esi+LD\_BPB+BPB\_Reserved+4], edx

1219 <1>

1220 0000C257 EB7D <1> jmp loc\_put\_fcc\_invalid\_sign

1221 <1>

1222 <1> loc\_check\_fat32\_ff\_cluster:

1223 0000C259 09C0 <1> or eax, eax

1224 0000C25B 750E <1> jnz short pass\_inc\_cfs\_fcc\_0

1225 0000C25D 58 <1> pop eax

1226 0000C25E A3[C4630100] <1> mov [CFS\_FAT32FC], eax

1227 <1> ;mov dword [FreeClusterCount], 1

1228 0000C263 FF05[33610100] <1> inc dword [FreeClusterCount]

1229 0000C269 EB27 <1> jmp short pass\_inc\_cfs\_fcc\_1

1230 <1>

1231 <1> pass\_inc\_cfs\_fcc\_0:

1232 0000C26B 58 <1> pop eax

1233 <1>

1234 <1> pass\_inc\_cfs\_fcc\_0c:

1235 0000C26C 40 <1> inc eax ; add eax, 1

1236 0000C26D 3B05[22610100] <1> cmp eax, [LastCluster]

1237 0000C273 76CC <1> jna short loc\_count\_fc\_next\_cluster\_0

1238 0000C275 EB6F <1> jmp short loc\_update\_FAT32\_fs\_info\_fcc

1239 <1>

1240 <1> loc\_count\_free\_fat\_clusters\_0:

1241 <1> ;mov eax, 2

1242 0000C277 B002 <1> mov al, 2

1243 <1>

1244 <1> loc\_count\_fc\_next\_cluster:

1245 0000C279 50 <1> push eax

1246 0000C27A E8C9F8FFFF <1> call get\_next\_cluster

1247 0000C27F 720C <1> jc short loc\_count\_fcc\_stc

1248 <1>

1249 <1> loc\_count\_free\_clusters\_1:

1250 0000C281 21C0 <1> and eax, eax

1251 0000C283 750C <1> jnz short pass\_inc\_cfs\_fcc

1252 <1>

1253 0000C285 FF05[33610100] <1> inc dword [FreeClusterCount]

1254 0000C28B EB04 <1> jmp short pass\_inc\_cfs\_fcc

1255 <1>

1256 <1> loc\_count\_fcc\_stc:

1257 0000C28D 09C0 <1> or eax, eax

1258 0000C28F 75BC <1> jnz short loc\_put\_fcc\_unknown\_sign ; 29/02/2016

1259 <1>

1260 <1> pass\_inc\_cfs\_fcc:

1261 0000C291 58 <1> pop eax

1262 <1>

1263 <1> pass\_inc\_cfs\_fcc\_1:

1264 0000C292 40 <1> inc eax ; add eax, 1

1265 0000C293 3B05[22610100] <1> cmp eax, [LastCluster]

1266 0000C299 76DE <1> jna short loc\_count\_fc\_next\_cluster

1267 <1>

1268 <1> loc\_set\_free\_sectors:

1269 0000C29B 807E0302 <1> cmp byte [esi+LD\_FATType], 2

1270 0000C29F 7745 <1> ja short loc\_update\_FAT32\_fs\_info\_fcc

1271 <1>

1272 <1> loc\_set\_free\_sectors\_FAT12\_FAT16:

1273 0000C2A1 803D[BA630100]00 <1> cmp byte [CFS\_OPType], 0

1274 0000C2A8 761C <1> jna short pass\_FAT\_add\_sub\_fcc

1275 0000C2AA A1[BC630100] <1> mov eax, [CFS\_CC]

1276 0000C2AF 803D[BA630100]01 <1> cmp byte [CFS\_OPType], 1

1277 0000C2B6 7708 <1> ja short pass\_FAT\_add\_fcc

1278 0000C2B8 0105[33610100] <1> add [FreeClusterCount], eax

1279 0000C2BE EB06 <1> jmp short pass\_FAT\_add\_sub\_fcc

1280 <1>

1281 <1> pass\_FAT\_add\_fcc:

1282 0000C2C0 2905[33610100] <1> sub [FreeClusterCount], eax

1283 <1>

1284 <1> pass\_FAT\_add\_sub\_fcc:

1285 0000C2C6 0FB64613 <1> movzx eax, byte [esi+LD\_BPB+SecPerClust]

1286 0000C2CA 8B15[33610100] <1> mov edx, [FreeClusterCount]

1287 0000C2D0 F7E2 <1> mul edx

1288 <1>

1289 0000C2D2 31C9 <1> xor ecx, ecx

1290 0000C2D4 EB05 <1> jmp short loc\_cfs\_retn\_params

1291 <1>

1292 <1> loc\_put\_fcc\_invalid\_sign:

1293 0000C2D6 29C0 <1> sub eax, eax ; 0

1294 0000C2D8 48 <1> dec eax ; FFFFFFFFh

1295 <1> loc\_fat32\_ffc\_recalc\_needed:

1296 0000C2D9 89C1 <1> mov ecx, eax

1297 <1>

1298 <1> loc\_cfs\_retn\_params:

1299 0000C2DB 894674 <1> mov [esi+LD\_FreeSectors], eax

1300 0000C2DE 0FB71D[BA630100] <1> movzx ebx, word [CFS\_OPType]

1301 0000C2E5 C3 <1> retn

1302 <1>

1303 <1> loc\_update\_FAT32\_fs\_info\_fcc:

1304 <1> loc\_check\_fcc\_FSINFO\_op:

1305 <1> ; 29/02/2016

1306 <1> ; EAX = Free cluster count (before this update) ; value from disk

1307 <1> ; EDX = First Free Cluster (before this update) ; value from disk

1308 0000C2E6 803D[BA630100]01 <1> cmp byte [CFS\_OPType], 1

1309 0000C2ED 7221 <1> jb short loc\_cfs\_FAT32\_get\_rcalc\_parms ; 0 = recalculated

1310 0000C2EF 7406 <1> je short loc\_check\_fcc\_FSINFO\_op1 ; 1 = add

1311 <1> loc\_check\_fcc\_FSINFO\_op2: ; subtract

1312 0000C2F1 F71D[BC630100] <1> neg dword [CFS\_CC] ; prepare to subtract ; 2 = sub (add negative)

1313 <1> loc\_check\_fcc\_FSINFO\_op1:

1314 <1> ; 01/03/2016

1315 0000C2F7 31D2 <1> xor edx, edx ; 0

1316 0000C2F9 4A <1> dec edx ; 0FFFFFFFFh

1317 0000C2FA 8B463A <1> mov eax, [esi+LD\_BPB+BPB\_Reserved]

1318 0000C2FD 39D0 <1> cmp eax, edx

1319 0000C2FF 73D5 <1> jnb short loc\_put\_fcc\_invalid\_sign

1320 0000C301 0305[BC630100] <1> add eax, [CFS\_CC] ; free cluster count on disk + current count

1321 0000C307 72CD <1> jc short loc\_put\_fcc\_invalid\_sign

1322 <1>

1323 0000C309 A3[33610100] <1> mov [FreeClusterCount], eax

1324 0000C30E EB0E <1> jmp short loc\_cfs\_write\_FSINFO\_sector

1325 <1>

1326 <1> loc\_cfs\_FAT32\_get\_rcalc\_parms:

1327 0000C310 8B15[C4630100] <1> mov edx, [CFS\_FAT32FC]

1328 0000C316 A1[33610100] <1> mov eax, [FreeClusterCount]

1329 0000C31B 89563E <1> mov [esi+LD\_BPB+BPB\_Reserved+4], edx ; First Free Cluster

1330 <1> loc\_cfs\_write\_FSINFO\_sector:

1331 0000C31E 89463A <1> mov [esi+LD\_BPB+BPB\_Reserved], eax ; Free cluster count

1332 <1> ; 01/03/2016

1333 0000C321 E8AA000000 <1> call set\_fat32\_fsinfo\_sector\_parms

1334 0000C326 72AE <1> jc short loc\_put\_fcc\_invalid\_sign

1335 <1>

1336 <1> loc\_set\_FAT32\_free\_sectors:

1337 <1> ; 29/02/2016

1338 <1> ;mov eax, [FreeClusterCount]

1339 <1> ;mov ecx, eax

1340 <1> ;cmp eax, 0FFFFFFFFh ; Invalid !

1341 <1> ;je short loc\_cfs\_retn\_params

1342 <1> ;

1343 0000C328 8B0D[33610100] <1> mov ecx, [FreeClusterCount]

1344 0000C32E 0FB64613 <1> movzx eax, byte [esi+LD\_BPB+SecPerClust]

1345 0000C332 F7E1 <1> mul ecx

1346 <1> ; 29/02/2016

1347 0000C334 31C9 <1> xor ecx, ecx ; 0

1348 0000C336 09D2 <1> or edx, edx ; 0 ?

1349 0000C338 759C <1> jnz loc\_put\_fcc\_invalid\_sign

1350 0000C33A 394670 <1> cmp [esi+LD\_TotalSectors], eax ; Volume size in sectors

1351 0000C33D 7697 <1> jna short loc\_put\_fcc\_invalid\_sign

1352 <1> ;

1353 <1> loc\_set\_FAT32\_free\_sectors\_ok:

1354 0000C33F 31D2 <1> xor edx, edx ; 0

1355 0000C341 EB98 <1> jmp short loc\_cfs\_retn\_params

1356 <1> ;

1357 <1>

1358 <1> get\_last\_cluster:

1359 <1> ; 22/10/2016

1360 <1> ; 27/02/2016 (TRDOS 386 = TRDOS v2.0)

1361 <1> ; 12/06/2010 (DRV\_FAT.ASM, 'proc\_get\_last\_custer')

1362 <1> ; 06/06/2010

1363 <1> ; INPUT ->

1364 <1> ; EAX = First Cluster Number

1365 <1> ; ESI = Logical Dos Drive Parameters Table

1366 <1> ; OUTPUT ->

1367 <1> ; cf = 0 -> No Error, EAX is valid

1368 <1> ; cf = 1 -> EAX > 0 -> Error

1369 <1> ; EAX = Last Cluster Number

1370 <1> ; ECX = Previous Cluster -just before the last cluster-

1371 <1> ; ; 22/10/2016

1372 <1> ; [glc\_index] = cluster index number of the last cluster

1373 <1> ;

1374 <1> ; (Modified registers: EAX, ECX, EBX, EDX)

1375 <1>

1376 0000C343 89C1 <1> mov ecx, eax

1377 <1>

1378 0000C345 C705[CC630100]FFFF- <1> mov dword [glc\_index], 0FFFFFFFFh ; 22/10/2016

1378 0000C34D FFFF <1>

1379 <1>

1380 <1> loc\_glc\_get\_next\_cluster\_1:

1381 0000C34F 890D[C8630100] <1> mov [glc\_prevcluster], ecx

1382 <1> ; 22/10/2016

1383 0000C355 FF05[CC630100] <1> inc dword [glc\_index]

1384 <1>

1385 <1> loc\_glc\_get\_next\_cluster\_2:

1386 0000C35B E8E8F7FFFF <1> call get\_next\_cluster

1387 <1> ; ecx = current/previous cluster

1388 <1> ; eax = next/last cluster

1389 0000C360 73ED <1> jnc short loc\_glc\_get\_next\_cluster\_1

1390 <1>

1391 0000C362 09C0 <1> or eax, eax

1392 0000C364 7509 <1> jnz short loc\_glc\_stc\_retn

1393 <1>

1394 <1> ; ecx = previous cluster

1395 0000C366 89C8 <1> mov eax, ecx

1396 <1>

1397 <1> ; previous cluster becomes last cluster (ecx -> eax)

1398 <1> ; previous of previous cluster becomes previous cluster (ecx)

1399 <1>

1400 <1> loc\_glc\_prev\_cluster\_retn:

1401 0000C368 8B0D[C8630100] <1> mov ecx, [glc\_prevcluster]

1402 0000C36E C3 <1> retn

1403 <1>

1404 <1> loc\_glc\_stc\_retn:

1405 0000C36F F5 <1> cmc ;stc

1406 0000C370 EBF6 <1> jmp short loc\_glc\_prev\_cluster\_retn

1407 <1>

1408 <1> truncate\_cluster\_chain:

1409 <1> ; 01/03/2016

1410 <1> ; 28/02/2016 (TRDOS 386 = TRDOS v2.0)

1411 <1> ; 22/01/2011 (DRV\_FAT.ASM, 'proc\_truncate\_cluster\_chain')

1412 <1> ; 11/09/2010

1413 <1> ; INPUT ->

1414 <1> ; ESI = Logical dos drive description table address

1415 <1> ; EAX = First cluster to be truncated/unlinked

1416 <1> ; OUTPUT ->

1417 <1> ; ESI = Logical dos drive description table address

1418 <1> ; ECX = Count of truncated/removed clusters

1419 <1> ; CF = 0 -> EAX = Free sectors

1420 <1> ; CF = 1 -> Error code in EAX (AL)

1421 <1>

1422 <1> ; NOTE: This procedure does not update lm date&time !

1423 <1>

1424 <1> loc\_truncate\_cc:

1425 0000C372 31C9 <1> xor ecx, ecx ; mov ecx, 0

1426 <1> ;mov byte [FAT\_BuffValidData], 0

1427 0000C374 890D[1E610100] <1> mov [FAT\_ClusterCounter], ecx ; 0 ; reset

1428 <1>

1429 <1> loc\_tcc\_unlink\_clusters:

1430 0000C37A E8F3FAFFFF <1> call update\_cluster

1431 <1> ; EAX = Next Cluster

1432 <1> ; ECX = Cluster Value

1433 <1> ; Note:

1434 <1> ; Returns count of unlinked clusters in

1435 <1> ; dword ptr FAT\_ClusterCounter

1436 0000C37F 73F9 <1> jnc short loc\_tcc\_unlink\_clusters

1437 <1>

1438 <1> pass\_tcc\_unlink\_clusters:

1439 0000C381 A2[D3630100] <1> mov byte [TCC\_FATErr], al

1440 0000C386 803D[16610100]02 <1> cmp byte [FAT\_BuffValidData], 2

1441 0000C38D 750E <1> jne short loc\_tcc\_calculate\_FAT\_freespace

1442 0000C38F E89BFDFFFF <1> call save\_fat\_buffer

1443 0000C394 7307 <1> jnc short loc\_tcc\_calculate\_FAT\_freespace

1444 0000C396 A2[D3630100] <1> mov byte [TCC\_FATErr], al ; Error

1445 <1> ;mov byte [FAT\_BuffValidData], 0

1446 <1>

1447 <1> ; 01/03/2016

1448 0000C39B EB12 <1> jmp short loc\_tcc\_recalculate\_FAT\_freespace

1449 <1>

1450 <1> loc\_tcc\_calculate\_FAT\_freespace:

1451 0000C39D A1[1E610100] <1> mov eax, [FAT\_ClusterCounter] ; signed (+-) number

1452 0000C3A2 66BB01FF <1> mov bx, 0FF01h ; BH = FFh -> ESI = Dos drv desc. table

1453 <1> ; BL = 1 -> add cluster

1454 0000C3A6 E819FEFFFF <1> call calculate\_fat\_freespace

1455 0000C3AB 21C9 <1> and ecx, ecx ; cx = 0 -> valid free sector count

1456 0000C3AD 7409 <1> jz short pass\_truncate\_cc\_recalc\_FAT\_freespace

1457 <1>

1458 <1> loc\_tcc\_recalculate\_FAT\_freespace:

1459 0000C3AF 66BB00FF <1> mov bx, 0FF00h ; recalculate !

1460 0000C3B3 E80CFEFFFF <1> call calculate\_fat\_freespace

1461 <1>

1462 <1> loc\_tcc\_calculate\_FAT\_freespace\_err:

1463 <1> pass\_truncate\_cc\_recalc\_FAT\_freespace:

1464 0000C3B8 8B0D[1E610100] <1> mov ecx, [FAT\_ClusterCounter]

1465 <1>

1466 0000C3BE 803D[D3630100]00 <1> cmp byte [TCC\_FATErr], 0

1467 0000C3C5 7608 <1> jna short loc\_tcc\_unlink\_clusters\_retn

1468 <1>

1469 <1> loc\_tcc\_unlink\_clusters\_error:

1470 0000C3C7 0FB605[D3630100] <1> movzx eax, byte [TCC\_FATErr]

1471 0000C3CE F9 <1> stc

1472 <1> loc\_tcc\_unlink\_clusters\_retn:

1473 0000C3CF C3 <1> retn

1474 <1>

1475 <1> set\_fat32\_fsinfo\_sector\_parms:

1476 <1> ; 15/10/2016

1477 <1> ; 23/03/2016

1478 <1> ; 29/02/2016 (TRDOS 386 = TRDOS v2.0)

1479 <1> ; INPUT ->

1480 <1> ; ESI = Logical dos drive description table address

1481 <1> ; [esi+LD\_BPB+BPB\_Reserved] = Free Cluster Count

1482 <1> ; [esi+LD\_BPB+BPB\_Reserved+4] = First Free Cluster

1483 <1> ; OUTPUT ->

1484 <1> ; ESI = Logical dos drive description table address

1485 <1> ; CF = 0 -> OK..

1486 <1> ; CF = 1 -> Error code in EAX (AL)

1487 <1> ;

1488 <1> ; (Modified registers: EAX, EBX, ECX, EDX)

1489 <1>

1490 0000C3D0 E824000000 <1> call get\_fat32\_fsinfo\_sector\_parms

1491 0000C3D5 7221 <1> jc short update\_fat32\_fsinfo\_sector\_retn

1492 <1>

1493 0000C3D7 8B463A <1> mov eax, [esi+LD\_BPB+BPB\_Reserved] ; Free Cluster Count

1494 0000C3DA 8B563E <1> mov edx, [esi+LD\_BPB+BPB\_Reserved+4] ; First free Cluster

1495 <1>

1496 <1> ;mov ebx, DOSBootSectorBuff

1497 0000C3DD 8983E8010000 <1> mov [ebx+488], eax

1498 0000C3E3 8993EC010000 <1> mov [ebx+492], edx

1499 <1>

1500 0000C3E9 A1[C0630100] <1> mov eax, [CFS\_FAT32FSINFOSEC]

1501 0000C3EE B901000000 <1> mov ecx, 1

1502 0000C3F3 E8D1330000 <1> call disk\_write

1503 <1> ;jnc short update\_fat32\_fsinfo\_sector\_retn

1504 <1>

1505 <1> ; 15/10/2016 (1Dh -> 18)

1506 <1> ; 23/03/2016 (1Dh)

1507 <1> ;mov eax, 18 ; Drive not ready or write error

1508 <1>

1509 <1> update\_fat32\_fsinfo\_sector\_retn:

1510 0000C3F8 C3 <1> retn

1511 <1>

1512 <1> get\_fat32\_fsinfo\_sector\_parms:

1513 <1> ; 15/10/2016

1514 <1> ; 23/03/2016

1515 <1> ; 01/03/2016

1516 <1> ; 29/02/2016 (TRDOS 386 = TRDOS v2.0)

1517 <1> ; INPUT ->

1518 <1> ; ESI = Logical dos drive description table address

1519 <1> ; OUTPUT ->

1520 <1> ; ESI = Logical dos drive description table address

1521 <1> ; EBX = FSINFO sector buffer address (DOSBootSectorBuff)

1522 <1> ; CF = 0 -> OK..

1523 <1> ; EAX = FsInfo sector address

1524 <1> ; ECX = Free cluster count

1525 <1> ; EDX = First free cluster

1526 <1> ; CF = 1 -> Error code in AL (EAX)

1527 <1> ; EBX = 0

1528 <1> ;

1529 <1> ; [CFS\_FAT32FSINFOSEC] = FAT32 FSINFO sector address

1530 <1> ;

1531 <1> ; (Modified registers: EAX, EBX, ECX, EDX)

1532 <1>

1533 0000C3F9 0FB74636 <1> movzx eax, word [esi+LD\_BPB+FAT32\_FSInfoSec]

1534 0000C3FD 03466C <1> add eax, [esi+LD\_StartSector]

1535 0000C400 A3[C0630100] <1> mov [CFS\_FAT32FSINFOSEC], eax

1536 <1>

1537 0000C405 BB[125F0100] <1> mov ebx, DOSBootSectorBuff

1538 0000C40A B901000000 <1> mov ecx, 1

1539 0000C40F E8C4330000 <1> call disk\_read

1540 0000C414 7232 <1> jc short loc\_read\_FAT32\_fsinfo\_sec\_err

1541 <1>

1542 0000C416 BB[125F0100] <1> mov ebx, DOSBootSectorBuff

1543 <1>

1544 0000C41B 813B52526141 <1> cmp dword [ebx], 41615252h

1545 0000C421 751E <1> jne short loc\_read\_FAT32\_fsinfo\_sec\_stc

1546 <1>

1547 0000C423 81BBE4010000727241- <1> cmp dword [ebx+484], 61417272h

1547 0000C42C 61 <1>

1548 0000C42D 7512 <1> jne short loc\_read\_FAT32\_fsinfo\_sec\_stc

1549 <1>

1550 0000C42F A1[C0630100] <1> mov eax, [CFS\_FAT32FSINFOSEC]

1551 0000C434 8B8BE8010000 <1> mov ecx, [ebx+488] ; free cluster count

1552 0000C43A 8B93EC010000 <1> mov edx, [ebx+492] ; first (next) free cluster

1553 <1>

1554 0000C440 C3 <1> retn

1555 <1>

1556 <1> loc\_read\_FAT32\_fsinfo\_sec\_stc:

1557 <1> ; 15/10/2016 (0Bh -> 28)

1558 0000C441 B81C000000 <1> mov eax, 28 ; Invalid format!

1559 0000C446 EB05 <1> jmp short loc\_read\_FAT32\_fsinfo\_sec\_stc\_retn

1560 <1>

1561 <1> loc\_read\_FAT32\_fsinfo\_sec\_err:

1562 <1> ; 15/10/2016 (15h -> 17)

1563 <1> ; 23/03/2016 (15h)

1564 0000C448 B811000000 <1> mov eax, 17 ; Drive not ready or read error

1565 <1>

1566 <1> loc\_read\_FAT32\_fsinfo\_sec\_stc\_retn:

1567 0000C44D 29DB <1> sub ebx, ebx ; 0

1568 0000C44F F9 <1> stc

1569 0000C450 C3 <1> retn

1570 <1>

1571 <1> add\_new\_cluster:

1572 <1> ; 15/10/2016

1573 <1> ; 16/05/2016

1574 <1> ; 18/03/2016, 24/03/2016

1575 <1> ; 11/03/2016 (TRDOS 386 = TRDOS v2.0)

1576 <1> ; 30/07/2011 (DRV\_FAT.ASM)

1577 <1> ; 11/09/2010

1578 <1> ; INPUT ->

1579 <1> ; ESI = Logical dos drv desc. table address

1580 <1> ; EAX = Last cluster

1581 <1> ; OUTPUT ->

1582 <1> ; ESI = Logical dos drv desc. table address

1583 <1> ; EAX = New Last cluster (next cluster)

1584 <1> ; cf = 1 -> error code in EAX (AL)

1585 <1> ; cf = 1 -> DX = sectors per cluster

1586 <1> ; ECX = Free sectors

1587 <1> ; NOTE:

1588 <1> ; This procedure does not update lm date&time !

1589 <1> ;

1590 <1> ; (Modified registers: EAX, EBX, ECX, EDX, EDI)

1591 <1> ;

1592 <1>

1593 0000C451 A3[F0640100] <1> mov [FAT\_anc\_LCluster], eax

1594 <1>

1595 0000C456 E844F9FFFF <1> call get\_first\_free\_cluster

1596 0000C45B 720B <1> jc short loc\_add\_new\_cluster\_retn

1597 <1> ; EAX >= 2 and EAX < FFFFFFFFh is valid

1598 <1>

1599 0000C45D 89C2 <1> mov edx, eax

1600 <1>

1601 0000C45F 42 <1> inc edx

1602 <1> ;jnz short loc\_add\_new\_cluster\_check\_ffc\_eax

1603 0000C460 7516 <1> jnz short loc\_add\_new\_cluster\_save\_fcc

1604 <1>

1605 <1> loc\_add\_new\_cluster\_no\_disk\_space\_retn:

1606 0000C462 B827000000 <1> mov eax, 27h ; MSDOS err => insufficient disk space

1607 <1> loc\_add\_new\_cluster\_stc\_retn:

1608 0000C467 F9 <1> stc

1609 <1> loc\_add\_new\_cluster\_retn:

1610 0000C468 0FB65E13 <1> movzx ebx, byte [esi+LD\_BPB+SecPerClust]

1611 0000C46C 8B4E74 <1> mov ecx, [esi+LD\_FreeSectors]

1612 <1> ;xor edx, edx

1613 <1> ;stc

1614 0000C46F C3 <1> retn

1615 <1>

1616 <1> loc\_anc\_invalid\_format\_stc\_retn:

1617 0000C470 F9 <1> stc

1618 <1> loc\_add\_new\_cluster\_invalid\_format\_retn:

1619 <1> ; 15/10/2016 (0Bh -> 28)

1620 0000C471 B81C000000 <1> mov eax, 28 ; Invalid format

1621 0000C476 EBF0 <1> jmp short loc\_add\_new\_cluster\_retn

1622 <1>

1623 <1> ;loc\_add\_new\_cluster\_check\_ffc\_eax:

1624 <1> ; cmp eax, 2

1625 <1> ; jb short loc\_add\_new\_cluster\_invalid\_format\_retn

1626 <1>

1627 <1> loc\_add\_new\_cluster\_save\_fcc:

1628 0000C478 A3[F4640100] <1> mov [FAT\_anc\_FFCluster], eax

1629 <1>

1630 0000C47D 83E802 <1> sub eax, 2

1631 0000C480 0FB65E13 <1> movzx ebx, byte [esi+LD\_BPB+SecPerClust]

1632 0000C484 F7E3 <1> mul ebx

1633 0000C486 09D2 <1> or edx, edx

1634 0000C488 75E6 <1> jnz short loc\_anc\_invalid\_format\_stc\_retn

1635 <1>

1636 <1> loc\_add\_new\_cluster\_allocate\_cluster:

1637 <1> ; 18/03/2016

1638 0000C48A 92 <1> xchg edx, eax ; eax = 0

1639 <1> ; 16/05/2016

1640 <1> ;cmp [ClusterBuffer\_Valid], al ; 0

1641 <1> ;jna short loc\_anc\_clear\_cluster\_buffer

1642 <1> ;; 'copy' command,

1643 <1> ;; writing destination file clust after reading source file clust

1644 <1> ;mov [ClusterBuffer\_Valid], al ; 0 ; reset

1645 <1> ;jmp short loc\_add\_new\_cluster\_write\_nc\_to\_disk

1646 <1>

1647 <1> loc\_anc\_clear\_cluster\_buffer:

1648 <1> ; 11/03/2016

1649 <1> ; Clear buffer

1650 0000C48B BF00000700 <1> mov edi, Cluster\_Buffer ; 70000h (for current TRDOS 386 version)

1651 0000C490 89D9 <1> mov ecx, ebx ; sector count

1652 0000C492 C1E107 <1> shl ecx, 7 ; 1 sector = 512 bytes -> 128 double words

1653 <1> ;xor eax, eax ; 0

1654 0000C495 F3AB <1> rep stosd

1655 <1>

1656 <1> loc\_add\_new\_cluster\_write\_nc\_to\_disk:

1657 <1> ; 11/03/2016

1658 <1> ;xchg eax, edx ; edx = 0, eax = sector offset

1659 0000C497 89D0 <1> mov eax, edx

1660 0000C499 034668 <1> add eax, [esi+LD\_DATABegin]

1661 0000C49C 72D3 <1> jc short loc\_add\_new\_cluster\_invalid\_format\_retn

1662 <1>

1663 0000C49E 89D9 <1> mov ecx, ebx ; ECX = sectors per cluster (<256)

1664 0000C4A0 BB00000700 <1> mov ebx, Cluster\_Buffer

1665 0000C4A5 E81F330000 <1> call disk\_write

1666 0000C4AA 7307 <1> jnc short loc\_add\_new\_cluster\_update\_fat\_nlc

1667 <1>

1668 <1> ; 15/10/2016 (1Dh -> 18)

1669 0000C4AC B812000000 <1> mov eax, 18 ; Write Error

1670 0000C4B1 EBB4 <1> jmp short loc\_add\_new\_cluster\_stc\_retn

1671 <1>

1672 <1> loc\_add\_new\_cluster\_update\_fat\_nlc:

1673 0000C4B3 A1[F4640100] <1> mov eax, [FAT\_anc\_FFCluster]

1674 0000C4B8 31C9 <1> xor ecx, ecx

1675 0000C4BA 890D[1E610100] <1> mov [FAT\_ClusterCounter], ecx ; 0 ; reset

1676 0000C4C0 49 <1> dec ecx ; 0FFFFFFFFh

1677 0000C4C1 E8ACF9FFFF <1> call update\_cluster

1678 0000C4C6 7304 <1> jnc short loc\_add\_new\_cluster\_update\_fat\_plc

1679 0000C4C8 09C0 <1> or eax, eax ;EAX = 0 -> cluster value is 0 or eocc

1680 0000C4CA 759B <1> jnz short loc\_add\_new\_cluster\_stc\_retn

1681 <1>

1682 <1> loc\_add\_new\_cluster\_update\_fat\_plc:

1683 0000C4CC A1[F0640100] <1> mov eax, [FAT\_anc\_LCluster]

1684 0000C4D1 8B0D[F4640100] <1> mov ecx, [FAT\_anc\_FFCluster]

1685 0000C4D7 E896F9FFFF <1> call update\_cluster

1686 0000C4DC 7314 <1> jnc short loc\_add\_new\_cluster\_save\_fat\_buffer

1687 0000C4DE 09C0 <1> or eax, eax ; EAX = 0 -> cluster value is 0 or eocc

1688 0000C4E0 7410 <1> jz short loc\_add\_new\_cluster\_save\_fat\_buffer

1689 <1>

1690 <1> loc\_anc\_save\_fat\_buffer\_err\_retn:

1691 <1> ;cmp byte [FAT\_ClusterCounter], 1

1692 <1> ;jb short loc\_add\_new\_cluster\_retn

1693 <1>

1694 0000C4E2 66BB00FF <1> mov bx, 0FF00h ; recalculate free space (BL = 0)

1695 <1> ; (BH = FFh -> Use ESI as Drv Param. Tbl.)

1696 0000C4E6 50 <1> push eax

1697 0000C4E7 E8D8FCFFFF <1> call calculate\_fat\_freespace

1698 0000C4EC 58 <1> pop eax

1699 0000C4ED E975FFFFFF <1> jmp loc\_add\_new\_cluster\_stc\_retn

1700 <1>

1701 <1> loc\_add\_new\_cluster\_save\_fat\_buffer:

1702 <1> ;cmp byte [FAT\_BuffValidData], 2

1703 <1> ;jne short loc\_add\_new\_cluster\_calc\_FAT\_freespace

1704 <1> ;Byte [FAT\_BuffValidData] = 2

1705 0000C4F2 E838FCFFFF <1> call save\_fat\_buffer

1706 0000C4F7 72E9 <1> jc short loc\_anc\_save\_fat\_buffer\_err\_retn

1707 <1>

1708 <1> loc\_add\_new\_cluster\_calc\_FAT\_freespace:

1709 <1> ;mov eax, 1 ; Only one Cluster

1710 0000C4F9 A1[1E610100] <1> mov eax, [FAT\_ClusterCounter]

1711 0000C4FE 66BB01FF <1> mov bx, 0FF01h ; BH = FFh -> ESI -> Dos drv desc. table

1712 <1> ; BL = 1 -> add cluster

1713 0000C502 B301 <1> mov bl, 01h ; BL = 1 -> add clusters

1714 <1> ; NOTE: EAX value will be added to Free Cluster Count

1715 <1> ; (Free Cluster Count is decreased when EAX value is negative)

1716 0000C504 E8BBFCFFFF <1> call calculate\_fat\_freespace

1717 <1> ;ECX = 0 -> no error, ECX > 0 -> error or invalid return

1718 0000C509 21C9 <1> and ecx, ecx ; ECX = 0 -> valid free sector count

1719 0000C50B 7409 <1> jz short loc\_add\_new\_cluster\_return\_cluster\_number

1720 <1>

1721 <1> loc\_add\_new\_cluster\_recalc\_FAT\_freespace:

1722 0000C50D 66BB00FF <1> mov bx, 0FF00h ; recalculate free space

1723 0000C511 E8AEFCFFFF <1> call calculate\_fat\_freespace

1724 <1> ; cf = 0

1725 <1> loc\_add\_new\_cluster\_return\_cluster\_number:

1726 0000C516 89C1 <1> mov ecx, eax ; Free sector count

1727 0000C518 A1[F4640100] <1> mov eax, [FAT\_anc\_FFCluster]

1728 0000C51D 0FB65E13 <1> movzx ebx, byte [esi+LD\_BPB+SecPerClust]

1729 <1> ;mov edi, Cluster\_Buffer

1730 0000C521 31D2 <1> xor edx, edx

1731 0000C523 C3 <1> retn

1732 <1>

1733 <1> write\_cluster:

1734 <1> ; 15/10/2016

1735 <1> ; 21/03/2016 (TRDOS 386 = TRDOS v2.0)

1736 <1> ;

1737 <1> ; INPUT ->

1738 <1> ; EAX = Cluster Number (Sector index for SINGLIX FS)

1739 <1> ; ESI = Logical DOS Drive Description Table address

1740 <1> ; EBX = Cluster (File R/W) Buffer address (max. 64KB)

1741 <1> ; Only for SINGLIX FS:

1742 <1> ; EDX = File Number (The 1st FDT address)

1743 <1> ; OUTPUT ->

1744 <1> ; cf = 1 -> Cluster can not be written onto disk

1745 <1> ; EAX > 0 -> Error number

1746 <1> ; cf = 0 -> Cluster has been written successfully

1747 <1> ;

1748 <1> ; (Modified registers: EAX, ECX, EBX, EDX)

1749 <1>

1750 0000C524 0FB64E13 <1> movzx ecx, byte [esi+LD\_BPB+BPB\_SecPerClust]

1751 <1> ; CL = 1 = [esi+LD\_FS\_Reserved2] ; SectPerClust for Singlix FS

1752 <1>

1753 <1> write\_file\_sectors: ; 16/03/2016

1754 0000C528 807E0300 <1> cmp byte [esi+LD\_FATType], 0

1755 0000C52C 761C <1> jna short write\_fs\_cluster

1756 <1>

1757 <1> write\_fat\_file\_sectors:

1758 0000C52E 83E802 <1> sub eax, 2 ; Beginning cluster number is always 2

1759 0000C531 0FB65613 <1> movzx edx, byte [esi+LD\_BPB+BPB\_SecPerClust] ; 18/03/2016

1760 0000C535 F7E2 <1> mul edx

1761 0000C537 034668 <1> add eax, [esi+LD\_DATABegin] ; absolute address of the cluster

1762 <1>

1763 <1> ; EAX = Disk sector address

1764 <1> ; ECX = Sector count

1765 <1> ; EBX = Buffer address

1766 <1> ; (EDX = 0)

1767 <1> ; ESI = Logical DOS drive description table address

1768 <1>

1769 0000C53A E88A320000 <1> call disk\_write

1770 0000C53F 7306 <1> jnc short wclust\_retn

1771 <1>

1772 <1> ; 15/10/2016 (1Dh -> 18)

1773 0000C541 B812000000 <1> mov eax, 18 ; Drive not ready or write error !

1774 0000C546 C3 <1> retn

1775 <1>

1776 <1> wclust\_retn:

1777 0000C547 29C0 <1> sub eax, eax ; 0

1778 0000C549 C3 <1> retn

1779 <1>

1780 <1> write\_fs\_cluster:

1781 <1> ; 21/03/2016 (TRDOS 386 = TRDOS v2.0)

1782 <1> ; Singlix FS

1783 <1>

1784 <1> ; EAX = Cluster number is sector index number of the file (eax)

1785 <1>

1786 <1> ; EDX = File number is the first File Descriptor Table address

1787 <1> ; of the file. (Absolute address of the FDT).

1788 <1>

1789 <1> ; eax = sector index (0 for the first sector)

1790 <1> ; edx = FDT0 address

1791 <1> ; 64 KB buffer = 128 sectors (limit)

1792 0000C54A B980000000 <1> mov ecx, 128 ; maximum count of sectors (before eof)

1793 0000C54F E801000000 <1> call write\_fs\_sectors

1794 0000C554 C3 <1> retn

1795 <1>

1796 <1> write\_fs\_sectors:

1797 <1> ; 21/03/2016 (TRDOS 386 = TRDOS v2.0)

1798 0000C555 F9 <1> stc

1799 0000C556 C3 <1> retn

1800 <1>

1801 <1> get\_cluster\_by\_index:

1802 <1> ; 29/04/2016 (TRDOS 386 = TRDOS v2.0)

1803 <1> ; INPUT ->

1804 <1> ; EAX = Beginning cluster

1805 <1> ; EDX = Sector index in disk/file section

1806 <1> ; (Only for SINGLIX file system!)

1807 <1> ; ECX = Cluster sequence number after the beginning cluster

1808 <1> ; ESI = Logical DOS Drive Description Table address

1809 <1> ; OUTPUT ->

1810 <1> ; EAX = Cluster number

1811 <1> ; cf = 1 -> Error code in AL (EAX)

1812 <1> ;

1813 <1> ;(Modified registers: EAX, ECX, EBX, EDX)

1814 <1> ;

1815 0000C557 807E0301 <1> cmp byte [esi+LD\_FATType], 1

1816 0000C55B 721E <1> jb short get\_fs\_section\_by\_index

1817 <1>

1818 0000C55D 3B4E78 <1> cmp ecx, [esi+LD\_Clusters]

1819 0000C560 7207 <1> jb short gcbi\_1

1820 <1> gcbi\_0:

1821 0000C562 F9 <1> stc

1822 0000C563 B823000000 <1> mov eax, 23h ; Cluster not available !

1823 <1> ; MSDOS error code: FCB unavailable

1824 0000C568 C3 <1> retn

1825 <1> gcbi\_1:

1826 0000C569 51 <1> push ecx

1827 0000C56A E8D9F5FFFF <1> call get\_next\_cluster

1828 0000C56F 59 <1> pop ecx

1829 0000C570 7203 <1> jc short gcbi\_3

1830 0000C572 E2F5 <1> loop gcbi\_1

1831 <1> gcbi\_2:

1832 0000C574 C3 <1> retn

1833 <1> gcbi\_3:

1834 0000C575 09C0 <1> or eax, eax

1835 0000C577 74E9 <1> jz short gcbi\_0

1836 0000C579 F5 <1> cmc ; stc

1837 0000C57A C3 <1> retn

1838 <1>

1839 <1> get\_fs\_section\_by\_index:

1840 <1> ; 29/04/2016 (TRDOS 386 = TRDOS v2.0)

1841 <1> ; INPUT ->

1842 <1> ; EAX = Beginning FDT number/address

1843 <1> ; EDX = Sector index in disk/file section

1844 <1> ; ECX = Sector sequence number after the beginning FDT

1845 <1> ; ESI = Logical DOS Drive Description Table address

1846 <1> ; OUTPUT ->

1847 <1> ; EAX = FDT number/address

1848 <1> ; EDX = Sector index of the section (0,1,2,3,4...)

1849 <1> ; cf = 1 -> Error code in AL (EAX)

1850 <1> ;

1851 <1> ;(Modified registers: EAX, ECX, EBX, EDX)

1852 <1> ;

1853 0000C57B B8FFFFFFFF <1> mov eax, 0FFFFFFFFh

1854 0000C580 C3 <1> retn

1855 <1>

1856 <1> get\_last\_section:

1857 <1> ; 22/10/2016 (TRDOS 386 = TRDOS v2.0)

1858 <1> ; INPUT ->

1859 <1> ; EAX = (The 1st) FDT number/address

1860 <1> ; ESI = Logical DOS Drive Description Table address

1861 <1> ; OUTPUT ->

1862 <1> ; EAX = FDT number/address of the last section

1863 <1> ; EDX = Last sector of the section (0,1,2,3,4...)

1864 <1> ; [glc\_index] = sector index number of the last sector

1865 <1> ; (for file, not for the last section)

1866 <1> ;

1867 <1> ; cf = 1 -> Error code in AL (EAX)

1868 <1> ;

1869 <1> ;(Modified registers: EAX, ECX, EBX, EDX)

1870 <1> ;

1871 0000C581 B800000000 <1> mov eax, 0

1872 0000C586 BA00000000 <1> mov edx, 0

1873 0000C58B C3 <1> retn

2310 %include 'trdosk6.s' ; 24/01/2016

1 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2 <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.0) - MAIN PROGRAM : trdosk6.s

3 <1> ; ----------------------------------------------------------------------------

4 <1> ; Last Update: 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> ; u1.s (27/17/2015), u2.s (03/01/2016)

12 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

13 <1> ; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan

14 <1> ; TRDOS2.ASM (09/11/2011)

15 <1> ; ----------------------------------------------------------------------------

16 <1> ; INT\_21H.ASM (c) 2009-2011 Erdogan TAN [14/11/2009] Last Update: 08/11/2011

17 <1>

18 <1> sysent: ; < enter to system call >

19 <1> ; 17/03/2017

20 <1> ; 03/03/2017

21 <1> ; 19/02/2017

22 <1> ; 13/01/2017

23 <1> ; 06/06/2016

24 <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)

25 <1> ; 16/04/2015 - 19/10/2015 (Retro UNIX 386 v1)

26 <1> ; 10/04/2013 - 18/01/2014 (Retro UNIX 8086 v1)

27 <1> ;

28 <1> ; 'unkni' or 'sysent' is sytem entry from various traps.

29 <1> ; The trap type is determined and an indirect jump is made to

30 <1> ; the appropriate system call handler. If there is a trap inside

31 <1> ; the system a jump to panic is made. All user registers are saved

32 <1> ; and u.sp points to the end of the users stack. The sys (trap)

33 <1> ; instructor is decoded to get the the system code part (see

34 <1> ; trap instruction in the PDP-11 handbook) and from this

35 <1> ; the indirect jump address is calculated. If a bad system call is

36 <1> ; made, i.e., the limits of the jump table are exceeded, 'badsys'

37 <1> ; is called. If the call is legitimate control passes to the

38 <1> ; appropriate system routine.

39 <1> ;

40 <1> ; Calling sequence:

41 <1> ; Through a trap caused by any sys call outside the system.

42 <1> ; Arguments:

43 <1> ; Arguments of particular system call.

44 <1> ; ...............................................................

45 <1> ;

46 <1> ; Retro UNIX 8086 v1 modification:

47 <1> ; System call number is in EAX register.

48 <1> ;

49 <1> ; Other parameters are in EDX, EBX, ECX, ESI, EDI, EBP

50 <1> ; registers depending of function details.

51 <1> ;

52 <1> ; 16/04/2015

53 0000C58C 368925[5C030300] <1> mov [ss:u.sp], esp ; Kernel stack points to return address

54 <1>

55 <1> ; save user registers

56 0000C593 1E <1> push ds

57 0000C594 06 <1> push es

58 0000C595 0FA0 <1> push fs

59 0000C597 0FA8 <1> push gs

60 0000C599 60 <1> pushad ; eax, ecx, edx, ebx, esp -before pushad-, ebp, esi, edi

61 <1> ;

62 <1> ; ESPACE = [ss:u.sp] - esp ; 4\*12 = 48 ; 17/09/2015 ; 06/06/2016

63 <1> ; (ESPACE is size of space in kernel stack

64 <1> ; for saving/restoring user registers.)

65 <1> ;

66 0000C59A 50 <1> push eax ; 01/07/2015

67 0000C59B 66B81000 <1> mov ax, KDATA

68 0000C59F 8ED8 <1> mov ds, ax

69 0000C5A1 8EC0 <1> mov es, ax

70 0000C5A3 8EE0 <1> mov fs, ax

71 0000C5A5 8EE8 <1> mov gs, ax

72 0000C5A7 A1[38580100] <1> mov eax, [k\_page\_dir]

73 0000C5AC 0F22D8 <1> mov cr3, eax

74 0000C5AF 58 <1> pop eax ; 01/07/2015

75 <1> ; 19/10/2015

76 0000C5B0 FC <1> cld

77 <1> ;

78 0000C5B1 FE05[5B030300] <1> inc byte [sysflg]

79 <1> ; incb sysflg / indicate a system routine is in progress

80 0000C5B7 FB <1> sti ; 18/01/2014

81 0000C5B8 0F85F39DFFFF <1> jnz panic ; 24/05/2013

82 <1> ; beq 1f

83 <1> ; jmp panic ; / called if trap inside system

84 <1> ;1:

85 <1> ; 17/03/2017

86 0000C5BE 80642438FE <1> and byte [esp+ESPACE+8], ~1 ; clear carry flag

87 <1>

88 <1> ; 16/04/2015

89 0000C5C3 A3[64030300] <1> mov [u.r0], eax

90 0000C5C8 8925[60030300] <1> mov [u.usp], esp ; kernel stack points to user's registers

91 <1>

92 <1> ; 13/01/2017 (TRDOS 386 Feaure only !)

93 0000C5CE 803D[D4030300]00 <1> cmp byte [u.t\_lock], 0 ; timer interrupt lock ?

94 0000C5D5 0F879D010000 <1> ja sysrele ; yes, sys release only !!!

95 <1>

96 <1> ; mov $s.syst+2,clockp

97 <1> ; mov r0,-(sp) / save user registers

98 <1> ; mov sp,u.r0 / pointer to bottom of users stack

99 <1> ; / in u.r0

100 <1> ; mov r1,-(sp)

101 <1> ; mov r2,-(sp)

102 <1> ; mov r3,-(sp)

103 <1> ; mov r4,-(sp)

104 <1> ; mov r5,-(sp)

105 <1> ; mov ac,-(sp) / "accumulator" register for extended

106 <1> ; / arithmetic unit

107 <1> ; mov mq,-(sp) / "multiplier quotient" register for the

108 <1> ; / extended arithmetic unit

109 <1> ; mov sc,-(sp) / "step count" register for the extended

110 <1> ; / arithmetic unit

111 <1> ; mov sp,u.sp / u.sp points to top of users stack

112 <1> ; mov 18.(sp),r0 / store pc in r0

113 <1> ; mov -(r0),r0 / sys inst in r0 10400xxx

114 <1> ; sub $sys,r0 / get xxx code

115 0000C5DB C1E002 <1> shl eax, 2

116 <1> ; asl r0 / multiply by 2 to jump indirect in bytes

117 0000C5DE 3DB8000000 <1> cmp eax, end\_of\_syscalls - syscalls

118 <1> ; cmp r0,$2f-1f / limit of table (35) exceeded

119 <1> ;jnb short badsys

120 <1> ; bhis badsys / yes, bad system call

121 0000C5E3 F5 <1> cmc

122 0000C5E4 9C <1> pushf

123 0000C5E5 50 <1> push eax

124 0000C5E6 8B2D[5C030300] <1> mov ebp, [u.sp] ; Kernel stack at the beginning of sys call

125 0000C5EC B0FE <1> mov al, 0FEh ; 11111110b

126 0000C5EE 1400 <1> adc al, 0 ; al = al + cf

127 0000C5F0 204508 <1> and [ebp+8], al ; flags (reset carry flag)

128 <1> ; bic $341,20.(sp) / set users processor priority to 0

129 <1> ; / and clear carry bit

130 0000C5F3 5D <1> pop ebp ; eax

131 0000C5F4 9D <1> popf

132 0000C5F5 0F8208020000 <1> jc badsys

133 0000C5FB A1[64030300] <1> mov eax, [u.r0]

134 <1> ; system call registers: EAX, EDX, ECX, EBX, ESI, EDI

135 0000C600 FFA5[06C60000] <1> jmp dword [ebp+syscalls]

136 <1> ; jmp \*1f(r0) / jump indirect thru table of addresses

137 <1> ; / to proper system routine.

138 <1> syscalls: ; 1:

139 <1> ; 31/12/2017

140 <1> ; 28/02/2017

141 <1> ; 20/02/2017

142 <1> ; 19/02/2017

143 <1> ; 15/10/2016

144 <1> ; 20/05/2016

145 <1> ; 19/05/2016

146 <1> ; 16/05/2016

147 <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)

148 <1> ; 21/09/2015

149 <1> ; 01/07/2015

150 <1> ; 16/04/2015 (32 bit address modification)

151 0000C606 [BDE60000] <1> dd sysver ; 0 ; Get TRDOS 386 version number (v2.0)

152 0000C60A [65C80000] <1> dd sysexit ; 1

153 0000C60E [3ACA0000] <1> dd sysfork ; 2

154 0000C612 [6DCE0000] <1> dd sysread ; 3

155 0000C616 [8CCE0000] <1> dd syswrite ; 4

156 0000C61A [23CC0000] <1> dd sysopen ; 5

157 0000C61E [44CE0000] <1> dd sysclose ; 6

158 0000C622 [BCC90000] <1> dd syswait ; 7

159 0000C626 [52CB0000] <1> dd syscreat ; 8

160 0000C62A [11F50000] <1> dd sysrename ; 9 ; TRDOS 386, Rename File (31/12/2017)

161 0000C62E [8CF00000] <1> dd sysdelete ; 10 ; TRDOS 386, Delete File (29/12/2017)

162 0000C632 [A0DA0000] <1> dd sysexec ; 11

163 0000C636 [B6F10000] <1> dd syschdir ; 12

164 0000C63A [7BF30000] <1> dd systime ; 13 ; TRDOS 386, Get Sys Date&Time (30/12/2017)

165 0000C63E [06CE0000] <1> dd sysmkdir ; 14

166 0000C642 [EAF10000] <1> dd syschmod ; 15 ; TRDOS 386, Change Attributes (30/12/2017)

167 0000C646 [F3F00000] <1> dd sysrmdir ; 16 ; TRDOS 386, Remove Directory (29/12/2017)

168 0000C64A [7BDD0000] <1> dd sysbreak ; 17

169 0000C64E [D0F20000] <1> dd sysdrive ; 18 ; TRDOS 386, Get/Set Current Drv (30/12/2017)

170 0000C652 [BCDD0000] <1> dd sysseek ; 19

171 0000C656 [CEDD0000] <1> dd systell ; 20

172 0000C65A [32F60000] <1> dd sysmem ; 21 ; TRDOS 386, Get Total&Free Mem (31/12/2017)

173 0000C65E [68F60000] <1> dd sysprompt ; 22 ; TRDOS 386, Change Cmd Prompt (31/12/2017)

174 0000C662 [AAF60000] <1> dd syspath ; 23 ; TRDOS 386, Get/Set Run Path (31/12/2017)

175 0000C666 [17F70000] <1> dd sysenv ; 24 ; TRDOS 386, Get/Set Env Vars (31/12/2017)

176 0000C66A [FCF30000] <1> dd sysstime ; 25 ; TRDOS 386, Set Sys Date&Time (30/12/2017)

177 0000C66E [34DE0000] <1> dd sysquit ; 26

178 0000C672 [28DE0000] <1> dd sysintr ; 27

179 0000C676 [1FF30000] <1> dd sysdir ; 28 ; TRDOS 386, Get Curr Drive&Dir (30/12/2017)

180 0000C67A [23CF0000] <1> dd sysemt ; 29

181 0000C67E [5AF30000] <1> dd sysldrvt ; 30 ; TRDOS 386, Get Logical DOS DDT (30/12/2017)

182 0000C682 [D4D00000] <1> dd sysvideo ; 31 ; TRDOS 386 Video Functions (16/05/2016)

183 0000C686 [E9000100] <1> dd sysaudio ; 32 ; TRDOS 386 Audio Functions (16/05/2016)

184 0000C68A [3CCF0000] <1> dd systimer ; 33 ; TRDOS 386 Timer Functions (18/05/2016)

185 0000C68E [75DE0000] <1> dd syssleep ; 34 ; Retro UNIX 8086 v1 feature only !

186 <1> ; 11/06/2014

187 0000C692 [A4DE0000] <1> dd sysmsg ; 35 ; Retro UNIX 386 v1 feature only !

188 <1> ; 01/07/2015

189 0000C696 [7BDF0000] <1> dd sysgeterr ; 36 ; Retro UNIX 386 v1 feature only !

190 <1> ; 21/09/2015 - get last error number

191 0000C69A [63F00000] <1> dd sysfpstat ; 37 ; TRDOS 386 FPU state option (28/02/2017)

192 0000C69E [CCE60000] <1> dd syspri ; 38 ; change priority - TRDOS 386 (20/05/2016)

193 0000C6A2 [78C70000] <1> dd sysrele ; 39 ; TRDOS 386 (19/05/2016) (0 -> 39)

194 0000C6A6 [FFE70000] <1> dd sysfff ; 40 ; Find First File - TRDOS 386 (15/10/2016)

195 0000C6AA [DEE80000] <1> dd sysfnf ; 41 ; Find Next File - TRDOS 386 (15/10/2016)

196 0000C6AE [4EEF0000] <1> dd sysalloc ; 42 ; Allocate contiguous memory block/pages

197 <1> ; TRDOS 386 (19/02/2017) DMA buff fuctions

198 0000C6B2 [0CF00000] <1> dd sysdalloc ; 43 ; Deallocate contiguous memory block/pages

199 <1> ; TRDOS 386 (19/02/2017) DMA buff fuctions

200 0000C6B6 [47F00000] <1> dd syscalbac ; 44 ; IRQ Callback and Signal Response Byte

201 <1> ; service setup - TRDOS 386 (20/02/2017)

202 <1> ; 28/08/2017 (20/08/2017)

203 0000C6BA [6D090100] <1> dd sysdma ; 45 ; TRDOS 386 - (ISA) DMA service

204 <1>

205 <1> end\_of\_syscalls:

206 <1>

207 <1> error:

208 <1> ; 18/05/2016

209 <1> ; 13/05/2016

210 <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)

211 <1> ; 16/04/2015 - 17/09/2015 (Retro UNIX 386 v1)

212 <1> ; 10/04/2013 - 07/08/2013 (Retro UNIX 8086 v1)

213 <1> ;

214 <1> ; 'error' merely sets the error bit off the processor status (c-bit)

215 <1> ; then falls right into the 'sysret', 'sysrele' return sequence.

216 <1> ;

217 <1> ; INPUTS -> none

218 <1> ; OUTPUTS ->

219 <1> ; processor status - carry (c) bit is set (means error)

220 <1> ;

221 <1> ; 26/05/2013 (Stack pointer must be reset here!

222 <1> ; Because, jumps to error procedure

223 <1> ; disrupts push-pop nesting balance)

224 <1> ;

225 0000C6BE 8B2D[5C030300] <1> mov ebp, [u.sp] ; interrupt (system call) return (iretd) address

226 0000C6C4 804D0801 <1> or byte [ebp+8], 1 ; set carry bit of flags register

227 <1> ; (system call will return with cf = 1)

228 <1> ; bis $1,20.(r1) / set c bit in processor status word below

229 <1> ; / users stack

230 <1> ; 17/09/2015

231 0000C6C8 83ED30 <1> sub ebp, ESPACE ; 48 ; total size of stack frame ('sysdefs.inc')

232 <1> ; for saving/restoring user registers

233 <1> ;cmp ebp, [u.usp]

234 <1> ;je short err0

235 0000C6CB 892D[60030300] <1> mov [u.usp], ebp

236 <1> ;err0:

237 <1> ; 01/09/2015

238 0000C6D1 8B25[60030300] <1> mov esp, [u.usp] ; Retro Unix 8086 v1 modification!

239 <1> ; 10/04/2013

240 <1> ; (If an I/O error occurs during disk I/O,

241 <1> ; related procedures will jump to 'error'

242 <1> ; procedure directly without returning to

243 <1> ; the caller procedure. So, stack pointer

244 <1> ; must be restored here.)

245 <1> ; 13/05/2016

246 <1> ; NOTE: (The last) error code is in 'u.error', it can be retrieved by

247 <1> ; 'get last error' system call later.

248 <1>

249 <1> ; 03/09/2015 - 09/06/2015 - 07/08/2013

250 0000C6D7 C605[C6030300]00 <1> mov byte [u.kcall], 0 ; namei\_r, mkdir\_w reset

251 <1>

252 <1> sysret: ; < return from system call>

253 <1> ; 01/03/2017

254 <1> ; 28/02/2017

255 <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)

256 <1> ; 16/04/2015 - 10/09/2015 (Retro UNIX 386 v1)

257 <1> ; 10/04/2013 - 23/02/2014 (Retro UNIX 8086 v1)

258 <1> ;

259 <1> ; 'sysret' first checks to see if process is about to be

260 <1> ; terminated (u.bsys). If it is, 'sysexit' is called.

261 <1> ; If not, following happens:

262 <1> ; 1) The user's stack pointer is restored.

263 <1> ; 2) r1=0 and 'iget' is called to see if last mentioned

264 <1> ; i-node has been modified. If it has, it is written out

265 <1> ; via 'ppoke'.

266 <1> ; 3) If the super block has been modified, it is written out

267 <1> ; via 'ppoke'.

268 <1> ; 4) If the dismountable file system's super block has been

269 <1> ; modified, it is written out to the specified device

270 <1> ; via 'ppoke'.

271 <1> ; 5) A check is made if user's time quantum (uquant) ran out

272 <1> ; during his execution. If so, 'tswap' is called to give

273 <1> ; another user a chance to run.

274 <1> ; 6) 'sysret' now goes into 'sysrele'.

275 <1> ; (See 'sysrele' for conclusion.)

276 <1> ;

277 <1> ; Calling sequence:

278 <1> ; jump table or 'br sysret'

279 <1> ; Arguments:

280 <1> ; -

281 <1> ; ...............................................................

282 <1> ;

283 <1> ; ((AX=r1 for 'iget' input))

284 <1> ;

285 0000C6DE 31C0 <1> xor eax, eax ; 28/02/2017

286 <1> sysret0: ; 29/07/2015 (eax = 0, jump from sysexec)

287 0000C6E0 FEC0 <1> inc al ; 04/05/2013

288 0000C6E2 3805[B2030300] <1> cmp [u.bsys], al ; 1

289 <1> ; tstb u.bsys / is a process about to be terminated because

290 0000C6E8 0F8377010000 <1> jnb sysexit ; 04/05/2013

291 <1> ; bne sysexit / of an error? yes, go to sysexit

292 <1> ;mov esp, [u.usp] ; 24/05/2013 (that is not needed here)

293 <1> ; mov u.sp,sp / no point stack to users stack

294 0000C6EE FEC8 <1> dec al ; mov ax, 0

295 <1> ; clr r1 / zero r1 to check last mentioned i-node

296 0000C6F0 E8CD300000 <1> call iget

297 <1> ; jsr r0,iget / if last mentioned i-node has been modified

298 <1> ; / it is written out

299 <1> ; 10/01/2017

300 <1> ; 09/01/2017

301 <1> ;sysrele: ; < release >

302 <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)

303 <1> ; 16/04/2015 - 14/10/2015 (Retro UNIX 386 v1)

304 <1> ; 10/04/2013 - 07/03/2014 (Retro UNIX 8086 v1)

305 <1> ;

306 <1> ; 'sysrele' first calls 'tswap' if the time quantum for a user is

307 <1> ; zero (see 'sysret'). It then restores the user's registers and

308 <1> ; turns off the system flag. It then checked to see if there is

309 <1> ; an interrupt from the user by calling 'isintr'. If there is,

310 <1> ; the output gets flashed (see isintr) and interrupt action is

311 <1> ; taken by a branch to 'intract'. If there is no interrupt from

312 <1> ; the user, a rti is made.

313 <1> ;

314 <1> ; Calling sequence:

315 <1> ; Fall through a 'bne' in 'sysret' & ?

316 <1> ; Arguments:

317 <1> ; -

318 <1> ; ...............................................................

319 <1> ;

320 <1> ; 23/02/2014 (swapret)

321 <1> ; 22/09/2013

322 <1> sysrel0: ;1:

323 0000C6F5 803D[A8030300]00 <1> cmp byte [u.quant], 0 ; 16/05/2013

324 <1> ; tstb uquant / is the time quantum 0?

325 0000C6FC 7705 <1> ja short swapret

326 <1> ; bne 1f / no, don't swap it out

327 <1> sysrelease: ; 07/12/2013 (jump from 'clock')

328 0000C6FE E8821E0000 <1> call tswap

329 <1> ; jsr r0,tswap / yes, swap it out

330 <1>

331 <1> ; Retro Unix 8086 v1 feature: return from 'swap' to 'swapret' address.

332 <1> swapret: ;1:

333 <1> ; 10/09/2015

334 <1> ; 01/09/2015

335 <1> ; 14/05/2015

336 <1> ; 16/04/2015 (Retro UNIX 386 v1 - 32 bit, pm modifications)

337 <1> ; 26/05/2013 (Retro UNIX 8086 v1)

338 <1> ; cli

339 <1> ; 24/07/2015

340 <1> ;

341 <1> ;; 'esp' must be already equal to '[u.usp]' here !

342 <1> ;; mov esp, [u.usp]

343 <1>

344 <1> ; 22/09/2013

345 0000C703 E8BB300000 <1> call isintr

346 <1> ; 20/10/2013

347 0000C708 7405 <1> jz short sysrel1

348 0000C70A E83F010000 <1> call intract

349 <1> ; jsr r0,isintr / is there an interrupt from the user

350 <1> ; br intract / yes, output gets flushed, take interrupt

351 <1> ; / action

352 <1> sysrel1:

353 0000C70F FA <1> cli ; 14/10/2015

354 <1> sysrel2:

355 <1> ; 28/02/2017

356 <1> ; Check if there is a (delayed) callback for current user/process

357 0000C710 A0[D7030300] <1> mov al, [u.irqwait]

358 0000C715 240F <1> and al, 0Fh ; is there a waiting IRQ callback service ?

359 0000C717 7444 <1> jz short sysrel8 ; no

360 <1>

361 <1> ; Set return to IRQ callback service and return from the service

362 0000C719 0FB6D8 <1> movzx ebx, al

363 0000C71C 883D[D7030300] <1> mov [u.irqwait], bh ; 0 ; reset

364 0000C722 8A9B[08160100] <1> mov bl, [ebx+IRQenum] ; (available) IRQ index +1 (1 to 9)

365 <1> ; 01/03/2017

366 0000C728 FECB <1> dec bl ; IRQ index number, 0 to 8

367 0000C72A 7831 <1> js short sysrel8 ; 0 -> FFh (not in use!?)

368 <1> ;

369 0000C72C A0[B3030300] <1> mov al, [u.uno] ; current process (user) number

370 0000C731 3883[6E6B0100] <1> cmp [ebx+IRQ.owner], al

371 0000C737 7524 <1> jne short sysrel8 ; it is not the current user/process !?

372 0000C739 F683[806B0100]01 <1> test byte [ebx+IRQ.method], 1 ; callback ?

373 0000C740 741B <1> jz short sysrel8 ; not a callback method !?

374 <1>

375 0000C742 8B93[926B0100] <1> mov edx, [ebx+IRQ.addr] ; IRQ callback service address (virtual)

376 0000C748 C605[D8030300]01 <1> mov byte [u.r\_lock], 1 ; IRQ callback service in progress flag

377 <1>

378 0000C74F E8D91E0000 <1> call wswap ; save user's registers & status

379 <1> ; (for return from IRQ callback service)

380 <1>

381 0000C754 8B2D[5C030300] <1> mov ebp, [u.sp]; kernel's stack, points to EIP (user)

382 0000C75A 895500 <1> mov [ebp], edx ; IRQ call back service address

383 <1> sysrel8:

384 0000C75D FE0D[5B030300] <1> dec byte [sysflg]

385 <1> ; decb sysflg / turn system flag off

386 <1>

387 0000C763 A1[B8030300] <1> mov eax, [u.pgdir]

388 0000C768 0F22D8 <1> mov cr3, eax ; 1st PDE points to Kernel Page Table 0 (1st 4 MB)

389 <1> ; (others are different than kernel page tables)

390 <1> ; 10/09/2015

391 0000C76B 61 <1> popad ; edi, esi, ebp, temp (icrement esp by 4), ebx, edx, ecx, eax

392 <1> ; mov (sp)+,sc / restore user registers

393 <1> ; mov (sp)+,mq

394 <1> ; mov (sp)+,ac

395 <1> ; mov (sp)+,r5

396 <1> ; mov (sp)+,r4

397 <1> ; mov (sp)+,r3

398 <1> ; mov (sp)+,r2

399 <1> ;

400 0000C76C A1[64030300] <1> mov eax, [u.r0] ; ((return value in EAX))

401 0000C771 0FA9 <1> pop gs

402 0000C773 0FA1 <1> pop fs

403 0000C775 07 <1> pop es

404 0000C776 1F <1> pop ds

405 <1> ;or word [esp+8], 200h ; 22/01/2017 ; force enabling interrupts

406 0000C777 CF <1> iretd

407 <1> ; rti / no, return from interrupt

408 <1>

409 <1> sysrele:

410 <1> ; 24/03/2017

411 <1> ; 28/02/2017

412 <1> ; 27/02/2017

413 <1> ; 29/01/2017

414 <1> ; 14/01/2017

415 <1> ; 13/01/2017

416 <1> ; 09/01/2017, 10/01/2017, 12/01/2017

417 <1> ; Major modification for TRDOS 386 (CallBack return)

418 <1> ;

419 <1> ; 'sysrele' system call restores previously saved

420 <1> ; registers and addresses of the process

421 <1> ; (Main purpose -in TRDOS 386- is to return from

422 <1> ; timer callback service routine in ring 3 -user mode-.)

423 <1> ;

424 <1> ; check if the process is in timer callback phase

425 0000C778 803D[D4030300]00 <1> cmp byte [u.t\_lock], 0 ; TIMER INT LOCK

426 <1> ;je short sysrel0 ; classic (Retro UNIX 386 type) sysrele

427 0000C77F 7734 <1> ja short sysrel3

428 <1> ; 27/02/2017

429 0000C781 803D[D8030300]00 <1> cmp byte [u.r\_lock], 0 ; IRQ callback lock

430 0000C788 0F8667FFFFFF <1> jna sysrel0 ; classic sysrele ; 24/03/2017

431 0000C78E E859000000 <1> call sysrel7

432 0000C793 803D[D8030300]00 <1> cmp byte [u.r\_lock], 0 ; IRQ callback service lock

433 0000C79A 7628 <1> jna short sysrel4

434 0000C79C C605[D8030300]00 <1> mov byte [u.r\_lock], 0 ; reset

435 <1> ;mov byte [u.irqwait], 0 ; reset ; 28/02/2017

436 0000C7A3 A0[D9030300] <1> mov al, [u.r\_mode]

437 0000C7A8 08C0 <1> or al, al

438 0000C7AA 7518 <1> jnz short sysrel4

439 0000C7AC FEC8 <1> dec al

440 0000C7AE A2[D9030300] <1> mov [u.r\_mode], al ; 0FFh ; not necessary !?

441 0000C7B3 EB32 <1> jmp short sysrel6

442 <1> sysrel3:

443 <1> ; 27/02/2017

444 0000C7B5 E832000000 <1> call sysrel7

445 <1> ; 14/01/2017

446 0000C7BA 28C0 <1> sub al, al

447 0000C7BC 3805[D4030300] <1> cmp [u.t\_lock], al ; 0 ; TIMER INT LOCK

448 0000C7C2 770E <1> ja short sysrel5 ; yes

449 <1> sysrel4:

450 <1> ; 29/01/2017

451 0000C7C4 8B44241C <1> mov eax, [esp+28] ; eax

452 0000C7C8 A3[64030300] <1> mov [u.r0], eax

453 0000C7CD E93EFFFFFF <1> jmp sysrel2

454 <1> sysrel5:

455 0000C7D2 A2[D4030300] <1> mov [u.t\_lock], al ; 0 ; reset

456 0000C7D7 A0[D5030300] <1> mov al, [u.t\_mode]

457 0000C7DC 20C0 <1> and al, al

458 <1> ;jnz short sysrel2 ; 0FFh ; user mode

459 0000C7DE 75E4 <1> jnz short sysrel4 ; 29/01/2017

460 0000C7E0 FEC8 <1> dec al

461 0000C7E2 A2[D5030300] <1> mov [u.t\_mode], al ; 0FFh ; not necessary !?

462 <1> sysrel6:

463 <1> ; cpu will continue from the interrupted sytem call addr

464 0000C7E7 61 <1> popad ; edi, esi, ebp, esp, ebx, edx, ecx, eax

465 0000C7E8 83C410 <1> add esp, 16 ; pass segment segisters: ds, es, fs, gs

466 0000C7EB CF <1> iretd ; eip, cs, eflags

467 <1>

468 <1> sysrel7:

469 0000C7EC 0FB61D[B3030300] <1> movzx ebx, byte [u.uno] ; current process number

470 0000C7F3 66C1E302 <1> shl bx, 2

471 <1> ;cmp [ebx+p.tcb-4], eax ; 0 ; is there callback address ?

472 <1> ;jna short sysrel0

473 <1> ; yes, reset callback address then restore process registers

474 <1> ;mov [ebx+p.tcb-4], eax ; 0 ; reset

475 0000C7F7 8B83[BC000300] <1> mov eax, [ebx+p.upage-4] ; UPAGE address

476 0000C7FD FA <1> cli ; disable interrupts till 'iretd'

477 0000C7FE E9621E0000 <1> jmp rswap ; restore process 'u' structure

478 <1>

479 <1> badsys:

480 <1> ; 25/12/2016

481 <1> ; 18/04/2016 (TRDOS 386 = TRDOS v2.0)

482 <1> ; 17/04/2011 (TRDOS v1.0, 'IFC.ASM')

483 <1> ; 03/02/2011 ('trdos\_ifc\_routine')

484 <1> ;

485 <1> ; 16/04/2015 (Retro UNIX 386 v1, 'badsys')

486 <1> ; (EIP, EAX values will be shown on screen with error message)

487 <1> ; (EIP = 'CD 40h' instruction address -INT 40h-)

488 <1> ; (EAX = Function number)

489 <1> ;

490 0000C803 FE05[B2030300] <1> inc byte [u.bsys]

491 <1> ;

492 0000C809 8B1D[5C030300] <1> mov ebx, [u.sp] ; esp at the beginning of 'sysent'

493 0000C80F 8B03 <1> mov eax, [ebx] ; EIP (return address, not 'INT 30h' address)

494 0000C811 83E802 <1> sub eax, 2 ; CDh, ##h

495 0000C814 E8F06AFFFF <1> call dwordtohex

496 0000C819 8915[E1130100] <1> mov [eip\_str], edx

497 0000C81F A3[E5130100] <1> mov [eip\_str+4], eax

498 0000C824 A1[64030300] <1> mov eax, [u.r0]

499 0000C829 E8DB6AFFFF <1> call dwordtohex

500 0000C82E 8915[D0130100] <1> mov [eax\_str], edx

501 0000C834 A3[D4130100] <1> mov [eax\_str+4], eax

502 <1>

503 0000C839 66C705[C5130100]34- <1> mov word [int\_num\_str], SYSCALL\_INT\_NUM ; 25/12/2016

503 0000C841 30 <1>

504 <1>

505 0000C842 BE[97130100] <1> mov esi, ifc\_msg ; "invalid funtion call !" msg (trdosk9.s)

506 0000C847 E8119BFFFF <1> call print\_msg

507 <1>

508 0000C84C EB17 <1> jmp sysexit

509 <1>

510 <1> intract: ; / interrupt action

511 <1> ; 14/10/2015

512 <1> ; 16/04/2015 (Retro UNIX 386 v1 - Beginning)

513 <1> ; 09/05/2013 - 07/12/2013 (Retro UNIX 8086 v1)

514 <1> ;

515 <1> ; Retro UNIX 8086 v1 modification !

516 <1> ; (Process/task switching and quit routine by using

517 <1> ; Retro UNIX 8086 v1 keyboard interrupt output.))

518 <1> ;

519 <1> ; input -> 'u.quit' (also value of 'u.intr' > 0)

520 <1> ; output -> If value of 'u.quit' = FFFFh ('ctrl+brk' sign)

521 <1> ; 'intract' will jump to 'sysexit'.

522 <1> ; Intract will return to the caller

523 <1> ; if value of 'u.quit' <> FFFFh.

524 <1> ; 14/10/2015

525 0000C84E FB <1> sti

526 <1> ; 07/12/2013

527 0000C84F 66FF05[AC030300] <1> inc word [u.quit]

528 0000C856 7408 <1> jz short intrct0 ; FFFFh -> 0

529 0000C858 66FF0D[AC030300] <1> dec word [u.quit]

530 <1> ; 16/04/2015

531 0000C85F C3 <1> retn

532 <1> intrct0:

533 0000C860 58 <1> pop eax ; call intract -> retn

534 <1> ;

535 0000C861 31C0 <1> xor eax, eax

536 0000C863 FEC0 <1> inc al ; mov ax, 1

537 <1> ;;;

538 <1> ; UNIX v1 original 'intract' routine...

539 <1> ; / interrupt action

540 <1> ;cmp \*(sp),$rti / are you in a clock interrupt?

541 <1> ; bne 1f / no, 1f

542 <1> ; cmp (sp)+,(sp)+ / pop clock pointer

543 <1> ; 1: / now in user area

544 <1> ; mov r1,-(sp) / save r1

545 <1> ; mov u.ttyp,r1

546 <1> ; / pointer to tty buffer in control-to r1

547 <1> ; cmpb 6(r1),$177

548 <1> ; / is the interrupt char equal to "del"

549 <1> ; beq 1f / yes, 1f

550 <1> ; clrb 6(r1)

551 <1> ; / no, clear the byte

552 <1> ; / (must be a quit character)

553 <1> ; mov (sp)+,r1 / restore r1

554 <1> ; clr u.quit / clear quit flag

555 <1> ; bis $20,2(sp)

556 <1> ; / set trace for quit (sets t bit of

557 <1> ; / ps-trace trap)

558 <1> ; rti ; / return from interrupt

559 <1> ; 1: / interrupt char = del

560 <1> ; clrb 6(r1) / clear the interrupt byte

561 <1> ; / in the buffer

562 <1> ; mov (sp)+,r1 / restore r1

563 <1> ; cmp u.intr,$core / should control be

564 <1> ; / transferred to loc core?

565 <1> ; blo 1f

566 <1> ; jmp \*u.intr / user to do rti yes,

567 <1> ; / transfer to loc core

568 <1> ; 1:

569 <1> ; sys 1 / exit

570 <1>

571 <1> sysexit: ; <terminate process>

572 <1> ; 14/11/2017

573 <1> ; 27/05/2017

574 <1> ; 10/04/2017

575 <1> ; 26/02/2017, 28/02/2017

576 <1> ; 02/01/2017, 23/01/2017

577 <1> ; 06/06/2016, 10/06/2016

578 <1> ; 19/05/2016, 23/05/2016

579 <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)

580 <1> ; 16/04/2015 - 01/09/2015 (Retro UNIX 386 v1)

581 <1> ; 19/04/2013 - 14/02/2014 (Retro UNIX 8086 v1)

582 <1> ;

583 <1> ; 'sysexit' terminates a process. First each file that

584 <1> ; the process has opened is closed by 'flose'. The process

585 <1> ; status is then set to unused. The 'p.pid' table is then

586 <1> ; searched to find children of the dying process. If any of

587 <1> ; children are zombies (died by not waited for), they are

588 <1> ; set free. The 'p.pid' table is then searched to find the

589 <1> ; dying process's parent. When the parent is found, it is

590 <1> ; checked to see if it is free or it is a zombie. If it is

591 <1> ; one of these, the dying process just dies. If it is waiting

592 <1> ; for a child process to die, it notified that it doesn't

593 <1> ; have to wait anymore by setting it's status from 2 to 1

594 <1> ; (waiting to active). It is awakened and put on runq by

595 <1> ; 'putlu'. The dying process enters a zombie state in which

596 <1> ; it will never be run again but stays around until a 'wait'

597 <1> ; is completed by it's parent process. If the parent is not

598 <1> ; found, process just dies. This means 'swap' is called with

599 <1> ; 'u.uno=0'. What this does is the 'wswap' is not called

600 <1> ; to write out the process and 'rswap' reads the new process

601 <1> ; over the one that dies..i.e., the dying process is

602 <1> ; overwritten and destroyed.

603 <1> ;

604 <1> ; Calling sequence:

605 <1> ; sysexit or conditional branch.

606 <1> ; Arguments:

607 <1> ; -

608 <1> ; ...............................................................

609 <1> ;

610 <1> ; Retro UNIX 8086 v1 modification:

611 <1> ; System call number (=1) is in EAX register.

612 <1> ;

613 <1> ; Other parameters are in EDX, EBX, ECX, ESI, EDI, EBP

614 <1> ; registers depending of function details.

615 <1> ;

616 <1> ; ('swap' procedure is mostly different than original UNIX v1.)

617 <1> ;

618 <1> ; / terminate process

619 <1> ; AX = 1

620 0000C865 6648 <1> dec ax ; 0

621 0000C867 66A3[AA030300] <1> mov [u.intr], ax ; 0

622 <1> ; clr u.intr / clear interrupt control word

623 <1> ; clr r1 / clear r1

624 <1> sysexit\_0:

625 <1> ; 23/01/2017

626 <1> ; 02/01/2017

627 <1> ; 10/06/2016

628 <1> ; 06/06/2016

629 <1> ; 23/05/2016

630 <1> ; 19/05/2016 - TRDOS 386 (TRDOS v2.0)

631 <1> ; Check and stop/clear timer event(s) of this (dying) process

632 <1> ; if there is.

633 <1>

634 <1> ; 02/01/2017

635 0000C86D FA <1> cli ; disable interrupts

636 <1> ; 23/01/2017 - reset timer frequency (to 18.2Hz)

637 0000C86E B036 <1> mov al, 00110110b ; 36h

638 0000C870 E643 <1> out 43h, al

639 0000C872 28C0 <1> sub al, al ; 0

640 0000C874 E640 <1> out 40h, al ; LB

641 0000C876 E640 <1> out 40h, al ; HB

642 <1> ;

643 0000C878 0FB61D[B3030300] <1> movzx ebx, byte [u.uno]

644 <1> ;mov bl, [u.uno] ; process number of dying process

645 0000C87F 3883[FF000300] <1> cmp byte [ebx+p.timer-1], al ; 0

646 0000C885 763A <1> jna short sysexit\_12 ; no timer events for this process

647 0000C887 8883[FF000300] <1> mov byte [ebx+p.timer-1], al ; 0 ; reset

648 <1> ;mov al, [timer\_events]

649 <1> ;or al, al

650 <1> ;jz short sysexit\_12 ; no timer events

651 <1> ;mov cl, al

652 0000C88D 8A0D[CF650100] <1> mov cl, [timer\_events] ; 14/11/2017

653 <1> ;cli ; disable interrupts

654 0000C893 B410 <1> mov ah, 16 ; number of available timer events

655 0000C895 BE[60040300] <1> mov esi, timer\_set ; beginning address of timer events

656 <1> sysexit\_7:

657 0000C89A 8A06 <1> mov al, [esi] ; process number (of timer event)

658 0000C89C 38D8 <1> cmp al, bl ; process number comparison

659 0000C89E 7411 <1> je short sysexit\_10

660 0000C8A0 20C0 <1> and al, al

661 0000C8A2 7404 <1> jz short sysexit\_9

662 <1> sysexit\_8:

663 0000C8A4 FEC9 <1> dec cl

664 0000C8A6 7416 <1> jz short sysexit\_11

665 <1> sysexit\_9:

666 0000C8A8 FECC <1> dec ah

667 0000C8AA 7415 <1> jz short sysexit\_12

668 0000C8AC 83C610 <1> add esi, 16

669 0000C8AF EBE9 <1> jmp short sysexit\_7

670 <1>

671 <1> sysexit\_10:

672 <1> ;mov byte [esi], 0

673 0000C8B1 66C7060000 <1> mov word [esi], 0

674 <1> ;mov dword [esi+12], 0

675 <1> ;

676 0000C8B6 FE0D[CF650100] <1> dec byte [timer\_events] ; 02/01/2017

677 <1> ;

678 0000C8BC EBE6 <1> jmp short sysexit\_8

679 <1>

680 <1> sysexit\_11:

681 0000C8BE 6629C0 <1> sub ax, ax ; 0 ; 26/02/2017

682 <1> sysexit\_12:

683 <1> ; 26/02/2017 (Unlink IRQ callbacks belong to the user)

684 0000C8C1 803D[D6030300]00 <1> cmp byte [u.irqc], 0 ; Count of IRQ callbacks

685 0000C8C8 7E2E <1> jng short sysexit\_16 ; zero or invalid

686 <1> ; 28/02/2017

687 <1> ; clear IRQ callback flags (for 'sysrele' and 'sysret')

688 0000C8CA A2[D7030300] <1> mov [u.irqwait], al ; 0 ; force to clear waiting flag

689 0000C8CF A2[D8030300] <1> mov [u.r\_lock], al ; 0 ; force to clear busy flag

690 0000C8D4 BE[6E6B0100] <1> mov esi, IRQ.owner

691 <1> sysexit\_13:

692 0000C8D9 AC <1> lodsb

693 0000C8DA 3A05[B3030300] <1> cmp al, [u.uno] ; owner = current user ?

694 0000C8E0 750C <1> jne short sysexit\_14

695 0000C8E2 C646FF00 <1> mov byte [esi-1], 0 ; owner = 0 : Free

696 0000C8E6 FE0D[D6030300] <1> dec byte [u.irqc]

697 0000C8EC 7408 <1> jz short sysexit\_15

698 <1> sysexit\_14:

699 0000C8EE 81FE[766B0100] <1> cmp esi, IRQ.owner + 8 ; the last IRQ index number ?

700 0000C8F4 76E3 <1> jna short sysexit\_13 ; no

701 <1> sysexit\_15:

702 0000C8F6 30C0 <1> xor al, al ; 0

703 <1> sysexit\_16: ; 2:

704 0000C8F8 FB <1> sti ; enable interrupts

705 <1> ;

706 <1> ; AX = 0

707 <1> sysexit\_1: ; 1:

708 <1> ; AX = File descriptor

709 <1> ; / r1 has file descriptor (index to u.fp list)

710 <1> ; / Search the whole list

711 0000C8F9 E89A130000 <1> call fclose

712 <1> ; jsr r0,fclose / close all files the process opened

713 <1> ;; ignore error return

714 <1> ; br .+2 / ignore error return

715 <1> ;inc ax

716 0000C8FE FEC0 <1> inc al

717 <1> ; inc r1 / increment file descriptor

718 <1> ;cmp ax, 10

719 0000C900 3C0A <1> cmp al, 10

720 <1> ; cmp r1,$10. / end of u.fp list?

721 0000C902 72F5 <1> jb short sysexit\_1

722 <1> ; blt 1b / no, go back

723 <1> ;movzx ebx, byte [u.uno]

724 0000C904 8A1D[B3030300] <1> mov bl, [u.uno] ; 02/01/2017

725 <1> ; movb u.uno,r1 / yes, move dying process's number to r1

726 0000C90A 88A3[AF000300] <1> mov [ebx+p.stat-1], ah ; 0, SFREE

727 <1> ; clrb p.stat-1(r1) / free the process

728 <1> ; 10/04/2017

729 0000C910 381D[E56B0100] <1> cmp [audio\_user], bl

730 0000C916 7518 <1> jne short sysexit\_17

731 <1> ; reset audio device (current) owner and 'initializated' flag

732 0000C918 883D[E56B0100] <1> mov [audio\_user], bh ; 0

733 <1> ; 27/05/2017

734 0000C91E 8B0D[D06B0100] <1> mov ecx, [audio\_buffer]

735 0000C924 09C9 <1> or ecx, ecx

736 0000C926 7408 <1> jz short sysexit\_17

737 <1> ; 'deallocate\_user\_pages' is not necessary in sysexit !!!

738 <1> ;push ebx

739 <1> ;mov ebx, ecx

740 <1> ;mov ecx, [audio\_buff\_size]

741 <1> ;call deallocate\_user\_pages

742 <1> ;; (Modified Registers -> EAX, EDX, ESI, EDI, EBX, ECX, EBP)

743 0000C928 29C9 <1> sub ecx, ecx

744 0000C92A 890D[D06B0100] <1> mov [audio\_buffer], ecx ; 0

745 <1> ;pop ebx

746 <1> sysexit\_17:

747 <1> ;shl bx, 1

748 0000C930 D0E3 <1> shl bl, 1

749 <1> ; asl r1 / use r1 for index into the below tables

750 0000C932 668B8B[1E000300] <1> mov cx, [ebx+p.pid-2]

751 <1> ; mov p.pid-2(r1),r3 / move dying process's name to r3

752 0000C939 668B93[3E000300] <1> mov dx, [ebx+p.ppid-2]

753 <1> ; mov p.ppid-2(r1),r4 / move its parents name to r4

754 <1> ; xor bx, bx ; 0

755 0000C940 30DB <1> xor bl, bl ; 0

756 <1> ; clr r2

757 0000C942 31F6 <1> xor esi, esi ; 0

758 <1> ; clr r5 / initialize reg

759 <1> sysexit\_2: ; 1:

760 <1> ; / find children of this dying process,

761 <1> ; / if they are zombies, free them

762 <1> ;add bx, 2

763 0000C944 80C302 <1> add bl, 2

764 <1> ; add $2,r2 / search parent process table

765 <1> ; / for dying process's name

766 0000C947 66398B[3E000300] <1> cmp [ebx+p.ppid-2], cx

767 <1> ; cmp p.ppid-2(r2),r3 / found it?

768 0000C94E 7513 <1> jne short sysexit\_4

769 <1> ; bne 3f / no

770 <1> ;shr bx, 1

771 0000C950 D0EB <1> shr bl, 1

772 <1> ; asr r2 / yes, it is a parent

773 0000C952 80BB[AF000300]03 <1> cmp byte [ebx+p.stat-1], 3 ; SZOMB

774 <1> ; cmpb p.stat-1(r2),$3 / is the child of this

775 <1> ; / dying process a zombie

776 0000C959 7506 <1> jne short sysexit\_3

777 <1> ; bne 2f / no

778 0000C95B 88A3[AF000300] <1> mov [ebx+p.stat-1], ah ; 0, SFREE

779 <1> ; clrb p.stat-1(r2) / yes, free the child process

780 <1> sysexit\_3: ; 2:

781 <1> ;shr bx, 1

782 0000C961 D0E3 <1> shl bl, 1

783 <1> ; asl r2

784 <1> sysexit\_4: ; 3:

785 <1> ; / search the process name table

786 <1> ; / for the dying process's parent

787 0000C963 663993[1E000300] <1> cmp [ebx+p.pid-2], dx

788 <1> ; cmp p.pid-2(r2),r4 / found it?

789 0000C96A 7502 <1> jne short sysexit\_5

790 <1> ; bne 3f / no

791 0000C96C 89DE <1> mov esi, ebx

792 <1> ; mov r2,r5 / yes, put index to p.pid table (parents

793 <1> ; / process # x2) in r5

794 <1> sysexit\_5: ; 3:

795 <1> ;cmp bx, nproc + nproc

796 0000C96E 80FB20 <1> cmp bl, nproc + nproc

797 <1> ; cmp r2,$nproc+nproc / has whole table been searched?

798 0000C971 72D1 <1> jb short sysexit\_2

799 <1> ; blt 1b / no, go back

800 <1> ; mov r5,r1 / yes, r1 now has parents process # x2

801 0000C973 21F6 <1> and esi, esi ; r5=r1

802 0000C975 7436 <1> jz short sysexit\_6

803 <1> ; beq 2f / no parent has been found.

804 <1> ; / The process just dies

805 0000C977 66D1EE <1> shr si, 1

806 <1> ; asr r1 / set up index to p.stat

807 0000C97A 8A86[AF000300] <1> mov al, [esi+p.stat-1]

808 <1> ; movb p.stat-1(r1),r2 / move status of parent to r2

809 0000C980 20C0 <1> and al, al

810 0000C982 7429 <1> jz short sysexit\_6

811 <1> ; beq 2f / if its been freed, 2f

812 0000C984 3C03 <1> cmp al, 3

813 <1> ; cmp r2,$3 / is parent a zombie?

814 0000C986 7425 <1> je short sysexit\_6

815 <1> ; beq 2f / yes, 2f

816 <1> ; BH = 0

817 0000C988 8A1D[B3030300] <1> mov bl, [u.uno]

818 <1> ; movb u.uno,r3 / move dying process's number to r3

819 0000C98E C683[AF000300]03 <1> mov byte [ebx+p.stat-1], 3 ; SZOMB

820 <1> ; movb $3,p.stat-1(r3) / make the process a zombie

821 0000C995 3C01 <1> cmp al, 1 ; SRUN

822 0000C997 7414 <1> je short sysexit\_6

823 <1> ;cmp al, 2

824 <1> ; cmp r2,$2 / is the parent waiting for

825 <1> ; / this child to die

826 <1> ;jne short sysexit\_6

827 <1> ; bne 2f / yes, notify parent not to wait any more

828 <1> ; p.stat = 2 --> waiting

829 <1> ; p.stat = 4 --> sleeping

830 0000C999 C686[AF000300]01 <1> mov byte [esi+p.stat-1], 1 ; SRUN

831 <1> ;dec byte [esi+p.stat-1]

832 <1> ; decb p.stat-1(r1) / awaken it by putting it (parent)

833 0000C9A0 6689F0 <1> mov ax, si ; r1 (process number in AL)

834 <1> ;

835 <1> ;mov ebx, runq + 4

836 <1> ; mov $runq+4,r2 / on the runq

837 0000C9A3 BB[54030300] <1> mov ebx, runq+2 ; normal run queue ; 02/01/2017

838 0000C9A8 E8F01C0000 <1> call putlu

839 <1> ; jsr r0, putlu

840 <1> sysexit\_6:

841 <1> ; / the process dies

842 0000C9AD C605[B3030300]00 <1> mov byte [u.uno], 0

843 <1> ; clrb u.uno / put zero as the process number,

844 <1> ; / so "swap" will

845 0000C9B4 E8E61B0000 <1> call swap

846 <1> ; jsr r0,swap / overwrite process with another process

847 <1>

848 <1> hlt\_sys:

849 <1> ;sti

850 <1> hlts0:

851 0000C9B9 F4 <1> hlt

852 0000C9BA EBFD <1> jmp short hlts0

853 <1> ; 0 / and thereby kill it; halt?

854 <1>

855 <1> syswait: ; < wait for a processs to die >

856 <1> ; 17/09/2015

857 <1> ; 02/09/2015

858 <1> ; 01/09/2015

859 <1> ; 16/04/2015 (Retro UNIX 386 v1 - Beginning)

860 <1> ; 24/05/2013 - 05/02/2014 (Retro UNIX 8086 v1)

861 <1> ;

862 <1> ; 'syswait' waits for a process die.

863 <1> ; It works in following way:

864 <1> ; 1) From the parent process number, the parent's

865 <1> ; process name is found. The p.ppid table of parent

866 <1> ; names is then searched for this process name.

867 <1> ; If a match occurs, r2 contains child's process

868 <1> ; number. The child status is checked to see if it is

869 <1> ; a zombie, i.e; dead but not waited for (p.stat=3)

870 <1> ; If it is, the child process is freed and it's name

871 <1> ; is put in (u.r0). A return is then made via 'sysret'.

872 <1> ; If the child is not a zombie, nothing happens and

873 <1> ; the search goes on through the p.ppid table until

874 <1> ; all processes are checked or a zombie is found.

875 <1> ; 2) If no zombies are found, a check is made to see if

876 <1> ; there are any children at all. If there are none,

877 <1> ; an error return is made. If there are, the parent's

878 <1> ; status is set to 2 (waiting for child to die),

879 <1> ; the parent is swapped out, and a branch to 'syswait'

880 <1> ; is made to wait on the next process.

881 <1> ;

882 <1> ; Calling sequence:

883 <1> ; ?

884 <1> ; Arguments:

885 <1> ; -

886 <1> ; Inputs: -

887 <1> ; Outputs: if zombie found, it's name put in u.r0.

888 <1> ; ...............................................................

889 <1> ;

890 <1>

891 <1> ; / wait for a process to die

892 <1>

893 <1> syswait\_0:

894 0000C9BC 0FB61D[B3030300] <1> movzx ebx, byte [u.uno] ; 01/09/2015

895 <1> ; movb u.uno,r1 / put parents process number in r1

896 0000C9C3 D0E3 <1> shl bl, 1

897 <1> ;shl bx, 1

898 <1> ; asl r1 / x2 to get index into p.pid table

899 0000C9C5 668B83[1E000300] <1> mov ax, [ebx+p.pid-2]

900 <1> ; mov p.pid-2(r1),r1 / get the name of this process

901 0000C9CC 31F6 <1> xor esi, esi

902 <1> ; clr r2

903 0000C9CE 31C9 <1> xor ecx, ecx ; 30/10/2013

904 <1> ;xor cl, cl

905 <1> ; clr r3 / initialize reg 3

906 <1> syswait\_1: ; 1:

907 0000C9D0 6683C602 <1> add si, 2

908 <1> ; add $2,r2 / use r2 for index into p.ppid table

909 <1> ; / search table of parent processes

910 <1> ; / for this process name

911 0000C9D4 663B86[3E000300] <1> cmp ax, [esi+p.ppid-2]

912 <1> ; cmp p.ppid-2(r2),r1 / r2 will contain the childs

913 <1> ; / process number

914 0000C9DB 7535 <1> jne short syswait\_3

915 <1> ;bne 3f / branch if no match of parent process name

916 <1> ;inc cx

917 0000C9DD FEC1 <1> inc cl

918 <1> ;inc r3 / yes, a match, r3 indicates number of children

919 0000C9DF 66D1EE <1> shr si, 1

920 <1> ; asr r2 / r2/2 to get index to p.stat table

921 <1> ; The possible states ('p.stat' values) of a process are:

922 <1> ; 0 = free or unused

923 <1> ; 1 = active

924 <1> ; 2 = waiting for a child process to die

925 <1> ; 3 = terminated, but not yet waited for (zombie).

926 0000C9E2 80BE[AF000300]03 <1> cmp byte [esi+p.stat-1], 3 ; SZOMB, 05/02/2014

927 <1> ; cmpb p.stat-1(r2),$3 / is the child process a zombie?

928 0000C9E9 7524 <1> jne short syswait\_2

929 <1> ; bne 2f / no, skip it

930 0000C9EB 88BE[AF000300] <1> mov [esi+p.stat-1], bh ; 0

931 <1> ; clrb p.stat-1(r2) / yes, free it

932 0000C9F1 66D1E6 <1> shl si, 1

933 <1> ; asl r2 / r2x2 to get index into p.pid table

934 0000C9F4 0FB786[1E000300] <1> movzx eax, word [esi+p.pid-2]

935 0000C9FB A3[64030300] <1> mov [u.r0], eax

936 <1> ; mov p.pid-2(r2),\*u.r0

937 <1> ; / put childs process name in (u.r0)

938 <1> ;

939 <1> ; Retro UNIX 386 v1 modification ! (17/09/2015)

940 <1> ;

941 <1> ; Parent process ID -p.ppid- field (of the child process)

942 <1> ; must be cleared in order to prevent infinitive 'syswait'

943 <1> ; system call loop from the application/program if it calls

944 <1> ; 'syswait' again (mistakenly) while there is not a zombie

945 <1> ; or running child process to wait. ('forktest.s', 17/09/2015)

946 <1> ;

947 <1> ; Note: syswait will return with error if there is not a

948 <1> ; zombie or running process to wait.

949 <1> ;

950 0000CA00 6629C0 <1> sub ax, ax

951 0000CA03 668986[3E000300] <1> mov [esi+p.ppid-2], ax ; 0 ; 17/09/2015

952 0000CA0A E9D1FCFFFF <1> jmp sysret0 ; ax = 0

953 <1> ;

954 <1> ;jmp sysret

955 <1> ; br sysret1 / return cause child is dead

956 <1> syswait\_2: ; 2:

957 0000CA0F 66D1E6 <1> shl si, 1

958 <1> ; asl r2 / r2x2 to get index into p.ppid table

959 <1> syswait\_3: ; 3:

960 0000CA12 6683FE20 <1> cmp si, nproc+nproc

961 <1> ; cmp r2,$nproc+nproc / have all processes been checked?

962 0000CA16 72B8 <1> jb short syswait\_1

963 <1> ; blt 1b / no, continue search

964 <1> ;and cx, cx

965 0000CA18 20C9 <1> and cl, cl

966 <1> ; tst r3 / one gets here if there are no children

967 <1> ; / or children that are still active

968 <1> ; 30/10/2013

969 0000CA1A 750B <1> jnz short syswait\_4

970 <1> ;jz error

971 <1> ; beq error1 / there are no children, error

972 0000CA1C 890D[64030300] <1> mov [u.r0], ecx ; 0

973 0000CA22 E997FCFFFF <1> jmp error

974 <1> syswait\_4:

975 0000CA27 8A1D[B3030300] <1> mov bl, [u.uno]

976 <1> ; movb u.uno,r1 / there are children so put

977 <1> ; / parent process number in r1

978 0000CA2D FE83[AF000300] <1> inc byte [ebx+p.stat-1] ; 2, SWAIT, 05/02/2014

979 <1> ; incb p.stat-1(r1) / it is waiting for

980 <1> ; / other children to die

981 <1> ; 04/11/2013

982 0000CA33 E8671B0000 <1> call swap

983 <1> ; jsr r0,swap / swap it out, because it's waiting

984 0000CA38 EB82 <1> jmp syswait\_0

985 <1> ; br syswait / wait on next process

986 <1>

987 <1> sysfork: ; < create a new process >

988 <1> ; 02/01/2017 (TRDOS 386 modification)

989 <1> ; 04/09/2015, 18/05/2015

990 <1> ; 28/08/2015, 01/09/2015, 02/09/2015

991 <1> ; 09/05/2015, 10/05/2015, 14/05/2015

992 <1> ; 06/05/2015 (Retro UNIX 386 v1 - Beginning)

993 <1> ; 24/05/2013 - 14/02/2014 (Retro UNIX 8086 v1)

994 <1> ;

995 <1> ; 'sysfork' creates a new process. This process is referred

996 <1> ; to as the child process. This new process core image is

997 <1> ; a copy of that of the caller of 'sysfork'. The only

998 <1> ; distinction is the return location and the fact that (u.r0)

999 <1> ; in the old process (parent) contains the process id (p.pid)

1000 <1> ; of the new process (child). This id is used by 'syswait'.

1001 <1> ; 'sysfork' works in the following manner:

1002 <1> ; 1) The process status table (p.stat) is searched to find

1003 <1> ; a process number that is unused. If none are found

1004 <1> ; an error occurs.

1005 <1> ; 2) when one is found, it becomes the child process number

1006 <1> ; and it's status (p.stat) is set to active.

1007 <1> ; 3) If the parent had a control tty, the interrupt

1008 <1> ; character in that tty buffer is cleared.

1009 <1> ; 4) The child process is put on the lowest priority run

1010 <1> ; queue via 'putlu'.

1011 <1> ; 5) A new process name is gotten from 'mpid' (actually

1012 <1> ; it is a unique number) and is put in the child's unique

1013 <1> ; identifier; process id (p.pid).

1014 <1> ; 6) The process name of the parent is then obtained and

1015 <1> ; placed in the unique identifier of the parent process

1016 <1> ; name is then put in 'u.r0'.

1017 <1> ; 7) The child process is then written out on disk by

1018 <1> ; 'wswap',i.e., the parent process is copied onto disk

1019 <1> ; and the child is born. (The child process is written

1020 <1> ; out on disk/drum with 'u.uno' being the child process

1021 <1> ; number.)

1022 <1> ; 8) The parent process number is then restored to 'u.uno'.

1023 <1> ; 9) The child process name is put in 'u.r0'.

1024 <1> ; 10) The pc on the stack sp + 18 is incremented by 2 to

1025 <1> ; create the return address for the parent process.

1026 <1> ; 11) The 'u.fp' list as then searched to see what files

1027 <1> ; the parent has opened. For each file the parent has

1028 <1> ; opened, the corresponding 'fsp' entry must be updated

1029 <1> ; to indicate that the child process also has opened

1030 <1> ; the file. A branch to 'sysret' is then made.

1031 <1> ;

1032 <1> ; Calling sequence:

1033 <1> ; from shell ?

1034 <1> ; Arguments:

1035 <1> ; -

1036 <1> ; Inputs: -

1037 <1> ; Outputs: \*u.r0 - child process name

1038 <1> ; ...............................................................

1039 <1> ;

1040 <1> ; Retro UNIX 8086 v1 modification:

1041 <1> ; AX = r0 = PID (>0) (at the return of 'sysfork')

1042 <1> ; = process id of child a parent process returns

1043 <1> ; = process id of parent when a child process returns

1044 <1> ;

1045 <1> ; In original UNIX v1, sysfork is called and returns as

1046 <1> ; in following manner: (with an example: c library, fork)

1047 <1> ;

1048 <1> ; 1:

1049 <1> ; sys fork

1050 <1> ; br 1f / child process returns here

1051 <1> ; bes 2f / parent process returns here

1052 <1> ; / pid of new process in r0

1053 <1> ; rts pc

1054 <1> ; 2: / parent process condionally branches here

1055 <1> ; mov $-1,r0 / pid = -1 means error return

1056 <1> ; rts pc

1057 <1> ;

1058 <1> ; 1: / child process brances here

1059 <1> ; clr r0 / pid = 0 in child process

1060 <1> ; rts pc

1061 <1> ;

1062 <1> ; In UNIX v7x86 (386) by Robert Nordier (1999)

1063 <1> ; // pid = fork();

1064 <1> ; //

1065 <1> ; // pid == 0 in child process;

1066 <1> ; // pid == -1 means error return

1067 <1> ; // in child,

1068 <1> ; // parents id is in par\_uid if needed

1069 <1> ;

1070 <1> ; \_fork:

1071 <1> ; mov $.fork,eax

1072 <1> ; int $0x30

1073 <1> ; jmp 1f

1074 <1> ; jnc 2f

1075 <1> ; jmp cerror

1076 <1> ; 1:

1077 <1> ; mov eax,\_par\_uid

1078 <1> ; xor eax,eax

1079 <1> ; 2:

1080 <1> ; ret

1081 <1> ;

1082 <1> ; In Retro UNIX 8086 v1,

1083 <1> ; 'sysfork' returns in following manner:

1084 <1> ;

1085 <1> ; mov ax, sys\_fork

1086 <1> ; mov bx, offset @f ; routine for child

1087 <1> ; int 20h

1088 <1> ; jc error

1089 <1> ;

1090 <1> ; ; Routine for parent process here (just after 'jc')

1091 <1> ; mov word ptr [pid\_of\_child], ax

1092 <1> ; jmp next\_routine\_for\_parent

1093 <1> ;

1094 <1> ; @@: ; routine for child process here

1095 <1> ; ....

1096 <1> ; NOTE: 'sysfork' returns to specified offset

1097 <1> ; for child process by using BX input.

1098 <1> ; (at first, parent process will return then

1099 <1> ; child process will return -after swapped in-

1100 <1> ; 'syswait' is needed in parent process

1101 <1> ; if return from child process will be waited for.)

1102 <1> ;

1103 <1>

1104 <1> ; / create a new process

1105 <1> ; EBX = return address for child process

1106 <1> ; (Retro UNIX 8086 v1 modification !)

1107 0000CA3A 31F6 <1> xor esi, esi

1108 <1> ; clr r1

1109 <1> sysfork\_1: ; 1: / search p.stat table for unused process number

1110 0000CA3C 46 <1> inc esi

1111 <1> ; inc r1

1112 0000CA3D 80BE[AF000300]00 <1> cmp byte [esi+p.stat-1], 0 ; SFREE, 05/02/2014

1113 <1> ; tstb p.stat-1(r1) / is process active, unused, dead

1114 0000CA44 760B <1> jna short sysfork\_2

1115 <1> ; beq 1f / it's unused so branch

1116 0000CA46 6683FE10 <1> cmp si, nproc

1117 <1> ; cmp r1,$nproc / all processes checked

1118 0000CA4A 72F0 <1> jb short sysfork\_1

1119 <1> ; blt 1b / no, branch back

1120 <1> ;

1121 <1> ; Retro UNIX 8086 v1. modification:

1122 <1> ; Parent process returns from 'sysfork' to address

1123 <1> ; which is just after 'sysfork' system call in parent

1124 <1> ; process. Child process returns to address which is put

1125 <1> ; in BX register by parent process for 'sysfork'.

1126 <1> ;

1127 <1> ;add $2,18.(sp) / add 2 to pc when trap occured, points

1128 <1> ; / to old process return

1129 <1> ; br error1 / no room for a new process

1130 0000CA4C E96DFCFFFF <1> jmp error

1131 <1> sysfork\_2: ; 1:

1132 0000CA51 E82481FFFF <1> call allocate\_page

1133 0000CA56 0F8262FCFFFF <1> jc error

1134 0000CA5C 50 <1> push eax ; UPAGE (user structure page) address

1135 <1> ; Retro UNIX 386 v1 modification!

1136 0000CA5D E82783FFFF <1> call duplicate\_page\_dir

1137 <1> ; EAX = New page directory

1138 0000CA62 730B <1> jnc short sysfork\_3

1139 0000CA64 58 <1> pop eax ; UPAGE (user structure page) address

1140 0000CA65 E8EE82FFFF <1> call deallocate\_page

1141 0000CA6A E94FFCFFFF <1> jmp error

1142 <1> sysfork\_3:

1143 <1> ; Retro UNIX 386 v1 modification !

1144 0000CA6F 56 <1> push esi

1145 0000CA70 E8B81B0000 <1> call wswap ; save current user (u) structure, user registers

1146 <1> ; and interrupt return components (for IRET)

1147 0000CA75 8705[B8030300] <1> xchg eax, [u.pgdir] ; page directory of the child process

1148 0000CA7B A3[BC030300] <1> mov [u.ppgdir], eax ; page directory of the parent process

1149 0000CA80 5E <1> pop esi

1150 0000CA81 58 <1> pop eax ; UPAGE (user structure page) address

1151 <1> ; [u.usp] = esp

1152 0000CA82 89F7 <1> mov edi, esi

1153 0000CA84 66C1E702 <1> shl di, 2

1154 0000CA88 8987[BC000300] <1> mov [edi+p.upage-4], eax ; memory page for 'user' struct

1155 0000CA8E A3[B4030300] <1> mov [u.upage], eax ; memory page for 'user' struct (child)

1156 <1> ; 28/08/2015

1157 0000CA93 0FB605[B3030300] <1> movzx eax, byte [u.uno] ; parent process number

1158 <1> ; movb u.uno,-(sp) / save parent process number

1159 0000CA9A 89C7 <1> mov edi, eax

1160 0000CA9C 50 <1> push eax ; \*\*

1161 0000CA9D 8A87[7F000300] <1> mov al, [edi+p.ttyc-1] ; console tty (parent)

1162 <1> ; 18/09/2015

1163 <1> ;mov [esi+p.ttyc-1], al ; set child's console tty

1164 <1> ;mov [esi+p.waitc-1], ah ; 0 ; reset child's wait channel

1165 0000CAA3 668986[7F000300] <1> mov [esi+p.ttyc-1], ax ; al - set child's console tty

1166 <1> ; ah - reset child's wait channel

1167 0000CAAA 89F0 <1> mov eax, esi

1168 0000CAAC A2[B3030300] <1> mov [u.uno], al ; child process number

1169 <1> ;movb r1,u.uno / set child process number to r1

1170 0000CAB1 FE86[AF000300] <1> inc byte [esi+p.stat-1] ; 1, SRUN, 05/02/2014

1171 <1> ; incb p.stat-1(r1) / set p.stat entry for child

1172 <1> ; / process to active status

1173 <1> ; mov u.ttyp,r2 / put pointer to parent process'

1174 <1> ; / control tty buffer in r2

1175 <1> ; beq 2f / branch, if no such tty assigned

1176 <1> ; clrb 6(r2) / clear interrupt character in tty buffer

1177 <1> ; 2:

1178 0000CAB7 53 <1> push ebx ; \* return address for the child process

1179 <1> ; \* Retro UNIX 8086 v1 feature only !

1180 <1> ; (Retro UNIX 8086 v1 modification!)

1181 <1> ; mov $runq+4,r2

1182 0000CAB8 BB[54030300] <1> mov ebx, runq+2 ; normal run queue ; 02/01/2017

1183 0000CABD E8DB1B0000 <1> call putlu

1184 <1> ; jsr r0,putlu / put child process on lowest priority

1185 <1> ; / run queue

1186 0000CAC2 66D1E6 <1> shl si, 1

1187 <1> ; asl r1 / multiply r1 by 2 to get index

1188 <1> ; / into p.pid table

1189 0000CAC5 66FF05[4E030300] <1> inc word [mpid]

1190 <1> ; inc mpid / increment m.pid; get a new process name

1191 0000CACC 66A1[4E030300] <1> mov ax, [mpid]

1192 0000CAD2 668986[1E000300] <1> mov [esi+p.pid-2], ax

1193 <1> ;mov mpid,p.pid-2(r1) / put new process name

1194 <1> ; / in child process' name slot

1195 0000CAD9 5A <1> pop edx ; \* return address for the child process

1196 <1> ; \* Retro UNIX 8086 v1 feature only !

1197 0000CADA 5B <1> pop ebx ; \*\*

1198 <1> ;mov ebx, [esp] ; \*\* parent process number

1199 <1> ; movb (sp),r2 / put parent process number in r2

1200 0000CADB 66D1E3 <1> shl bx, 1

1201 <1> ;asl r2 / multiply by 2 to get index into below tables

1202 <1> ;movzx eax, word [ebx+p.pid-2]

1203 0000CADE 668B83[1E000300] <1> mov ax, [ebx+p.pid-2]

1204 <1> ; mov p.pid-2(r2),r2 / get process name of parent

1205 <1> ; / process

1206 0000CAE5 668986[3E000300] <1> mov [esi+p.ppid-2], ax

1207 <1> ; mov r2,p.ppid-2(r1) / put parent process name

1208 <1> ; / in parent process slot for child

1209 0000CAEC A3[64030300] <1> mov [u.r0], eax

1210 <1> ; mov r2,\*u.r0 / put parent process name on stack

1211 <1> ; / at location where r0 was saved

1212 0000CAF1 8B2D[5C030300] <1> mov ebp, [u.sp] ; points to return address (EIP for IRET)

1213 0000CAF7 895500 <1> mov [ebp], edx ; \*, CS:EIP -> EIP

1214 <1> ; \* return address for the child process

1215 <1> ; mov $sysret1,-(sp) /

1216 <1> ; mov sp,u.usp / contents of sp at the time when

1217 <1> ; / user is swapped out

1218 <1> ; mov $sstack,sp / point sp to swapping stack space

1219 <1> ; 04/09/2015 - 01/09/2015

1220 <1> ; [u.usp] = esp

1221 0000CAFA 68[DEC60000] <1> push sysret ; \*\*\*

1222 0000CAFF 8925[60030300] <1> mov [u.usp], esp ; points to 'sysret' address (\*\*\*)

1223 <1> ; (for child process)

1224 0000CB05 31C0 <1> xor eax, eax

1225 0000CB07 66A3[94030300] <1> mov [u.ttyp], ax ; 0

1226 <1> ;

1227 0000CB0D E81B1B0000 <1> call wswap ; Retro UNIX 8086 v1 modification !

1228 <1> ;jsr r0,wswap / put child process out on drum

1229 <1> ;jsr r0,unpack / unpack user stack

1230 <1> ;mov u.usp,sp / restore user stack pointer

1231 <1> ; tst (sp)+ / bump stack pointer

1232 <1> ; Retro UNIX 386 v1 modification !

1233 0000CB12 58 <1> pop eax ; \*\*\*

1234 0000CB13 66D1E3 <1> shl bx, 1

1235 0000CB16 8B83[BC000300] <1> mov eax, [ebx+p.upage-4] ; UPAGE address ; 14/05/2015

1236 0000CB1C E8441B0000 <1> call rswap ; restore parent process 'u' structure,

1237 <1> ; registers and return address (for IRET)

1238 <1> ;movb (sp)+,u.uno / put parent process number in u.uno

1239 0000CB21 0FB705[4E030300] <1> movzx eax, word [mpid]

1240 0000CB28 A3[64030300] <1> mov [u.r0], eax

1241 <1> ; mov mpid,\*u.r0 / put child process name on stack

1242 <1> ; / where r0 was saved

1243 <1> ; add $2,18.(sp) / add 2 to pc on stack; gives parent

1244 <1> ; / process return

1245 <1> ;xor ebx, ebx

1246 0000CB2D 31F6 <1> xor esi, esi

1247 <1> ;clr r1

1248 <1> sysfork\_4: ; 1: / search u.fp list to find the files

1249 <1> ; / opened by the parent process

1250 <1> ; 01/09/2015

1251 <1> ;xor bh, bh

1252 <1> ;mov bl, [esi+u.fp]

1253 0000CB2F 8A86[6A030300] <1> mov al, [esi+u.fp]

1254 <1> ; movb u.fp(r1),r2 / get an open file for this process

1255 <1> ;or bl, bl

1256 0000CB35 08C0 <1> or al, al

1257 0000CB37 740D <1> jz short sysfork\_5

1258 <1> ; beq 2f / file has not been opened by parent,

1259 <1> ; / so branch

1260 0000CB39 B40A <1> mov ah, 10 ; Retro UNIX 386 v1 fsp structure size = 10 bytes

1261 0000CB3B F6E4 <1> mul ah

1262 <1> ;movzx ebx, ax

1263 0000CB3D 6689C3 <1> mov bx, ax

1264 <1> ;shl bx, 3

1265 <1> ; asl r2 / multiply by 8

1266 <1> ; asl r2 / to get index into fsp table

1267 <1> ; asl r2

1268 0000CB40 FE83[4E010300] <1> inc byte [ebx+fsp-2]

1269 <1> ; incb fsp-2(r2) / increment number of processes

1270 <1> ; / using file, because child will now be

1271 <1> ; / using this file

1272 <1> sysfork\_5: ; 2:

1273 0000CB46 46 <1> inc esi

1274 <1> ; inc r1 / get next open file

1275 0000CB47 6683FE0A <1> cmp si, 10

1276 <1> ; cmp r1,$10. / 10. files is the maximum number which

1277 <1> ; / can be opened

1278 0000CB4B 72E2 <1> jb short sysfork\_4

1279 <1> ; blt 1b / check next entry

1280 0000CB4D E98CFBFFFF <1> jmp sysret

1281 <1> ; br sysret1

1282 <1>

1283 <1> syscreat: ; < create file >

1284 <1> ; 13/11/2017

1285 <1> ; 27/10/2016

1286 <1> ; 25/10/2016, 26/10/2016

1287 <1> ; 15/10/2016, 16/10/2016, 17/10/2016

1288 <1> ; 10/10/2016 (TRDOS 386 = TRDOS v2.0)

1289 <1> ; -derived from INT\_21H.ASM-

1290 <1> ; ("loc\_INT21h\_create\_file")

1291 <1> ; 10/07/2011 (12/03/2011)

1292 <1> ; INT 21h Function AH = 3Ch

1293 <1> ; Create File

1294 <1> ; INPUT

1295 <1> ; CX = Attributes

1296 <1> ; DS:DX= Address of zero terminaned path name

1297 <1> ;

1298 <1> ; 27/12/2015 (Retro UNIX 386 v1.1)

1299 <1> ; 14/05/2015 (Retro UNIX 386 v1 - Beginning)

1300 <1> ; 27/05/2013 (Retro UNIX 8086 v1)

1301 <1> ;

1302 <1> ; 'syscreat' called with two arguments; name and mode.

1303 <1> ; u.namep points to name of the file and mode is put

1304 <1> ; on the stack. 'namei' is called to get i-number of the file.

1305 <1> ; If the file aready exists, it's mode and owner remain

1306 <1> ; unchanged, but it is truncated to zero length. If the file

1307 <1> ; did not exist, an i-node is created with the new mode via

1308 <1> ; 'maknod' whether or not the file already existed, it is

1309 <1> ; open for writing. The fsp table is then searched for a free

1310 <1> ; entry. When a free entry is found, proper data is placed

1311 <1> ; in it and the number of this entry is put in the u.fp list.

1312 <1> ; The index to the u.fp (also know as the file descriptor)

1313 <1> ; is put in the user's r0.

1314 <1> ;

1315 <1> ; Calling sequence:

1316 <1> ; syscreate; name; mode

1317 <1> ; Arguments:

1318 <1> ; name - name of the file to be created

1319 <1> ; mode - mode of the file to be created

1320 <1> ; Inputs: (arguments)

1321 <1> ; Outputs: \*u.r0 - index to u.fp list

1322 <1> ; (the file descriptor of new file)

1323 <1> ; ...............................................................

1324 <1> ;

1325 <1> ; Retro UNIX 8086 v1 modification:

1326 <1> ; 'syscreate' system call has two arguments; so,

1327 <1> ; \* 1st argument, name is pointed to by BX register

1328 <1> ; \* 2nd argument, mode is in CX register

1329 <1> ;

1330 <1> ; AX register (will be restored via 'u.r0') will return

1331 <1> ; to the user with the file descriptor/number

1332 <1> ; (index to u.fp list).

1333 <1> ;

1334 <1> ;call arg2

1335 <1> ; \* name - 'u.namep' points to address of file/path name

1336 <1> ; in the user's program segment ('u.segmnt')

1337 <1> ; with offset in BX register (as sysopen argument 1).

1338 <1> ; \* mode - sysopen argument 2 is in CX register

1339 <1> ; which is on top of stack.

1340 <1> ;

1341 <1> ; TRDOS 386 (10/10/2016)

1342 <1> ;

1343 <1> ; INPUT ->

1344 <1> ; CL = File Attributes

1345 <1> ; bit 0 (1) - Read only file (R)

1346 <1> ; bit 1 (1) - Hidden file (H)

1347 <1> ; bit 2 (1) - System file (R)

1348 <1> ; bit 3 (1) - Volume label/name (V)

1349 <1> ; bit 4 (1) - Subdirectory (D)

1350 <1> ; bit 5 (1) - File has been archived (A)

1351 <1> ; EBX = Pointer to filename (ASCIIZ) -path-

1352 <1> ;

1353 <1> ; OUTPUT ->

1354 <1> ; eax = File/Device Handle/Number (index) (AL)

1355 <1> ; cf = 1 -> Error code in AL

1356 <1> ;

1357 <1> ; Modified Registers: EAX (at the return of system call)

1358 <1> ;

1359 <1> ; Note: If the file is existing and it has not any one

1360 <1> ; of S,H,R,V,D attributes, it will be truncated

1361 <1> ; to zero length; otherwise, access error will be

1362 <1> ; returned.

1363 <1>

1364 <1> sysmkdir\_0:

1365 0000CB52 F6C108 <1> test cl, 08h ; Volume name

1366 0000CB55 740A <1> jz short syscreat\_0

1367 <1>

1368 <1> ; Volume name or long name creation

1369 <1> ; is not permitted (in TRDOS 386)!

1370 0000CB57 B80B000000 <1> mov eax, ERR\_FILE\_ACCESS ; 11 ; 'permission denied !'

1371 0000CB5C E926020000 <1> jmp sysopen\_dev\_err

1372 <1>

1373 <1> syscreat\_0:

1374 <1> ;mov [u.namep], ebx

1375 0000CB61 51 <1> push ecx

1376 0000CB62 89DE <1> mov esi, ebx

1377 <1> ; file name is forced, change directory as temporary

1378 <1> ;mov ax, 1

1379 <1> ;mov [FFF\_Valid], ah ; 0 ; reset ; 17/10/2016

1380 <1> ;call set\_working\_path

1381 0000CB64 E892300000 <1> call set\_working\_path\_x ; 17/10/2016

1382 0000CB69 0F82D7000000 <1> jc syscreat\_err

1383 <1>

1384 <1> ; 16/10/2016

1385 0000CB6F 803D[F3650100]00 <1> cmp byte [SWP\_inv\_fname], 0

1386 0000CB76 776C <1> ja short syscreat\_inv\_fname ; invalid file name !

1387 <1>

1388 <1> ; Here, we have a valid path and also a valid file name

1389 <1> ; (Working dir has been changed if the path

1390 <1> ; -file name string- had contained a dir name.)

1391 <1>

1392 0000CB78 6631C0 <1> xor ax, ax

1393 <1> ;mov esi, FindFile\_Name

1394 0000CB7B E8E3B6FFFF <1> call find\_first\_file

1395 0000CB80 59 <1> pop ecx

1396 <1> ; ESI = Directory Entry (FindFile\_DirEntry) Location

1397 <1> ; EDI = Directory Buffer Directory Entry Location

1398 <1> ; EAX = File Size

1399 <1> ; BL = Attributes of The File/Directory

1400 <1> ; BH = Long Name Yes/No Status (>0 is YES)

1401 <1> ; DX > 0 : Ambiguous filename chars are used

1402 0000CB81 7269 <1> jc short syscreat\_1 ; file not found (the good!)

1403 <1> ; or another error (the bad')

1404 <1>

1405 <1> ; (& the uggly!) truncate file to zero length before open

1406 <1>

1407 <1> ;'\*' and '?' already checked at 'set\_working\_path' stage

1408 <1> ;and dx, dx

1409 <1> ;jnz short sysmkdir\_err ; permission denied

1410 <1> ; invalid filename chars

1411 <1>

1412 <1> ;test cl, 10h ; subdirectory ?

1413 <1> ;jnz short sysmkdir\_err

1414 <1>

1415 <1> ; BL = File Attributes:

1416 <1> ; bit 0 (1) - Read only file (R)

1417 <1> ; bit 1 (1) - Hidden file (H)

1418 <1> ; bit 2 (1) - System file (R)

1419 <1> ; bit 3 (1) - Volume label/name (V)

1420 <1> ; bit 4 (1) - Subdirectory (D)

1421 <1> ; bit 5 (1) - File has been archived

1422 <1>

1423 <1> ; \* existing directory must not be truncated

1424 <1> ; (we don't know it is empty or not, at this stage)

1425 <1> ; \* existing volume name (or a long name) can not be

1426 <1> ; re-created or truncated by 'syscreat'

1427 <1> ; \* A file with S, H, R attributes must not be truncated

1428 <1> ; (change attributes to normal, if you need truncate it)

1429 <1>

1430 0000CB83 F6C31F <1> test bl, 00011111b ; check attributes of existing file

1431 0000CB86 754E <1> jnz short sysmkdir\_err

1432 <1>

1433 <1> ;; normal file, OK to continue...

1434 <1>

1435 <1> ; ESI = FindFile\_DirEntry

1436 0000CB88 668B4614 <1> mov ax, [esi+DirEntry\_FstClusHI] ; 20

1437 0000CB8C C1E010 <1> shl eax, 16 ; 13/11/2017

1438 0000CB8F 668B461A <1> mov ax, [esi+DirEntry\_FstClusLO] ; 26

1439 <1> ; EAX = First cluster to be truncated/unlinked

1440 0000CB93 57 <1> push edi

1441 0000CB94 51 <1> push ecx

1442 0000CB95 BE00010900 <1> mov esi, Logical\_DOSDisks

1443 0000CB9A 29C9 <1> sub ecx, ecx

1444 0000CB9C 8A2D[FE580100] <1> mov ch, [Current\_Drv]

1445 0000CBA2 01CE <1> add esi, ecx

1446 <1> ; ESI = Logical dos drive description table address

1447 0000CBA4 E8C9F7FFFF <1> call truncate\_cluster\_chain

1448 0000CBA9 59 <1> pop ecx

1449 0000CBAA 5F <1> pop edi

1450 0000CBAB 7230 <1> jc short syscreate\_truncate\_err

1451 <1>

1452 <1> ; 26/10/2016

1453 <1> ; EDI = Directory entry address in directory buffer

1454 <1> ; Update directory entry

1455 0000CBAD E848DCFFFF <1> call convert\_current\_date\_time

1456 <1> ; OUTPUT -> DX = Date in dos dir entry format

1457 <1> ; AX = Time in dos dir entry format

1458 0000CBB2 66894716 <1> mov [edi+DirEntry\_WrtTime], ax

1459 0000CBB6 66895718 <1> mov [edi+DirEntry\_WrtDate], dx

1460 0000CBBA 66895712 <1> mov [edi+DirEntry\_LastAccDate], dx

1461 0000CBBE 31C0 <1> xor eax, eax ; file size = 0

1462 0000CBC0 89471C <1> mov [edi+DirEntry\_FileSize], eax ; 0

1463 0000CBC3 C605[28610100]02 <1> mov byte [DirBuff\_ValidData], 2 ; data changed sign

1464 0000CBCA BE[F4620100] <1> mov esi, FindFile\_DirEntry

1465 0000CBCF B201 <1> mov dl, 1 ; open file for writing

1466 0000CBD1 E9AA000000 <1> jmp sysopen\_2

1467 <1>

1468 <1> sysmkdir\_err:

1469 <1> ; 1 = write, 2 = read & write, >2 = invalid

1470 0000CBD6 B80B000000 <1> mov eax, ERR\_FILE\_ACCESS ; 11 ; 'permission denied !'

1471 0000CBDB EB73 <1> jmp short sysopen\_err

1472 <1>

1473 <1> syscreate\_truncate\_err:

1474 0000CBDD B812000000 <1> mov eax, ERR\_DRV\_WRITE ; 18 ; 'disk write error !'

1475 0000CBE2 EB6C <1> jmp short sysopen\_err

1476 <1>

1477 <1> syscreat\_inv\_fname: ; invalid file name chars

1478 <1> ; 16/10/2016

1479 0000CBE4 B81A000000 <1> mov eax, ERR\_INV\_FILE\_NAME ; 26 ; invalid file name chars

1480 0000CBE9 59 <1> pop ecx

1481 0000CBEA EB64 <1> jmp sysopen\_err

1482 <1>

1483 <1> syscreat\_1:

1484 <1> ; Error code in EAX

1485 0000CBEC 3C02 <1> cmp al, 02h ; 'File not found' error

1486 0000CBEE 7560 <1> jne sysopen\_err

1487 <1>

1488 0000CBF0 F6C110 <1> test cl, 10h ; Directory

1489 0000CBF3 0F852C020000 <1> jnz sysmkdir\_2

1490 <1>

1491 <1> syscreat\_2:

1492 0000CBF9 BE[E4620100] <1> mov esi, FindFile\_Name

1493 <1> ;xor edx, edx

1494 0000CBFE 31C0 <1> xor eax, eax ; File Size = 0

1495 0000CC00 31DB <1> xor ebx, ebx

1496 0000CC02 4B <1> dec ebx ; FFFFFFFFh -> create empty file

1497 <1> ; (only for FAT fs)

1498 <1> ; CL = File Attributes

1499 0000CC03 E8F8EBFFFF <1> call create\_file

1500 0000CC08 7246 <1> jc sysopen\_err

1501 <1> ; EAX = New file's first cluster

1502 <1> ; ESI = Logical Dos Drv Descr. Table Addr.

1503 <1> ; EBX = offset CreateFile\_Size

1504 <1> ; ECX = Sectors per cluster (<256)

1505 <1> ; EDX = Directory entry index/number (<65536)

1506 <1> ; 26/10/2016

1507 <1> ;mov esi, Directory\_Buffer

1508 <1> ;shl dx, 5 ; \*32

1509 <1> ;add esi, edx

1510 <1> ;; esi = directory entry address in directory buffer

1511 <1>

1512 <1> ; Here, directory entry has been created but last

1513 <1> ; modification date & time of the parent dir has not

1514 <1> ; been updated, yet!

1515 <1> ; (Note: Directory and FAT buffers have been updated...)

1516 <1>

1517 0000CC0A E824DDFFFF <1> call update\_parent\_dir\_lmdt ; now, it is OK too!

1518 <1>

1519 <1> ; 25/10/2016

1520 0000CC0F 66B80018 <1> mov ax, 1800h

1521 0000CC13 BE[E4620100] <1> mov esi, FindFile\_Name

1522 0000CC18 E846B6FFFF <1> call find\_first\_file

1523 0000CC1D 7231 <1> jc short sysopen\_err

1524 <1>

1525 <1> ; Only possible error after here is

1526 <1> ; "too many open files !" error.

1527 <1> ;

1528 <1> ; If "syscreat" will return with that error,

1529 <1> ; (the file has been created but it could not be opened)

1530 <1> ; the user must retry to open this file again

1531 <1> ; or must close another file before using

1532 <1> ; "sysopen" system call.

1533 <1>

1534 0000CC1F B201 <1> mov dl, 1 ; open file for writing

1535 <1> ; ESI = Directory Entry (FindFile\_DirEntry) Location

1536 <1> ; EAX = File Size (= 0)

1537 0000CC21 EB5D <1> jmp short sysopen\_2

1538 <1>

1539 <1> sysopen: ;<open file>

1540 <1> ; 26/10/2016

1541 <1> ; 24/10/2016

1542 <1> ; 17/10/2016

1543 <1> ; 15/10/2016

1544 <1> ; 06/10/2016, 07/10/2016, 08/10/2016

1545 <1> ; 05/10/2016 (TRDOS 386 = TRDOS v2.0)

1546 <1> ; -derived from INT\_21H.ASM-

1547 <1> ; ("loc\_INT21h\_open\_file")

1548 <1> ; 26/02/2011

1549 <1> ; INT 21h Function AH = 3Dh

1550 <1> ; Open File

1551 <1> ; INPUT

1552 <1> ; AL= File Access Value

1553 <1> ; 0- Open for reading

1554 <1> ; 1- Open for writing

1555 <1> ; 2- Open for reading and writing

1556 <1> ; DS:DX= Pointer to filename (ASCIIZ)

1557 <1> ;

1558 <1> ; 14/05/2015 (Retro UNIX 386 v1 - Beginning)

1559 <1> ; 22/05/2013 - 27/05/2013 (Retro UNIX 8086 v1)

1560 <1> ;

1561 <1> ; 'sysopen' opens a file in following manner:

1562 <1> ; 1) The second argument in a sysopen says whether to

1563 <1> ; open the file ro read (0) or write (>0).

1564 <1> ; 2) I-node of the particular file is obtained via 'namei'.

1565 <1> ; 3) The file is opened by 'iopen'.

1566 <1> ; 4) Next housekeeping is performed on the fsp table

1567 <1> ; and the user's open file list - u.fp.

1568 <1> ; a) u.fp and fsp are scanned for the next available slot.

1569 <1> ; b) An entry for the file is created in the fsp table.

1570 <1> ; c) The number of this entry is put on u.fp list.

1571 <1> ; d) The file descriptor index to u.fp list is pointed

1572 <1> ; to by u.r0.

1573 <1> ;

1574 <1> ; Calling sequence:

1575 <1> ; sysopen; name; mode

1576 <1> ; Arguments:

1577 <1> ; name - file name or path name

1578 <1> ; mode - 0 to open for reading

1579 <1> ; 1 to open for writing

1580 <1> ; Inputs: (arguments)

1581 <1> ; Outputs: \*u.r0 - index to u.fp list (the file descriptor)

1582 <1> ; is put into r0's location on the stack.

1583 <1> ; ...............................................................

1584 <1> ;

1585 <1> ; Retro UNIX 8086 v1 modification:

1586 <1> ; 'sysopen' system call has two arguments; so,

1587 <1> ; \* 1st argument, name is pointed to by BX register

1588 <1> ; \* 2nd argument, mode is in CX register

1589 <1> ;

1590 <1> ; AX register (will be restored via 'u.r0') will return

1591 <1> ; to the user with the file descriptor/number

1592 <1> ; (index to u.fp list).

1593 <1> ;

1594 <1> ;call arg2

1595 <1> ; \* name - 'u.namep' points to address of file/path name

1596 <1> ; in the user's program segment ('u.segmnt')

1597 <1> ; with offset in BX register (as sysopen argument 1).

1598 <1> ; \* mode - sysopen argument 2 is in CX register

1599 <1> ; which is on top of stack.

1600 <1> ;

1601 <1> ; jsr r0,arg2 / get sys args into u.namep and on stack

1602 <1> ;

1603 <1> ; system call registers: ebx, ecx (through 'sysenter')

1604 <1> ;

1605 <1> ; TRDOS 386 (05/10/2016)

1606 <1> ;

1607 <1> ; INPUT ->

1608 <1> ; CL = File Access Value (Open Mode)

1609 <1> ; 0 - Open file for reading

1610 <1> ; 1 - Open file for writing

1611 <1> ; 2 - Open device for reading

1612 <1> ; 3 - Open device for writing

1613 <1> ; EBX = Pointer to filename/devicename (ASCIIZ)

1614 <1> ; OUTPUT ->

1615 <1> ; eax = File/Device Handle/Number (index) (AL)

1616 <1> ; cf = 1 -> Error code in AL

1617 <1> ;

1618 <1> ; Modified Registers: EAX (at the return of system call)

1619 <1> ;

1620 <1>

1621 0000CC23 80F901 <1> cmp cl, 1 ; read file (0), write file (1)

1622 0000CC26 7614 <1> jna short sysopen\_0

1623 <1>

1624 0000CC28 80F903 <1> cmp cl, 3

1625 0000CC2B 0F8640010000 <1> jna sysopen\_device

1626 <1>

1627 <1> ; Invalid access code

1628 0000CC31 B817000000 <1> mov eax, ERR\_INV\_PARAMETER

1629 0000CC36 0F874B010000 <1> ja sysopen\_dev\_err

1630 <1>

1631 <1> sysopen\_0:

1632 <1> ;mov [u.namep], ebx

1633 0000CC3C 51 <1> push ecx

1634 0000CC3D 89DE <1> mov esi, ebx

1635 <1> ; file name is forced, change directory as temporary

1636 <1> ;mov ax, 1

1637 <1> ;mov [FFF\_Valid], ah ; 0 ; reset ; 17/10/2016

1638 <1> ;call set\_working\_path

1639 0000CC3F E8B72F0000 <1> call set\_working\_path\_x ; 17/10/2016

1640 0000CC44 731E <1> jnc short sysopen\_1

1641 <1>

1642 <1> syscreat\_err: ; ecx = file attributes (for 'syscreat')

1643 0000CC46 59 <1> pop ecx ; open mode

1644 0000CC47 21C0 <1> and eax, eax ; 0 -> Bad Path!

1645 0000CC49 7505 <1> jnz short sysopen\_err

1646 <1> ; eax = 0

1647 0000CC4B B80C000000 <1> mov eax, ERR\_DIR\_NOT\_FOUND ; Directory not found !

1648 <1> sysopen\_err:

1649 0000CC50 A3[64030300] <1> mov [u.r0], eax

1650 0000CC55 A3[C8030300] <1> mov [u.error], eax

1651 0000CC5A E871300000 <1> call reset\_working\_path

1652 0000CC5F E95AFAFFFF <1> jmp error

1653 <1>

1654 <1> sysopen\_1:

1655 <1> ;mov esi, FindFile\_Name

1656 0000CC64 66B80018 <1> mov ax, 1800h ; Only files

1657 0000CC68 E8F6B5FFFF <1> call find\_first\_file

1658 0000CC6D 5A <1> pop edx

1659 0000CC6E 72E0 <1> jc short sysopen\_err ; eax = 2 (File not found !)

1660 <1>

1661 <1> ; check\_open\_file\_attr\_access\_code

1662 <1>

1663 0000CC70 F6C307 <1> test bl, 7 ; system, hidden, readonly

1664 0000CC73 740B <1> jz short sysopen\_2

1665 <1>

1666 0000CC75 20D2 <1> and dl, dl ; 0 = read mode

1667 0000CC77 7407 <1> jz short sysopen\_2

1668 <1>

1669 <1> ; 1 = write, 2 = read & write, >2 = invalid

1670 0000CC79 B80B000000 <1> mov eax, ERR\_FILE\_ACCESS ; 11 = 'permission denied !'

1671 0000CC7E EBD0 <1> jmp short sysopen\_err

1672 <1>

1673 <1> sysopen\_2:

1674 <1> ; esi = Directory Entry (FindFile\_DirEntry) Location

1675 0000CC80 89F3 <1> mov ebx, esi

1676 0000CC82 31F6 <1> xor esi, esi ; 0

1677 0000CC84 31FF <1> xor edi, edi ; 0

1678 <1> sysopen\_3: ; scan the list of entries in fsp table

1679 0000CC86 80BE[6A030300]00 <1> cmp byte [esi+u.fp], 0

1680 0000CC8D 760F <1> jna short sysopen\_4 ; empty slot

1681 0000CC8F 6646 <1> inc si

1682 0000CC91 6683FE0A <1> cmp si, 10

1683 0000CC95 72EF <1> jb short sysopen\_3

1684 <1> toomanyf:

1685 0000CC97 B80D000000 <1> mov eax, ERR\_TOO\_MANY\_FILES ; too many open files !

1686 0000CC9C EBB2 <1> jmp short sysopen\_err

1687 <1>

1688 <1> sysopen\_4:

1689 0000CC9E 80BF[62690100]00 <1> cmp byte [edi+OF\_MODE], 0 ; Scan open files table

1690 0000CCA5 760A <1> jna short sysopen\_5

1691 0000CCA7 6647 <1> inc di

1692 0000CCA9 6683FF0A <1> cmp di, OPENFILES ; max. number of open files (=10)

1693 0000CCAD 72EF <1> jb short sysopen\_4

1694 0000CCAF EBE6 <1> jmp short toomanyf

1695 <1>

1696 <1> sysopen\_5:

1697 0000CCB1 FEC2 <1> inc dl

1698 0000CCB3 8897[62690100] <1> mov [edi+OF\_MODE], dl

1699 0000CCB9 8A15[A2620100] <1> mov dl, [FindFile\_Drv]

1700 0000CCBF 8897[58690100] <1> mov [edi+OF\_DRIVE], dl ; Logical DOS drive number

1701 0000CCC5 66C1E702 <1> shl di, 2 ; \*4 (dword offset)

1702 <1>

1703 0000CCC9 8987[A8690100] <1> mov [edi+OF\_SIZE], eax ; File size in bytes

1704 <1>

1705 0000CCCF 668B4314 <1> mov ax, [ebx+DirEntry\_FstClusHI]

1706 0000CCD3 C1E010 <1> shl eax, 16

1707 0000CCD6 668B431A <1> mov ax, [ebx+DirEntry\_FstClusLO]

1708 0000CCDA 8987[30690100] <1> mov [edi+OF\_FCLUSTER], eax ; First cluster

1709 0000CCE0 8987[486A0100] <1> mov [edi+OF\_CCLUSTER], eax ; Current cluster

1710 <1>

1711 0000CCE6 31DB <1> xor ebx, ebx

1712 0000CCE8 899F[80690100] <1> mov [edi+OF\_POINTER], ebx ; offset pointer (0)

1713 0000CCEE 899F[706A0100] <1> mov [edi+OF\_CCINDEX], ebx ; cluster index (0)

1714 <1>

1715 0000CCF4 A1[14630100] <1> mov eax, [FindFile\_DirFirstCluster]

1716 0000CCF9 8987[D0690100] <1> mov [edi+OF\_DIRFCLUSTER], eax

1717 <1>

1718 0000CCFF A1[18630100] <1> mov eax, [FindFile\_DirCluster]

1719 0000CD04 8987[F8690100] <1> mov [edi+OF\_DIRCLUSTER], eax

1720 <1>

1721 <1> ; Get (& Save) Volume ID

1722 <1> ; Important for files of removable drives

1723 <1> ; (In order to check the drive has same volume/disk)

1724 0000CD0A 88D7 <1> mov bh, dl

1725 0000CD0C 81C300010900 <1> add ebx, Logical\_DOSDisks

1726 0000CD12 8A4303 <1> mov al, [ebx+LD\_FATType]

1727 0000CD15 3C01 <1> cmp al, 1

1728 0000CD17 7209 <1> jb short sysopen\_6\_fs

1729 0000CD19 3C02 <1> cmp al, 2

1730 0000CD1B 770A <1> ja short sysopen\_6\_fat32

1731 <1> sysopen\_6\_fat:

1732 0000CD1D 8B432D <1> mov eax, [ebx+LD\_BPB+VolumeID]

1733 0000CD20 EB08 <1> jmp short sysopen\_7

1734 <1> sysopen\_6\_fs:

1735 0000CD22 8B4328 <1> mov eax, [ebx+LD\_FS\_VolumeSerial]

1736 0000CD25 EB03 <1> jmp short sysopen\_7

1737 <1> sysopen\_6\_fat32:

1738 0000CD27 8B4349 <1> mov eax, [ebx+LD\_BPB+FAT32\_VolID]

1739 <1> sysopen\_7:

1740 0000CD2A A3[F4580100] <1> mov [Current\_VolSerial], eax

1741 <1>

1742 0000CD2F 8987[206A0100] <1> mov [edi+OF\_VOLUMEID], eax

1743 <1>

1744 <1> ; 24/10/2016

1745 0000CD35 66D1EF <1> shr di, 1 ; 4/2, word offset

1746 0000CD38 668B1D[1C630100] <1> mov bx, [FindFile\_DirEntryNumber]

1747 0000CD3F 66899F[986A0100] <1> mov [edi+OF\_DIRENTRY], bx

1748 <1>

1749 0000CD46 31D2 <1> xor edx, edx

1750 <1> ;shr di, 2 ; /4 (byte offset)

1751 0000CD48 66D1EF <1> shr di, 1 ; 2/2, byte offset

1752 0000CD4B 8897[76690100] <1> mov byte [edi+OF\_OPENCOUNT], dl ; 0

1753 0000CD51 8897[6C690100] <1> mov byte [edi+OF\_STATUS], dl ; 0

1754 <1>

1755 0000CD57 89FB <1> mov ebx, edi

1756 0000CD59 FEC3 <1> inc bl

1757 <1>

1758 0000CD5B 889E[6A030300] <1> mov [esi+u.fp], bl ; Open File Entry Number

1759 0000CD61 8935[64030300] <1> mov [u.r0], esi ; move index to u.fp list

1760 <1> ; into eax on stack

1761 <1>

1762 0000CD67 E8642F0000 <1> call reset\_working\_path

1763 <1>

1764 0000CD6C E96DF9FFFF <1> jmp sysret

1765 <1>

1766 <1> ; (Retro UNIX 386 v1.0)

1767 <1> ; 'fsp' table (10 bytes/entry)

1768 <1> ; bit 15 bit 0

1769 <1> ; ---|-------------------------------------------

1770 <1> ; r/w| i-number of open file

1771 <1> ; ---|-------------------------------------------

1772 <1> ; device number

1773 <1> ; -----------------------------------------------

1774 <1> ; offset pointer, r/w pointer to file (bit 0-15)

1775 <1> ; -----------------------------------------------

1776 <1> ; offset pointer, r/w pointer to file (bit 16-31)

1777 <1> ; ----------------------|------------------------

1778 <1> ; flag that says file | number of processes

1779 <1> ; has been deleted | that have file open

1780 <1> ; ----------------------|------------------------

1781 <1>

1782 <1> sysopen\_device:

1783 <1> ; 15/10/2016

1784 <1> ; 08/10/2016

1785 <1> ; 07/10/2016 (TRDOS 386 = TRDOS v2.0)

1786 0000CD71 51 <1> push ecx ; open mode

1787 0000CD72 89E5 <1> mov ebp, esp

1788 0000CD74 B910000000 <1> mov ecx, 16 ; transfer length = 16 bytes

1789 0000CD79 29CC <1> sub esp, ecx

1790 0000CD7B 89E7 <1> mov edi, esp ; destination address

1791 0000CD7D 89DE <1> mov esi, ebx ; dev name in user's memory space

1792 0000CD7F E83F1A0000 <1> call transfer\_from\_user\_buffer

1793 0000CD84 7310 <1> jnc short sysopen\_dev\_0

1794 <1> ; eax = ERR\_OUT\_OF\_MEMORY = 4 = ERR\_MINOR\_IM

1795 0000CD86 59 <1> pop ecx

1796 <1> sysopen\_dev\_err:

1797 0000CD87 A3[64030300] <1> mov [u.r0], eax

1798 0000CD8C A3[C8030300] <1> mov [u.error], eax

1799 0000CD91 E928F9FFFF <1> jmp error

1800 <1> sysopen\_dev\_0:

1801 0000CD96 89FE <1> mov esi, edi ; Device name addr (max. 16 bytes, ASCIIZ)

1802 <1> ; for example: "tty, TTY, /dev/tty"

1803 0000CD98 E8DB310000 <1> call get\_device\_number

1804 0000CD9D 89EC <1> mov esp, ebp

1805 0000CD9F 59 <1> pop ecx

1806 0000CDA0 7307 <1> jnc short sysopen\_dev\_1

1807 0000CDA2 B818000000 <1> mov eax, ERR\_INV\_DEV\_NAME ; 24 ; 'invalid device name !'

1808 0000CDA7 EBDE <1> jmp short sysopen\_dev\_err

1809 <1> sysopen\_dev\_1:

1810 <1> ; eax = Device Number (AL)

1811 <1> ; cl = Open mode (2 = device read, 3 = device write)

1812 0000CDA9 31DB <1> xor ebx, ebx ; 0

1813 <1> sysopen\_dev\_2: ; scan the list of entries

1814 0000CDAB 389B[6A030300] <1> cmp [ebx+u.fp], bl ; 0

1815 0000CDB1 760E <1> jna short sysopen\_dev\_3 ; empty slot

1816 0000CDB3 FEC3 <1> inc bl

1817 0000CDB5 80FB0A <1> cmp bl, 10

1818 0000CDB8 72F1 <1> jb short sysopen\_dev\_2

1819 <1> ;

1820 0000CDBA B80D000000 <1> mov eax, ERR\_TOO\_MANY\_FILES ; too many open files !

1821 0000CDBF EBC6 <1> jmp short sysopen\_dev\_err

1822 <1> sysopen\_dev\_3:

1823 0000CDC1 891D[64030300] <1> mov [u.r0], ebx ; File/Device index/handle/descriptor

1824 <1> ; eax = device number (entry offset)

1825 0000CDC7 8AA8[F4660100] <1> mov ch, [eax+DEV\_ACCESS] ; bit 0 = accessable by users

1826 <1> ; bit 1 = read access perm

1827 <1> ; bit 2 = write access perm

1828 <1> ; bit 3 = IOCTL permit to users

1829 <1> ; bit 4 = block device if set

1830 <1> ; bit 5 = 16 bit or 1024 byte

1831 <1> ; bit 6 = 32 bit or 2048 byte

1832 <1> ; bit 7 = installable device drv

1833 0000CDCD F6C501 <1> test ch, 1 ; accessable by normal users (except root)

1834 0000CDD0 7510 <1> jnz short sysopen\_dev\_4 ; yes, permission has been given

1835 0000CDD2 803D[B0030300]00 <1> cmp byte [u.uid], 0 ; root?

1836 0000CDD9 7607 <1> jna short sysopen\_dev\_4 ; superuser can open all devices

1837 <1> sysopen\_dev\_perm\_err:

1838 0000CDDB B80B000000 <1> mov eax, ERR\_DEV\_ACCESS ; 11 = 'permission denied !'

1839 0000CDE0 EBA5 <1> jmp short sysopen\_dev\_err

1840 <1> sysopen\_dev\_4:

1841 0000CDE2 D0ED <1> shr ch, 1 ; result: 1 = read, 2 = write, 3 = r & w

1842 0000CDE4 FEC9 <1> dec cl ; result: 1 = read, 2 = write

1843 0000CDE6 84E9 <1> test cl, ch

1844 0000CDE8 74F1 <1> jz short sysopen\_dev\_perm\_err

1845 <1>

1846 0000CDEA D0E5 <1> shl ch, 1 ; bit 0 = 0

1847 <1> ; eax = device number (entry offset)

1848 0000CDEC E8A3320000 <1> call device\_open

1849 0000CDF1 72E8 <1> jc short sysopen\_dev\_perm\_err

1850 <1>

1851 <1> ; eax = device number (entry offset)

1852 0000CDF3 0C80 <1> or al, 80h ; set device bit (set bit 7 to 1)

1853 0000CDF5 8B1D[64030300] <1> mov ebx, [u.r0]

1854 0000CDFB 8883[6A030300] <1> mov [ebx+u.fp], al ; bit 7 (=1) points to device

1855 <1>

1856 0000CE01 E9D8F8FFFF <1> jmp sysret

1857 <1>

1858 <1> sysmkdir: ; < make directory >

1859 <1> ; 15/10/2016

1860 <1> ; 10/10/2016 (TRDOS 386 = TRDOS v2.0)

1861 <1> ; -derived from INT\_21H.ASM-

1862 <1> ; ("loc\_INT21h\_create\_file")

1863 <1> ; 10/07/2011 (12/03/2011)

1864 <1> ; INT 21h Function AH = 3Ch

1865 <1> ; Create File

1866 <1> ; INPUT

1867 <1> ; CX = Attributes

1868 <1> ; DS:DX= Address of zero terminaned path name

1869 <1> ;

1870 <1> ;

1871 <1> ; 14/05/2015 (Retro UNIX 386 v1 - Beginning)

1872 <1> ; 27/05/2013 - 02/08/2013 (Retro UNIX 8086 v1)

1873 <1> ;

1874 <1> ; 'sysmkdir' creates an empty directory whose name is

1875 <1> ; pointed to by arg 1. The mode of the directory is arg 2.

1876 <1> ; The special entries '.' and '..' are not present.

1877 <1> ; Errors are indicated if the directory already exists or

1878 <1> ; user is not the super user.

1879 <1> ;

1880 <1> ; Calling sequence:

1881 <1> ; sysmkdir; name; mode

1882 <1> ; Arguments:

1883 <1> ; name - points to the name of the directory

1884 <1> ; mode - mode of the directory

1885 <1> ; Inputs: (arguments)

1886 <1> ; Outputs: -

1887 <1> ; (sets 'directory' flag to 1;

1888 <1> ; 'set user id on execution' and 'executable' flags to 0)

1889 <1> ; ...............................................................

1890 <1> ;

1891 <1> ; Retro UNIX 8086 v1 modification:

1892 <1> ; 'sysmkdir' system call has two arguments; so,

1893 <1> ; \* 1st argument, name is pointed to by BX register

1894 <1> ; \* 2nd argument, mode is in CX register

1895 <1> ;

1896 <1> ; TRDOS 386 (10/10/2016)

1897 <1> ;

1898 <1> ; INPUT ->

1899 <1> ; CL = Directory Attributes

1900 <1> ; bit 0 (1) - Read only file/dir (R)

1901 <1> ; bit 1 (1) - Hidden file/dir (H)

1902 <1> ; bit 2 (1) - System file/dir (R)

1903 <1> ; bit 3 (1) - Volume label/name (V)

1904 <1> ; bit 4 (1) - Subdirectory (D)

1905 <1> ; bit 5 (1) - File/Dir has been archived (A)

1906 <1> ; CX = 0 -> create normal directory

1907 <1> ; EBX = Pointer to directory name (ASCIIZ) -path-

1908 <1> ;

1909 <1> ; OUTPUT ->

1910 <1> ; eax = First cluster of the new directory

1911 <1> ; cf = 1 -> Error code in AL

1912 <1> ;

1913 <1> ; Modified Registers: EAX (at the return of system call)

1914 <1> ;

1915 <1> ; Note: If the file or directory is existing

1916 <1> ; an access error will be returned.

1917 <1>

1918 0000CE06 6621C9 <1> and cx, cx ; if cx = 0 -> create a normal subdir

1919 0000CE09 7413 <1> jz short sysmkdir\_1

1920 <1>

1921 0000CE0B F6C110 <1> test cl, 10h ; if dir flags set, also use other flags

1922 0000CE0E 0F853EFDFFFF <1> jnz sysmkdir\_0 ; jump to head of 'syscreat'

1923 <1>

1924 <1> ; CX has wrong flags

1925 0000CE14 B817000000 <1> mov eax, ERR\_INV\_FLAGS

1926 0000CE19 E969FFFFFF <1> jmp sysopen\_dev\_err

1927 <1>

1928 <1> sysmkdir\_1:

1929 0000CE1E B110 <1> mov cl, 10h ; set subdir flag and reset other flags

1930 0000CE20 E92DFDFFFF <1> jmp sysmkdir\_0 ; jump to head of 'syscreat'

1931 <1> sysmkdir\_2:

1932 <1> ; jump from 'syscreat' ; from 'syscreat\_1'

1933 <1> ; CL = Directory attributes/flags

1934 0000CE25 BE[E4620100] <1> mov esi, FindFile\_Name

1935 0000CE2A E804D7FFFF <1> call make\_sub\_directory

1936 0000CE2F 0F821BFEFFFF <1> jc sysopen\_err ; NOTE: Old type (TRDOS 8086)

1937 <1> ; error codes must be modified

1938 <1> ; for next TRDOS 386 versions

1939 <1> ; (10/10/2016)

1940 <1> ; Old (MSDOS type)

1941 <1> ; error codes (2011):

1942 <1> ; 2 = file not found

1943 <1> ; 3 = directory not found

1944 <1> ; 5 = access denied

1945 <1> ; 12 = no more files

1946 <1> ; 19 = disk write protected

1947 <1> ; 39 = insufficient disk space

1948 <1> ; 'sysdefs.s' ; 10/10/2016

1949 <1>

1950 0000CE35 A3[64030300] <1> mov [u.r0], eax ; New sub dir's first cluster

1951 <1>

1952 0000CE3A E8912E0000 <1> call reset\_working\_path

1953 <1>

1954 0000CE3F E99AF8FFFF <1> jmp sysret

1955 <1>

1956 <1> sysclose: ;<close file>

1957 <1> ; 06/10/2016 (TRDOS 386 = TRDOS v2.0)

1958 <1> ;

1959 <1> ; 14/05/2015 (Retro UNIX 386 v1 - Beginning)

1960 <1> ; 22/05/2013 - 26/05/2013 (Retro UNIX 8086 v1)

1961 <1> ;

1962 <1> ; 'sysclose', given a file descriptor in 'u.r0', closes the

1963 <1> ; associated file. The file descriptor (index to 'u.fp' list)

1964 <1> ; is put in r1 and 'fclose' is called.

1965 <1> ;

1966 <1> ; Calling sequence:

1967 <1> ; sysclose

1968 <1> ; Arguments:

1969 <1> ; -

1970 <1> ; Inputs: \*u.r0 - file descriptor

1971 <1> ; Outputs: -

1972 <1> ; ...............................................................

1973 <1> ;

1974 <1> ; Retro UNIX 8086 v1 modification:

1975 <1> ; The user/application program puts file descriptor

1976 <1> ; in BX register as 'sysclose' system call argument.

1977 <1> ; (argument transfer method 1)

1978 <1>

1979 <1> ; TRDOS 386 (06/10/2016)

1980 <1> ;

1981 <1> ; INPUT ->

1982 <1> ; EBX = File Handle/Number (file index) (AL)

1983 <1> ; OUTPUT ->

1984 <1> ; cf = 0 -> EAX = 0

1985 <1> ; cf = 1 -> Error code in EAX (ERR\_FILE\_NOT\_OPEN)

1986 <1> ;

1987 <1> ; Modified Registers: EAX (at the return of system call)

1988 <1> ;

1989 <1>

1990 0000CE44 89D8 <1> mov eax, ebx

1991 0000CE46 31DB <1> xor ebx, ebx

1992 0000CE48 891D[64030300] <1> mov [u.r0], ebx ; 0 ; return value of EAX

1993 0000CE4E E8450E0000 <1> call fclose

1994 0000CE53 0F8385F8FFFF <1> jnc sysret

1995 0000CE59 B80A000000 <1> mov eax, ERR\_FILE\_NOT\_OPEN ; file not open !

1996 0000CE5E A3[C8030300] <1> mov [u.error], eax ;

1997 0000CE63 A3[64030300] <1> mov [u.r0], eax ; ! invalid handle !

1998 0000CE68 E951F8FFFF <1> jmp error

1999 <1>

2000 <1> sysread: ; < read from file >

2001 <1> ; 11/10/2016 (TRDOS 386 = TRDOS v2.0)

2002 <1> ; -derived from INT\_21H.ASM-

2003 <1> ; ("loc\_INT21h\_read\_file")

2004 <1> ; 13/03/2011 (05/03/2011)

2005 <1> ; INT 21h Function AH = 3Fh

2006 <1> ; Read from a File

2007 <1> ; INPUT

2008 <1> ; BX = File Handle

2009 <1> ; CX = Number of bytes to read

2010 <1> ; DS:DX= Buffer address

2011 <1> ;

2012 <1> ; Note: TRDOS 386 'sysread' has been derived from

2013 <1> ; Retro UNIX 386 v1 'sysread', except a few

2014 <1> ; code modifications.

2015 <1> ;

2016 <1> ; 13/05/2015 (Retro UNIX 386 v1)

2017 <1> ; 11/05/2015 (Retro UNIX 386 v1 - Beginning)

2018 <1> ; 23/05/2013 (Retro UNIX 8086 v1)

2019 <1> ;

2020 <1> ; 'sysread' is given a buffer to read into and the number of

2021 <1> ; characters to be read. If finds the file from the file

2022 <1> ; descriptor located in \*u.r0 (r0). This file descriptor

2023 <1> ; is returned from a successful open call (sysopen).

2024 <1> ; The i-number of file is obtained via 'rw1' and the data

2025 <1> ; is read into core via 'readi'.

2026 <1> ;

2027 <1> ; Calling sequence:

2028 <1> ; sysread; buffer; nchars

2029 <1> ; Arguments:

2030 <1> ; buffer - location of contiguous bytes where

2031 <1> ; input will be placed.

2032 <1> ; nchars - number of bytes or characters to be read.

2033 <1> ; Inputs: \*u.r0 - file descriptor (& arguments)

2034 <1> ; Outputs: \*u.r0 - number of bytes read.

2035 <1> ; ...............................................................

2036 <1> ;

2037 <1> ; Retro UNIX 8086 v1 modification:

2038 <1> ; 'sysread' system call has three arguments; so,

2039 <1> ; \* 1st argument, file descriptor is in BX register

2040 <1> ; \* 2nd argument, buffer address/offset in CX register

2041 <1> ; \* 3rd argument, number of bytes is in DX register

2042 <1> ;

2043 <1> ; AX register (will be restored via 'u.r0') will return

2044 <1> ; to the user with number of bytes read.

2045 <1> ;

2046 <1> ; TRDOS 386 (05/10/2016)

2047 <1> ;

2048 <1> ; INPUT ->

2049 <1> ; EBX = File handle (descriptor/index)

2050 <1> ; ECX = Buffer address

2051 <1> ; EDX = Number of bytes

2052 <1> ; OUTPUT ->

2053 <1> ; EAX = Number of bytes have been read

2054 <1> ; cf = 1 -> Error code in AL

2055 <1> ;

2056 <1> ; Modified Registers: EAX (at the return of system call)

2057 <1> ;

2058 <1>

2059 <1> ; EBX = File descriptor

2060 0000CE6D E8740E0000 <1> call getf1

2061 0000CE72 7277 <1> jc short device\_read ; read data from device

2062 <1> ; EAX = First cluster of the file

2063 <1>

2064 0000CE74 E83F000000 <1> call rw1

2065 0000CE79 730A <1> jnc short sysread\_0

2066 <1>

2067 0000CE7B A3[64030300] <1> mov [u.r0], eax ; error code

2068 0000CE80 E939F8FFFF <1> jmp error

2069 <1>

2070 <1> sysread\_0:

2071 0000CE85 E825140000 <1> call readi

2072 0000CE8A EB1D <1> jmp short rw0

2073 <1>

2074 <1> syswrite: ; < write to file >

2075 <1> ; 23/10/2016

2076 <1> ; 11/10/2016 (TRDOS 386 = TRDOS v2.0)

2077 <1> ; -derived from INT\_21H.ASM-

2078 <1> ; ("loc\_INT21h\_write\_file")

2079 <1> ; 13/03/2011 (05/03/2011)

2080 <1> ; INT 21h Function AH = 40h

2081 <1> ; Write to a File

2082 <1> ; INPUT

2083 <1> ; BX = File Handle

2084 <1> ; CX = Number of bytes to write

2085 <1> ; DS:DX= Buffer address

2086 <1> ;

2087 <1> ; Note: TRDOS 386 'sysrwrite' has been derived from

2088 <1> ; Retro UNIX 386 v1 'syswrite', except a few

2089 <1> ; code modifications.

2090 <1> ;

2091 <1>

2092 <1> ; 13/05/2015 (Retro UNIX 386 v1)

2093 <1> ; 11/05/2015 (Retro UNIX 386 v1 - Beginning)

2094 <1> ; 23/05/2013 (Retro UNIX 8086 v1)

2095 <1> ;

2096 <1> ; 'syswrite' is given a buffer to write onto an output file

2097 <1> ; and the number of characters to write. If finds the file

2098 <1> ; from the file descriptor located in \*u.r0 (r0). This file

2099 <1> ; descriptor is returned from a successful open or create call

2100 <1> ; (sysopen or syscreat). The i-number of file is obtained via

2101 <1> ; 'rw1' and buffer is written on the output file via 'write'.

2102 <1> ;

2103 <1> ; Calling sequence:

2104 <1> ; syswrite; buffer; nchars

2105 <1> ; Arguments:

2106 <1> ; buffer - location of contiguous bytes to be writtten.

2107 <1> ; nchars - number of characters to be written.

2108 <1> ; Inputs: \*u.r0 - file descriptor (& arguments)

2109 <1> ; Outputs: \*u.r0 - number of bytes written.

2110 <1> ; ...............................................................

2111 <1> ;

2112 <1> ; Retro UNIX 8086 v1 modification:

2113 <1> ; 'syswrite' system call has three arguments; so,

2114 <1> ; \* 1st argument, file descriptor is in BX register

2115 <1> ; \* 2nd argument, buffer address/offset in CX register

2116 <1> ; \* 3rd argument, number of bytes is in DX register

2117 <1> ;

2118 <1> ; AX register (will be restored via 'u.r0') will return

2119 <1> ; to the user with number of bytes written.

2120 <1> ;

2121 <1> ; INPUT ->

2122 <1> ; EBX = File handle (descriptor/index)

2123 <1> ; ECX = Buffer address

2124 <1> ; EDX = Number of bytes

2125 <1> ; OUTPUT ->

2126 <1> ; EAX = Number of bytes have been written

2127 <1> ; cf = 1 -> Error code in AL

2128 <1> ;

2129 <1> ; Modified Registers: EAX (at the return of system call)

2130 <1> ;

2131 <1>

2132 <1> ; EBX = File descriptor

2133 0000CE8C E8550E0000 <1> call getf1

2134 0000CE91 7274 <1> jc short device\_write ; write data to device

2135 <1> ; EAX = First cluster of the file

2136 <1> ; EBX = File number (Open file number) ; 23/10/2016

2137 <1>

2138 0000CE93 E820000000 <1> call rw1

2139 0000CE98 730A <1> jnc short syswrite\_0

2140 0000CE9A A3[64030300] <1> mov [u.r0], eax ; error code

2141 0000CE9F E91AF8FFFF <1> jmp error

2142 <1>

2143 <1> syswrite\_0:

2144 0000CEA4 E8321B0000 <1> call writei

2145 <1> rw0: ; 1:

2146 0000CEA9 A1[8C030300] <1> mov eax, [u.nread]

2147 0000CEAE A3[64030300] <1> mov [u.r0], eax

2148 0000CEB3 E926F8FFFF <1> jmp sysret

2149 <1>

2150 <1> rw1:

2151 <1> ; 11/10/2016 (TRDOS 386 = TRDOS v2.0)

2152 <1> ; 14/05/2015 (Retro UNIX 386 v1)

2153 <1> ; 11/05/2015 (Retro UNIX 386 v1 - Beginning)

2154 <1> ; 23/05/2013 - 24/05/2013 (Retro UNIX 8086 v1)

2155 <1> ; System call registers: ebx, ecx, edx (through 'sysenter')

2156 <1> ;

2157 <1> ; EBX = File descriptor

2158 <1> ;call getf1 ; calling point in 'getf' from 'rw1'

2159 <1> ;jc short device\_rw ; read/write data from/to device

2160 <1> ; EAX = First cluster of the file

2161 <1>

2162 0000CEB8 83F802 <1> cmp eax, 2

2163 0000CEBB 7217 <1> jb short rw2

2164 <1> ;

2165 0000CEBD 890D[84030300] <1> mov [u.base], ecx ; buffer address/offset

2166 <1> ;(in the user's virtual memory space)

2167 0000CEC3 8915[88030300] <1> mov [u.count], edx

2168 <1>

2169 0000CEC9 C705[C8030300]0000- <1> mov dword [u.error], 0 ; reset the last error code

2169 0000CED1 0000 <1>

2170 0000CED3 C3 <1> retn

2171 <1>

2172 <1> rw2:

2173 0000CED4 B80A000000 <1> mov eax, ERR\_FILE\_NOT\_OPEN ; file not open !

2174 0000CED9 A3[C8030300] <1> mov dword [u.error], eax

2175 0000CEDE C3 <1> retn

2176 <1> rw3:

2177 0000CEDF B80B000000 <1> mov eax, ERR\_FILE\_ACCESS ; permission denied !

2178 0000CEE4 A3[C8030300] <1> mov dword [u.error], eax

2179 0000CEE9 F9 <1> stc

2180 0000CEEA C3 <1> retn

2181 <1>

2182 <1> device\_read:

2183 <1> ; 11/10/2016 (TRDOS 386 = TRDOS v2.0)

2184 <1> ; cl = DEV\_OPENMODE ; open mode

2185 <1> ; ch = DEV\_ACCESS ; access flags

2186 <1> ; al = DEV\_DRIVER ; device number (eax)

2187 <1>

2188 0000CEEB F6C101 <1> test cl, 1 ; 1 = read, 2 = write, 3 = read&write

2189 0000CEEE 74EF <1> jz short rw3

2190 <1>

2191 0000CEF0 89C3 <1> mov ebx, eax

2192 0000CEF2 66C1E302 <1> shl bx, 2 ; \*4

2193 <1>

2194 0000CEF6 F6C580 <1> test ch, 80h ; bit 7, installable device driver flag

2195 0000CEF9 7406 <1> jz short d\_read\_2 ; Kernel device

2196 <1> ; installable device

2197 <1> d\_read\_1:

2198 0000CEFB FFA3[B0660100] <1> jmp dword [ebx+IDEV\_RADDR-4]

2199 <1> d\_read\_2:

2200 0000CF01 FFA3[50150100] <1> jmp dword [ebx+KDEV\_RADDR-4]

2201 <1>

2202 <1> device\_write:

2203 <1> ; 11/10/2016 (TRDOS 386 = TRDOS v2.0)

2204 <1> ; cl = DEV\_OPENMODE ; open mode

2205 <1> ; ch = DEV\_ACCESS ; access flags

2206 <1> ; al = DEV\_DRIVER ; device number (eax)

2207 <1>

2208 0000CF07 F6C102 <1> test cl, 2 ; 1 = read, 2 = write, 3 = read&write

2209 0000CF0A 74D3 <1> jz short rw3

2210 <1>

2211 0000CF0C 89C3 <1> mov ebx, eax

2212 0000CF0E 66C1E302 <1> shl bx, 2 ; \*4

2213 <1>

2214 0000CF12 F6C580 <1> test ch, 80h ; bit 7, installable device driver flag

2215 0000CF15 7406 <1> jz short d\_write\_2 ; Kernel device

2216 <1> ; installable device

2217 <1> d\_write\_1:

2218 0000CF17 FFA3[D0660100] <1> jmp dword [ebx+IDEV\_WADDR-4]

2219 <1> d\_write\_2:

2220 0000CF1D FFA3[A0150100] <1> jmp dword [ebx+KDEV\_WADDR-4]

2221 <1>

2222 <1>

2223 <1> sysemt: ; enable (or disable) multi tasking -time sharing-

2224 <1> ;

2225 <1> ; 23/05/2016 - TRDOS 386 (TRDOS v2.0)

2226 <1> ; 14/05/2015 (Retro UNIX 386 v1)

2227 <1> ; 10/12/2013 - 20/04/2014 (Retro UNIX 8086 v1)

2228 <1> ;

2229 <1> ; Retro UNIX 8086 v1 modification:

2230 <1> ; 'Enable Multi Tasking' system call instead

2231 <1> ; of 'Emulator Trap' in original UNIX v1 for PDP-11.

2232 <1> ;

2233 <1> ; Retro UNIX 8086 v1 feature only!

2234 <1> ; Using purpose: Kernel will start without time-out

2235 <1> ; (internal clock/timer) functionality.

2236 <1> ; Then etc/init will enable clock/timer for

2237 <1> ; multi tasking.

2238 <1> ;

2239 <1> ; INPUT ->

2240 <1> ; BL = 0 -> disable multi tasking

2241 <1> ; BL > 1 -> enable multi tasking (time sharing)

2242 <1> ; OUTPUT ->

2243 <1> ; none

2244 <1> ;

2245 <1> ; Note: Multi tasking is disabled during system

2246 <1> ; initialization, it must be enabled by using

2247 <1> ; this system call. (Otherwise, running proces

2248 <1> ; will not be changed by another process within

2249 <1> ; run time sequence/schedule, if running process

2250 <1> ; will not 'release' itself. Only 'wakeup' procedure

2251 <1> ; for waiting processes and programmed timer events

2252 <1> ; for other processes can change running process

2253 <1> ; while multi tasking is disabled.) \*\* 23/05/2016 \*\*

2254 <1>

2255 0000CF23 803D[B0030300]00 <1> cmp byte [u.uid], 0 ; root ?

2256 <1> ;ja error

2257 0000CF2A 0F87D3F8FFFF <1> ja badsys ; 14/05/2015

2258 <1> ;

2259 0000CF30 FA <1> cli

2260 0000CF31 881D[CE650100] <1> mov [multi\_tasking], bl ; 0 to disable, >0 to enable

2261 0000CF37 E9A2F7FFFF <1> jmp sysret

2262 <1>

2263 <1> systimer:

2264 <1> ; 02/01/2017

2265 <1> ; 21/12/2016

2266 <1> ; 19/12/2016

2267 <1> ; 10/12/2016 (callback)

2268 <1> ; 10/06/2016

2269 <1> ; 07/06/2016

2270 <1> ; 06/06/2016

2271 <1> ; 21/05/2016

2272 <1> ; 19/05/2016

2273 <1> ; 18/05/2016 - TRDOS 386 (TRDOS v2.0)

2274 <1> ; (TRDOS 386 feature only!)

2275 <1> ;

2276 <1> ; (start or stop timer event(s))

2277 <1> ;

2278 <1> ; INPUT ->

2279 <1> ; BL = Signal return byte (response byte)

2280 <1> ; (Any requested value between 0 and 255)

2281 <1> ; (Kernel will put it at the requested address)

2282 <1> ; BH = Time count unit

2283 <1> ; 0 = Stop timer event

2284 <1> ; 1 = 18.2 ticks per second

2285 <1> ; 2 = 10 milliseconds

2286 <1> ; 3 = 1 second (for real time clock interrupt)

2287 <1> ; 4 = time/tick count in current time count unit

2288 <1> ; // 10/12/2016

2289 <1> ; 80h = Stop timer event (callback method)

2290 <1> ; 81h = 18.2 ticks per second, callback method

2291 <1> ; 82h = 10 milliseconds, callback method

2292 <1> ; 83h = 1 second (for RTC int), callback method

2293 <1> ; 84h = current time count unit, callback method

2294 <1> ;

2295 <1> ; Note: Only 03h or 83h will set real time clock

2296 <1> ; (RTC) events (Others are for PIT events)!

2297 <1> ;

2298 <1> ; NOTE: If callback (user service) method is used,

2299 <1> ; EDX will point to the return address (of service

2300 <1> ; procedure) in user's space instead of signal

2301 <1> ; response byte address. (TRDOS 386 kernel will

2302 <1> ; direct the cpu to that address -in user's space-

2303 <1> ; at the return of system call or interrupt

2304 <1> ; just after the adjusted count/time is elapsed.)

2305 <1> ; User's sevice routine must be ended with a

2306 <1> ; 'iret'. Normal return addresses from system

2307 <1> ; calls or and interrupts will be kept same except

2308 <1> ; the timer returns.

2309 <1> ;

2310 <1> ; BH = 0 -> Stop timer event

2311 <1> ; BL = Timer event number (1 to 255) if BH = 0

2312 <1> ; If BL = 0, all timer events (which are belongs

2313 <1> ; to running process) will be stopped

2314 <1> ; ECX = Time/Tick count (depending on time count unit)

2315 <1> ; EDX = Signal return (Response) byte address

2316 <1> ; (virtual address in user's memory space)

2317 <1> ; OUTPUT ->

2318 <1> ; AL = Timer event number (1 to 255) (max. value = 16)

2319 <1> ; IF BH Input = 0 & CF = 0 & AL = 0 ->

2320 <1> ; timer event(s) has/have been stopped/finished

2321 <1> ; CF = 1 & AL = 0 -> no timer setting space to set

2322 <1> ; CF = 1 & AL > 0 -> timer count unit is not usable

2323 <1> ;

2324 <1> ; NOTE: To modify a time count for a user function,

2325 <1> ; at first, current timer event must be stopped

2326 <1> ; then a new timer event (which is related with

2327 <1> ; same user function) must be started.

2328 <1> ;

2329 <1> ; Signal return (response) byte may be used for

2330 <1> ; several purposes. Kernel will put this value

2331 <1> ; to requested address during timer interrupt,

2332 <1> ; program/user can check this value to understand

2333 <1> ; which event has been occurred and what is changed.

2334 <1> ; (Multi timer events can share same signal address)

2335 <1> ;

2336 <1> ; NOTE: If the process is running while the time count

2337 <1> ; is reached, kernel will put signal return (response)

2338 <1> ; byte value at requested address during timer

2339 <1> ; interrupt and the process will continue to run.

2340 <1> ; Program/process must call (jump to) it's timer event

2341 <1> ; function as required, for checking the timer event

2342 <1> ; status via signal return (response) byte address.

2343 <1> ;

2344 <1> ; If the process is not running (waiting or sleeping

2345 <1> ; or released) while the time count is reached,

2346 <1> ; it is restarted from where it left, to ensure

2347 <1> ; proper multi media (video, audio, clock, timer)

2348 <1> ; functionality.

2349 <1> ;

2350 <1> ; (It is better to use 'syswait' or 'syssleep',

2351 <1> ; or 'sysrele' system call just after the timer

2352 <1> ; function. Otherwise, timer events may block other

2353 <1> ; processes which are not using timer events.)

2354 <1> ;

2355 <1> ; Timer Event Structure: (max. 16 timer events, 16\*16 bytes)

2356 <1> ; Owner: resb 1 ; 0 = free

2357 <1> ; ;>0 = process number (u.uno)

2358 <1> ; Calback: resb 1 ; 1 = callback, 0 = response byte

2359 <1> ; Interrupt: resb 1 ; 0 = Timer interrupt (or none)

2360 <1> ; ; 1 = Real Time Clock interrupt

2361 <1> ; Response: resb 1 ; 0 to 255, signal return value

2362 <1> ; Count Limit: resd 1 ; count of ticks (total/set)

2363 <1> ; Current Count: resd 1 ; count of ticks (current)

2364 <1> ; Response Addr: resd 1 ; response byte (pointer) address

2365 <1> ;

2366 <1>

2367 <1> ; 19/12/2016 (timer callback)

2368 0000CF3C C605[0C6B0100]00 <1> mov byte [tcallback], 0

2369 0000CF43 C605[0D6B0100]00 <1> mov byte [trtc], 0

2370 0000CF4A C705[D0030300]0000- <1> mov dword [u.tcb], 0 ; this is not necessary...

2370 0000CF52 0000 <1>

2371 <1>

2372 0000CF54 80FF80 <1> cmp bh, 80h

2373 0000CF57 7225 <1> jb short systimer\_cb2

2374 0000CF59 7704 <1> ja short systimer\_cb0

2375 <1>

2376 0000CF5B 31D2 <1> xor edx, edx ; 0, reset callback address

2377 0000CF5D EB0B <1> jmp short systimer\_cb1

2378 <1>

2379 <1> systimer\_cb0:

2380 0000CF5F 80FF84 <1> cmp bh, 84h

2381 0000CF62 7764 <1> ja short systimer\_5 ; undefined, error

2382 <1>

2383 <1> ;mov byte [tcallback], 1 ; 19/12/2016

2384 0000CF64 FE05[0C6B0100] <1> inc byte [tcallback]

2385 <1>

2386 <1> systimer\_cb1:

2387 0000CF6A 0FB635[B3030300] <1> movzx esi, byte [u.uno] ; process number

2388 0000CF71 66C1E602 <1> shl si, 2

2389 0000CF75 8996[0C010300] <1> mov [esi+p.tcb-4], edx ; set process timer callback address

2390 <1> ; (overwrite prev value if it is set!)

2391 0000CF7B 80E77F <1> and bh, 7Fh

2392 <1>

2393 <1> systimer\_cb2:

2394 0000CF7E 80FF02 <1> cmp bh, 2

2395 0000CF81 7445 <1> je short systimer\_5 ; only 18.2 ticks per second is usable

2396 <1> ; 10 milliseconds (100 Hertz) timer

2397 <1> ; will be set later (18/05/2016)

2398 0000CF83 774B <1> ja short systimer\_6

2399 <1>

2400 0000CF85 20FF <1> and bh, bh

2401 0000CF87 0F84BA000000 <1> jz systimer\_9 ; stop timer event(s)

2402 <1>

2403 <1> ; bh = 1 (timer interrupt, 18.2 Hz, IBM PC/AT ROMBIOS default)

2404 <1>

2405 <1> systimer\_19:

2406 0000CF8D B00A <1> mov al, 10 ; (\*)

2407 <1>

2408 <1> systimer\_0:

2409 0000CF8F B710 <1> mov bh, 16

2410 <1> ;

2411 0000CF91 383D[CF650100] <1> cmp [timer\_events], bh ; 16 ; 07/06/2016

2412 0000CF97 7319 <1> jnb short systimer\_3 ; max. 16 timer events

2413 <1> ;

2414 0000CF99 50 <1> push eax ; (\*)

2415 <1>

2416 0000CF9A BF[60040300] <1> mov edi, timer\_set ; beginning address of timer events

2417 <1> ; setting space

2418 0000CF9F 30C0 <1> xor al, al ; 0

2419 <1> systimer\_1:

2420 0000CFA1 FEC0 <1> inc al

2421 0000CFA3 803F00 <1> cmp byte [edi], 0 ; is it free space ?

2422 0000CFA6 7639 <1> jna short systimer\_7 ; yes

2423 0000CFA8 FECF <1> dec bh

2424 0000CFAA 7405 <1> jz short systimer\_2

2425 0000CFAC 83C710 <1> add edi, 16

2426 0000CFAF EBF0 <1> jmp short systimer\_1 ; next event space

2427 <1>

2428 <1> systimer\_2:

2429 0000CFB1 58 <1> pop eax ; (\*) discard

2430 <1> systimer\_3:

2431 0000CFB2 C605[64030300]00 <1> mov byte [u.r0], 0

2432 <1> systimer\_4:

2433 0000CFB9 C705[C8030300]1B00- <1> mov dword [u.error], ERR\_MISC

2433 0000CFC1 0000 <1>

2434 <1> ; one of miscellaneous/other errors

2435 0000CFC3 E9F6F6FFFF <1> jmp error ; cf -> 1

2436 <1>

2437 <1> systimer\_5:

2438 0000CFC8 883D[64030300] <1> mov [u.r0], bh ; Time count unit (=2 or >3)

2439 0000CFCE EBE9 <1> jmp short systimer\_4 ; 07/06/2016

2440 <1>

2441 <1> systimer\_6:

2442 0000CFD0 80FF04 <1> cmp bh, 4

2443 0000CFD3 77F3 <1> ja short systimer\_5 ; undefined time count unit

2444 <1> ;jb short systimer\_16

2445 <1>

2446 <1> ;mov al, 1 ; default (use current timer unit)

2447 <1> ; countdown value is in ECX !

2448 <1> ; max. value of ecx = 4294967296/10

2449 <1> ;jmp short systimer\_0

2450 <1> ;jmp short systimer\_19

2451 0000CFD5 74B6 <1> je short systimer\_19

2452 <1>

2453 <1> systimer\_16:

2454 <1> ; bh = 3

2455 <1> ; timer event via real time clock interrupt

2456 <1> ; interrupt/update frequency: 1 Hz (1 tick per second)

2457 <1>

2458 0000CFD7 B0B6 <1> mov al, 182 ; (\*) ; 18.2 \* 10

2459 0000CFD9 FE05[0D6B0100] <1> inc byte [trtc] ; timer event via real time clock

2460 0000CFDF EBAE <1> jmp short systimer\_0

2461 <1>

2462 <1> systimer\_7:

2463 0000CFE1 A2[64030300] <1> mov [u.r0], al ; timer event number

2464 <1> ;

2465 <1> ; edi = address of empty timer event area

2466 0000CFE6 A0[B3030300] <1> mov al, [u.uno]

2467 0000CFEB FA <1> cli ; disable interrupts

2468 0000CFEC AA <1> stosb ; process number

2469 0000CFED A0[0C6B0100] <1> mov al, [tcallback] ; timer callback flag

2470 0000CFF2 AA <1> stosb ; 1= callback method, 0= signal response byte method

2471 0000CFF3 A0[0D6B0100] <1> mov al, [trtc] ; timer interrupt type

2472 0000CFF8 AA <1> stosb ; 1= real time clock, 0= programmable interval timer

2473 0000CFF9 88D8 <1> mov al, bl ; Signal return (Response) value

2474 0000CFFB AA <1> stosb ; response byte

2475 0000CFFC 58 <1> pop eax ; (\*) ; 10 or 182

2476 0000CFFD 89D3 <1> mov ebx, edx ; virtual address for response/signal byte

2477 0000CFFF F7E1 <1> mul ecx

2478 <1> ; (eax = 10 \* count of 18.2 Hz timer ticks)

2479 <1> ; (count down step = 10)

2480 0000D001 AB <1> stosd ; count limit (reset value)

2481 0000D002 AB <1> stosd ; current count value

2482 <1>

2483 <1> ; 19/12/2016

2484 0000D003 803D[0C6B0100]00 <1> cmp byte [tcallback], 0 ; timer callback method ?

2485 0000D00A 7604 <1> jna short systimer\_17 ; no

2486 0000D00C 89D8 <1> mov eax, ebx ; virtual address for callback routine

2487 0000D00E EB0D <1> jmp short systimer\_18

2488 <1>

2489 <1> systimer\_17: ; signal response byte method

2490 <1> ; ebx = virtual address

2491 <1> ; [u.pgdir] = page directory's physical address

2492 <1> ; 20/02/2017

2493 0000D010 FE05[0E6B0100] <1> inc byte [no\_page\_swap] ; 1

2494 <1> ; Do not add this page to swap queue

2495 <1> ; and remove it from swap queue if it is

2496 <1> ; on the queue.

2497 0000D016 E87482FFFF <1> call get\_physical\_addr

2498 0000D01B 721A <1> jc short systimer\_8 ; 07/06/2016

2499 <1> ; eax = physical address of the virtual address in user's space

2500 <1> systimer\_18:

2501 0000D01D AB <1> stosd ; response addr (physical) or callback addr (virtual)

2502 0000D01E FE05[CF650100] <1> inc byte [timer\_events] ; 07/06/201

2503 <1> ; 02/01/2017

2504 0000D024 0FB605[B3030300] <1> movzx eax, byte [u.uno]

2505 0000D02B FE80[FF000300] <1> inc byte [eax+p.timer-1]

2506 <1> ;

2507 0000D031 FB <1> sti ; enable interrupts

2508 0000D032 E9A7F6FFFF <1> jmp sysret

2509 <1>

2510 <1> systimer\_8:

2511 <1> ; 10/06/2016

2512 <1> ; 07/06/2016

2513 0000D037 28C0 <1> sub al, al ; 0

2514 0000D039 8847F4 <1> mov [edi-12], al ; clear process number (free timer event)

2515 <1> ;mov dword [edi], eax ; 0

2516 0000D03C FB <1> sti

2517 0000D03D A2[64030300] <1> mov [u.r0], al ; 0

2518 0000D042 E977F6FFFF <1> jmp error

2519 <1>

2520 <1> systimer\_9:

2521 <1> ; 10/06/2016

2522 <1> ; 07/06/2016

2523 0000D047 28C0 <1> sub al, al

2524 0000D049 A2[64030300] <1> mov byte [u.r0], al ; 0

2525 0000D04E 3805[CF650100] <1> cmp byte [timer\_events], al ; 0

2526 0000D054 7631 <1> jna short systimer\_12

2527 <1>

2528 <1> ; Note: ecx and edx are undefined here

2529 <1> ; (for stop timer function)

2530 <1>

2531 0000D056 BE[60040300] <1> mov esi, timer\_set ; beginning address of timer events

2532 <1> ; setting space

2533 0000D05B A0[B3030300] <1> mov al, [u.uno]

2534 <1>

2535 0000D060 B710 <1> mov bh, 16

2536 <1>

2537 0000D062 08DB <1> or bl, bl

2538 0000D064 7544 <1> jnz short systimer\_15

2539 <1>

2540 <1> ; clear timer event areas belong to current process

2541 <1> ; (for stopping all timer events belong to current process)

2542 0000D066 FA <1> cli ; disable interrupts

2543 <1> systimer\_10:

2544 <1> ; 10/06/2016

2545 <1> ; 07/06/2016

2546 0000D067 8A26 <1> mov ah, [esi]

2547 0000D069 08E4 <1> or ah, ah ; 0 ?

2548 0000D06B 7411 <1> jz short systimer\_11

2549 0000D06D 38C4 <1> cmp ah, al ; is the process number (owner) same ?

2550 0000D06F 750D <1> jne short systimer\_11 ; no

2551 <1>

2552 <1> ;mov byte [esi], 0

2553 0000D071 66C7060000 <1> mov word [esi], 0 ; clear

2554 <1> ;mov dword [esi+12], 0 ; clear

2555 <1>

2556 0000D076 FE0D[CF650100] <1> dec byte [timer\_events]

2557 0000D07C 7409 <1> jz short systimer\_12

2558 <1>

2559 <1> systimer\_11:

2560 0000D07E FECF <1> dec bh

2561 0000D080 7405 <1> jz short systimer\_12

2562 0000D082 83C610 <1> add esi, 16

2563 0000D085 EBE0 <1> jmp short systimer\_10

2564 <1>

2565 <1> systimer\_12:

2566 0000D087 0FB635[B3030300] <1> movzx esi, byte [u.uno]

2567 0000D08E 08DB <1> or bl, bl ; all timer events or one timer event ?

2568 0000D090 740C <1> jz short systimer\_13

2569 0000D092 8A9E[FF000300] <1> mov bl, [esi+p.timer-1]

2570 0000D098 20DB <1> and bl, bl ; previous number of timer events for the process

2571 0000D09A 7408 <1> jz short systimer\_14

2572 0000D09C FECB <1> dec bl ; previous number of timer events for the process - 1

2573 <1> systimer\_13:

2574 0000D09E 889E[FF000300] <1> mov [esi+p.timer-1], bl ; 0 ; no timer events for process

2575 <1> systimer\_14:

2576 0000D0A4 FB <1> sti ; enable interrupts

2577 0000D0A5 E934F6FFFF <1> jmp sysret

2578 <1>

2579 <1> systimer\_15:

2580 0000D0AA 38FB <1> cmp bl, bh ; 16

2581 0000D0AC 0F8707FFFFFF <1> ja systimer\_4 ; max. 16 timer events !

2582 <1> ;

2583 0000D0B2 88DA <1> mov dl, bl

2584 0000D0B4 FECA <1> dec dl ; 16 -> 15 ... 1 -> 0

2585 0000D0B6 C0E204 <1> shl dl, 4 ; \* 16

2586 0000D0B9 0FB6FA <1> movzx edi, dl

2587 0000D0BC 01F7 <1> add edi, esi ; timer\_set

2588 <1>

2589 0000D0BE 3A07 <1> cmp al, [edi] ; process number

2590 0000D0C0 0F85F3FEFFFF <1> jne systimer\_4

2591 <1>

2592 <1> ; same process ID

2593 0000D0C6 FA <1> cli ; disable interrupts

2594 <1> ; 10/06/2016 ; 02/01/2017

2595 <1> ;mov byte [edi], 0

2596 0000D0C7 66C7070000 <1> mov word [edi], 0 ; clear

2597 <1> ;mov dword [edi+12], 0 ; clear

2598 0000D0CC FE0D[CF650100] <1> dec byte [timer\_events]

2599 0000D0D2 EBB3 <1> jmp short systimer\_12

2600 <1>

2601 <1> sysvideo: ; VIDEO DATA TRANSFER FUNCTIONS

2602 <1> ; 12/05/2017

2603 <1> ; 11/07/2016

2604 <1> ; 13/06/2016

2605 <1> ; 16/05/2016 - TRDOS 386 (TRDOS v2.0)

2606 <1> ;

2607 <1> ;

2608 <1> ; VIDEO DATA TRANSFER FUNCTIONS:

2609 <1> ;

2610 <1> ; Inputs:

2611 <1> ; BH = 0 = VIDEO BIOS Mode 3, tty/text mode data transfers

2612 <1> ; BL =

2613 <1> ; Bits 0&1, Transfer direction

2614 <1> ; 0 - System to system

2615 <1> ; 1 - User to system

2616 <1> ; 2 - System to user

2617 <1> ; 3 - User to user

2618 <1> ; Bits 2&3, Transfer Type

2619 <1> ; 0 - Display page transfer

2620 <1> ; 1 - Display page window transfer

2621 <1> ; 2 - Frame/Viewport/Window address transfer

2622 <1> ; 3 - Window handle transfer

2623 <1> ;

2624 <1> ; /// BL = 0 -> System to system (display page) transfer

2625 <1> ; CL = Source page

2626 <1> ; DL = Destination page

2627 <1> ; /// BL = 1&2 -> user to system & system to user transfer

2628 <1> ; ECX = User buffer

2629 <1> ; DL = Video page

2630 <1> ; /// BL = 5&6 -> user to system, system to user transfer

2631 <1> ; (window in current display page and in current mode)

2632 <1> ; ESI = User's buffer address

2633 <1> ; ECX Low 16 bits = Top left column (X1 position)

2634 <1> ; ECX High 16 bits = Top row (Y1 position)

2635 <1> ; EDX Low 16 bits = Bottom right column (X2 position)

2636 <1> ; EDX High 16 bits = Bottom row (Y2 position)

2637 <1> ; If BL = 5 ->

2638 <1> ; EDI = Swap address (in user's memory space)

2639 <1> ; (If swap address > 0, previous content of the window

2640 <1> ; will be saved into swap area in user's memory space)

2641 <1> ; /// BL = 4 -> system to system transfer

2642 <1> ; ESI = System's source buffer (video page) address

2643 <1> ; ECX Low 16 bits = Top left column (X1 position)

2644 <1> ; ECX High 16 bits = Top row (Y1 position)

2645 <1> ; EDX Low 16 bits = Bottom right column (X2 position)

2646 <1> ; EDX High 16 bits = Bottom row (Y2 position)

2647 <1> ; EDI = System's destination buffer (video page) address

2648 <1> ;

2649 <1> ; BH = 1 = CGA Graphics (0B8000h) data transfers

2650 <1> ; BL =

2651 <1> ; 0 = Fill color (color in CL] (32K)

2652 <1> ; 1 = User to system display page transfer

2653 <1> ; 2 = System to user display page transfer

2654 <1> ; 3 = NOT bits in window (ECX, EDX)

2655 <1> ; 4 = Window copy (system to system)

2656 <1> ; 5 = User to system window transfer

2657 <1> ; 6 = System to user window transfer

2658 <1> ; 7 = AND display page bytes with CL

2659 <1> ; 8 = OR display page bytes with CL

2660 <1> ; 9 = XOR display page bytes with CL

2661 <1> ;

2662 <1> ; /// BL = 0 -> Fill color (all screen pixels)

2663 <1> ; CL = Color value

2664 <1> ; /// BL = 1&2 -> user to system & system to user transfer

2665 <1> ; ECX = User buffer

2666 <1> ; /// BL = 5&6 -> user to system, system to user transfer

2667 <1> ; (window in current display page and in current mode)

2668 <1> ; ESI = User's buffer address

2669 <1> ; ECX Low 16 bits = Top left column (X1 position)

2670 <1> ; ECX High 16 bits = Top row (Y1 position)

2671 <1> ; EDX Low 16 bits = Bottom right column (X2 position)

2672 <1> ; EDX High 16 bits = Bottom row (Y2 position)

2673 <1> ; /// BL = 4 -> system to system (window) transfer

2674 <1> ; ESI = System's source buffer (video page) address

2675 <1> ; ECX Low 16 bits = Top left column (X1 position)

2676 <1> ; ECX High 16 bits = Top row (Y1 position)

2677 <1> ; EDX Low 16 bits = Bottom right column (X2 position)

2678 <1> ; EDX High 16 bits = Bottom row (Y2 position)

2679 <1> ; EDI = System's destination buffer (video page) address

2680 <1> ; /// BL = 3 -> NOT byte in display page/memory

2681 <1> ; ECX Low 16 bits = Top left column (X1 position)

2682 <1> ; ECX High 16 bits = Top row (Y1 position)

2683 <1> ; EDX Low 16 bits = Bottom right column (X2 position)

2684 <1> ; EDX High 16 bits = Bottom row (Y2 position)

2685 <1> ;

2686 <1> ; BH = 2 = VGA Graphics (0A0000h) data transfers

2687 <1> ; BL =

2688 <1> ; x0h = Fill color (color in CL] (64K)

2689 <1> ; x1h = User to system display page transfer

2690 <1> ; x2h = System to user display page transfer

2691 <1> ; x3h = NOT bits in window (ECX, EDX)

2692 <1> ; x4h = Window copy (system to system)

2693 <1> ; x5h = User to system window transfer

2694 <1> ; x6h = System to user window transfer

2695 <1> ; x7h = AND display page bytes with CL

2696 <1> ; x8h = OR display page bytes with CL

2697 <1> ; x9h = XOR display page bytes with CL

2698 <1> ; x = 0 -> screen width = 320

2699 <1> ; x = 1 -> screen width = 640

2700 <1> ; x = 2 -> screen width = 800

2701 <1> ;

2702 <1> ; /// BL = 0 -> Fill color (all screen pixels)

2703 <1> ; CL = Color value

2704 <1> ; /// BL = 1&2 -> user to system & system to user transfer

2705 <1> ; ECX = User buffer

2706 <1> ; /// BL = 5&6 -> user to system, system to user transfer

2707 <1> ; (window in current display page and in current mode)

2708 <1> ; ESI = User's buffer address

2709 <1> ; ECX Low 16 bits = Top left column (X1 position)

2710 <1> ; ECX High 16 bits = Top row (Y1 position)

2711 <1> ; EDX Low 16 bits = Bottom right column (X2 position)

2712 <1> ; EDX High 16 bits = Bottom row (Y2 position)

2713 <1> ; /// BL = 4 -> system to system (window) transfer

2714 <1> ; ESI = System's source buffer (video page) address

2715 <1> ; ECX Low 16 bits = Top left column (X1 position)

2716 <1> ; ECX High 16 bits = Top row (Y1 position)

2717 <1> ; EDX Low 16 bits = Bottom right column (X2 position)

2718 <1> ; EDX High 16 bits = Bottom row (Y2 position)

2719 <1> ; EDI = System's destination buffer (video page) address

2720 <1> ; /// BL = 3 -> NOT byte in display page/memory

2721 <1> ; ECX Low 16 bits = Top left column (X1 position)

2722 <1> ; ECX High 16 bits = Top row (Y1 position)

2723 <1> ; EDX Low 16 bits = Bottom right column (X2 position)

2724 <1> ; EDX High 16 bits = Bottom row (Y2 position)

2725 <1> ;

2726 <1> ; BH = 3 = Super VGA, LINEAR FRAME BUFFER data transfers

2727 <1> ; BL =

2728 <1> ; 0 = Fill color (color in ECX] (Frame buffer size)

2729 <1> ; 1 = User to system display page transfer

2730 <1> ; 2 = System to user display page transfer

2731 <1> ; 3 = NOT bits in window (ECX, EDX)

2732 <1> ; 4 = Window copy (system to system)

2733 <1> ; 5 = User to system window transfer

2734 <1> ; 6 = System to user window transfer

2735 <1> ; 7 = AND display page bytes with ECX

2736 <1> ; 8 = OR display page bytes with ECX

2737 <1> ; 9 = XOR display page bytes with ECX

2738 <1> ;

2739 <1> ; /// BL = 0 -> Fill color (all screen pixels)

2740 <1> ; CL = Color value

2741 <1> ; /// BL = 1&2 -> user to system & system to user transfer

2742 <1> ; ECX = User buffer

2743 <1> ; /// BL = 5&6 -> user to system, system to user transfer

2744 <1> ; (window in current display page and in current mode)

2745 <1> ; ESI = User's buffer address

2746 <1> ; ECX Low 16 bits = Top left column (X1 position)

2747 <1> ; ECX High 16 bits = Top row (Y1 position)

2748 <1> ; EDX Low 16 bits = Bottom right column (X2 position)

2749 <1> ; EDX High 16 bits = Bottom row (Y2 position)

2750 <1> ; /// BL = 4 -> system to system (window) transfer

2751 <1> ; ESI = System's source buffer (video page) address

2752 <1> ; ECX Low 16 bits = Top left column (X1 position)

2753 <1> ; ECX High 16 bits = Top row (Y1 position)

2754 <1> ; EDX Low 16 bits = Bottom right column (X2 position)

2755 <1> ; EDX High 16 bits = Bottom row (Y2 position)

2756 <1> ; EDI = System's destination buffer (video page) address

2757 <1> ; /// BL = 3 -> NOT byte in display page/memory

2758 <1> ; ECX Low 16 bits = Top left column (X1 position)

2759 <1> ; ECX High 16 bits = Top row (Y1 position)

2760 <1> ; EDX Low 16 bits = Bottom right column (X2 position)

2761 <1> ; EDX High 16 bits = Bottom row (Y2 position)

2762 <1> ;

2763 <1> ; Outputs:

2764 <1> ; EAX = transfer/byte count

2765 <1> ;

2766 <1> ; NOTE: If the source or destination address passes out of

2767 <1> ; video pages (display memory limits), data will not be transferred

2768 <1> ; and EAX will return as 0.

2769 <1> ;

2770 <1> ;

2771 <1> ; DIRECT (STANDARD VGA/CGA) DISPLAY MEMORY ACCESS FUNCTIONS:

2772 <1> ;

2773 <1> ; BH = 4 = CGA direct video memory (0B8000h, 32K) access

2774 <1> ; Page directory & page tables of the user's

2775 <1> ; program will be updated to direct access to

2776 <1> ; 0B8000h (32K) video (CGA, color) memory; if

2777 <1> ; there is not a permission conflict or lock!

2778 <1> ; (User's program/process will have permision to

2779 <1> ; access locked display memory if the owner is

2780 <1> ; it's parent.)

2781 <1> ;

2782 <1> ; Screen width = 320

2783 <1> ;

2784 <1> ; BH = 5 = VGA direct video memory (0A0000h, 64K) access

2785 <1> ; Page directory & page tables of the user's

2786 <1> ; program will be updated to direct access to

2787 <1> ; 0A0000h (64K) video (VGA) memory; if there is not

2788 <1> ; a permission conflict or lock!

2789 <1> ; (User's program/process will have permision to

2790 <1> ; access locked display memory if the owner is

2791 <1> ; it's parent.)

2792 <1> ;

2793 <1> ; BL = Screen width (320, 640, 800)

2794 <1> ;

2795 <1> ; Outputs:

2796 <1> ; EAX = Display mmory address for direct access

2797 <1> ; 0A0000h for VGA, 0B8000h for CGA

2798 <1> ; (Display memory size: 32K for CGA, 64K for VGA)

2799 <1> ; EAX = 0 if display page access permission has been denied.

2800 <1> ; (Locked!)

2801 <1> ;

2802 <1> ; LINEAR FRAME BUFFER ACCESS FUNCTIONS:

2803 <1> ;

2804 <1> ; BH = 6 = Linear Frame Buffer direct video memory access

2805 <1> ;

2806 <1> ; Page directory & page tables of the user's

2807 <1> ; program will be updated to direct access to

2808 <1> ; the configured LFB (Linear Frame Buffer) address,

2809 <1> ; if there is not a permission conflict or lock!

2810 <1> ; (User's program/process will have permision to

2811 <1> ; access locked display memory if the owner is

2812 <1> ; it's parent.)

2813 <1> ;

2814 <1> ; Return: EAX = Linear Frame Buffer address

2815 <1> ; EDX = Frame Buffer Size in bytes

2816 <1> ;

2817 <1> ; BH = 7 = Get Linear Frame Buffer info (for current mode)

2818 <1> ;

2819 <1> ; Return:

2820 <1> ; EAX = Frame Buffer Address (0 = is not in use)

2821 <1> ; EDX = Frame Buffer Size in bytes

2822 <1> ; BL = Current Video Mode

2823 <1> ; BL = 0FFh -> Super VGA (Extended VGA)

2824 <1> ; If BL = 0FFh,

2825 <1> ; BH = 0 = 16 colors

2826 <1> ; BH = 1 = 256 colors

2827 <1> ; BH = 2 = 66536 colors

2828 <1> ; BH = 3 = 24 bits TRUE (16M) colors

2829 <1> ; BH = 4 = 32 bits TRUE (16M) colors

2830 <1> ; ECX = Pixel resolution

2831 <1> ; CX = Width (640, 800, 1024, 1366, 1920)

2832 <1> ; High 16 bits of ECX = Height

2833 <1> ;

2834 <1> ; NOTE: Each process will have it's own frame buffer

2835 <1> ; address and resolution parameters in 'u' area.

2836 <1> ; Then, if the current frame buffer & resolution

2837 <1> ; is different, frame buffer r/w functions

2838 <1> ; will use scale factor to convert process's

2839 <1> ; pixel coordinates to actual screen coordinates.

2840 <1> ; resolution -> dimensional scale

2841 <1> ; color size -> color scale

2842 <1> ; \* RGB (TRUE) colors to 256 colors conversion:

2843 <1> ; TRUE Colors -> 8,8,8 (R,G,B; byte 0 is R)

2844 <1> ; 256 colors -> 2,2,2,2 (R,G,B,L; bit 0&1 is R)

2845 <1> ; bit 6&7 -> luminosity base level (0,1,2,3)

2846 <1> ; bit 4&5 -> blue level (0,1,2,3)

2847 <1> ; bit 2%3 -> green level (0,1,2,3)

2848 <1> ; bit 0&1 -> red level (0,1,2,3)

2849 <1> ; Example: total red level : luminosity + red level

2850 <1> ; Luminosity base level: 0 -> 16

2851 <1> ; 1 -> 32

2852 <1> ; 2 -> 64

2853 <1> ; 3 -> 128

2854 <1> ; Color level:

2855 <1> ; 0 -> 0

2856 <1> ; 1 -> luminosity level

2857 <1> ; 2 -> luminosity level + 64

2858 <1> ; 3 -> 255

2859 <1> ; Luminosity base level = min (R,G,B)

2860 <1> ; if it is <16, it will be set to 16

2861 <1> ; Color levels: Color values are fixed to (nearest)

2862 <1> ; one of all possible set level (step) values

2863 <1> ; (according to luminosity base level); then

2864 <1> ; color levels are set to R-L, G-L, B-L.

2865 <1> ; For example: If luminosity base level is 32

2866 <1> ; all possible set values are 0, 32, 96, 255.

2867 <1> ;

2868 <1> ; \* RGB (TRUE) colors to 16 colors conversion:

2869 <1> ; 16 colors: R, B,G, L bits (4 bits)

2870 <1> ; If any one of R,G,B >= 128 L = 1

2871 <1> ; If max. value of (R,G,B) >= 32, it is 1

2872 <1> ; else all color bits (R&G&B&L) are 0

2873 <1> ; If the second value >= max. value / 2

2874 <1> ; it is 1

2875 <1> ; If third value value >= max. value / 2

2876 <1> ; it is 1

2877 <1> ; Example: R = 132, G = 64, B = 78

2878 <1> ; L = 1, R = 1

2879 <1> ; G < 66 --> G = 0

2880 <1> ; B >= 66 --> B = 1

2881 <1>

2882 <1> ; 16/05/2016

2883 0000D0D4 31C0 <1> xor eax, eax

2884 0000D0D6 A3[64030300] <1> mov [u.r0], eax

2885 <1>

2886 0000D0DB 20FF <1> and bh, bh

2887 0000D0DD 0F8572020000 <1> jnz sysvideo\_13 ; 11/07/2016

2888 <1>

2889 <1> ; Video mode 0, 80\*25 text mode, CGA 16 colors ; [CRT\_MODE] = 3

2890 0000D0E3 88DF <1> mov bh, bl

2891 0000D0E5 C0EF02 <1> shr bh, 2

2892 0000D0E8 20FF <1> and bh, bh

2893 0000D0EA 0F8598000000 <1> jnz sysvideo\_4

2894 0000D0F0 BF00800B00 <1> mov edi, 0B8000h

2895 0000D0F5 20D2 <1> and dl, dl

2896 0000D0F7 7413 <1> jz short sysvideo\_1

2897 0000D0F9 80FA07 <1> cmp dl, 7

2898 0000D0FC 0F87DCF5FFFF <1> ja sysret

2899 <1> sysvideo\_0:

2900 0000D102 81C7A00F0000 <1> add edi, 80\*25\*2

2901 0000D108 FECA <1> dec dl

2902 0000D10A 75F6 <1> jnz short sysvideo\_0

2903 <1> sysvideo\_1:

2904 0000D10C 80E303 <1> and bl, 3

2905 0000D10F 7530 <1> jnz short sysvideo\_2

2906 0000D111 80F907 <1> cmp cl, 7

2907 0000D114 0F87C4F5FFFF <1> ja sysret

2908 <1> ; system to system video/display page transfer (mode 0)

2909 0000D11A BE00800B00 <1> mov esi, 0B8000h

2910 0000D11F 0FB6C1 <1> movzx eax, cl

2911 0000D122 BAA00F0000 <1> mov edx, 80\*25\*2

2912 0000D127 F7E2 <1> mul edx

2913 0000D129 01C6 <1> add esi, eax

2914 0000D12B B9A00F0000 <1> mov ecx, (80\*25\*2)

2915 0000D130 890D[64030300] <1> mov [u.r0], ecx

2916 0000D136 66C1E902 <1> shr cx, 2 ; /4

2917 0000D13A F3A5 <1> rep movsd

2918 0000D13C E99DF5FFFF <1> jmp sysret

2919 <1> sysvideo\_2:

2920 0000D141 80FB02 <1> cmp bl, 2

2921 0000D144 0F8794F5FFFF <1> ja sysret

2922 0000D14A 721F <1> jb short sysvideo\_3

2923 <1> ; system to user video/display page transfer (mode 0)

2924 0000D14C 89FE <1> mov esi, edi

2925 0000D14E 89CF <1> mov edi, ecx ; user buffer

2926 0000D150 B9A00F0000 <1> mov ecx, 80\*25\*2

2927 0000D155 E81F160000 <1> call transfer\_to\_user\_buffer ; fast transfer

2928 0000D15A 0F827EF5FFFF <1> jc sysret

2929 0000D160 890D[64030300] <1> mov [u.r0], ecx

2930 0000D166 E973F5FFFF <1> jmp sysret

2931 <1> sysvideo\_3:

2932 <1> ; user to system video/display page transfer (mode 0)

2933 0000D16B 89CE <1> mov esi, ecx ; user buffer

2934 <1> ; edi = video page address

2935 0000D16D B9A00F0000 <1> mov ecx, 80\*25\*2

2936 0000D172 E84C160000 <1> call transfer\_from\_user\_buffer ; fast transfer

2937 0000D177 0F8261F5FFFF <1> jc sysret

2938 0000D17D 890D[64030300] <1> mov [u.r0], ecx

2939 0000D183 E956F5FFFF <1> jmp sysret

2940 <1> sysvideo\_4:

2941 0000D188 80E303 <1> and bl, 3

2942 0000D18B 0F85F6000000 <1> jnz sysvideo\_9

2943 0000D191 80F907 <1> cmp cl, 7

2944 0000D194 0F8744F5FFFF <1> ja sysret

2945 <1> ; system to system video/display page window transfer (mode 0)

2946 0000D19A 81FE00800B00 <1> cmp esi, 0B8000h

2947 0000D1A0 0F8238F5FFFF <1> jb sysret

2948 0000D1A6 81FE00FD0B00 <1> cmp esi, 0B8000h+(80\*25\*2\*8)

2949 0000D1AC 0F832CF5FFFF <1> jnb sysret

2950 0000D1B2 81FF00800B00 <1> cmp edi, 0B8000h

2951 0000D1B8 0F8220F5FFFF <1> jb sysret

2952 0000D1BE 81FF00FD0B00 <1> cmp edi, 0B8000h+(80\*25\*2\*8)

2953 0000D1C4 0F8314F5FFFF <1> jnb sysret

2954 <1> ;

2955 0000D1CA 51 <1> push ecx

2956 0000D1CB 52 <1> push edx

2957 0000D1CC 0FB7C1 <1> movzx eax, cx ; top left column

2958 0000D1CF 50 <1> push eax

2959 0000D1D0 C1E910 <1> shr ecx, 16 ; top row

2960 0000D1D3 66B8A000 <1> mov ax, 80\*2 ; 80 colums, 160 bytes per row

2961 0000D1D7 F7E1 <1> mul ecx

2962 0000D1D9 01C6 <1> add esi, eax

2963 0000D1DB 01C7 <1> add edi, eax

2964 0000D1DD 58 <1> pop eax

2965 0000D1DE 66D1E0 <1> shl ax, 1 ; \*2

2966 0000D1E1 01C6 <1> add esi, eax

2967 0000D1E3 01C7 <1> add edi, eax

2968 0000D1E5 5A <1> pop edx

2969 0000D1E6 59 <1> pop ecx

2970 0000D1E7 B800FD0B00 <1> mov eax, 0B8000h+(80\*25\*2\*8)

2971 0000D1EC 39C6 <1> cmp esi, eax

2972 0000D1EE 0F83EAF4FFFF <1> jnb sysret

2973 0000D1F4 39C6 <1> cmp esi, eax

2974 0000D1F6 0F83E2F4FFFF <1> jnb sysret

2975 <1>

2976 0000D1FC 56 <1> push esi ; \*\*\*\*

2977 0000D1FD 57 <1> push edi ; \*\*\*

2978 0000D1FE 52 <1> push edx ; \*\*

2979 0000D1FF 51 <1> push ecx ; \*

2980 0000D200 C1E910 <1> shr ecx, 16 ; top row

2981 0000D203 C1EA10 <1> shr edx, 16 ; bottom row

2982 0000D206 83F918 <1> cmp ecx, 24 ; max. 25 rows

2983 0000D209 7773 <1> ja short sysvideo\_6

2984 0000D20B 83FA18 <1> cmp edx, 24 ; max. 25 rows

2985 0000D20E 776E <1> ja short sysvideo\_6

2986 0000D210 28CA <1> sub dl, cl

2987 0000D212 726A <1> jc short sysvideo\_6

2988 0000D214 50 <1> push eax ; \*\*\*\*\*

2989 0000D215 89D3 <1> mov ebx, edx ; row count - 1

2990 0000D217 B8A0000000 <1> mov eax, 80\*2

2991 0000D21C F7E0 <1> mul eax

2992 0000D21E 01C6 <1> add esi, eax

2993 0000D220 01C7 <1> add edi, eax

2994 0000D222 58 <1> pop eax ; \*\*\*\*\*

2995 0000D223 39C6 <1> cmp esi, eax

2996 0000D225 7757 <1> ja short sysvideo\_6

2997 0000D227 39C7 <1> cmp edi, eax

2998 0000D229 7753 <1> ja short sysvideo\_6

2999 0000D22B 59 <1> pop ecx ; \*

3000 0000D22C 5A <1> pop edx ; \*\*

3001 0000D22D 81E1FFFF0000 <1> and ecx, 0FFFFh

3002 0000D233 81E2FFFF0000 <1> and edx, 0FFFFh

3003 0000D239 83F94F <1> cmp ecx, 79 ; max. 80 columns

3004 0000D23C 7742 <1> ja short sysvideo\_7

3005 0000D23E 83FA4F <1> cmp edx, 79 ; max. 80 columns

3006 0000D241 773D <1> ja short sysvideo\_7

3007 0000D243 28CA <1> sub dl, cl

3008 0000D245 7639 <1> jna short sysvideo\_7

3009 <1> ; edx = column count (width) - 1

3010 0000D247 D0E2 <1> shl dl, 1

3011 0000D249 01D6 <1> add esi, edx

3012 0000D24B 01D7 <1> add edi, edx

3013 0000D24D 39C6 <1> cmp esi, eax

3014 0000D24F 772F <1> ja short sysvideo\_7

3015 0000D251 39C7 <1> cmp edi, eax

3016 0000D253 772B <1> ja short sysvideo\_7

3017 0000D255 5F <1> pop edi ; \*\*\*

3018 0000D256 5E <1> pop esi ; \*\*\*\*

3019 0000D257 FEC3 <1> inc bl

3020 0000D259 FEC2 <1> inc dl ; column count

3021 0000D25B 88D7 <1> mov bh, dl

3022 0000D25D D0E2 <1> shl dl, 1

3023 0000D25F B8A0000000 <1> mov eax, 80\*2

3024 0000D264 28D0 <1> sub al, dl ; (80 - columns) \* 2

3025 <1> sysvideo\_5:

3026 0000D266 88F9 <1> mov cl, bh

3027 0000D268 0115[64030300] <1> add [u.r0], edx

3028 0000D26E F366A5 <1> rep movsw

3029 0000D271 01C6 <1> add esi, eax ; next row

3030 0000D273 01C7 <1> add edi, eax ; next row

3031 0000D275 FECB <1> dec bl

3032 0000D277 75ED <1> jnz short sysvideo\_5

3033 0000D279 E960F4FFFF <1> jmp sysret

3034 <1>

3035 <1> sysvideo\_6:

3036 0000D27E 59 <1> pop ecx ; \*

3037 0000D27F 5A <1> pop edx ; \*\*

3038 <1> sysvideo\_7:

3039 0000D280 5F <1> pop edi ; \*\*\*

3040 0000D281 5E <1> pop esi ; \*\*\*\*

3041 0000D282 E957F4FFFF <1> jmp sysret

3042 <1>

3043 <1> sysvideo\_9:

3044 0000D287 80FB02 <1> cmp bl, 2

3045 0000D28A 0F874EF4FFFF <1> ja sysret

3046 <1>

3047 0000D290 56 <1> push esi ; \*\*\*\*

3048 0000D291 57 <1> push edi ; \*\*\*

3049 0000D292 52 <1> push edx ; \*\*

3050 0000D293 51 <1> push ecx ; \*

3051 <1>

3052 0000D294 C1E910 <1> shr ecx, 16 ; top row

3053 0000D297 C1EA10 <1> shr edx, 16 ; bottom row

3054 0000D29A 83F918 <1> cmp ecx, 24 ; max. 25 rows

3055 0000D29D 77DF <1> ja short sysvideo\_6

3056 0000D29F 83FA18 <1> cmp edx, 24 ; max. 25 rows

3057 0000D2A2 77DA <1> ja short sysvideo\_6

3058 0000D2A4 28CA <1> sub dl, cl

3059 0000D2A6 72D6 <1> jc short sysvideo\_6

3060 <1>

3061 0000D2A8 88CD <1> mov ch, cl ; top row

3062 0000D2AA 8A0D[66580100] <1> mov cl, [ACTIVE\_PAGE]

3063 0000D2B0 BFA00F0000 <1> mov edi, 80\*25\*2

3064 0000D2B5 D3E7 <1> shl edi, cl

3065 0000D2B7 81C760700B00 <1> add edi, 0B8000h - 80\*25\*2

3066 <1>

3067 0000D2BD 88D7 <1> mov bh, dl ; row count - 1

3068 0000D2BF 88EA <1> mov dl, ch ; top row

3069 0000D2C1 B8A0000000 <1> mov eax, 80\*2

3070 0000D2C6 F7E2 <1> mul edx

3071 0000D2C8 01C7 <1> add edi, eax

3072 <1>

3073 0000D2CA 59 <1> pop ecx ; \*

3074 0000D2CB 5A <1> pop edx ; \*\*

3075 0000D2CC 81E1FFFF0000 <1> and ecx, 0FFFFh

3076 0000D2D2 81E2FFFF0000 <1> and edx, 0FFFFh

3077 0000D2D8 83F94F <1> cmp ecx, 79 ; max. 80 columns

3078 0000D2DB 77A3 <1> ja short sysvideo\_7

3079 0000D2DD 83FA4F <1> cmp edx, 79 ; max. 80 columns

3080 0000D2E0 779E <1> ja short sysvideo\_7

3081 <1>

3082 0000D2E2 28CA <1> sub dl, cl

3083 0000D2E4 769A <1> jna short sysvideo\_7

3084 <1>

3085 0000D2E6 0FB6C1 <1> movzx eax, cl ; left column

3086 0000D2E9 D0E0 <1> shl al, 1 ; column \* 2

3087 0000D2EB 01C7 <1> add edi, eax

3088 <1>

3089 0000D2ED FEC2 <1> inc dl ; column count

3090 0000D2EF D0E2 <1> shl dl, 1

3091 0000D2F1 88D1 <1> mov cl, dl ; column count \* 2

3092 0000D2F3 B2A0 <1> mov dl, 80\*2

3093 0000D2F5 58 <1> pop eax ; \*\*\* (swap address)

3094 0000D2F6 5E <1> pop esi ; \*\*\*\*

3095 0000D2F7 FEC7 <1> inc bh

3096 <1>

3097 <1> ;mov edx, 80\*2

3098 0000D2F9 B2A0 <1> mov dl, 80\*2

3099 <1> ;

3100 0000D2FB 80FB01 <1> cmp bl, 1

3101 0000D2FE 7735 <1> ja short sysvideo\_11

3102 <1>

3103 <1> ; user to system video/display page window transfer (mode 0)

3104 0000D300 21C0 <1> and eax, eax ; swap address

3105 0000D302 7413 <1> jz short sysvideo\_10 ; no window swap

3106 <1> ; save previous window content in user's buffer (swap address)

3107 0000D304 56 <1> push esi ; user buffer

3108 0000D305 57 <1> push edi ; beginning address of the window

3109 0000D306 89FE <1> mov esi, edi

3110 0000D308 89C7 <1> mov edi, eax

3111 0000D30A E86A140000 <1> call transfer\_to\_user\_buffer ; fast transfer

3112 0000D30F 5F <1> pop edi

3113 0000D310 5E <1> pop esi

3114 0000D311 0F82C7F3FFFF <1> jc sysret

3115 <1> sysvideo\_10:

3116 <1> ; user to system video/display page window transfer (mode 0)

3117 <1> ; esi = user buffer

3118 0000D317 E8A7140000 <1> call transfer\_from\_user\_buffer ; fast transfer

3119 0000D31C 0F82BCF3FFFF <1> jc sysret

3120 0000D322 010D[64030300] <1> add [u.r0], ecx

3121 0000D328 01D7 <1> add edi, edx ; next row

3122 0000D32A 01CE <1> add esi, ecx

3123 0000D32C FECF <1> dec bh

3124 0000D32E 75E7 <1> jnz short sysvideo\_10

3125 0000D330 E9A9F3FFFF <1> jmp sysret

3126 <1>

3127 <1> sysvideo\_11:

3128 <1> ; system to user video/display page window transfer (mode 0)

3129 0000D335 87FE <1> xchg edi, esi

3130 <1> sysvideo\_12:

3131 <1> ; esi = beginning address of the window

3132 <1> ; edi = user buffer

3133 0000D337 E83D140000 <1> call transfer\_to\_user\_buffer ; fast transfer

3134 0000D33C 0F829CF3FFFF <1> jc sysret

3135 0000D342 010D[64030300] <1> add [u.r0], ecx

3136 0000D348 01D6 <1> add esi, edx ; next row

3137 0000D34A 01CF <1> add edi, ecx

3138 0000D34C FECF <1> dec bh

3139 0000D34E 75E7 <1> jnz short sysvideo\_12

3140 0000D350 E989F3FFFF <1> jmp sysret

3141 <1>

3142 <1> sysvideo\_13:

3143 0000D355 80FF01 <1> cmp bh, 1

3144 0000D358 0F871F030000 <1> ja sysvideo\_38

3145 <1> ; BH = 1 = CGA Graphics (0B8000h) data transfers

3146 <1>

3147 0000D35E 20DB <1> and bl, bl

3148 0000D360 751A <1> jnz short sysvideo\_14

3149 <1>

3150 <1> ; BL = 0 = Fill color (color in CL] (32K)

3151 <1>

3152 0000D362 88C8 <1> mov al, cl

3153 0000D364 B900800000 <1> mov ecx, 32768

3154 0000D369 66890D[64030300] <1> mov [u.r0], cx

3155 0000D370 BF00800B00 <1> mov edi, 0B8000h

3156 0000D375 F3AB <1> rep stosd

3157 0000D377 E962F3FFFF <1> jmp sysret

3158 <1>

3159 <1> sysvideo\_14:

3160 0000D37C 80FB01 <1> cmp bl, 1

3161 0000D37F 7723 <1> ja short sysvideo\_16

3162 <1>

3163 0000D381 89CE <1> mov esi, ecx ; user buffer

3164 <1> ; BL = 1 = user to system video/display page transfer

3165 <1> sysvideo\_15:

3166 0000D383 BF00800B00 <1> mov edi, 0B8000h

3167 <1> ; edi = video page address

3168 0000D388 B900800000 <1> mov ecx, 32768

3169 0000D38D E831140000 <1> call transfer\_from\_user\_buffer ; fast transfer

3170 0000D392 0F8246F3FFFF <1> jc sysret ; [u.r0] = 0

3171 0000D398 66890D[64030300] <1> mov [u.r0], cx

3172 0000D39F E93AF3FFFF <1> jmp sysret

3173 <1>

3174 <1> sysvideo\_16:

3175 0000D3A4 80FB02 <1> cmp bl, 2

3176 0000D3A7 7723 <1> ja short sysvideo\_18

3177 <1>

3178 0000D3A9 89CF <1> mov edi, ecx ; user buffer

3179 <1> ; BL = 2 = system to user video/display page transfer

3180 <1> sysvideo\_17:

3181 0000D3AB BE00800B00 <1> mov esi, 0B8000h

3182 0000D3B0 B900800000 <1> mov ecx, 32768

3183 0000D3B5 E8BF130000 <1> call transfer\_to\_user\_buffer ; fast transfer

3184 0000D3BA 0F821EF3FFFF <1> jc sysret ; [u.r0] = 0

3185 0000D3C0 66890D[64030300] <1> mov [u.r0], cx

3186 0000D3C7 E912F3FFFF <1> jmp sysret

3187 <1>

3188 <1> sysvideo\_18:

3189 0000D3CC 80FB03 <1> cmp bl, 3

3190 0000D3CF 777E <1> ja short sysvideo\_23

3191 <1>

3192 <1> ; BL = 3 = NOT bits in window (ECX, EDX)

3193 <1>

3194 0000D3D1 BF00800B00 <1> mov edi, 0B8000h

3195 0000D3D6 89FE <1> mov esi, edi

3196 <1>

3197 0000D3D8 39CA <1> cmp edx, ecx ; bottom-right > top-left ?

3198 0000D3DA 7716 <1> ja short sysvideo\_20 ; window

3199 <1> ; full screen (update)

3200 0000D3DC B900800000 <1> mov ecx, 32768

3201 0000D3E1 66890D[64030300] <1> mov [u.r0], cx

3202 <1> sysvideo\_19:

3203 0000D3E8 F616 <1> not byte [esi] ; NOT operation

3204 0000D3EA 46 <1> inc esi

3205 0000D3EB E2FB <1> loop sysvideo\_19

3206 0000D3ED E9ECF2FFFF <1> jmp sysret

3207 <1> sysvideo\_20:

3208 0000D3F2 0FB7C2 <1> movzx eax, dx ; bottom right column

3209 0000D3F5 6629C8 <1> sub ax, cx ; - top left column

3210 0000D3F8 0F82E0F2FFFF <1> jb sysret ; invalid

3211 0000D3FE 6640 <1> inc ax ; same column no == 1 column

3212 0000D400 50 <1> push eax ; byte count per window row

3213 0000D401 52 <1> push edx

3214 0000D402 BB40010000 <1> mov ebx, 320 ; screen width

3215 0000D407 89C8 <1> mov eax, ecx

3216 0000D409 C1E810 <1> shr eax, 16 ; top row

3217 0000D40C F7E3 <1> mul ebx

3218 0000D40E 6689CA <1> mov dx, cx ; top left column

3219 0000D411 01D0 <1> add eax, edx

3220 0000D413 01C6 <1> add esi, eax ; start address

3221 0000D415 59 <1> pop ecx ; edx

3222 0000D416 89C8 <1> mov eax, ecx

3223 0000D418 C1E810 <1> shr eax, 16 ; bottom row

3224 0000D41B F7E3 <1> mul ebx

3225 0000D41D 6689CA <1> mov dx, cx ; bottom right column

3226 0000D420 01D0 <1> add eax, edx

3227 0000D422 01C7 <1> add edi, eax ; stop address (included)

3228 0000D424 5A <1> pop edx ; byte count per window row

3229 0000D425 81FFFFFF0B00 <1> cmp edi, 0BFFFFh

3230 0000D42B 0F87ADF2FFFF <1> ja sysret

3231 0000D431 56 <1> push esi

3232 0000D432 4E <1> dec esi

3233 <1> sysvideo\_21:

3234 0000D433 89D1 <1> mov ecx, edx

3235 <1> sysvideo\_22:

3236 0000D435 46 <1> inc esi

3237 0000D436 F616 <1> not byte [esi]

3238 0000D438 E2FB <1> loop sysvideo\_22

3239 0000D43A 01DE <1> add esi, ebx ; bytes per screen row

3240 <1> ;

3241 0000D43C 39FE <1> cmp esi, edi ; stop address (included in loop)

3242 0000D43E 76F3 <1> jna short sysvideo\_21

3243 0000D440 5E <1> pop esi

3244 0000D441 29F7 <1> sub edi, esi

3245 0000D443 66893D[64030300] <1> mov [u.r0], di

3246 0000D44A E98FF2FFFF <1> jmp sysret

3247 <1>

3248 <1> sysvideo\_23:

3249 0000D44F 80FB04 <1> cmp bl, 4

3250 0000D452 0F87A7000000 <1> ja sysvideo\_26

3251 <1>

3252 <1> ; BL = 4 = window copy (system to system)

3253 <1>

3254 0000D458 B800800B00 <1> mov eax, 0B8000h

3255 0000D45D 39C6 <1> cmp esi, eax

3256 0000D45F 0F8279F2FFFF <1> jb sysret

3257 0000D465 39C7 <1> cmp edi, eax

3258 0000D467 0F8271F2FFFF <1> jb sysret

3259 0000D46D 6605FF7F <1> add ax, 7FFFh ; 32767

3260 0000D471 39C6 <1> cmp esi, eax

3261 0000D473 0F8765F2FFFF <1> ja sysret

3262 0000D479 39C7 <1> cmp edi, eax

3263 0000D47B 0F875DF2FFFF <1> ja sysret

3264 <1>

3265 0000D481 39CA <1> cmp edx, ecx ; bottom-right > top-left ?

3266 0000D483 7714 <1> ja short sysvideo\_24 ; window

3267 <1> ; full screen copy

3268 0000D485 89C1 <1> mov ecx, eax

3269 0000D487 29F9 <1> sub ecx, edi

3270 0000D489 6641 <1> inc cx

3271 0000D48B 66890D[64030300] <1> mov [u.r0], cx

3272 0000D492 F3A4 <1> rep movsb

3273 0000D494 E945F2FFFF <1> jmp sysret

3274 <1> sysvideo\_24:

3275 0000D499 0FB7C2 <1> movzx eax, dx ; bottom right column

3276 0000D49C 6629C8 <1> sub ax, cx ; - top left column

3277 0000D49F 0F8239F2FFFF <1> jb sysret ; invalid

3278 0000D4A5 6640 <1> inc ax ; same column no == 1 column

3279 0000D4A7 50 <1> push eax ; byte count per window row

3280 <1> ;

3281 0000D4A8 52 <1> push edx

3282 0000D4A9 BB40010000 <1> mov ebx, 320 ; screen width

3283 0000D4AE 89C8 <1> mov eax, ecx

3284 0000D4B0 C1E810 <1> shr eax, 16 ; top row

3285 0000D4B3 F7E3 <1> mul ebx

3286 0000D4B5 6689CA <1> mov dx, cx ; top left column

3287 0000D4B8 01D0 <1> add eax, edx

3288 0000D4BA 01C7 <1> add edi, eax ; start address

3289 0000D4BC 01C6 <1> add esi, eax

3290 0000D4BE 59 <1> pop ecx ; edx

3291 0000D4BF 89C8 <1> mov eax, ecx

3292 0000D4C1 C1E810 <1> shr eax, 16 ; bottom row

3293 0000D4C4 F7E3 <1> mul ebx

3294 0000D4C6 6689CA <1> mov dx, cx ; bottom right column

3295 0000D4C9 01D0 <1> add eax, edx

3296 0000D4CB 5A <1> pop edx ; byte count per window row

3297 0000D4CC 0500800B00 <1> add eax, 0B8000h

3298 0000D4D1 3DFFFF0B00 <1> cmp eax, 0BFFFFh

3299 0000D4D6 0F8702F2FFFF <1> ja sysret

3300 0000D4DC 57 <1> push edi ; start address

3301 0000D4DD 50 <1> push eax ; stop address (included)

3302 <1> sysvideo\_25:

3303 0000D4DE 89D1 <1> mov ecx, edx

3304 0000D4E0 F3A4 <1> rep movsb

3305 0000D4E2 4F <1> dec edi

3306 0000D4E3 4E <1> dec esi

3307 0000D4E4 01DF <1> add edi, ebx ; bytes per screen row

3308 0000D4E6 01DE <1> add esi, ebx

3309 <1> ;

3310 0000D4E8 3B3C24 <1> cmp edi, [esp] ; stop addr(included in loop)

3311 0000D4EB 76F1 <1> jna short sysvideo\_25

3312 0000D4ED 5B <1> pop ebx ; stop address

3313 0000D4EE 5F <1> pop edi ; start address

3314 0000D4EF 29FB <1> sub ebx, edi

3315 0000D4F1 6643 <1> inc bx

3316 0000D4F3 66891D[64030300] <1> mov [u.r0], bx

3317 0000D4FA E9DFF1FFFF <1> jmp sysret

3318 <1>

3319 <1> sysvideo\_26:

3320 0000D4FF 80FB05 <1> cmp bl, 5

3321 0000D502 0F8795000000 <1> ja sysvideo\_29

3322 <1>

3323 <1> ; BL = 5 = window copy (user to system)

3324 <1>

3325 0000D508 B800800B00 <1> mov eax, 0B8000h

3326 0000D50D 39C7 <1> cmp edi, eax

3327 0000D50F 0F82C9F1FFFF <1> jb sysret

3328 0000D515 6605FF7F <1> add ax, 7FFFh ; 32767

3329 0000D519 39C7 <1> cmp edi, eax

3330 0000D51B 0F87BDF1FFFF <1> ja sysret

3331 <1>

3332 <1> ; esi = user buffer (in user's memory space)

3333 0000D521 39CA <1> cmp edx, ecx ; bottom-right > top-left ?

3334 0000D523 0F865AFEFFFF <1> jna sysvideo\_15 ; full screen copy

3335 <1>

3336 0000D529 0FB7C2 <1> movzx eax, dx ; bottom right column

3337 0000D52C 6629C8 <1> sub ax, cx ; - top left column

3338 0000D52F 0F82A9F1FFFF <1> jb sysret ; invalid

3339 0000D535 6640 <1> inc ax ; same column no == 1 column

3340 0000D537 50 <1> push eax ; byte count per window row

3341 <1>

3342 0000D538 52 <1> push edx

3343 0000D539 BB40010000 <1> mov ebx, 320 ; screen width

3344 0000D53E 89C8 <1> mov eax, ecx

3345 0000D540 C1E810 <1> shr eax, 16 ; top row

3346 0000D543 F7E3 <1> mul ebx

3347 0000D545 6689CA <1> mov dx, cx ; top left column

3348 0000D548 01D0 <1> add eax, edx

3349 0000D54A 01C7 <1> add edi, eax ; start address

3350 0000D54C 59 <1> pop ecx ; edx

3351 0000D54D 89C8 <1> mov eax, ecx

3352 0000D54F C1E810 <1> shr eax, 16 ; bottom row

3353 0000D552 F7E3 <1> mul ebx

3354 0000D554 6689CA <1> mov dx, cx ; bottom right column

3355 0000D557 01D0 <1> add eax, edx

3356 0000D559 5A <1> pop edx ; byte count per window row

3357 0000D55A 0500800B00 <1> add eax, 0B8000h

3358 0000D55F 3DFFFF0B00 <1> cmp eax, 0BFFFFh

3359 0000D564 0F8774F1FFFF <1> ja sysret

3360 0000D56A 57 <1> push edi ; start address

3361 0000D56B 50 <1> push eax ; stop address (included)

3362 <1> sysvideo\_27:

3363 0000D56C 89D1 <1> mov ecx, edx ; byte count

3364 <1> ; user to system video/display page window transfer

3365 <1> ; esi = user buffer

3366 0000D56E E850120000 <1> call transfer\_from\_user\_buffer ; fast transfer

3367 0000D573 7221 <1> jc short sysvideo\_28

3368 0000D575 010D[64030300] <1> add [u.r0], ecx

3369 0000D57B 01DF <1> add edi, ebx ; next row

3370 0000D57D 01CE <1> add esi, ecx

3371 0000D57F 3B3C24 <1> cmp edi, [esp] ; stop addr(included in loop)

3372 0000D582 76E8 <1> jna short sysvideo\_27

3373 0000D584 5B <1> pop ebx ; stop address

3374 0000D585 5F <1> pop edi ; start address

3375 0000D586 29FB <1> sub ebx, edi

3376 0000D588 6643 <1> inc bx

3377 0000D58A 66891D[64030300] <1> mov [u.r0], bx

3378 0000D591 E948F1FFFF <1> jmp sysret

3379 <1> sysvideo\_28:

3380 0000D596 58 <1> pop eax

3381 0000D597 5A <1> pop edx

3382 0000D598 E941F1FFFF <1> jmp sysret

3383 <1>

3384 <1> sysvideo\_29:

3385 0000D59D 80FB06 <1> cmp bl, 6

3386 0000D5A0 0F8797000000 <1> ja sysvideo\_32

3387 <1>

3388 <1> ; BL = 6 = window copy (system to user)

3389 <1>

3390 0000D5A6 89F7 <1> mov edi, esi ; user buffer

3391 <1>

3392 0000D5A8 B800800B00 <1> mov eax, 0B8000h

3393 0000D5AD 39C6 <1> cmp esi, eax

3394 0000D5AF 0F8229F1FFFF <1> jb sysret

3395 0000D5B5 6605FF7F <1> add ax, 7FFFh ; 32767

3396 0000D5B9 39C6 <1> cmp esi, eax

3397 0000D5BB 0F871DF1FFFF <1> ja sysret

3398 <1>

3399 <1> ; edi = user buffer (in user's memory space)

3400 0000D5C1 39CA <1> cmp edx, ecx ; bottom-right > top-left ?

3401 0000D5C3 0F86E2FDFFFF <1> jna sysvideo\_17 ; full screen copy

3402 <1>

3403 0000D5C9 0FB7C2 <1> movzx eax, dx ; bottom right column

3404 0000D5CC 6629C8 <1> sub ax, cx ; - top left column

3405 0000D5CF 0F8209F1FFFF <1> jb sysret ; invalid

3406 0000D5D5 6640 <1> inc ax ; same column no == 1 column

3407 0000D5D7 50 <1> push eax ; byte count per window row

3408 <1>

3409 0000D5D8 52 <1> push edx

3410 0000D5D9 BB40010000 <1> mov ebx, 320 ; screen width

3411 0000D5DE 89C8 <1> mov eax, ecx

3412 0000D5E0 C1E810 <1> shr eax, 16 ; top row

3413 0000D5E3 F7E3 <1> mul ebx

3414 0000D5E5 6689CA <1> mov dx, cx ; top left column

3415 0000D5E8 01D0 <1> add eax, edx

3416 0000D5EA 01C6 <1> add esi, eax ; start address

3417 0000D5EC 59 <1> pop ecx ; edx

3418 0000D5ED 89C8 <1> mov eax, ecx

3419 0000D5EF C1E810 <1> shr eax, 16 ; bottom row

3420 0000D5F2 F7E3 <1> mul ebx

3421 0000D5F4 6689CA <1> mov dx, cx ; bottom right column

3422 0000D5F7 01D0 <1> add eax, edx

3423 0000D5F9 5A <1> pop edx ; byte count per window row

3424 0000D5FA 0500800B00 <1> add eax, 0B8000h

3425 0000D5FF 3DFFFF0B00 <1> cmp eax, 0BFFFFh

3426 0000D604 0F87D4F0FFFF <1> ja sysret

3427 0000D60A 56 <1> push esi ; start address

3428 0000D60B 50 <1> push eax ; stop address (included)

3429 <1> sysvideo\_30:

3430 0000D60C 89D1 <1> mov ecx, edx ; byte count

3431 <1> ; user to system video/display page window transfer

3432 <1> ; esi = user buffer

3433 0000D60E E866110000 <1> call transfer\_to\_user\_buffer ; fast transfer

3434 0000D613 7221 <1> jc short sysvideo\_31

3435 0000D615 010D[64030300] <1> add [u.r0], ecx

3436 0000D61B 01DF <1> add edi, ebx ; next row

3437 0000D61D 01CE <1> add esi, ecx

3438 0000D61F 3B3C24 <1> cmp edi, [esp] ; stop addr(included in loop)

3439 0000D622 76E8 <1> jna short sysvideo\_30

3440 0000D624 5B <1> pop ebx ; stop address

3441 0000D625 5F <1> pop edi ; start address

3442 0000D626 29FB <1> sub ebx, edi

3443 0000D628 6643 <1> inc bx

3444 0000D62A 66891D[64030300] <1> mov [u.r0], bx

3445 0000D631 E9A8F0FFFF <1> jmp sysret

3446 <1> sysvideo\_31:

3447 0000D636 58 <1> pop eax

3448 0000D637 5A <1> pop edx

3449 0000D638 E9A1F0FFFF <1> jmp sysret

3450 <1>

3451 <1> sysvideo\_32:

3452 0000D63D 80FB07 <1> cmp bl, 7

3453 0000D640 770F <1> ja short sysvideo\_34

3454 <1>

3455 <1> ; BL = 7 = AND display page bytes with CL

3456 <1>

3457 0000D642 BE00800B00 <1> mov esi, 0B8000h

3458 0000D647 B900800000 <1> mov ecx, 32768

3459 <1> sysvideo\_33:

3460 0000D64C 200E <1> and byte [esi], cl

3461 0000D64E 46 <1> inc esi

3462 0000D64F E2FB <1> loop sysvideo\_33

3463 <1>

3464 <1> sysvideo\_34:

3465 0000D651 80FB08 <1> cmp bl, 8

3466 0000D654 770F <1> ja short sysvideo\_36

3467 <1>

3468 <1> ; BL = 8 = OR display page bytes with CL

3469 <1>

3470 0000D656 BE00800B00 <1> mov esi, 0B8000h

3471 0000D65B B900800000 <1> mov ecx, 32768

3472 <1> sysvideo\_35:

3473 0000D660 080E <1> or byte [esi], cl

3474 0000D662 46 <1> inc esi

3475 0000D663 E2FB <1> loop sysvideo\_35

3476 <1>

3477 <1> sysvideo\_36:

3478 0000D665 80FB09 <1> cmp bl, 9

3479 0000D668 0F8770F0FFFF <1> ja sysret ; nothing to do

3480 <1>

3481 <1> ; BL = 9 = XOR display page bytes with CL

3482 <1>

3483 0000D66E BE00800B00 <1> mov esi, 0B8000h

3484 0000D673 B900800000 <1> mov ecx, 32768

3485 <1> sysvideo\_37:

3486 0000D678 300E <1> xor byte [esi], cl

3487 0000D67A 46 <1> inc esi

3488 0000D67B E2FB <1> loop sysvideo\_37

3489 <1>

3490 <1> sysvideo\_38:

3491 0000D67D 80FF02 <1> cmp bh, 2

3492 0000D680 0F8733030000 <1> ja sysvideo\_64

3493 <1> ; BH = 2 = VGA Graphics (0A0000h) data transfers

3494 <1>

3495 0000D686 88DC <1> mov ah, bl

3496 0000D688 80E30F <1> and bl, 0Fh

3497 0000D68B C0EC04 <1> shr ah, 4

3498 0000D68E C1E310 <1> shl ebx, 16

3499 0000D691 66BB4001 <1> mov bx, 320 ; 320\*200, 320\*240

3500 0000D695 20E4 <1> and ah, ah

3501 0000D697 7413 <1> jz short sysvideo\_39

3502 0000D699 66D1E3 <1> shl bx, 1 ; 640\*200, 640 \* 400, 640\*480

3503 0000D69C 80FC02 <1> cmp ah, 2

3504 0000D69F 720B <1> jb short sysvideo\_39

3505 0000D6A1 0F8737F0FFFF <1> ja sysret ; invalid

3506 <1> ; 800\*600

3507 0000D6A7 6681C3A000 <1> add bx, 160 ; 800

3508 <1> sysvideo\_39:

3509 0000D6AC C1CB10 <1> ror ebx, 16

3510 <1>

3511 0000D6AF 20DB <1> and bl, bl

3512 0000D6B1 7519 <1> jnz short sysvideo\_40

3513 <1>

3514 <1> ; BL = 0 = Fill color (color in CL] (64K)

3515 <1>

3516 0000D6B3 88C8 <1> mov al, cl

3517 0000D6B5 B900000100 <1> mov ecx, 65536

3518 0000D6BA 890D[64030300] <1> mov [u.r0], ecx

3519 0000D6C0 BF00000A00 <1> mov edi, 0A0000h

3520 0000D6C5 F3AB <1> rep stosd

3521 0000D6C7 E912F0FFFF <1> jmp sysret

3522 <1>

3523 <1> sysvideo\_40:

3524 0000D6CC 80FB01 <1> cmp bl, 1

3525 0000D6CF 7722 <1> ja short sysvideo\_42

3526 <1>

3527 0000D6D1 89CE <1> mov esi, ecx ; user buffer

3528 <1> ; BL = 1 = user to system video/display page transfer

3529 <1> sysvideo\_41:

3530 0000D6D3 BF00000A00 <1> mov edi, 0A0000h

3531 <1> ; edi = video page address

3532 0000D6D8 B900000100 <1> mov ecx, 65536

3533 0000D6DD E8E1100000 <1> call transfer\_from\_user\_buffer ; fast transfer

3534 0000D6E2 0F82F6EFFFFF <1> jc sysret ; [u.r0] = 0

3535 0000D6E8 890D[64030300] <1> mov [u.r0], ecx

3536 0000D6EE E9EBEFFFFF <1> jmp sysret

3537 <1>

3538 <1> sysvideo\_42:

3539 0000D6F3 80FB02 <1> cmp bl, 2

3540 0000D6F6 7722 <1> ja short sysvideo\_44

3541 <1>

3542 0000D6F8 89CF <1> mov edi, ecx ; user buffer

3543 <1> ; BL = 2 = system to user video/display page transfer

3544 <1> sysvideo\_43:

3545 0000D6FA BE00000A00 <1> mov esi, 0A0000h

3546 0000D6FF B900000100 <1> mov ecx, 65536

3547 0000D704 E870100000 <1> call transfer\_to\_user\_buffer ; fast transfer

3548 0000D709 0F82CFEFFFFF <1> jc sysret ; [u.r0] = 0

3549 0000D70F 890D[64030300] <1> mov [u.r0], ecx

3550 0000D715 E9C4EFFFFF <1> jmp sysret

3551 <1>

3552 <1> sysvideo\_44:

3553 0000D71A 80FB03 <1> cmp bl, 3

3554 0000D71D 777A <1> ja short sysvideo\_49

3555 <1>

3556 <1> ; BL = 3 = NOT bits in window (ECX, EDX)

3557 <1>

3558 0000D71F BF00000A00 <1> mov edi, 0A0000h

3559 0000D724 89FE <1> mov esi, edi

3560 <1>

3561 0000D726 39CA <1> cmp edx, ecx ; bottom-right > top-left ?

3562 0000D728 770B <1> ja short sysvideo\_45 ; window

3563 <1> ; full screen (update)

3564 0000D72A B900000100 <1> mov ecx, 65536

3565 0000D72F 890D[64030300] <1> mov [u.r0], ecx

3566 <1> sysvideo\_45:

3567 0000D735 F616 <1> not byte [esi] ; NOT operation

3568 0000D737 46 <1> inc esi

3569 0000D738 E2FB <1> loop sysvideo\_45

3570 0000D73A E99FEFFFFF <1> jmp sysret

3571 <1> sysvideo\_46:

3572 0000D73F 0FB7C2 <1> movzx eax, dx ; bottom right column

3573 0000D742 6629C8 <1> sub ax, cx ; - top left column

3574 0000D745 0F8293EFFFFF <1> jb sysret ; invalid

3575 0000D74B 6640 <1> inc ax ; same column no == 1 column

3576 0000D74D 50 <1> push eax ; byte count per window row

3577 0000D74E 52 <1> push edx

3578 0000D74F C1EB10 <1> shr ebx, 16 ; 320,640,800 : screen width

3579 0000D752 89C8 <1> mov eax, ecx

3580 0000D754 C1E810 <1> shr eax, 16 ; top row

3581 0000D757 F7E3 <1> mul ebx

3582 0000D759 6689CA <1> mov dx, cx ; top left column

3583 0000D75C 01D0 <1> add eax, edx

3584 0000D75E 01C6 <1> add esi, eax ; start address

3585 0000D760 59 <1> pop ecx ; edx

3586 0000D761 89C8 <1> mov eax, ecx

3587 0000D763 C1E810 <1> shr eax, 16 ; bottom row

3588 0000D766 F7E3 <1> mul ebx

3589 0000D768 6689CA <1> mov dx, cx ; bottom right column

3590 0000D76B 01D0 <1> add eax, edx

3591 0000D76D 01C7 <1> add edi, eax ; stop address (included)

3592 0000D76F 5A <1> pop edx ; byte count per window row

3593 0000D770 81FFFFFF0A00 <1> cmp edi, 0AFFFFh

3594 0000D776 0F8762EFFFFF <1> ja sysret

3595 0000D77C 56 <1> push esi

3596 0000D77D 4E <1> dec esi

3597 <1> sysvideo\_47:

3598 0000D77E 89D1 <1> mov ecx, edx

3599 <1> sysvideo\_48:

3600 0000D780 46 <1> inc esi

3601 0000D781 F616 <1> not byte [esi]

3602 0000D783 E2FB <1> loop sysvideo\_48

3603 0000D785 01DE <1> add esi, ebx ; bytes per screen row

3604 <1> ;

3605 0000D787 39FE <1> cmp esi, edi ; stop address (included in loop)

3606 0000D789 76F3 <1> jna short sysvideo\_47

3607 0000D78B 5E <1> pop esi

3608 0000D78C 29F7 <1> sub edi, esi

3609 0000D78E 893D[64030300] <1> mov [u.r0], edi

3610 0000D794 E945EFFFFF <1> jmp sysret

3611 <1>

3612 <1> sysvideo\_49:

3613 0000D799 80FB04 <1> cmp bl, 4

3614 0000D79C 0F87A1000000 <1> ja sysvideo\_52

3615 <1>

3616 <1> ; BL = 4 = window copy (system to system)

3617 <1>

3618 0000D7A2 B800000A00 <1> mov eax, 0A0000h

3619 0000D7A7 39C6 <1> cmp esi, eax

3620 0000D7A9 0F822FEFFFFF <1> jb sysret

3621 0000D7AF 39C7 <1> cmp edi, eax

3622 0000D7B1 0F8227EFFFFF <1> jb sysret

3623 0000D7B7 6683C0FF <1> add ax, 0FFFFh ; 65535

3624 0000D7BB 39C6 <1> cmp esi, eax

3625 0000D7BD 0F871BEFFFFF <1> ja sysret

3626 0000D7C3 39C7 <1> cmp edi, eax

3627 0000D7C5 0F8713EFFFFF <1> ja sysret

3628 <1>

3629 0000D7CB 39CA <1> cmp edx, ecx ; bottom-right > top-left ?

3630 0000D7CD 7712 <1> ja short sysvideo\_50 ; window

3631 <1> ; full screen copy

3632 0000D7CF 89C1 <1> mov ecx, eax

3633 0000D7D1 29F9 <1> sub ecx, edi

3634 0000D7D3 41 <1> inc ecx

3635 0000D7D4 890D[64030300] <1> mov [u.r0], ecx

3636 0000D7DA F3A4 <1> rep movsb

3637 0000D7DC E9FDEEFFFF <1> jmp sysret

3638 <1> sysvideo\_50:

3639 0000D7E1 0FB7C2 <1> movzx eax, dx ; bottom right column

3640 0000D7E4 6629C8 <1> sub ax, cx ; - top left column

3641 0000D7E7 0F82F1EEFFFF <1> jb sysret ; invalid

3642 0000D7ED 6640 <1> inc ax ; same column no == 1 column

3643 0000D7EF 50 <1> push eax ; byte count per window row

3644 <1> ;

3645 0000D7F0 52 <1> push edx

3646 0000D7F1 C1EB10 <1> shr ebx, 16 ; 320,640,800 : screen width

3647 0000D7F4 89C8 <1> mov eax, ecx

3648 0000D7F6 C1E810 <1> shr eax, 16 ; top row

3649 0000D7F9 F7E3 <1> mul ebx

3650 0000D7FB 6689CA <1> mov dx, cx ; top left column

3651 0000D7FE 01D0 <1> add eax, edx

3652 0000D800 01C7 <1> add edi, eax ; start address

3653 0000D802 01C6 <1> add esi, eax

3654 0000D804 59 <1> pop ecx ; edx

3655 0000D805 89C8 <1> mov eax, ecx

3656 0000D807 C1E810 <1> shr eax, 16 ; bottom row

3657 0000D80A F7E3 <1> mul ebx

3658 0000D80C 6689CA <1> mov dx, cx ; bottom right column

3659 0000D80F 01D0 <1> add eax, edx

3660 0000D811 5A <1> pop edx ; byte count per window row

3661 0000D812 0500000A00 <1> add eax, 0A0000h

3662 0000D817 3DFFFF0A00 <1> cmp eax, 0AFFFFh

3663 0000D81C 0F87BCEEFFFF <1> ja sysret

3664 0000D822 57 <1> push edi ; start address

3665 0000D823 50 <1> push eax ; stop address (included)

3666 <1> sysvideo\_51:

3667 0000D824 89D1 <1> mov ecx, edx

3668 0000D826 F3A4 <1> rep movsb

3669 0000D828 4F <1> dec edi

3670 0000D829 4E <1> dec esi

3671 0000D82A 01DF <1> add edi, ebx ; bytes per screen row

3672 0000D82C 01DE <1> add esi, ebx

3673 <1> ;

3674 0000D82E 3B3C24 <1> cmp edi, [esp] ; stop addr(included in loop)

3675 0000D831 76F1 <1> jna short sysvideo\_51

3676 0000D833 5B <1> pop ebx ; stop address

3677 0000D834 5F <1> pop edi ; start address

3678 0000D835 29FB <1> sub ebx, edi

3679 0000D837 43 <1> inc ebx

3680 0000D838 891D[64030300] <1> mov [u.r0], ebx

3681 0000D83E E99BEEFFFF <1> jmp sysret

3682 <1>

3683 <1> sysvideo\_52:

3684 0000D843 80FB05 <1> cmp bl, 5

3685 0000D846 0F8791000000 <1> ja sysvideo\_55

3686 <1>

3687 <1> ; BL = 5 = window copy (user to system)

3688 <1>

3689 0000D84C B800000A00 <1> mov eax, 0A0000h

3690 0000D851 39C7 <1> cmp edi, eax

3691 0000D853 0F8285EEFFFF <1> jb sysret

3692 0000D859 6683C0FF <1> add ax, 0FFFFh ; 65535

3693 0000D85D 39C7 <1> cmp edi, eax

3694 0000D85F 0F8779EEFFFF <1> ja sysret

3695 <1>

3696 <1> ; esi = user buffer (in user's memory space)

3697 0000D865 39CA <1> cmp edx, ecx ; bottom-right > top-left ?

3698 0000D867 0F8666FEFFFF <1> jna sysvideo\_41 ; full screen copy

3699 <1>

3700 0000D86D 0FB7C2 <1> movzx eax, dx ; bottom right column

3701 0000D870 6629C8 <1> sub ax, cx ; - top left column

3702 0000D873 0F8265EEFFFF <1> jb sysret ; invalid

3703 0000D879 6640 <1> inc ax ; same column no == 1 column

3704 0000D87B 50 <1> push eax ; byte count per window row

3705 <1>

3706 0000D87C 52 <1> push edx

3707 0000D87D C1EB10 <1> shr ebx, 16 ; 320,640,800 : screen width

3708 0000D880 89C8 <1> mov eax, ecx

3709 0000D882 C1E810 <1> shr eax, 16 ; top row

3710 0000D885 F7E3 <1> mul ebx

3711 0000D887 6689CA <1> mov dx, cx ; top left column

3712 0000D88A 01D0 <1> add eax, edx

3713 0000D88C 01C7 <1> add edi, eax ; start address

3714 0000D88E 59 <1> pop ecx ; edx

3715 0000D88F 89C8 <1> mov eax, ecx

3716 0000D891 C1E810 <1> shr eax, 16 ; bottom row

3717 0000D894 F7E3 <1> mul ebx

3718 0000D896 6689CA <1> mov dx, cx ; bottom right column

3719 0000D899 01D0 <1> add eax, edx

3720 0000D89B 5A <1> pop edx ; byte count per window row

3721 0000D89C 0500000A00 <1> add eax, 0A0000h

3722 0000D8A1 3DFFFF0A00 <1> cmp eax, 0AFFFFh

3723 0000D8A6 0F8732EEFFFF <1> ja sysret

3724 0000D8AC 57 <1> push edi ; start address

3725 0000D8AD 50 <1> push eax ; stop address (included)

3726 <1> sysvideo\_53:

3727 0000D8AE 89D1 <1> mov ecx, edx ; byte count

3728 <1> ; user to system video/display page window transfer

3729 <1> ; esi = user buffer

3730 0000D8B0 E80E0F0000 <1> call transfer\_from\_user\_buffer ; fast transfer

3731 0000D8B5 721F <1> jc short sysvideo\_54

3732 0000D8B7 010D[64030300] <1> add [u.r0], ecx

3733 0000D8BD 01DF <1> add edi, ebx ; next row

3734 0000D8BF 01CE <1> add esi, ecx

3735 0000D8C1 3B3C24 <1> cmp edi, [esp] ; stop addr(included in loop)

3736 0000D8C4 76E8 <1> jna short sysvideo\_53

3737 0000D8C6 5B <1> pop ebx ; stop address

3738 0000D8C7 5F <1> pop edi ; start address

3739 0000D8C8 29FB <1> sub ebx, edi

3740 0000D8CA 43 <1> inc ebx

3741 0000D8CB 891D[64030300] <1> mov [u.r0], ebx

3742 0000D8D1 E908EEFFFF <1> jmp sysret

3743 <1> sysvideo\_54:

3744 0000D8D6 58 <1> pop eax

3745 0000D8D7 5A <1> pop edx

3746 0000D8D8 E901EEFFFF <1> jmp sysret

3747 <1>

3748 <1> sysvideo\_55:

3749 0000D8DD 80FB06 <1> cmp bl, 6

3750 0000D8E0 0F8793000000 <1> ja sysvideo\_58

3751 <1>

3752 <1> ; BL = 6 = window copy (system to user)

3753 <1>

3754 0000D8E6 89F7 <1> mov edi, esi ; user buffer

3755 <1>

3756 0000D8E8 B800000A00 <1> mov eax, 0A0000h

3757 0000D8ED 39C6 <1> cmp esi, eax

3758 0000D8EF 0F82E9EDFFFF <1> jb sysret

3759 0000D8F5 6683C0FF <1> add ax, 0FFFFh ; 65535

3760 0000D8F9 39C6 <1> cmp esi, eax

3761 0000D8FB 0F87DDEDFFFF <1> ja sysret

3762 <1>

3763 <1> ; edi = user buffer (in user's memory space)

3764 0000D901 39CA <1> cmp edx, ecx ; bottom-right > top-left ?

3765 0000D903 0F86A2FAFFFF <1> jna sysvideo\_17 ; full screen copy

3766 <1>

3767 0000D909 0FB7C2 <1> movzx eax, dx ; bottom right column

3768 0000D90C 6629C8 <1> sub ax, cx ; - top left column

3769 0000D90F 0F82C9EDFFFF <1> jb sysret ; invalid

3770 0000D915 6640 <1> inc ax ; same column no == 1 column

3771 0000D917 50 <1> push eax ; byte count per window row

3772 <1>

3773 0000D918 52 <1> push edx

3774 0000D919 C1EB10 <1> shr ebx, 16 ; 320, 640,800 ; screen width

3775 0000D91C 89C8 <1> mov eax, ecx

3776 0000D91E C1E810 <1> shr eax, 16 ; top row

3777 0000D921 F7E3 <1> mul ebx

3778 0000D923 6689CA <1> mov dx, cx ; top left column

3779 0000D926 01D0 <1> add eax, edx

3780 0000D928 01C6 <1> add esi, eax ; start address

3781 0000D92A 59 <1> pop ecx ; edx

3782 0000D92B 89C8 <1> mov eax, ecx

3783 0000D92D C1E810 <1> shr eax, 16 ; bottom row

3784 0000D930 F7E3 <1> mul ebx

3785 0000D932 6689CA <1> mov dx, cx ; bottom right column

3786 0000D935 01D0 <1> add eax, edx

3787 0000D937 5A <1> pop edx ; byte count per window row

3788 0000D938 0500000A00 <1> add eax, 0A0000h

3789 0000D93D 3DFFFF0A00 <1> cmp eax, 0AFFFFh

3790 0000D942 0F8796EDFFFF <1> ja sysret

3791 0000D948 56 <1> push esi ; start address

3792 0000D949 50 <1> push eax ; stop address (included)

3793 <1> sysvideo\_56:

3794 0000D94A 89D1 <1> mov ecx, edx ; byte count

3795 <1> ; user to system video/display page window transfer

3796 <1> ; esi = user buffer

3797 0000D94C E8280E0000 <1> call transfer\_to\_user\_buffer ; fast transfer

3798 0000D951 721F <1> jc short sysvideo\_57

3799 0000D953 010D[64030300] <1> add [u.r0], ecx

3800 0000D959 01DF <1> add edi, ebx ; next row

3801 0000D95B 01CE <1> add esi, ecx

3802 0000D95D 3B3C24 <1> cmp edi, [esp] ; stop addr(included in loop)

3803 0000D960 76E8 <1> jna short sysvideo\_56

3804 0000D962 5B <1> pop ebx ; stop address

3805 0000D963 5F <1> pop edi ; start address

3806 0000D964 29FB <1> sub ebx, edi

3807 0000D966 43 <1> inc ebx

3808 0000D967 891D[64030300] <1> mov [u.r0], ebx

3809 0000D96D E96CEDFFFF <1> jmp sysret

3810 <1> sysvideo\_57:

3811 0000D972 58 <1> pop eax

3812 0000D973 5A <1> pop edx

3813 0000D974 E965EDFFFF <1> jmp sysret

3814 <1>

3815 <1> sysvideo\_58:

3816 0000D979 80FB07 <1> cmp bl, 7

3817 0000D97C 770F <1> ja short sysvideo\_60

3818 <1>

3819 <1> ; BL = 7 = AND display page bytes with CL

3820 <1>

3821 0000D97E BE00000A00 <1> mov esi, 0A0000h

3822 0000D983 B900000100 <1> mov ecx, 65536

3823 <1> sysvideo\_59:

3824 0000D988 200E <1> and byte [esi], cl

3825 0000D98A 46 <1> inc esi

3826 0000D98B E2FB <1> loop sysvideo\_59

3827 <1>

3828 <1> sysvideo\_60:

3829 0000D98D 80FB08 <1> cmp bl, 8

3830 0000D990 770F <1> ja short sysvideo\_62

3831 <1>

3832 <1> ; BL = 8 = OR display page bytes with CL

3833 <1>

3834 0000D992 BE00000A00 <1> mov esi, 0A0000h

3835 0000D997 B900000100 <1> mov ecx, 65536

3836 <1> sysvideo\_61:

3837 0000D99C 080E <1> or byte [esi], cl

3838 0000D99E 46 <1> inc esi

3839 0000D99F E2FB <1> loop sysvideo\_61

3840 <1>

3841 <1> sysvideo\_62:

3842 0000D9A1 80FB09 <1> cmp bl, 9

3843 0000D9A4 0F8734EDFFFF <1> ja sysret ; nothing to do

3844 <1>

3845 <1> ; BL = 9 = XOR display page bytes with CL

3846 <1>

3847 0000D9AA BE00000A00 <1> mov esi, 0A0000h

3848 0000D9AF B900000100 <1> mov ecx, 65536

3849 <1> sysvideo\_63:

3850 0000D9B4 300E <1> xor byte [esi], cl

3851 0000D9B6 46 <1> inc esi

3852 0000D9B7 E2FB <1> loop sysvideo\_63

3853 <1>

3854 <1> sysvideo\_64:

3855 0000D9B9 80FF03 <1> cmp bh, 3

3856 0000D9BC 7464 <1> je short sysvideo\_68

3857 0000D9BE 80FF04 <1> cmp bh, 4

3858 0000D9C1 7721 <1> ja short sysvideo\_65

3859 <1>

3860 <1> ; BH = 4

3861 <1> ; Direct User Access for CGA video memory.

3862 <1> ; Setup user's page tables for direct access to 0B8000h.

3863 <1> ;

3864 <1> ; Permission checks are not implemented yet !

3865 <1> ; (11/07/2016)

3866 <1>

3867 0000D9C3 B800800B00 <1> mov eax, 0B8000h

3868 0000D9C8 B908000000 <1> mov ecx, 8 ; 8 pages (8\*4K=32K)

3869 0000D9CD 89C3 <1> mov ebx, eax ; 12/05/2017 ; virtual = physical

3870 0000D9CF E8CD7CFFFF <1> call direct\_memory\_access

3871 0000D9D4 0F8204EDFFFF <1> jc sysret

3872 <1> ; eax = 0B8000h if there is not an error

3873 0000D9DA A3[64030300] <1> mov [u.r0], eax

3874 0000D9DF E9FAECFFFF <1> jmp sysret

3875 <1>

3876 <1> sysvideo\_65:

3877 0000D9E4 80FF05 <1> cmp bh, 5

3878 0000D9E7 7721 <1> ja short sysvideo\_66

3879 <1>

3880 <1> ; BH = 5

3881 <1> ; Direct User Access for VGA video memory.

3882 <1> ; Setup user's page tables for direct access to 0A0000h.

3883 <1> ;

3884 <1> ; Permission checks are not implemented yet !

3885 <1> ; (11/07/2016)

3886 <1>

3887 0000D9E9 B800000A00 <1> mov eax, 0A0000h

3888 0000D9EE B910000000 <1> mov ecx, 16 ; 16 pages (16\*4K=64K)

3889 0000D9F3 89C3 <1> mov ebx, eax ; 12/05/2017 ; virtual = physical

3890 0000D9F5 E8A77CFFFF <1> call direct\_memory\_access

3891 0000D9FA 0F82DEECFFFF <1> jc sysret

3892 <1> ; eax = 0A0000h if there is not an error

3893 0000DA00 A3[64030300] <1> mov [u.r0], eax

3894 0000DA05 E9D4ECFFFF <1> jmp sysret

3895 <1>

3896 <1> sysvideo\_66:

3897 0000DA0A 80FF06 <1> cmp bh, 6

3898 0000DA0D 7705 <1> ja short sysvideo\_67

3899 <1> ; BH = 6

3900 <1> ; Direct User Access for (Super VGA) Linear Frame Buffer.

3901 <1> ; Setup user's page tables for direct access to LFB.

3902 <1> ;

3903 <1> ; Not implemented yet !

3904 <1> ; (11/07/2016)

3905 0000DA0F E9CAECFFFF <1> jmp sysret

3906 <1>

3907 <1> sysvideo\_67:

3908 0000DA14 80FF07 <1> cmp bh, 7

3909 0000DA17 0F87C1ECFFFF <1> ja sysret ; invalid !

3910 <1>

3911 <1> ; BH = 7

3912 <1> ; Get (Super/Extended VGA) Linear Frame Buffer info.

3913 <1> ;

3914 <1> ; Not implemented yet !

3915 <1> ; (11/07/2016)

3916 0000DA1D E9BCECFFFF <1> jmp sysret

3917 <1>

3918 <1> sysvideo\_68:

3919 <1> ; BH = 3

3920 <1> ; Super VGA, LINEAR FRAME BUFFER data transfers

3921 <1> ; Not implemented for yet ! (11/07/2016)

3922 0000DA22 E9B7ECFFFF <1> jmp sysret

3923 <1>

3924 <1> mkdir:

3925 <1> ; 04/12/2015 (14 byte directory names)

3926 <1> ; 12/10/2015

3927 <1> ; 17/06/2015 (Retro UNIX 386 v1 - Beginning)

3928 <1> ; 29/04/2013 - 01/08/2013 (Retro UNIX 8086 v1)

3929 <1> ;

3930 <1> ; 'mkdir' makes a directory entry from the name pointed to

3931 <1> ; by u.namep into the current directory.

3932 <1> ;

3933 <1> ; INPUTS ->

3934 <1> ; u.namep - points to a file name

3935 <1> ; that is about to be a directory entry.

3936 <1> ; ii - current directory's i-number.

3937 <1> ; OUTPUTS ->

3938 <1> ; u.dirbuf+2 - u.dirbuf+10 - contains file name.

3939 <1> ; u.off - points to entry to be filled

3940 <1> ; in the current directory

3941 <1> ; u.base - points to start of u.dirbuf.

3942 <1> ; r1 - contains i-number of current directory

3943 <1> ;

3944 <1> ; ((AX = R1)) output

3945 <1> ;

3946 <1> ; (Retro UNIX Prototype : 11/11/2012, UNIXCOPY.ASM)

3947 <1> ; ((Modified registers: eAX, eDX, eBX, eCX, eSI, eDI, eBP))

3948 <1> ;

3949 <1>

3950 <1> ; 17/06/2015 - 32 bit modifications (Retro UNIX 386 v1)

3951 0000DA27 31C0 <1> xor eax, eax

3952 0000DA29 BF[9A030300] <1> mov edi, u.dirbuf+2

3953 0000DA2E 89FE <1> mov esi, edi

3954 0000DA30 AB <1> stosd

3955 0000DA31 AB <1> stosd

3956 <1> ; 04/12/2015 (14 byte directory names)

3957 0000DA32 AB <1> stosd

3958 0000DA33 66AB <1> stosw

3959 <1> ; jsr r0,copyz; u.dirbuf+2; u.dirbuf+10. / clear this

3960 0000DA35 89F7 <1> mov edi, esi ; offset to u.dirbuf

3961 <1> ; 12/10/2015 ([u.namep] -> ebp)

3962 <1> ;mov ebp, [u.namep]

3963 0000DA37 E80D030000 <1> call trans\_addr\_nmbp ; convert virtual address to physical

3964 <1> ; esi = physical address (page start + offset)

3965 <1> ; ecx = byte count in the page (1 - 4096)

3966 <1> ; edi = offset to u.dirbuf (edi is not modified in trans\_addr\_nm)

3967 <1> ; mov u.namep,r2 / r2 points to name of directory entry

3968 <1> ; mov $u.dirbuf+2,r3 / r3 points to u.dirbuf+2

3969 <1> mkdir\_1: ; 1:

3970 0000DA3C 45 <1> inc ebp ; 12/10/2015

3971 <1> ;

3972 <1> ; / put characters in the directory name in u.dirbuf+2 - u.dirbuf+10

3973 <1> ; 01/08/2013

3974 0000DA3D AC <1> lodsb

3975 <1> ; movb (r2)+,r1 / move character in name to r1

3976 0000DA3E 20C0 <1> and al, al

3977 0000DA40 7427 <1> jz short mkdir\_3

3978 <1> ; beq 1f / if null, done

3979 0000DA42 3C2F <1> cmp al, '/'

3980 <1> ; cmp r1,$'/ / is it a "/"?

3981 0000DA44 7414 <1> je short mkdir\_err

3982 <1> ;je error

3983 <1> ; beq error9 / yes, error

3984 <1> ; 12/10/2015

3985 0000DA46 6649 <1> dec cx

3986 0000DA48 7505 <1> jnz short mkdir\_2

3987 <1> ; 12/10/2015 ([u.namep] -> ebp)

3988 0000DA4A E800030000 <1> call trans\_addr\_nm ; convert virtual address to physical

3989 <1> ; esi = physical address (page start + offset)

3990 <1> ; ecx = byte count in the page

3991 <1> ; edi = offset to u.dirbuf (edi is not modified in trans\_addr\_nm)

3992 <1> mkdir\_2:

3993 0000DA4F 81FF[A8030300] <1> cmp edi, u.dirbuf+16 ; ; 04/12/2015 (10 -> 16)

3994 <1> ; cmp r3,$u.dirbuf+10. / have we reached the last slot for

3995 <1> ; / a char?

3996 0000DA55 74E5 <1> je short mkdir\_1

3997 <1> ; beq 1b / yes, go back

3998 0000DA57 AA <1> stosb

3999 <1> ; movb r1,(r3)+ / no, put the char in the u.dirbuf

4000 0000DA58 EBE2 <1> jmp short mkdir\_1

4001 <1> ; br 1b / get next char

4002 <1> mkdir\_err:

4003 <1> ; 17/06/2015

4004 0000DA5A C705[C8030300]1300- <1> mov dword [u.error], ERR\_NOT\_DIR ; 'not a valid directory !'

4004 0000DA62 0000 <1>

4005 0000DA64 E955ECFFFF <1> jmp error

4006 <1>

4007 <1> mkdir\_3: ; 1:

4008 0000DA69 A1[78030300] <1> mov eax, [u.dirp]

4009 0000DA6E A3[80030300] <1> mov [u.off], eax

4010 <1> ; mov u.dirp,u.off / pointer to empty current directory

4011 <1> ; / slot to u.off

4012 <1> wdir: ; 29/04/2013

4013 0000DA73 C705[84030300]- <1> mov dword [u.base], u.dirbuf

4013 0000DA79 [98030300] <1>

4014 <1> ; mov $u.dirbuf,u.base / u.base points to created file name

4015 0000DA7D C705[88030300]1000- <1> mov dword [u.count], 16 ; 04/12/2015 (10 -> 16)

4015 0000DA85 0000 <1>

4016 <1> ; mov $10.,u.count / u.count = 10

4017 0000DA87 66A1[51040300] <1> mov ax, [ii]

4018 <1> ; mov ii,r1 / r1 has i-number of current directory

4019 0000DA8D B201 <1> mov dl, 1 ; owner flag mask ; RETRO UNIX 8086 v1 modification !

4020 0000DA8F E8331D0000 <1> call access

4021 <1> ; jsr r0,access; 1 / get i-node and set its file up

4022 <1> ; / for writing

4023 <1> ; AX = i-number of current directory

4024 <1> ; 01/08/2013

4025 0000DA94 FE05[C6030300] <1> inc byte [u.kcall] ; the caller is 'mkdir' sign

4026 0000DA9A E83C0F0000 <1> call writei

4027 <1> ; jsr r0,writei / write into directory

4028 0000DA9F C3 <1> retn

4029 <1> ; rts r0

4030 <1>

4031 <1> sysexec:

4032 <1> ; 18/11/2017

4033 <1> ; 14/11/2017

4034 <1> ; 13/11/2017

4035 <1> ; 24/10/2016, 04/01/2017

4036 <1> ; 24/04/2016 - TRDOS 386 (TRDOS v2.0)

4037 <1> ; 23/06/2015 - 23/10/2015 (Retro UNIX 386 v1)

4038 <1> ; 03/06/2013 - 06/12/2013 (Retro UNIX 8086 v1)

4039 <1> ;

4040 <1> ; 'sysexec' initiates execution of a file whose path name if

4041 <1> ; pointed to by 'name' in the sysexec call.

4042 <1> ; 'sysexec' performs the following operations:

4043 <1> ; 1. obtains i-number of file to be executed via 'namei'.

4044 <1> ; 2. obtains i-node of file to be exceuted via 'iget'.

4045 <1> ; 3. sets trap vectors to system routines.

4046 <1> ; 4. loads arguments to be passed to executing file into

4047 <1> ; highest locations of user's core

4048 <1> ; 5. puts pointers to arguments in locations immediately

4049 <1> ; following arguments.

4050 <1> ; 6. saves number of arguments in next location.

4051 <1> ; 7. intializes user's stack area so that all registers

4052 <1> ; will be zeroed and the PS is cleared and the PC set

4053 <1> ; to core when 'sysret' restores registers

4054 <1> ; and does an rti.

4055 <1> ; 8. inializes u.r0 and u.sp

4056 <1> ; 9. zeros user's core down to u.r0

4057 <1> ; 10. reads executable file from storage device into core

4058 <1> ; starting at location 'core'.

4059 <1> ; 11. sets u.break to point to end of user's code with

4060 <1> ; data area appended.

4061 <1> ; 12. calls 'sysret' which returns control at location

4062 <1> ; 'core' via 'rti' instruction.

4063 <1> ;

4064 <1> ; Calling sequence:

4065 <1> ; sysexec; namep; argp

4066 <1> ; Arguments:

4067 <1> ; namep - points to pathname of file to be executed

4068 <1> ; argp - address of table of argument pointers

4069 <1> ; argp1... argpn - table of argument pointers

4070 <1> ; argp1:<...0> ... argpn:<...0> - argument strings

4071 <1> ; Inputs: (arguments)

4072 <1> ; Outputs: -

4073 <1> ; ...............................................................

4074 <1> ;

4075 <1> ; Retro UNIX 386 v1 modification:

4076 <1> ; User application runs in it's own virtual space

4077 <1> ; which is izolated from kernel memory (and other

4078 <1> ; memory pages) via 80386 paging in ring 3

4079 <1> ; privilige mode. Virtual start address is always 0.

4080 <1> ; User's core memory starts at linear address 400000h

4081 <1> ; (the end of the 1st 4MB).

4082 <1> ;

4083 <1> ; Retro UNIX 8086 v1 modification:

4084 <1> ; user/application segment and system/kernel segment

4085 <1> ; are different and sysenter/sysret/sysrele routines

4086 <1> ; are different (user's registers are saved to

4087 <1> ; and then restored from system's stack.)

4088 <1> ;

4089 <1> ; NOTE: Retro UNIX 8086 v1 'arg2' routine gets these

4090 <1> ; arguments which were in these registers;

4091 <1> ; but, it returns by putting the 1st argument

4092 <1> ; in 'u.namep' and the 2nd argument

4093 <1> ; on top of stack. (1st argument is offset of the

4094 <1> ; file/path name in the user's program segment.)

4095 <1>

4096 <1> ;call arg2

4097 <1> ; \* name - 'u.namep' points to address of file/path name

4098 <1> ; in the user's program segment ('u.segmnt')

4099 <1> ; with offset in BX register (as sysopen argument 1).

4100 <1> ; \* argp - sysexec argument 2 is in CX register

4101 <1> ; which is on top of stack.

4102 <1> ;

4103 <1> ; jsr r0,arg2 / arg0 in u.namep,arg1 on top of stack

4104 <1>

4105 <1> ; 23/06/2015 (32 bit modifications)

4106 <1>

4107 <1> ;; 13/11/2017

4108 <1> ;;mov [u.namep], ebx ; argument 1

4109 <1> ; 18/10/2015

4110 0000DAA0 890D[4C040300] <1> mov [argv], ecx ; \* ; argument 2

4111 <1>

4112 <1> ; 13/11/2017

4113 0000DAA6 89DE <1> mov esi, ebx

4114 0000DAA8 E84E210000 <1> call set\_working\_path\_x

4115 0000DAAD 7319 <1> jnc short sysexec\_0

4116 <1>

4117 <1> ;; 'bad command or file name'

4118 <1> ;mov eax, ERR\_BAD\_CMD\_ARG ; 01h ; TRDOS 8086

4119 <1>

4120 <1> ; 'file not found !' error

4121 0000DAAF B802000000 <1> mov eax, ERR\_NOT\_FOUND ; 02h ; TRDOS 8086

4122 <1> sysexec\_not\_found\_err:

4123 <1> sysexec\_access\_error:

4124 <1> sysexec\_ext\_error:

4125 0000DAB4 A3[64030300] <1> mov [u.r0], eax

4126 0000DAB9 A3[C8030300] <1> mov [u.error], eax

4127 0000DABE E80D220000 <1> call reset\_working\_path

4128 0000DAC3 E9F6EBFFFF <1> jmp error

4129 <1>

4130 <1> sysexec\_0:

4131 <1> ; 13/11/2017

4132 <1> ;mov esi, FindFile\_Name

4133 0000DAC8 66B80018 <1> mov ax, 1800h ; Only files

4134 0000DACC E892A7FFFF <1> call find\_first\_file

4135 0000DAD1 72E1 <1> jc short sysexec\_not\_found\_err ; eax = 2

4136 <1>

4137 <1> ; check\_ file attributes

4138 <1> ; (attribute bits = 00ADVSHR) ; 18h = Directory+Volume

4139 <1> ; BL = Attributes byte

4140 <1>

4141 0000DAD3 F6C306 <1> test bl, 6 ; system file or hidden file (S+H)

4142 <1> ;jz short sysexec\_0ext

4143 0000DAD6 7417 <1> jz short sysexec\_1 ; yes

4144 <1>

4145 <1> ; 13/11/2017

4146 <1> ; /// TRDOS386 permission check for multiuser mode ///

4147 <1> ; SYSTEM file or HIDDEN file !!

4148 <1> ; (Only super user has permission to run this file.)

4149 <1>

4150 <1> ; ([u.uid]=0 for super user or root in multiuser mode)

4151 <1> ; ([u.uid]=0 for any users in singleuser mode)

4152 0000DAD8 803D[B0030300]00 <1> cmp byte [u.uid], 0 ; Super User ([u.uid]=0) ?

4153 <1> ;jna short sysexec\_0ext

4154 0000DADF 760E <1> jna short sysexec\_1 ; yes

4155 <1>

4156 <1> ; 'permission denied !' error

4157 0000DAE1 B80B000000 <1> mov eax, ERR\_FILE\_ACCESS ; 11 = ERR\_PERM\_DENIED

4158 0000DAE6 EBCC <1> jmp short sysexec\_access\_error

4159 <1>

4160 <1> sysexec\_not\_exf:

4161 <1> ; 'not executable file !' error

4162 0000DAE8 B816000000 <1> mov eax, ERR\_NOT\_EXECUTABLE

4163 0000DAED EBC5 <1> jmp sysexec\_ext\_error

4164 <1>

4165 <1> ;sysexec\_0ext:

4166 <1> sysexec\_1:

4167 <1> ; 18/11/2017

4168 0000DAEF BE[E4620100] <1> mov esi, FindFile\_Name

4169 <1> ; 13/11/2017

4170 <1> ; check program file name extension

4171 <1> ; ('.PRG' for current TRDOS version)

4172 0000DAF4 E80DC2FFFF <1> call check\_prg\_filename\_ext

4173 0000DAF9 72ED <1> jc short sysexec\_not\_exf

4174 <1>

4175 <1> ; 18/11/2017

4176 0000DAFB 3C50 <1> cmp al, 'P'

4177 0000DAFD 75E9 <1> jne short sysexec\_not\_exf

4178 <1>

4179 <1> ; '.PRG' extension is OK.

4180 <1> ; Only '.PRG' files are valid program files

4181 <1> ; for current TRDOS 386 version.

4182 <1>

4183 0000DAFF 8B15[10630100] <1> mov edx, [FindFile\_DirEntry+DirEntry\_FileSize]

4184 0000DB05 66A1[08630100] <1> mov ax, [FindFile\_DirEntry+DirEntry\_FstClusHI]

4185 0000DB0B C1E010 <1> shl eax, 16

4186 0000DB0E 66A1[0E630100] <1> mov ax, [FindFile\_DirEntry+DirEntry\_FstClusLO]

4187 <1> ; EAX = First Cluster number

4188 <1> ; EDX = File Size

4189 <1>

4190 0000DB14 A3[51040300] <1> mov [ii], eax

4191 0000DB19 8915[55040300] <1> mov [i.size], edx

4192 <1>

4193 <1> ;sysexec\_1:

4194 <1> ; 13/11/2017 - TRDOS 386 (TRDOS v2.0)

4195 <1> ; 24/06/2015 - 23/10/2015 (Retro UNIX 386 v1)

4196 <1> ; Moving arguments to the end of [u.upage]

4197 <1> ; (by regarding page borders in user's memory space)

4198 <1> ;

4199 <1> ; 10/10/2015

4200 <1> ; 21/07/2015

4201 0000DB1F 89E5 <1> mov ebp, esp ; (\*\*)

4202 <1> ; 18/10/2015

4203 0000DB21 89EF <1> mov edi, ebp

4204 0000DB23 B900010000 <1> mov ecx, MAX\_ARG\_LEN ; 256

4205 <1> ;sub edi, MAX\_ARG\_LEN ; 256

4206 0000DB28 29CF <1> sub edi, ecx

4207 0000DB2A 89FC <1> mov esp, edi ; \*!\*

4208 0000DB2C 31C0 <1> xor eax, eax

4209 0000DB2E A3[8C030300] <1> mov [u.nread], eax ; 0

4210 0000DB33 66A3[4A040300] <1> mov [argc], ax ; 0 ; 13/11/2017

4211 0000DB39 49 <1> dec ecx ; 256 - 1

4212 0000DB3A 890D[88030300] <1> mov [u.count], ecx ; MAX\_ARG\_LEN - 1 ; 255

4213 <1> ;mov dword [u.count], MAX\_ARG\_LEN - 1 ; 255

4214 <1> sysexec\_2:

4215 0000DB40 8B35[4C040300] <1> mov esi, [argv] ; 18/10/2015

4216 0000DB46 E866000000 <1> call get\_argp

4217 0000DB4B B904000000 <1> mov ecx, 4 ; mov ecx, 4

4218 <1> sysexec\_3:

4219 0000DB50 21C0 <1> and eax, eax

4220 0000DB52 0F8429050000 <1> jz sysexec\_6

4221 <1> ; 18/10/2015

4222 0000DB58 010D[4C040300] <1> add [argv], ecx ; 4

4223 0000DB5E 66FF05[4A040300] <1> inc word [argc]

4224 <1> ;

4225 0000DB65 A3[84030300] <1> mov [u.base], eax

4226 <1> ; 23/10/2015

4227 0000DB6A 66C705[C4030300]00- <1> mov word [u.pcount], 0

4227 0000DB72 00 <1>

4228 <1> sysexec\_4:

4229 0000DB73 E8A10B0000 <1> call cpass ; get a character from user's core memory

4230 0000DB78 750E <1> jnz short sysexec\_5

4231 <1> ; (max. 255 chars + null)

4232 <1> ; 18/10/2015

4233 0000DB7A 28C0 <1> sub al, al

4234 0000DB7C AA <1> stosb

4235 0000DB7D FF05[8C030300] <1> inc dword [u.nread]

4236 0000DB83 E9F9040000 <1> jmp sysexec\_6 ; 24/04/2016

4237 <1> sysexec\_5:

4238 0000DB88 AA <1> stosb

4239 0000DB89 20C0 <1> and al, al

4240 0000DB8B 75E6 <1> jnz short sysexec\_4

4241 0000DB8D B904000000 <1> mov ecx, 4

4242 0000DB92 390D[48040300] <1> cmp [ncount], ecx ; 4

4243 0000DB98 72A6 <1> jb short sysexec\_2

4244 0000DB9A 8B35[44040300] <1> mov esi, [nbase]

4245 0000DBA0 010D[44040300] <1> add [nbase], ecx ; 4

4246 0000DBA6 66290D[48040300] <1> sub [ncount], cx

4247 0000DBAD 8B06 <1> mov eax, [esi]

4248 0000DBAF EB9F <1> jmp short sysexec\_3

4249 <1>

4250 <1> get\_argp:

4251 <1> ; 14/11/2017 - TRDOS 386 (TRDOS v2.0)

4252 <1> ; 18/10/2015 (nbase, ncount)

4253 <1> ; 21/07/2015

4254 <1> ; 24/06/2015 (Retro UNIX 386 v1)

4255 <1> ; Get (virtual) address of argument from user's core memory

4256 <1> ;

4257 <1> ; INPUT:

4258 <1> ; esi = virtual address of argument pointer

4259 <1> ; OUTPUT:

4260 <1> ; eax = virtual address of argument

4261 <1> ;

4262 <1> ; Modified registers: EAX, EBX, ECX, EDX, ESI

4263 <1> ;

4264 0000DBB1 833D[BC030300]00 <1> cmp dword [u.ppgdir], 0 ; /etc/init ?

4265 <1> ; (the caller is kernel)

4266 0000DBB8 7667 <1> jna short get\_argpk

4267 <1> ;

4268 0000DBBA 89F3 <1> mov ebx, esi

4269 0000DBBC E8CE76FFFF <1> call get\_physical\_addr ; get physical address

4270 0000DBC1 0F8289000000 <1> jc get\_argp\_err

4271 0000DBC7 A3[44040300] <1> mov [nbase], eax ; physical address

4272 0000DBCC 66890D[48040300] <1> mov [ncount], cx ; remain byte count in page (1-4096)

4273 0000DBD3 B804000000 <1> mov eax, 4 ; 21/07/2015

4274 0000DBD8 6639C1 <1> cmp cx, ax ; 4

4275 0000DBDB 735D <1> jnb short get\_argp2

4276 0000DBDD 89F3 <1> mov ebx, esi

4277 0000DBDF 01CB <1> add ebx, ecx

4278 0000DBE1 E8A976FFFF <1> call get\_physical\_addr ; get physical address

4279 0000DBE6 7268 <1> jc short get\_argp\_err

4280 <1> ;push esi

4281 0000DBE8 89C6 <1> mov esi, eax

4282 0000DBEA 66870D[48040300] <1> xchg cx, [ncount]

4283 0000DBF1 8735[44040300] <1> xchg esi, [nbase]

4284 0000DBF7 B504 <1> mov ch, 4

4285 0000DBF9 28CD <1> sub ch, cl

4286 <1> get\_argp0:

4287 0000DBFB AC <1> lodsb

4288 0000DBFC 6650 <1> push ax

4289 0000DBFE FEC9 <1> dec cl

4290 0000DC00 75F9 <1> jnz short get\_argp0

4291 0000DC02 8B35[44040300] <1> mov esi, [nbase]

4292 <1> ; 21/07/2015

4293 0000DC08 0FB6C5 <1> movzx eax, ch

4294 0000DC0B 0105[44040300] <1> add [nbase], eax

4295 0000DC11 662905[48040300] <1> sub [ncount], ax

4296 <1> get\_argp1:

4297 0000DC18 AC <1> lodsb

4298 0000DC19 FECD <1> dec ch

4299 0000DC1B 7447 <1> jz short get\_argp3

4300 0000DC1D 6650 <1> push ax

4301 0000DC1F EBF7 <1> jmp short get\_argp1

4302 <1> get\_argpk:

4303 <1> ; Argument is in kernel's memory space

4304 0000DC21 66C705[48040300]00- <1> mov word [ncount], PAGE\_SIZE ; 4096

4304 0000DC29 10 <1>

4305 0000DC2A 8935[44040300] <1> mov [nbase], esi

4306 0000DC30 8305[44040300]04 <1> add dword [nbase], 4

4307 0000DC37 8B06 <1> mov eax, [esi] ; virtual addr. = physcal addr.

4308 0000DC39 C3 <1> retn

4309 <1> get\_argp2:

4310 <1> ; 21/07/2015

4311 <1> ;mov eax, 4

4312 0000DC3A 8B15[44040300] <1> mov edx, [nbase] ; 18/10/2015

4313 0000DC40 0105[44040300] <1> add [nbase], eax

4314 0000DC46 662905[48040300] <1> sub [ncount], ax

4315 <1> ;

4316 0000DC4D 8B02 <1> mov eax, [edx]

4317 0000DC4F C3 <1> retn

4318 <1> get\_argp\_err:

4319 0000DC50 A3[C8030300] <1> mov [u.error], eax

4320 <1> ; 14/11/2017

4321 0000DC55 B801000000 <1> mov eax, ERR\_BAD\_CMD\_ARG ; 01h ; TRDOS 8086

4322 0000DC5A A3[64030300] <1> mov [u.r0], eax

4323 0000DC5F E95AEAFFFF <1> jmp error

4324 <1> get\_argp3:

4325 0000DC64 B103 <1> mov cl, 3

4326 <1> get\_argp4:

4327 0000DC66 C1E008 <1> shl eax, 8

4328 0000DC69 665A <1> pop dx

4329 0000DC6B 88D0 <1> mov al, dl

4330 0000DC6D E2F7 <1> loop get\_argp4

4331 <1> ;pop esi

4332 0000DC6F C3 <1> retn

4333 <1>

4334 <1> sysstat:

4335 <1> ; 13/01/2017 - TRDOS 386 (TRDOS v2.0)

4336 <1> ; temporary !

4337 0000DC70 B801000000 <1> mov eax, ERR\_INV\_FNUMBER ; 'invalid function number !'

4338 0000DC75 A3[C8030300] <1> mov [u.error], eax

4339 0000DC7A A3[64030300] <1> mov [u.r0], eax

4340 0000DC7F E93AEAFFFF <1> jmp error

4341 <1>

4342 <1> sysfstat:

4343 <1> ; 13/01/2017 - TRDOS 386 (TRDOS v2.0)

4344 <1> ; temporary !

4345 0000DC84 B801000000 <1> mov eax, ERR\_INV\_FNUMBER ; 'invalid function number !'

4346 0000DC89 A3[C8030300] <1> mov [u.error], eax

4347 0000DC8E A3[64030300] <1> mov [u.r0], eax

4348 0000DC93 E926EAFFFF <1> jmp error

4349 <1>

4350 <1> fclose:

4351 <1> ; 06/10/2016 (TRDOS 386 = TRDOS v2.0)

4352 <1> ;

4353 <1> ; 18/06/2015 (Retro UNIX 386 v1 - Beginning)

4354 <1> ; (32 bit offset pointer modification)

4355 <1> ; 19/04/2013 - 12/01/2014 (Retro UNIX 8086 v1)

4356 <1> ;

4357 <1> ; Given the file descriptor (index to the u.fp list)

4358 <1> ; 'fclose' first gets the i-number of the file via 'getf'.

4359 <1> ; If i-node is active (i-number > 0) the entry in

4360 <1> ; u.fp list is cleared. If all the processes that opened

4361 <1> ; that file close it, then fsp etry is freed and the file

4362 <1> ; is closed. If not a return is taken.

4363 <1> ; If the file has been deleted while open, 'anyi' is called

4364 <1> ; to see anyone else has it open, i.e., see if it is appears

4365 <1> ; in another entry in the fsp table. Upon return from 'anyi'

4366 <1> ; a check is made to see if the file is special.

4367 <1> ;

4368 <1> ; INPUTS ->

4369 <1> ; r1 - contains the file descriptor (value=0,1,2...)

4370 <1> ; u.fp - list of entries in the fsp table

4371 <1> ; fsp - table of entries (4 words/entry) of open files.

4372 <1> ; OUTPUTS ->

4373 <1> ; r1 - contains the same file descriptor

4374 <1> ; r2 - contains i-number

4375 <1> ;

4376 <1> ; ((AX = R1))

4377 <1> ; ((Modified registers: eDX, eBX, eCX, eSI, eDI, eBP))

4378 <1> ;

4379 <1> ; Retro UNIX 8086 v1 modification : CF = 1

4380 <1> ; if i-number of the file is 0. (error)

4381 <1> ;

4382 <1> ; TRDOS 386 (06/10/2016)

4383 <1> ;

4384 <1> ; INPUT:

4385 <1> ; EAX = File Handle (File Descriptor, File Index)

4386 <1> ;

4387 <1> ; OUTPUT:

4388 <1> ; CF = 1 -> File not open !

4389 <1> ; CF = 0 -> OK!

4390 <1> ; EBX = File Number (System)

4391 <1> ; [cdev] = Logical DOS Drive Number

4392 <1> ; EAX = File Handle/Number (user)

4393 <1> ;

4394 <1> ; Modified Registers: EBX

4395 <1>

4396 0000DC98 50 <1> push eax ; File handle

4397 <1>

4398 0000DC99 E846000000 <1> call getf

4399 0000DC9E 0F8207240000 <1> jc device\_close ; eax = device number

4400 <1>

4401 0000DCA4 80BB[62690100]01 <1> cmp byte [ebx+OF\_MODE], 1 ; open mode ; 0 = empty entry

4402 0000DCAB 722E <1> jb short fclose\_1 ; 1 = read, 2 = write

4403 <1>

4404 0000DCAD 83F801 <1> cmp eax, 1 ; is the first cluster number > 0

4405 0000DCB0 7229 <1> jb short fclose\_1 ; no, this is empty entry

4406 <1>

4407 <1> fclose\_0:

4408 0000DCB2 FE8B[76690100] <1> dec byte [ebx+OF\_OPENCOUNT] ; decrement the number of processes

4409 <1> ; that have opened the file

4410 0000DCB8 7921 <1> jns short fclose\_1 ; jump if not negative (jump if bit 7 is 0)

4411 <1> ; if all processes haven't closed the file, return

4412 <1> ;

4413 <1> ; eax ; First cluster

4414 0000DCBA 31C0 <1> xor eax, eax ; 0

4415 0000DCBC 8883[62690100] <1> mov [ebx+OF\_MODE], al ; 0 = empty entry

4416 <1> ;mov [ebx+OF\_STATUS], al ; 0 = empty entry

4417 0000DCC2 66C1E302 <1> shl bx, 2

4418 0000DCC6 8983[30690100] <1> mov [ebx+OF\_FCLUSTER], eax ; 0

4419 0000DCCC 8983[486A0100] <1> mov [ebx+OF\_CCLUSTER], eax ; 0

4420 <1> ;mov [ebx+OF\_CCINDEX], eax ; 0

4421 0000DCD2 A3[74030300] <1> mov [u.fofp], eax ; 0

4422 0000DCD7 66C1EB02 <1> shr bx, 2

4423 <1> fclose\_1: ; 1:

4424 0000DCDB 58 <1> pop eax ; File handle (File Descriptor, File Index)

4425 0000DCDC C680[6A030300]00 <1> mov byte [eax+u.fp], 0 ; clear that entry in the u.fp list

4426 0000DCE3 C3 <1> retn

4427 <1>

4428 <1> getf:

4429 <1> ; 12/10/2016

4430 <1> ; 11/10/2016

4431 <1> ; 08/10/2016

4432 <1> ; 06/10/2016 (TRDOS 386 = TRDOS v2.0)

4433 <1> ; / get the device number and the i-number of an open file

4434 <1> ; 13/05/2015

4435 <1> ; 11/05/2015 (Retro UNIX 386 v1 - Beginning)

4436 <1> ; 19/04/2013 - 18/11/2013 (Retro UNIX 8086 v1)

4437 <1> ;

4438 0000DCE4 89C3 <1> mov ebx, eax

4439 <1> getf1:

4440 0000DCE6 83FB0A <1> cmp ebx, 10

4441 0000DCE9 730A <1> jnb short getf2

4442 0000DCEB 8A9B[6A030300] <1> mov bl, [ebx+u.fp]

4443 0000DCF1 08DB <1> or bl, bl

4444 0000DCF3 7503 <1> jnz short getf3

4445 <1> getf2:

4446 <1> ; 'File not open !' error (ax=0)

4447 0000DCF5 29C0 <1> sub eax, eax

4448 0000DCF7 C3 <1> retn

4449 <1> getf3:

4450 0000DCF8 F6C380 <1> test bl, 80h

4451 0000DCFB 7530 <1> jnz short getf5 ; device

4452 0000DCFD FECB <1> dec bl ; 0 based

4453 0000DCFF 8A83[58690100] <1> mov al, [ebx+OF\_DRIVE]

4454 0000DD05 A2[46030300] <1> mov [cdev], al

4455 0000DD0A C0E302 <1> shl bl, 2 ; \*4 (dword offset)

4456 0000DD0D 8B83[A8690100] <1> mov eax, [ebx+OF\_SIZE]

4457 0000DD13 A3[55040300] <1> mov [i.size], eax ; file size

4458 0000DD18 8D83[80690100] <1> lea eax, [ebx+OF\_POINTER] ;12/10/2016

4459 0000DD1E A3[74030300] <1> mov [u.fofp], eax

4460 0000DD23 8B83[30690100] <1> mov eax, [ebx+OF\_FCLUSTER]

4461 0000DD29 C0EB02 <1> shr bl, 2 ; /4 (byte offset)

4462 <1> getf4:

4463 0000DD2C C3 <1> retn

4464 <1> getf5:

4465 <1> ; get device number

4466 0000DD2D 80E37F <1> and bl, 7Fh ; 1 to 7Fh

4467 0000DD30 FECB <1> dec bl ; 0 based (0 to 7Eh)

4468 0000DD32 8A83[8A670100] <1> mov al, [ebx+DEV\_DRIVER]

4469 0000DD38 8AAB[F4660100] <1> mov ch, [ebx+DEV\_ACCESS]

4470 0000DD3E 8A8B[A8670100] <1> mov cl, [ebx+DEV\_OPENMODE]

4471 0000DD44 80E5FE <1> and ch, 0FEh ; reset bit 0 ; dev\_close

4472 0000DD47 F9 <1> stc ; cf = 1

4473 0000DD48 C3 <1> retn

4474 <1>

4475 <1> trans\_addr\_nmbp:

4476 <1> ; 18/10/2015

4477 <1> ; 12/10/2015

4478 0000DD49 8B2D[7C030300] <1> mov ebp, [u.namep]

4479 <1> trans\_addr\_nm:

4480 <1> ; Convert virtual (pathname) address to physical address

4481 <1> ; (Retro UNIX 386 v1 feature only !)

4482 <1> ; 18/10/2015

4483 <1> ; 12/10/2015 (u.pnbase & u.pncount has been removed from code)

4484 <1> ; 02/07/2015

4485 <1> ; 17/06/2015

4486 <1> ; 16/06/2015

4487 <1> ;

4488 <1> ; INPUTS:

4489 <1> ; ebp = pathname address (virtual) ; [u.namep]

4490 <1> ; [u.pgdir] = user's page directory

4491 <1> ; OUTPUT:

4492 <1> ; esi = physical address of the pathname

4493 <1> ; ecx = remain byte count in the page

4494 <1> ;

4495 <1> ; (Modified registers: EAX, EBX, ECX, EDX, ESI)

4496 <1> ;

4497 0000DD4F 833D[BC030300]00 <1> cmp dword [u.ppgdir], 0 ; /etc/init ? (sysexec)

4498 0000DD56 7618 <1> jna short trans\_addr\_nmk ; the caller is os kernel;

4499 <1> ; it is already physical address

4500 0000DD58 50 <1> push eax

4501 0000DD59 89EB <1> mov ebx, ebp ; [u.namep] ; pathname address (virtual)

4502 0000DD5B E82F75FFFF <1> call get\_physical\_addr ; get physical address

4503 0000DD60 7204 <1> jc short tr\_addr\_nm\_err

4504 <1> ; 18/10/2015

4505 <1> ; eax = physical address

4506 <1> ; cx = remain byte count in page (1-4096)

4507 <1> ; 12/10/2015 (cx = [u.pncount])

4508 0000DD62 89C6 <1> mov esi, eax ; 12/10/2015 (esi=[u.pnbase])

4509 0000DD64 58 <1> pop eax

4510 0000DD65 C3 <1> retn

4511 <1>

4512 <1> tr\_addr\_nm\_err:

4513 0000DD66 A3[C8030300] <1> mov [u.error], eax

4514 <1> ;pop eax

4515 0000DD6B E94EE9FFFF <1> jmp error

4516 <1>

4517 <1> trans\_addr\_nmk:

4518 <1> ; 12/10/2015

4519 <1> ; 02/07/2015

4520 0000DD70 8B35[7C030300] <1> mov esi, [u.namep] ; [u.pnbase]

4521 0000DD76 66B90010 <1> mov cx, PAGE\_SIZE ; 4096 ; [u.pncount]

4522 0000DD7A C3 <1> retn

4523 <1>

4524 <1>

4525 <1> sysbreak:

4526 <1> ; 18/10/2015

4527 <1> ; 07/10/2015

4528 <1> ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)

4529 <1> ; 20/06/2013 - 24/03/2014 (Retro UNIX 8086 v1)

4530 <1> ;

4531 <1> ; 'sysbreak' sets the programs break points.

4532 <1> ; It checks the current break point (u.break) to see if it is

4533 <1> ; between "core" and the stack (sp). If it is, it is made an

4534 <1> ; even address (if it was odd) and the area between u.break

4535 <1> ; and the stack is cleared. The new breakpoint is then put

4536 <1> ; in u.break and control is passed to 'sysret'.

4537 <1> ;

4538 <1> ; Calling sequence:

4539 <1> ; sysbreak; addr

4540 <1> ; Arguments: -

4541 <1> ;

4542 <1> ; Inputs: u.break - current breakpoint

4543 <1> ; Outputs: u.break - new breakpoint

4544 <1> ; area between old u.break and the stack (sp) is cleared.

4545 <1> ; ...............................................................

4546 <1> ;

4547 <1> ; Retro UNIX 8086 v1 modification:

4548 <1> ; The user/application program puts breakpoint address

4549 <1> ; in BX register as 'sysbreak' system call argument.

4550 <1> ; (argument transfer method 1)

4551 <1> ;

4552 <1> ; NOTE: Beginning of core is 0 in Retro UNIX 8086 v1 !

4553 <1> ; ((!'sysbreak' is not needed in Retro UNIX 8086 v1!))

4554 <1> ; NOTE:

4555 <1> ; 'sysbreak' clears extended part (beyond of previous

4556 <1> ; 'u.break' address) of user's memory for original unix's

4557 <1> ; 'bss' compatibility with Retro UNIX 8086 v1 (19/11/2013)

4558 <1>

4559 <1> ; mov u.break,r1 / move users break point to r1

4560 <1> ; cmp r1,$core / is it the same or lower than core?

4561 <1> ; blos 1f / yes, 1f

4562 <1> ; 23/06/2015

4563 0000DD7B 8B2D[90030300] <1> mov ebp, [u.break] ; virtual address (offset)

4564 <1> ;and ebp, ebp

4565 <1> ;jz short sysbreak\_3

4566 <1> ; Retro UNIX 386 v1 NOTE: u.break points to virtual address !!!

4567 <1> ; (Even break point address is not needed for Retro UNIX 386 v1)

4568 0000DD81 8B15[5C030300] <1> mov edx, [u.sp] ; kernel stack at the beginning of sys call

4569 0000DD87 83C20C <1> add edx, 12 ; EIP -4-> CS -4-> EFLAGS -4-> ESP (user)

4570 <1> ; 07/10/2015

4571 0000DD8A 891D[90030300] <1> mov [u.break], ebx ; virtual address !!!

4572 <1> ;

4573 0000DD90 3B1A <1> cmp ebx, [edx] ; compare new break point with

4574 <1> ; with top of user's stack (virtual!)

4575 0000DD92 7323 <1> jnb short sysbreak\_3

4576 <1> ; cmp r1,sp / is it the same or higher

4577 <1> ; / than the stack?

4578 <1> ; bhis 1f / yes, 1f

4579 0000DD94 89DE <1> mov esi, ebx

4580 0000DD96 29EE <1> sub esi, ebp ; new break point - old break point

4581 0000DD98 761D <1> jna short sysbreak\_3

4582 <1> ;push ebx

4583 <1> sysbreak\_1:

4584 0000DD9A 89EB <1> mov ebx, ebp

4585 0000DD9C E8EE74FFFF <1> call get\_physical\_addr ; get physical address

4586 0000DDA1 72C3 <1> jc tr\_addr\_nm\_err

4587 <1> ; 18/10/2015

4588 0000DDA3 89C7 <1> mov edi, eax

4589 0000DDA5 29C0 <1> sub eax, eax ; 0

4590 <1> ; ECX = remain byte count in page (1-4096)

4591 0000DDA7 39CE <1> cmp esi, ecx

4592 0000DDA9 7302 <1> jnb short sysbreak\_2

4593 0000DDAB 89F1 <1> mov ecx, esi

4594 <1> sysbreak\_2:

4595 0000DDAD 29CE <1> sub esi, ecx

4596 0000DDAF 01CD <1> add ebp, ecx

4597 0000DDB1 F3AA <1> rep stosb

4598 0000DDB3 09F6 <1> or esi, esi

4599 0000DDB5 75E3 <1> jnz short sysbreak\_1

4600 <1> ;

4601 <1> ; bit $1,r1 / is it an odd address

4602 <1> ; beq 2f / no, its even

4603 <1> ; clrb (r1)+ / yes, make it even

4604 <1> ; 2: / clear area between the break point and the stack

4605 <1> ; cmp r1,sp / is it higher or same than the stack

4606 <1> ; bhis 1f / yes, quit

4607 <1> ; clr (r1)+ / clear word

4608 <1> ; br 2b / go back

4609 <1> ;pop ebx

4610 <1> sysbreak\_3: ; 1:

4611 <1> ;mov [u.break], ebx ; virtual address !!!

4612 <1> ; jsr r0,arg; u.break / put the "address"

4613 <1> ; / in u.break (set new break point)

4614 <1> ; br sysret4 / br sysret

4615 0000DDB7 E922E9FFFF <1> jmp sysret

4616 <1>

4617 <1> sysseek: ; / moves read write pointer in an fsp entry

4618 <1> ; 06/11/2016 - TRDOS 386 (TRDOS v2.0)

4619 <1> ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)

4620 <1> ; 07/07/2013 - 05/08/2013 (Retro UNIX 8086 v1)

4621 <1> ;

4622 <1> ; 'sysseek' changes the r/w pointer of (3rd word of in an

4623 <1> ; fsp entry) of an open file whose file descriptor is in u.r0.

4624 <1> ; The file descriptor refers to a file open for reading or

4625 <1> ; writing. The read (or write) pointer is set as follows:

4626 <1> ; \* if 'ptrname' is 0, the pointer is set to offset.

4627 <1> ; \* if 'ptrname' is 1, the pointer is set to its

4628 <1> ; current location plus offset.

4629 <1> ; \* if 'ptrname' is 2, the pointer is set to the

4630 <1> ; size of file plus offset.

4631 <1> ; The error bit (e-bit) is set for an undefined descriptor.

4632 <1> ;

4633 <1> ; Calling sequence:

4634 <1> ; sysseek; offset; ptrname

4635 <1> ; Arguments:

4636 <1> ; offset - number of bytes desired to move

4637 <1> ; the r/w pointer

4638 <1> ; ptrname - a switch indicated above

4639 <1> ;

4640 <1> ; Inputs: r0 - file descriptor

4641 <1> ; Outputs: -

4642 <1> ; ...............................................................

4643 <1> ;

4644 <1> ; Retro UNIX 8086 v1 modification:

4645 <1> ; 'sysseek' system call has three arguments; so,

4646 <1> ; \* 1st argument, file descriptor is in BX (BL) register

4647 <1> ; \* 2nd argument, offset is in CX register

4648 <1> ; \* 3rd argument, ptrname/switch is in DX (DL) register

4649 <1>

4650 0000DDBC E821000000 <1> call seektell

4651 <1> ; EAX = Current R/W pointer of the file

4652 <1> ; EBX = [u.fofp]

4653 <1> ; [u.base] = offset (ECX input)

4654 <1>

4655 0000DDC1 0305[84030300] <1> add eax, [u.base]

4656 0000DDC7 8903 <1> mov [ebx], eax

4657 0000DDC9 E910E9FFFF <1> jmp sysret

4658 <1>

4659 <1> systell: ; / get the r/w pointer

4660 <1> ; 06/11/2016 - TRDOS 386 (TRDOS v2.0) - temporary !-

4661 <1> ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)

4662 <1> ; 07/07/2013 - 05/08/2013 (Retro UNIX 8086 v1)

4663 <1> ;

4664 <1> ; Retro UNIX 8086 v1 modification:

4665 <1> ; ! 'systell' does not work in original UNIX v1,

4666 <1> ; it returns with error !

4667 <1> ; Inputs: r0 - file descriptor

4668 <1> ; Outputs: r0 - file r/w pointer

4669 <1>

4670 <1> ;xor ecx, ecx ; 0

4671 0000DDCE BA01000000 <1> mov edx, 1 ; 05/08/2013

4672 <1> ;call seektell

4673 0000DDD3 E810000000 <1> call seektell0 ; 05/08/2013

4674 <1> ;; 06/11/2016

4675 <1> ;; mov eax, [ebx]

4676 0000DDD8 A3[64030300] <1> mov [u.r0], eax

4677 0000DDDD E9FCE8FFFF <1> jmp sysret

4678 <1>

4679 <1> ; Original unix v1 'systell' system call:

4680 <1> ; jsr r0,seektell

4681 <1> ; br error4

4682 <1>

4683 <1> seektell:

4684 <1> ; 06/11/2016 - TRDOS 386 (TRDOS v2.0)

4685 <1> ; 03/01/2016

4686 <1> ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)

4687 <1> ; 07/07/2013 - 05/08/2013 (Retro UNIX 8086 v1)

4688 <1> ;

4689 <1> ; 'seektell' puts the arguments from sysseek and systell

4690 <1> ; call in u.base and u.count. It then gets the i-number of

4691 <1> ; the file from the file descriptor in u.r0 and by calling

4692 <1> ; getf. The i-node is brought into core and then u.count

4693 <1> ; is checked to see it is a 0, 1, or 2.

4694 <1> ; If it is 0 - u.count stays the same

4695 <1> ; 1 - u.count = offset (u.fofp)

4696 <1> ; 2 - u.count = i.size (size of file)

4697 <1> ;

4698 <1> ; !! Retro UNIX 8086 v1 modification:

4699 <1> ; Argument 1, file descriptor is in BX;

4700 <1> ; Argument 2, offset is in CX;

4701 <1> ; Argument 3, ptrname/switch is in DX register.

4702 <1> ;

4703 <1> ; ((Return -> eax = base for offset (position= base+offset))

4704 <1> ;

4705 0000DDE2 890D[84030300] <1> mov [u.base], ecx ; offset

4706 <1> seektell0:

4707 0000DDE8 8915[88030300] <1> mov [u.count], edx

4708 <1> ; EBX = file descriptor (file number)

4709 0000DDEE E8F3FEFFFF <1> call getf1

4710 <1> ; EAX = First cluster of the file

4711 <1> ; EBX = File number (Open file number)

4712 <1> ; [u.fofp] = Pointer to File pointer

4713 <1> ; [i.size] = File size

4714 <1>

4715 0000DDF3 09C0 <1> or eax, eax

4716 0000DDF5 7514 <1> jnz short seektell1

4717 <1>

4718 0000DDF7 B80A000000 <1> mov eax, ERR\_FILE\_NOT\_OPEN

4719 0000DDFC A3[64030300] <1> mov [u.r0], eax

4720 0000DE01 A3[C8030300] <1> mov dword [u.error], eax ; 'file not open !'

4721 0000DE06 E9B3E8FFFF <1> jmp error

4722 <1>

4723 <1> seektell1:

4724 0000DE0B 8B1D[74030300] <1> mov ebx, [u.fofp]

4725 0000DE11 803D[88030300]01 <1> cmp byte [u.count], 1

4726 0000DE18 7705 <1> ja short seektell2

4727 0000DE1A 7409 <1> je short seektell3

4728 0000DE1C 31C0 <1> xor eax, eax

4729 0000DE1E C3 <1> retn

4730 <1>

4731 <1> seektell2:

4732 0000DE1F A1[55040300] <1> mov eax, [i.size]

4733 0000DE24 C3 <1> retn

4734 <1>

4735 <1> seektell3:

4736 0000DE25 8B03 <1> mov eax, [ebx]

4737 0000DE27 C3 <1> retn

4738 <1>

4739 <1> sysintr: ; / set interrupt handling

4740 <1> ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)

4741 <1> ; 07/07/2013 (Retro UNIX 8086 v1)

4742 <1> ;

4743 <1> ; 'sysintr' sets the interrupt handling value. It puts

4744 <1> ; argument of its call in u.intr then branches into 'sysquit'

4745 <1> ; routine. u.tty is checked if to see if a control tty exists.

4746 <1> ; If one does the interrupt character in the tty buffer is

4747 <1> ; cleared and 'sysret'is called. If one does not exits

4748 <1> ; 'sysret' is just called.

4749 <1> ;

4750 <1> ; Calling sequence:

4751 <1> ; sysintr; arg

4752 <1> ; Argument:

4753 <1> ; arg - if 0, interrupts (ASCII DELETE) are ignored.

4754 <1> ; - if 1, intterupts cause their normal result

4755 <1> ; i.e force an exit.

4756 <1> ; - if arg is a location within the program,

4757 <1> ; control is passed to that location when

4758 <1> ; an interrupt occurs.

4759 <1> ; Inputs: -

4760 <1> ; Outputs: -

4761 <1> ; ...............................................................

4762 <1> ;

4763 <1> ; Retro UNIX 8086 v1 modification:

4764 <1> ; 'sysintr' system call sets u.intr to value of BX

4765 <1> ; then branches into sysquit.

4766 <1> ;

4767 0000DE28 66891D[AA030300] <1> mov [u.intr], bx

4768 <1> ; jsr r0,arg; u.intr / put the argument in u.intr

4769 <1> ; br 1f / go into quit routine

4770 0000DE2F E9AAE8FFFF <1> jmp sysret

4771 <1>

4772 <1> sysquit:

4773 <1> ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)

4774 <1> ; 07/07/2013 (Retro UNIX 8086 v1)

4775 <1> ;

4776 <1> ; 'sysquit' turns off the quit signal. it puts the argument of

4777 <1> ; the call in u.quit. u.tty is checked if to see if a control

4778 <1> ; tty exists. If one does the interrupt character in the tty

4779 <1> ; buffer is cleared and 'sysret'is called. If one does not exits

4780 <1> ; 'sysret' is just called.

4781 <1> ;

4782 <1> ; Calling sequence:

4783 <1> ; sysquit; arg

4784 <1> ; Argument:

4785 <1> ; arg - if 0, this call diables quit signals from the

4786 <1> ; typewriter (ASCII FS)

4787 <1> ; - if 1, quits are re-enabled and cause execution to

4788 <1> ; cease and a core image to be produced.

4789 <1> ; i.e force an exit.

4790 <1> ; - if arg is an addres in the program,

4791 <1> ; a quit causes control to sent to that

4792 <1> ; location.

4793 <1> ; Inputs: -

4794 <1> ; Outputs: -

4795 <1> ; ...............................................................

4796 <1> ;

4797 <1> ; Retro UNIX 8086 v1 modification:

4798 <1> ; 'sysquit' system call sets u.quit to value of BX

4799 <1> ; then branches into 'sysret'.

4800 <1> ;

4801 0000DE34 66891D[AC030300] <1> mov [u.quit], bx

4802 0000DE3B E99EE8FFFF <1> jmp sysret

4803 <1> ; jsr r0,arg; u.quit / put argument in u.quit

4804 <1> ;1:

4805 <1> ; mov u.ttyp,r1 / move pointer to control tty buffer

4806 <1> ; / to r1

4807 <1> ; beq sysret4 / return to user

4808 <1> ; clrb 6(r1) / clear the interrupt character

4809 <1> ; / in the tty buffer

4810 <1> ; br sysret4 / return to user

4811 <1>

4812 <1> anyi:

4813 <1> ; 06/10/2016 (TRDOS 386 = TRDOS v2.0)

4814 <1> ; Major Modification!

4815 <1> ; TRDOS 386 does not permit to delete a file while it is open

4816 <1> ; The role of 'anyi' procedure has beeen changed to ensure that.

4817 <1> ;

4818 <1> ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)

4819 <1> ; 25/04/2013 (Retro UNIX 8086 v1)

4820 <1> ;

4821 <1> ; 'anyi' is called if a file deleted while open.

4822 <1> ; "anyi" checks to see if someone else has opened this file.

4823 <1> ;

4824 <1> ; INPUTS ->

4825 <1> ; r1 - contains an i-number

4826 <1> ; fsp - start of table containing open files

4827 <1> ;

4828 <1> ; OUTPUTS ->

4829 <1> ; "deleted" flag set in fsp entry of another occurrence of

4830 <1> ; this file and r2 points 1st word of this fsp entry.

4831 <1> ; if file not found - bit in i-node map is cleared

4832 <1> ; (i-node is freed)

4833 <1> ; all blocks related to i-node are freed

4834 <1> ; all flags in i-node are cleared

4835 <1> ; ((AX = R1)) input

4836 <1> ;

4837 <1> ; (Retro UNIX Prototype : 02/12/2012, UNIXCOPY.ASM)

4838 <1> ; ((Modified registers: eDX, eCX, eBX, eSI, eDI, eBP))

4839 <1> ;

4840 <1> ; / r1 contains an i-number

4841 <1>

4842 <1> ; TRDOS 386 (06/10/2016)

4843 <1> ;

4844 <1> ; INPUT:

4845 <1> ; EAX = First Cluster

4846 <1> ; DL = Logical DOS Drive Number

4847 <1> ;

4848 <1> ; OUTPUT:

4849 <1> ; CF = 1 -> EBX = File Handle/Number/Index

4850 <1> ; CF = 0 -> EBX = 0

4851 <1> ;

4852 <1> ; Modified Registers: EBX

4853 <1>

4854 0000DE40 31DB <1> xor ebx, ebx

4855 <1> anyi\_0:

4856 0000DE42 80BB[62690100]00 <1> cmp byte [ebx+OF\_MODE], 0 ; 0 = empty entry

4857 0000DE49 770A <1> ja short anyi\_2 ; 1 (r), 2 (w) or 3 (r&w)

4858 <1> anyi\_1:

4859 0000DE4B FEC3 <1> inc bl

4860 0000DE4D 80FB0A <1> cmp bl, OPENFILES ; max. count of open files

4861 0000DE50 72F0 <1> jb short anyi\_0

4862 0000DE52 31C0 <1> xor eax, eax

4863 0000DE54 C3 <1> retn

4864 <1> anyi\_2:

4865 0000DE55 3A93[58690100] <1> cmp dl, [ebx+OF\_DRIVE]

4866 0000DE5B 75EE <1> jne short anyi\_1

4867 0000DE5D 66C1E302 <1> shl bx, 2 ; \*4 (dword offset)

4868 0000DE61 3B83[30690100] <1> cmp eax, [ebx+OF\_FCLUSTER]

4869 0000DE67 7406 <1> je short anyi\_3

4870 0000DE69 66C1EB02 <1> shr bx, 2 ; /4 (byte offset)

4871 0000DE6D EBDC <1> jmp short anyi\_1

4872 <1> anyi\_3:

4873 0000DE6F 66C1EB02 <1> shr bx, 2 ; /4 (bytes offset) (index)

4874 0000DE73 F9 <1> stc

4875 0000DE74 C3 <1> retn

4876 <1>

4877 <1> ; Retro UNIX 386 v1 Kernel (v0.2) - SYS9.INC

4878 <1> ; Last Modification: 09/12/2015

4879 <1>

4880 <1> syssleep:

4881 <1> ; 29/06/2015 - (Retro UNIX 386 v1)

4882 <1> ; 11/06/2014 - (Retro UNIX 8086 v1)

4883 <1> ;

4884 <1> ; Retro UNIX 8086 v1 feature only

4885 <1> ; (INPUT -> none)

4886 <1> ;

4887 0000DE75 0FB61D[B3030300] <1> movzx ebx, byte [u.uno] ; process number

4888 0000DE7C 8AA3[7F000300] <1> mov ah, [ebx+p.ttyc-1] ; current/console tty

4889 0000DE82 E841190000 <1> call sleep

4890 0000DE87 E952E8FFFF <1> jmp sysret

4891 <1>

4892 <1> \_vp\_clr:

4893 <1> ; Reset/Clear Video Page

4894 <1> ;

4895 <1> ; 30/06/2015 - (Retro UNIX 386 v1)

4896 <1> ; 21/05/2013 - 30/10/2013(Retro UNIX 8086 v1) (U0.ASM)

4897 <1> ;

4898 <1> ; Retro UNIX 8086 v1 feature only !

4899 <1> ;

4900 <1> ; INPUTS ->

4901 <1> ; BH = video page number

4902 <1> ;

4903 <1> ; OUTPUT ->

4904 <1> ; none

4905 <1> ; ((Modified registers: eAX, BH, eCX, eDX, eSI, eDI))

4906 <1> ;

4907 <1> ; 04/12/2013

4908 0000DE8C 28C0 <1> sub al, al

4909 <1> ; al = 0 (clear video page)

4910 <1> ; bh = video page ; 13/05/2016

4911 0000DE8E B407 <1> mov ah, 07h

4912 <1> ; ah = 7 (attribute/color)

4913 0000DE90 6631C9 <1> xor cx, cx ; 0, left upper column (cl) & row (cl)

4914 0000DE93 66BA4F18 <1> mov dx, 184Fh ; right lower column & row (dl=24, dh=79)

4915 0000DE97 E86E3BFFFF <1> call \_scroll\_up

4916 <1> ; bh = video page

4917 0000DE9C 6631D2 <1> xor dx, dx ; 0 (cursor position)

4918 0000DE9F E9A43EFFFF <1> jmp \_set\_cpos

4919 <1>

4920 <1> sysmsg:

4921 <1> ; 13/05/2016

4922 <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)

4923 <1> ; 01/07/2015 - 11/11/2015 (Retro UNIX 386 v1)

4924 <1> ; Print user-application message on user's console tty

4925 <1> ;

4926 <1> ; Input -> EBX = Message address

4927 <1> ; ECX = Message length (max. 255)

4928 <1> ; DL = Color (IBM PC Rombios color attributes)

4929 <1> ;

4930 0000DEA4 81F9FF000000 <1> cmp ecx, MAX\_MSG\_LEN ; 255

4931 0000DEAA 0F872EE8FFFF <1> ja sysret ; nothing to do with big message size

4932 0000DEB0 08C9 <1> or cl, cl

4933 0000DEB2 0F8426E8FFFF <1> jz sysret

4934 0000DEB8 20D2 <1> and dl, dl

4935 0000DEBA 7502 <1> jnz short sysmsg0

4936 0000DEBC B207 <1> mov dl, 07h ; default color

4937 <1> ; (black background, light gray character)

4938 <1> sysmsg0:

4939 0000DEBE 891D[84030300] <1> mov [u.base], ebx

4940 0000DEC4 8815[67580100] <1> mov [ccolor], dl ; color attributes

4941 0000DECA 89E5 <1> mov ebp, esp

4942 0000DECC 31DB <1> xor ebx, ebx ; 0

4943 0000DECE 891D[8C030300] <1> mov [u.nread], ebx ; 0

4944 <1> ;

4945 0000DED4 381D[C6030300] <1> cmp [u.kcall], bl ; 0

4946 0000DEDA 7769 <1> ja short sysmsgk ; Temporary (01/07/2015)

4947 <1> ;

4948 0000DEDC 890D[88030300] <1> mov [u.count], ecx

4949 0000DEE2 41 <1> inc ecx ; + 00h ; ASCIIZ

4950 0000DEE3 29CC <1> sub esp, ecx

4951 0000DEE5 89E7 <1> mov edi, esp

4952 0000DEE7 89E6 <1> mov esi, esp

4953 0000DEE9 66891D[C4030300] <1> mov [u.pcount], bx ; reset page (phy. addr.) counter

4954 <1> ; 11/11/2015

4955 0000DEF0 8A25[94030300] <1> mov ah, [u.ttyp] ; recent open tty

4956 <1> ; 0 = none

4957 0000DEF6 FECC <1> dec ah

4958 0000DEF8 790C <1> jns short sysmsg1

4959 0000DEFA 8A1D[B3030300] <1> mov bl, [u.uno] ; process number

4960 0000DF00 8AA3[7F000300] <1> mov ah, [ebx+p.ttyc-1] ; user's (process's) console tty

4961 <1> sysmsg1:

4962 0000DF06 8825[96030300] <1> mov [u.ttyn], ah

4963 <1> sysmsg2:

4964 0000DF0C E808080000 <1> call cpass

4965 0000DF11 7416 <1> jz short sysmsg5

4966 0000DF13 AA <1> stosb

4967 0000DF14 20C0 <1> and al, al

4968 0000DF16 75F4 <1> jnz short sysmsg2

4969 <1> sysmsg3:

4970 0000DF18 80FC07 <1> cmp ah, 7 ; tty number

4971 0000DF1B 7711 <1> ja short sysmsg6 ; serial port

4972 0000DF1D E83E000000 <1> call print\_cmsg

4973 <1> sysmsg4:

4974 0000DF22 89EC <1> mov esp, ebp

4975 0000DF24 E9B5E7FFFF <1> jmp sysret

4976 <1> sysmsg5:

4977 0000DF29 C60700 <1> mov byte [edi], 0

4978 0000DF2C EBEA <1> jmp short sysmsg3

4979 <1> sysmsg6:

4980 0000DF2E 8A06 <1> mov al, [esi]

4981 0000DF30 E891180000 <1> call sndc

4982 0000DF35 72EB <1> jc short sysmsg4

4983 0000DF37 803E00 <1> cmp byte [esi], 0 ; 0 is stop character

4984 0000DF3A 76E6 <1> jna short sysmsg4

4985 0000DF3C 46 <1> inc esi

4986 0000DF3D 8A25[96030300] <1> mov ah, [u.ttyn]

4987 0000DF43 EBE9 <1> jmp short sysmsg6

4988 <1>

4989 <1> sysmsgk: ; Temporary (01/07/2015)

4990 <1> ; The message has been sent by Kernel (ASCIIZ string)

4991 <1> ; (ECX -character count- will not be considered)

4992 0000DF45 8B35[84030300] <1> mov esi, [u.base]

4993 0000DF4B 8A25[66580100] <1> mov ah, [ptty] ; present/current screen (video page)

4994 0000DF51 8825[96030300] <1> mov [u.ttyn], ah

4995 0000DF57 C605[C6030300]00 <1> mov byte [u.kcall], 0

4996 0000DF5E EBB8 <1> jmp short sysmsg3

4997 <1>

4998 <1> print\_cmsg:

4999 <1> ; 18/11/2017

5000 <1> ; 13/05/2016 - TRDOS 386 (TRDOS v2.0)

5001 <1> ; 01/07/2015 (Retro UNIX 386 v1)

5002 <1> ;

5003 <1> ; print message (on user's console tty)

5004 <1> ; with requested color

5005 <1> ;

5006 <1> ; INPUTS:

5007 <1> ; esi = message address

5008 <1> ; [u.ttyn] = tty number (0 to 7)

5009 <1> ; [ccolor] = color attributes (IBM PC BIOS colors)

5010 <1>

5011 <1> ;mov bh, ah

5012 0000DF60 8A3D[96030300] <1> mov bh, [u.ttyn]

5013 <1> ;mov bl, [ccolor] ; \*

5014 <1> pcmsg1:

5015 0000DF66 AC <1> lodsb

5016 0000DF67 20C0 <1> and al, al ; 0

5017 0000DF69 740F <1> jz short pcmsg2

5018 0000DF6B 56 <1> push esi

5019 0000DF6C 8A1D[67580100] <1> mov bl, [ccolor] ; \* (video.s 'u11'&'beep' change BL)

5020 <1> ;mov bh, [u.ttyn]

5021 0000DF72 E83B3DFFFF <1> call \_write\_tty

5022 0000DF77 5E <1> pop esi

5023 0000DF78 EBEC <1> jmp short pcmsg1

5024 <1> pcmsg2:

5025 0000DF7A C3 <1> retn

5026 <1>

5027 <1> sysgeterr:

5028 <1> ; 09/12/2015

5029 <1> ; 21/09/2015 - (Retro UNIX 386 v1 feature only!)

5030 <1> ; Get last error number or page fault count

5031 <1> ; (for debugging)

5032 <1> ;

5033 <1> ; Input -> EBX = return type

5034 <1> ; 0 = last error code (which is in 'u.error')

5035 <1> ; FFFFFFFFh = page fault count for running process

5036 <1> ; FFFFFFFEh = total page fault count

5037 <1> ; 1 .. FFFFFFFDh = undefined

5038 <1> ;

5039 <1> ; Output -> EAX = last error number or page fault count

5040 <1> ; (depending on EBX input)

5041 <1> ;

5042 0000DF7B 21DB <1> and ebx, ebx

5043 0000DF7D 750B <1> jnz short glerr\_2

5044 <1> glerr\_0:

5045 0000DF7F A1[C8030300] <1> mov eax, [u.error]

5046 <1> glerr\_1:

5047 0000DF84 A3[64030300] <1> mov [u.r0], eax

5048 0000DF89 C3 <1> retn

5049 <1> glerr\_2:

5050 0000DF8A 43 <1> inc ebx ; FFFFFFFFh -> 0, FFFFFFFEh -> FFFFFFFFh

5051 0000DF8B 74FD <1> jz short glerr\_2 ; page fault count for process

5052 0000DF8D 43 <1> inc ebx ; FFFFFFFFh -> 0

5053 0000DF8E 75EF <1> jnz short glerr\_0

5054 0000DF90 A1[80050300] <1> mov eax, [PF\_Count] ; total page fault count

5055 0000DF95 EBED <1> jmp short glerr\_1

5056 <1> glerr\_3:

5057 0000DF97 A1[CC030300] <1> mov eax, [u.pfcount]

5058 0000DF9C EBE6 <1> jmp short glerr\_1

5059 <1>

5060 <1> load\_and\_run\_file:

5061 <1> ; 18/11/2017

5062 <1> ; 22/01/2017

5063 <1> ; 04/01/2017, 07/01/2017

5064 <1> ; 24/10/2016

5065 <1> ; 24/04/2016, 02/05/2016, 03/05/2016, 06/05/2016

5066 <1> ; 23/04/2016 (TRDOS 386 = TRDOS v2.0)

5067 <1> ; 23/10/2015 (Retro UNIX 386 v1, 'sysexec')

5068 <1> ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)

5069 <1> ; 03/06/2013 - 06/12/2013 (Retro UNIX 8086 v1)

5070 <1> ; EAX = First Cluster number

5071 <1> ; EDX = File Size

5072 <1> ; ESI = Argument list address

5073 <1> ; [argc] = argument count

5074 <1> ; [u.nread] = argument list length

5075 <1> ; [esp] = return address to the caller (\*)

5076 <1> ;

5077 0000DF9E 8935[4C040300] <1> mov [argv], esi

5078 0000DFA4 8915[55040300] <1> mov [i.size], edx

5079 0000DFAA A3[51040300] <1> mov [ii], eax

5080 <1>

5081 <1> ;sti ; 07/01/2017

5082 <1> ;mov eax, [k\_page\_dir]

5083 <1> ;mov [u.pgdir], eax

5084 0000DFAF 31C0 <1> xor eax, eax ; clc ; \*\*\* ; 04/01/2017

5085 <1> ;mov [u.r0], eax ; 0 ; 07/01/2017

5086 <1>

5087 <1> ; 06/05/2016

5088 <1> ; Set 'sysexit' return order to MainProg

5089 <1> ;

5090 0000DFB1 58 <1> pop eax ; \* 'loc\_load\_and\_run\_file\_8:' address

5091 <1> ;; 22/01/2017

5092 <1> ;;cli ; 07/01/2017

5093 0000DFB2 8B25[D4570100] <1> mov esp, [tss.esp0]

5094 <1> ;

5095 <1> ; 'loc\_load\_run\_file\_8' address has

5096 <1> ; 'jmp loc\_file\_rw\_restore\_retn' instruction

5097 <1> ; 'loc\_file\_rw\_restore\_retn:' will return to

5098 <1> ; [mainprog\_return\_addr]

5099 <1> ; just after 'call command\_interpreter'

5100 <1> ;

5101 0000DFB8 68[3B630000] <1> push \_end\_of\_mainprog ; we must not return to here !

5102 0000DFBD FF35[BC650100] <1> push dword [mainprog\_return\_addr]

5103 0000DFC3 89E5 <1> mov ebp, esp ; \*\*

5104 <1> ;

5105 0000DFC5 9C <1> pushfd ; EFLAGS ; IRETD ; \*\*\*

5106 0000DFC6 6A08 <1> push KCODE ; cs ; IRETD

5107 0000DFC8 50 <1> push eax ; \* (eip) ; IRETD

5108 0000DFC9 8925[5C030300] <1> mov [u.sp], esp

5109 <1> ;mov byte [u.quant], time\_count

5110 0000DFCF 1E <1> push ds

5111 0000DFD0 06 <1> push es

5112 0000DFD1 0FA0 <1> push fs

5113 0000DFD3 0FA8 <1> push gs

5114 <1> ;mov eax, [u.r0]

5115 0000DFD5 29C0 <1> sub eax, eax

5116 0000DFD7 60 <1> pushad

5117 0000DFD8 68[DEC60000] <1> push sysret

5118 <1> ;push sysrel1 ; 07/01/2017

5119 0000DFDD 8925[60030300] <1> mov [u.usp], esp

5120 <1> ;

5121 0000DFE3 E845060000 <1> call wswap ; Save MainProg (process 1) 'u' structure

5122 <1> ; and registers for return (from program)

5123 0000DFE8 89EC <1> mov esp, ebp ; \*\*

5124 <1> ;;22/01/2017

5125 <1> ;;sti ; 07/01/2017

5126 0000DFEA 50 <1> push eax ; \* 'loc\_load\_and\_run\_file\_8:' address

5127 <1> ;

5128 <1> ;;; 02/05/2016

5129 <1> ;;; Create a new process (parent: MainProg)

5130 0000DFEB 31F6 <1> xor esi, esi

5131 <1> cnpm\_1: ; search p.stat table for unused process number

5132 0000DFED 46 <1> inc esi

5133 0000DFEE 80BE[AF000300]00 <1> cmp byte [esi+p.stat-1], 0 ; SFREE

5134 <1> ; is process active, unused, dead

5135 0000DFF5 760B <1> jna short cnpm\_2 ; it's unused so branch

5136 0000DFF7 6683FE10 <1> cmp si, nproc ; all processes checked

5137 0000DFFB 72F0 <1> jb short cnpm\_1 ; no, branch back

5138 0000DFFD E9AF83FFFF <1> jmp panic

5139 <1> cnpm\_2:

5140 0000E002 A1[B8030300] <1> mov eax, [u.pgdir] ; page directory of MainProg

5141 0000E007 A3[BC030300] <1> mov [u.ppgdir], eax ; parent's page directory

5142 0000E00C E8696BFFFF <1> call allocate\_page

5143 0000E011 0F829A83FFFF <1> jc panic

5144 <1> ; EAX = UPAGE (user structure page) address

5145 0000E017 A3[B4030300] <1> mov [u.upage], eax ; memory page for 'user' struct (child)

5146 0000E01C 89F7 <1> mov edi, esi

5147 0000E01E 66C1E702 <1> shl di, 2

5148 0000E022 8987[BC000300] <1> mov [edi+p.upage-4], eax ; memory page for 'user' struct

5149 0000E028 E8C76BFFFF <1> call clear\_page ; 03/05/2016

5150 <1> ;movzx eax, byte [p.ttyc] ; console tty (for MainProg)

5151 0000E02D 6629C0 <1> sub ax, ax ; 0

5152 0000E030 668986[7F000300] <1> mov [esi+p.ttyc-1], ax ; al - set child's console tty

5153 <1> ; ah - reset child's wait channel

5154 0000E037 89F0 <1> mov eax, esi

5155 0000E039 A2[B3030300] <1> mov [u.uno], al ; child process number

5156 0000E03E FE86[AF000300] <1> inc byte [esi+p.stat-1] ; 1, SRUN

5157 0000E044 66D1E6 <1> shl si, 1 ; multiply si by 2 to get index into p.pid table

5158 0000E047 66FF05[4E030300] <1> inc word [mpid] ; increment m.pid; get a new process name

5159 0000E04E 66A1[4E030300] <1> mov ax, [mpid]

5160 0000E054 668986[1E000300] <1> mov [esi+p.pid-2], ax ; put new process name

5161 <1> ; in child process' name slot

5162 <1> ;mov ax, [p.pid] ; get process name of MainProg

5163 0000E05B 66B80100 <1> mov ax, 1

5164 0000E05F 668986[3E000300] <1> mov [esi+p.ppid-2], ax ; put parent process name

5165 <1> ; in parent process slot for child

5166 0000E066 6648 <1> dec ax ; 0

5167 0000E068 66A3[94030300] <1> mov [u.ttyp], ax ; 0

5168 <1> ;;;

5169 0000E06E A1[51040300] <1> mov eax, [ii]

5170 <1> ; Retro UNIX 386 v1, 'sysexec' (u2.s)

5171 0000E073 E84C170000 <1> call iopen

5172 <1> ; 06/06/2016

5173 0000E078 C605[A9030300]01 <1> mov byte [u.pri], 1 ; normal priority

5174 <1> ;

5175 0000E07F EB16 <1> jmp short sysexec\_7 ; 02/05/2016

5176 <1>

5177 <1> sysexec\_6:

5178 <1> ; 19/11/2017

5179 <1> ; 18/11/2017

5180 <1> ; 14/11/2017

5181 <1> ; 13/11/2017

5182 0000E081 8925[4C040300] <1> mov [argv], esp ; \*!\* ; start address of argument list

5183 <1>

5184 <1> ; 04/01/2017

5185 <1> ; 24/10/2016

5186 <1> ;;02/05/2016

5187 <1> ; 23/04/2016 (TRDOS 386)

5188 <1> ; 18/10/2015 ('sysexec\_6')

5189 <1> ; 23/06/2015

5190 0000E087 A1[B8030300] <1> mov eax, [u.pgdir] ; physical address of page directory

5191 <1> ;cmp eax, [k\_page\_dir] ; TRDOS MainProg ?

5192 <1> ;je short sysexec\_7

5193 <1> ; 19/11/2017

5194 0000E08C 8B1D[BC030300] <1> mov ebx, [u.ppgdir] ; phy addr of the parent's page dir

5195 0000E092 E81C6CFFFF <1> call deallocate\_page\_dir

5196 <1> sysexec\_7:

5197 0000E097 E84C6BFFFF <1> call make\_page\_dir

5198 0000E09C 0F820F83FFFF <1> jc panic ; allocation error

5199 <1> ; after a deallocation would be nonsence !?

5200 <1> ; 24/07/2015

5201 <1> ; map kernel pages (1st 4MB) to PDE 0

5202 <1> ; of the user's page directory

5203 <1> ; (It is needed for interrupts!)

5204 <1> ; 18/10/2015

5205 0000E0A2 8B15[38580100] <1> mov edx, [k\_page\_dir] ; Kernel's page directory

5206 0000E0A8 8B02 <1> mov eax, [edx] ; physical address of

5207 <1> ; kernel's first page table (1st 4 MB)

5208 <1> ; (PDE 0 of kernel's page directory)

5209 0000E0AA 8B15[B8030300] <1> mov edx, [u.pgdir]

5210 0000E0B0 8902 <1> mov [edx], eax ; PDE 0 (1st 4MB)

5211 <1> ;

5212 <1> ; 20/07/2015

5213 0000E0B2 BB00004000 <1> mov ebx, CORE ; start address = 0 (virtual) + CORE

5214 <1> ; 18/10/2015

5215 0000E0B7 BE[3C040300] <1> mov esi, pcore ; physical start address

5216 <1> sysexec\_8:

5217 0000E0BC B907000000 <1> mov ecx, PDE\_A\_USER + PDE\_A\_WRITE + PDE\_A\_PRESENT

5218 0000E0C1 E8406BFFFF <1> call make\_page\_table

5219 0000E0C6 0F82E582FFFF <1> jc panic

5220 <1> ;mov ecx, PTE\_A\_USER + PTE\_A\_WRITE + PTE\_A\_PRESENT

5221 0000E0CC E8436BFFFF <1> call make\_page ; make new page, clear and set the pte

5222 0000E0D1 0F82DA82FFFF <1> jc panic

5223 <1> ;

5224 0000E0D7 8906 <1> mov [esi], eax ; 24/06/2015

5225 <1> ; ebx = virtual address (24/07/2015)

5226 0000E0D9 E8DB70FFFF <1> call add\_to\_swap\_queue

5227 <1> ; 18/10/2015

5228 0000E0DE 81FE[40040300] <1> cmp esi, ecore ; user's stack (last) page ?

5229 0000E0E4 740C <1> je short sysexec\_9 ; yes

5230 0000E0E6 BE[40040300] <1> mov esi, ecore ; physical address of the last page

5231 <1> ; 20/07/2015

5232 0000E0EB BB00F0FFFF <1> mov ebx, (ECORE - PAGE\_SIZE) + CORE

5233 <1> ; ebx = virtual end address + segment base address - 4K

5234 0000E0F0 EBCA <1> jmp short sysexec\_8

5235 <1> sysexec\_9:

5236 <1> ; 19/11/2017

5237 <1> ; 24/04/2016 (TRDOS 386 = TRDOS v2.0)

5238 <1> ; 25/06/2015, 26/08/2015, 18/10/2015

5239 <1> ; move arguments from kernel stack to [ecore]

5240 <1> ; (argument list/line will be copied from kernel stack

5241 <1> ; frame to the last (stack) page of user's core memory)

5242 <1> ; 18/10/2015

5243 0000E0F2 8B3D[40040300] <1> mov edi, [ecore]

5244 0000E0F8 81C700100000 <1> add edi, PAGE\_SIZE

5245 <1> ; 19/11/2017

5246 0000E0FE 83EF04 <1> sub edi, 4

5247 0000E101 C70700000000 <1> mov dword [edi], 0

5248 0000E107 89FB <1> mov ebx, edi

5249 <1> ;

5250 0000E109 0FB705[4A040300] <1> movzx eax, word [argc]

5251 0000E110 09C0 <1> or eax, eax

5252 0000E112 7445 <1> jz short sysexec\_13 ; 19/11/2017

5253 <1> ;jnz short sysexec\_10

5254 <1> ;mov ebx, edi

5255 <1> ;sub ebx, 4

5256 <1> ;mov [ebx], eax ; 0

5257 <1> ;jmp short sysexec\_13

5258 <1> sysexec\_10:

5259 0000E114 8B0D[8C030300] <1> mov ecx, [u.nread]

5260 <1> ; 13/11/2017

5261 <1> ;mov esi, TextBuffer ; 'load\_and\_execute\_file'

5262 <1> ;mov esi, esp ; 'sysexec'

5263 0000E11A 8B35[4C040300] <1> mov esi, [argv] ; 24/04/2016 (TRDOS 386 = TRDOS v2.0)

5264 <1> ;sub edi, ecx ; page end address - argument list length

5265 0000E120 29CB <1> sub ebx, ecx ; 19/11/2017

5266 0000E122 89C2 <1> mov edx, eax

5267 0000E124 FEC2 <1> inc dl ; argument count + 1 for argc value

5268 0000E126 C0E202 <1> shl dl, 2 ; 4 \* (argument count + 1)

5269 <1> ;mov ebx, edi

5270 0000E129 89DF <1> mov edi, ebx ; 19//11/2017

5271 0000E12B 80E3FC <1> and bl, 0FCh ; 32 bit (dword) alignment

5272 0000E12E 29D3 <1> sub ebx, edx

5273 0000E130 89FA <1> mov edx, edi

5274 0000E132 F3A4 <1> rep movsb

5275 0000E134 89D6 <1> mov esi, edx

5276 0000E136 89DF <1> mov edi, ebx

5277 0000E138 BA00F0BFFF <1> mov edx, ECORE - PAGE\_SIZE ; virtual addr. of the last page

5278 0000E13D 2B15[40040300] <1> sub edx, [ecore] ; difference (virtual - physical)

5279 0000E143 AB <1> stosd ; eax = argument count

5280 <1> sysexec\_11:

5281 0000E144 89F0 <1> mov eax, esi

5282 0000E146 01D0 <1> add eax, edx

5283 0000E148 AB <1> stosd ; eax = virtual address

5284 <1> ;dec byte [argc]

5285 0000E149 66FF0D[4A040300] <1> dec word [argc] ; 14/11/2017

5286 0000E150 7407 <1> jz short sysexec\_13

5287 <1> sysexec\_12:

5288 0000E152 AC <1> lodsb

5289 0000E153 20C0 <1> and al, al

5290 0000E155 75FB <1> jnz short sysexec\_12

5291 0000E157 EBEB <1> jmp short sysexec\_11

5292 <1> sysexec\_13:

5293 <1> ; 24/10/2016

5294 <1> ; 24/04/2016 - TRDOS 386 (TRDOS v2.0)

5295 <1> ; 23/06/2015 - 19/10/2015 (Retro UNIX 386 v1, 'sysexec\_13')

5296 <1> ;

5297 <1> ; moving arguments to [ecore] is OK here..

5298 <1> ;

5299 <1> ; ebx = beginning addres of argument list pointers

5300 <1> ; in user's stack

5301 0000E159 2B1D[40040300] <1> sub ebx, [ecore]

5302 0000E15F 81C300F0BFFF <1> add ebx, (ECORE - PAGE\_SIZE)

5303 <1> ; end of core - 4096 (last page)

5304 <1> ; (virtual address)

5305 0000E165 891D[4C040300] <1> mov [argv], ebx

5306 0000E16B 891D[90030300] <1> mov [u.break], ebx ; available user memory

5307 <1> ;

5308 0000E171 29C0 <1> sub eax, eax

5309 0000E173 C705[88030300]2000- <1> mov dword [u.count], 32 ; Executable file header size

5309 0000E17B 0000 <1>

5310 0000E17D C705[74030300]- <1> mov dword [u.fofp], u.off

5310 0000E183 [80030300] <1>

5311 0000E187 A3[80030300] <1> mov [u.off], eax ; 0

5312 0000E18C A3[84030300] <1> mov [u.base], eax ; 0, start of user's core (virtual)

5313 <1> ; 24/10/2016

5314 0000E191 A0[FE580100] <1> mov al, [Current\_Drv]

5315 0000E196 A2[46030300] <1> mov [cdev], al

5316 <1> ;

5317 0000E19B A1[51040300] <1> mov eax, [ii] ; Fist Cluster of the Program (PRG) file

5318 <1> ; EAX = First cluster of the executable file

5319 0000E1A0 E80A010000 <1> call readi

5320 <1>

5321 0000E1A5 8B0D[90030300] <1> mov ecx, [u.break] ; top of user's stack (physical addr.)

5322 0000E1AB 890D[88030300] <1> mov [u.count], ecx ; save for overrun check

5323 <1> ;

5324 0000E1B1 8B0D[8C030300] <1> mov ecx, [u.nread]

5325 0000E1B7 890D[90030300] <1> mov [u.break], ecx ; virtual address (offset from start)

5326 0000E1BD 80F920 <1> cmp cl, 32

5327 0000E1C0 7540 <1> jne short sysexec\_15

5328 <1> ;:

5329 <1> ; Retro UNIX 386 v1 (32 bit) executable file header format

5330 0000E1C2 8B35[3C040300] <1> mov esi, [pcore] ; start address of user's core memory

5331 <1> ; (phys. start addr. of the exec. file)

5332 0000E1C8 AD <1> lodsd

5333 0000E1C9 663DEB1E <1> cmp ax, 1EEBh ; EBH, 1Eh -> jump to +32

5334 0000E1CD 7533 <1> jne short sysexec\_15

5335 0000E1CF AD <1> lodsd

5336 0000E1D0 89C1 <1> mov ecx, eax ; text (code) section size

5337 0000E1D2 AD <1> lodsd

5338 0000E1D3 01C1 <1> add ecx, eax ; + data section size (initialized data)

5339 0000E1D5 89CB <1> mov ebx, ecx

5340 0000E1D7 AD <1> lodsd

5341 0000E1D8 01C3 <1> add ebx, eax ; + bss section size (for overrun checking)

5342 0000E1DA 3B1D[88030300] <1> cmp ebx, [u.count]

5343 0000E1E0 7711 <1> ja short sysexec\_14 ; program overruns stack !

5344 <1> ;

5345 <1> ; add bss section size to [u.break]

5346 0000E1E2 0105[90030300] <1> add [u.break], eax

5347 <1> ;

5348 0000E1E8 83E920 <1> sub ecx, 32 ; header size (already loaded)

5349 <1> ;cmp ecx, [u.count]

5350 <1> ;jnb short sysexec\_16

5351 0000E1EB 890D[88030300] <1> mov [u.count], ecx ; required read count

5352 0000E1F1 EB29 <1> jmp short sysexec\_16

5353 <1> sysexec\_14:

5354 <1> ; insufficient (out of) memory

5355 0000E1F3 C705[C8030300]0400- <1> mov dword [u.error], ERR\_MINOR\_IM ; 1

5355 0000E1FB 0000 <1>

5356 0000E1FD E9BCE4FFFF <1> jmp error

5357 <1> sysexec\_15:

5358 0000E202 8B15[55040300] <1> mov edx, [i.size] ; file size

5359 0000E208 29CA <1> sub edx, ecx ; file size - loaded bytes

5360 0000E20A 7626 <1> jna short sysexec\_17 ; no need to next read

5361 0000E20C 01D1 <1> add ecx, edx ; [i.size]

5362 0000E20E 3B0D[88030300] <1> cmp ecx, [u.count] ; overrun check (!)

5363 0000E214 77DD <1> ja short sysexec\_14

5364 0000E216 8915[88030300] <1> mov [u.count], edx

5365 <1> sysexec\_16:

5366 0000E21C A1[51040300] <1> mov eax, [ii] ; first cluster

5367 0000E221 E889000000 <1> call readi

5368 0000E226 8B0D[8C030300] <1> mov ecx, [u.nread]

5369 0000E22C 010D[90030300] <1> add [u.break], ecx

5370 <1> sysexec\_17:

5371 0000E232 A1[51040300] <1> mov eax, [ii] ; first cluster

5372 0000E237 E889150000 <1> call iclose

5373 0000E23C 31C0 <1> xor eax, eax

5374 0000E23E FEC0 <1> inc al

5375 0000E240 66A3[AA030300] <1> mov [u.intr], ax ; 1 (interrupt/time-out is enabled)

5376 0000E246 66A3[AC030300] <1> mov [u.quit], ax ; 1 ('crtl+brk' signal is enabled)

5377 0000E24C 833D[BC030300]00 <1> cmp dword [u.ppgdir], 0 ; is the caller MainProg (kernel) ?

5378 0000E253 770C <1> ja short sysexec\_18 ; no, the caller is user process

5379 <1> ; If the caller is kernel (MainProg), 'sysexec' will come here

5380 0000E255 8B15[38580100] <1> mov edx, [k\_page\_dir] ; kernel's page directory

5381 0000E25B 8915[BC030300] <1> mov [u.ppgdir], edx ; next time 'sysexec' must not come here

5382 <1> sysexec\_18:

5383 <1> ; 02/05/2016

5384 <1> ; 24/04/2016 (TRDOS 386 = TRDOS v2.0)

5385 <1> ; 18/10/2015 (Retro UNIX 386 v1)

5386 <1> ; 05/08/2015

5387 <1> ; 29/07/2015

5388 <1>

5389 <1> ; ; \*\*\*\* arguments list test start - 19/11/2017

5390 <1> ; mov ebp, [argv]

5391 <1> ; sub ebp, ECORE - 4096

5392 <1> ; add ebp, [ecore]

5393 <1> ;

5394 <1> ; mov ebx, [ebp]

5395 <1> ; mov [argc], bx

5396 <1> ; add ebp, 4

5397 <1> ; mov byte [ccolor], 1Fh

5398 <1> ;\_zx0:

5399 <1> ; cmp word [argc], 0

5400 <1> ; jna short \_zx2

5401 <1> ;\_zx1:

5402 <1> ; push ebp

5403 <1> ; mov esi, [ebp]

5404 <1> ;

5405 <1> ; sub esi, ECORE - 4096

5406 <1> ; add esi, [ecore]

5407 <1> ;

5408 <1> ; call print\_cmsg

5409 <1> ;

5410 <1> ; dec word [argc]

5411 <1> ; jz short \_zx2

5412 <1> ;

5413 <1> ; mov al, '.'

5414 <1> ; mov bl, 07h

5415 <1> ; mov bh, [u.ttyn]

5416 <1> ; call \_write\_tty

5417 <1> ;

5418 <1> ; pop ebp

5419 <1> ; add ebp, 4

5420 <1> ; jmp short \_zx1

5421 <1> ;\_zx2:

5422 <1> ; pop ebp

5423 <1> ; mov byte [ccolor], 07h

5424 <1> ; mov eax, 1

5425 <1> ; ; \*\*\*\* arguments list test stop

5426 <1> ; Test result is OK! (there is not a wrong thing) - 19/11/2017

5427 <1>

5428 0000E261 8B2D[4C040300] <1> mov ebp, [argv] ; user's stack pointer must point to argument

5429 <1> ; list pointers (argument count)

5430 0000E267 FA <1> cli

5431 0000E268 8B25[D4570100] <1> mov esp, [tss.esp0] ; ring 0 (kernel) stack pointer

5432 <1> ;mov esp, [u.sp] ; Restore Kernel stack

5433 <1> ; for this process

5434 <1> ;add esp, 20 ; --> EIP, CS, EFLAGS, ESP, SS

5435 <1> ;xor eax, eax ; 0

5436 0000E26E FEC8 <1> dec al ; eax = 0

5437 <1> ;mov edx, UDATA

5438 <1> ; 18/11/2017

5439 0000E270 6A23 <1> push UDATA ; user's stack segment

5440 <1> ;push edx

5441 0000E272 55 <1> push ebp ; user's stack pointer

5442 <1> ; (points to number of arguments)

5443 <1>

5444 <1> ; 04/01/2017

5445 <1> ; MainProg comes here while [sysflg]= 0FFh

5446 <1> ; (but sysexec comes here while [sysflg]= 0)

5447 0000E273 C605[5B030300]00 <1> mov byte [sysflg], 0 ; 04/01/2017

5448 <1> ; (timer\_int sysflg control)

5449 0000E27A FB <1> sti

5450 0000E27B 9C <1> pushfd ; EFLAGS

5451 <1> ; Set IF for enabling interrupts in user mode

5452 <1> ;or dword [esp], 200h

5453 <1> ;

5454 <1> ;mov bx, UCODE

5455 <1> ;push bx ; user's code segment

5456 0000E27C 6A1B <1> push UCODE

5457 <1> ;push 0

5458 0000E27E 50 <1> push eax ; EIP (=0) - start address -

5459 0000E27F 8925[5C030300] <1> mov [u.sp], esp ; 29/07/2015

5460 <1> ; 05/08/2015

5461 <1> ; Remedy of a General Protection Fault during 'iretd' is here !

5462 <1> ; ('push dx' would cause to general protection fault,

5463 <1> ; after 'pop ds' etc.)

5464 <1> ;

5465 <1> ;; push dx ; ds (UDATA)

5466 <1> ;; push dx ; es (UDATA)

5467 <1> ;; push dx ; fs (UDATA)

5468 <1> ;; push dx ; gs (UDATA)

5469 <1> ;

5470 <1> ; This is a trick to prevent general protection fault

5471 <1> ; during 'iretd' intruction at the end of 'sysrele' (in u1.s):

5472 0000E285 66BA2300 <1> mov dx, UDATA ; 19/11/2017

5473 0000E289 8EC2 <1> mov es, dx ; UDATA

5474 0000E28B 06 <1> push es ; ds (UDATA)

5475 0000E28C 06 <1> push es ; es (UDATA)

5476 0000E28D 06 <1> push es ; fs (UDATA)

5477 0000E28E 06 <1> push es ; gs (UDATA)

5478 0000E28F 66BA1000 <1> mov dx, KDATA

5479 0000E293 8EC2 <1> mov es, dx

5480 <1> ;

5481 <1> ;; pushad simulation

5482 0000E295 89E5 <1> mov ebp, esp ; esp before pushad

5483 0000E297 50 <1> push eax ; eax (0)

5484 0000E298 50 <1> push eax ; ecx (0)

5485 0000E299 50 <1> push eax ; edx (0)

5486 0000E29A 50 <1> push eax ; ebx (0)

5487 0000E29B 55 <1> push ebp ; esp before pushad

5488 0000E29C 50 <1> push eax ; ebp (0)

5489 0000E29D 50 <1> push eax ; esi (0)

5490 0000E29E 50 <1> push eax ; edi (0)

5491 <1> ;

5492 0000E29F A3[64030300] <1> mov [u.r0], eax ; eax = 0

5493 0000E2A4 8925[60030300] <1> mov [u.usp], esp

5494 <1>

5495 <1> ; 14/11/2017

5496 0000E2AA E931E4FFFF <1> jmp sysret0

5497 <1>

5498 <1> ; ; 02/05/2016

5499 <1> ; ;inc byte [sysflg] ; 0FFh -> 0

5500 <1> ; ;mov byte [sysflg], 0 ; 04/01/2017

5501 <1> ; movzx ebx, byte [u.uno]

5502 <1> ; shl bl, 1 ; 13/11/2017

5503 <1> ; cmp word [ebx+p.ppid-2], 1 ; MainProg

5504 <1> ; ja sysret0 ; 03/05/2016

5505 <1> ; push sysret ; \*

5506 <1> ; mov [u.usp], esp

5507 <1> ; call wswap ; save child process 'u' structure and

5508 <1> ; ; registers

5509 <1> ; add dword [u.usp], 4 ; 03/05/2016

5510 <1> ;sysexec\_19: ; 02/05/2016

5511 <1> ; retn ; \* 'sysret' ; byte [sysflg] -> 0FFh

5512 <1>

5513 <1> readi:

5514 <1> ; 01/05/2016

5515 <1> ; 25/04/2016 - TRDOS 386 (TRDOS v2.0)

5516 <1> ; 20/05/2015 - Retro UNIX 386 v1

5517 <1> ; 11/03/2013 - 31/07/2013 (Retro UNIX 8086 v1)

5518 <1> ;

5519 <1> ; Reads from a file whose the first cluster number in EAX

5520 <1> ;

5521 <1> ; INPUTS ->

5522 <1> ; EAX - First cluster number of the file

5523 <1> ; u.count - byte count user desires

5524 <1> ; u.base - points to user buffer

5525 <1> ; u.fofp - points to dword with current file offset

5526 <1> ; i.size - file size

5527 <1> ; cdev - logical dos drive number of the file

5528 <1> ; OUTPUTS ->

5529 <1> ; u.count - cleared

5530 <1> ; u.nread - accumulates total bytes passed back

5531 <1> ;

5532 <1> ; ((EAX)) input/output

5533 <1> ; (Retro UNIX Prototype : 14/12/2012 - 01/03/2013, UNIXCOPY.ASM)

5534 <1> ; ((Modified registers: edx, ebx, ecx, esi, edi))

5535 <1>

5536 0000E2AF 31D2 <1> xor edx, edx ; 0

5537 0000E2B1 8915[8C030300] <1> mov [u.nread], edx ; 0

5538 0000E2B7 668915[C4030300] <1> mov [u.pcount], dx ; 19/05/2015

5539 0000E2BE 3915[88030300] <1> cmp [u.count], edx ; 0

5540 0000E2C4 7701 <1> ja short readi\_1

5541 0000E2C6 C3 <1> retn

5542 <1> readi\_1:

5543 <1> dskr:

5544 <1> ; 01/05/2016

5545 <1> ; 25/04/2016 - TRDOS 386 (TRDOS v2.0)

5546 <1> ; 24/05/2015 - 12/10/2015 (Retro UNIX 386 v1)

5547 <1> ; 26/04/2013 - 03/08/2013 (Retro UNIX 8086 v1)

5548 <1> dskr\_0:

5549 0000E2C7 8B15[55040300] <1> mov edx, [i.size]

5550 0000E2CD 8B1D[74030300] <1> mov ebx, [u.fofp]

5551 0000E2D3 2B13 <1> sub edx, [ebx]

5552 0000E2D5 7647 <1> jna short dskr\_4

5553 <1> ;

5554 0000E2D7 50 <1> push eax ; 01/05/2016

5555 0000E2D8 3B15[88030300] <1> cmp edx, [u.count]

5556 0000E2DE 7306 <1> jnb short dskr\_1

5557 0000E2E0 8915[88030300] <1> mov [u.count], edx

5558 <1> dskr\_1:

5559 <1> ; EAX = First Cluster

5560 <1> ; [Current\_Drv] = Physical drive number

5561 0000E2E6 E83B000000 <1> call mget\_r

5562 <1> ; NOTE: in 'mget\_r', relevant sector will be read in buffer

5563 <1> ; if it is not already in buffer !

5564 0000E2EB BB[8C050300] <1> mov ebx, readi\_buffer

5565 0000E2F0 803D[C6030300]00 <1> cmp byte [u.kcall], 0 ; the caller is 'namei' sign (=1)

5566 0000E2F7 770F <1> ja short dskr\_3 ; zf=0 -> the caller is 'namei'

5567 0000E2F9 66833D[C4030300]00 <1> cmp word [u.pcount], 0

5568 0000E301 7705 <1> ja short dskr\_3

5569 <1> dskr\_2:

5570 <1> ; [u.base] = virtual address to transfer (as destination address)

5571 0000E303 E894010000 <1> call trans\_addr\_w ; translate virtual address to physical (w)

5572 <1> dskr\_3:

5573 <1> ; EBX (r5) = system (I/O) buffer address -physical-

5574 0000E308 E8F7010000 <1> call sioreg

5575 0000E30D 87F7 <1> xchg esi, edi

5576 <1> ; EDI = file (user data) offset

5577 <1> ; ESI = sector (I/O) buffer offset

5578 <1> ; ECX = byte count

5579 0000E30F F3A4 <1> rep movsb

5580 <1> ; eax = remain bytes in buffer

5581 <1> ; (check if remain bytes in the buffer > [u.pcount])

5582 0000E311 09C0 <1> or eax, eax

5583 0000E313 75EE <1> jnz short dskr\_2 ; (page end before system buffer end!)

5584 0000E315 58 <1> pop eax ; (first cluster number)

5585 0000E316 390D[88030300] <1> cmp [u.count], ecx ; 0

5586 0000E31C 77A9 <1> ja short dskr\_0

5587 <1> dskr\_4:

5588 0000E31E C605[C6030300]00 <1> mov byte [u.kcall], 0

5589 0000E325 C3 <1> retn

5590 <1>

5591 <1> mget\_r:

5592 <1> ; 24/10/2016

5593 <1> ; 22/10/2016

5594 <1> ; 12/10/2016

5595 <1> ; 29/04/2016

5596 <1> ; 25/04/2016 - TRDOS 386 (TRDOS v2.0)

5597 <1> ; 03/06/2015 (Retro UNIX 386 v1, 'mget', u.5s)

5598 <1> ; 22/03/2013 - 31/07/2013 (Retro UNIX 8086 v1)

5599 <1> ;

5600 <1> ; Get existing or (allocate) a new disk block for file

5601 <1> ;

5602 <1> ; INPUTS ->

5603 <1> ; [u.fofp] = file offset pointer

5604 <1> ; EAX = First Cluster

5605 <1> ; [cdev] = Logical dos drive number

5606 <1> ; ([u.off] = file offset)

5607 <1> ; OUTPUTS ->

5608 <1> ; EAX = logical sector number

5609 <1> ; ESI = Logical Dos Drive Description Table address

5610 <1> ;

5611 <1> ; Modified registers: EDX, EBX, ECX, ESI, EDI

5612 <1>

5613 0000E326 8B35[74030300] <1> mov esi, [u.fofp]

5614 0000E32C 8B1E <1> mov ebx, [esi] ; (u.off)

5615 <1>

5616 0000E32E 29C9 <1> sub ecx, ecx

5617 0000E330 8A2D[46030300] <1> mov ch, [cdev]

5618 <1>

5619 0000E336 BE00010900 <1> mov esi, Logical\_DOSDisks

5620 0000E33B 01CE <1> add esi, ecx

5621 <1>

5622 0000E33D 380D[70650100] <1> cmp [readi.valid], cl ; 0

5623 0000E343 7649 <1> jna short mget\_r\_0

5624 <1>

5625 0000E345 3A2D[71650100] <1> cmp ch, [readi.drv]

5626 0000E34B 7541 <1> jne short mget\_r\_0

5627 <1>

5628 0000E34D 3B05[84650100] <1> cmp eax, [readi.fclust]

5629 0000E353 7565 <1> jne short mget\_r\_3

5630 <1>

5631 0000E355 89D8 <1> mov eax, ebx ; file offset

5632 0000E357 668B0D[78650100] <1> mov cx, [readi.bpc]

5633 0000E35E 41 <1> inc ecx ; <= 65536

5634 0000E35F 29D2 <1> sub edx, edx

5635 0000E361 F7F1 <1> div ecx

5636 <1>

5637 0000E363 8B3D[80650100] <1> mov edi, [readi.c\_index] ; cluster index

5638 <1>

5639 0000E369 39F8 <1> cmp eax, edi

5640 0000E36B 757A <1> jne short mget\_r\_4 ; (\*)

5641 <1>

5642 <1> ; edx = byte offset in cluster (<= 65535)

5643 0000E36D 668915[7A650100] <1> mov [readi.offset], dx

5644 0000E374 66C1EA09 <1> shr dx, 9 ; / 512

5645 0000E378 8815[73650100] <1> mov [readi.s\_index], dl ; sector index in cluster (0 to spc -1)

5646 <1>

5647 0000E37E A1[7C650100] <1> mov eax, [readi.cluster] ; > 0 if [readi.valid] = 1

5648 0000E383 8B15[88650100] <1> mov edx, [readi.fs\_index]

5649 0000E389 E99A000000 <1> jmp mget\_r\_7

5650 <1>

5651 <1> mget\_r\_0:

5652 0000E38E 882D[71650100] <1> mov [readi.drv], ch ; physical drive number

5653 0000E394 807E0300 <1> cmp byte [esi+LD\_FATType], 0

5654 0000E398 7707 <1> ja short mget\_r\_1

5655 0000E39A 8A4E12 <1> mov cl, [esi+LD\_FS\_BytesPerSec+1]

5656 0000E39D D0E9 <1> shr cl, 1 ; ; 1 for 512 bytes, 4 for 2048 bytes

5657 0000E39F EB03 <1> jmp short mget\_r\_2

5658 <1> mget\_r\_1:

5659 0000E3A1 8A4E13 <1> mov cl, [esi+LD\_BPB+BPB\_SecPerClust]

5660 <1> mget\_r\_2:

5661 0000E3A4 880D[72650100] <1> mov [readi.spc], cl ; sectors per cluster

5662 <1> ; NOTE: readi bytes per sector value is always 512 !

5663 0000E3AA 66C1E109 <1> shl cx, 9 ; \* 512

5664 0000E3AE 6649 <1> dec cx ; bytes per cluster - 1

5665 0000E3B0 66890D[78650100] <1> mov [readi.bpc], cx

5666 0000E3B7 6629C9 <1> sub cx, cx

5667 <1> mget\_r\_3:

5668 0000E3BA A3[84650100] <1> mov [readi.fclust], eax ; first cluster (or FDT address)

5669 0000E3BF 880D[70650100] <1> mov [readi.valid], cl ; 0

5670 <1> ;mov [readi.s\_index], cl ; 0

5671 <1> ;mov [readi.offset], cx ; 0

5672 0000E3C5 890D[80650100] <1> mov [readi.c\_index], ecx ; 0

5673 0000E3CB 890D[7C650100] <1> mov [readi.cluster], ecx ; 0

5674 0000E3D1 890D[74650100] <1> mov [readi.sector], ecx ; 0

5675 <1>

5676 0000E3D7 89D8 <1> mov eax, ebx ; file offset

5677 0000E3D9 668B0D[78650100] <1> mov cx, [readi.bpc]

5678 0000E3E0 41 <1> inc ecx ; <= 65536

5679 0000E3E1 29D2 <1> sub edx, edx

5680 0000E3E3 F7F1 <1> div ecx

5681 <1> ;mov edi, [readi.c\_index] ; previous cluster index

5682 0000E3E5 29FF <1> sub edi, edi

5683 <1> mget\_r\_4:

5684 0000E3E7 A3[80650100] <1> mov [readi.c\_index], eax ; cluster index

5685 <1> ; edx = byte offset in cluster (<= 65535)

5686 0000E3EC 668915[7A650100] <1> mov [readi.offset], dx

5687 0000E3F3 66C1EA09 <1> shr dx, 9 ; / 512

5688 0000E3F7 8815[73650100] <1> mov [readi.s\_index], dl ; sector index in cluster (0 to spc -1)

5689 <1>

5690 0000E3FD 89C1 <1> mov ecx, eax ; current cluster index

5691 0000E3FF A1[84650100] <1> mov eax, [readi.fclust]

5692 0000E404 09C9 <1> or ecx, ecx ; cluster index

5693 0000E406 741B <1> jz short mget\_r\_6

5694 <1>

5695 0000E408 39CF <1> cmp edi, ecx

5696 0000E40A 7710 <1> ja short mget\_r\_5 ; old cluster index is higher

5697 0000E40C 8B15[7C650100] <1> mov edx, [readi.cluster]

5698 0000E412 21D2 <1> and edx, edx

5699 0000E414 7406 <1> jz short mget\_r\_5

5700 <1> ; valid 'readi' parameters (\*)

5701 0000E416 89D0 <1> mov eax, edx

5702 0000E418 29F9 <1> sub ecx, edi

5703 0000E41A 740C <1> jz short mget\_r\_7

5704 <1> mget\_r\_5:

5705 <1> ; EAX = Beginning cluster

5706 <1> ; EDX = Sector index in disk/file section

5707 <1> ; (Only for SINGLIX file system!)

5708 <1> ; ECX = Cluster sequence number after the beginning cluster

5709 <1> ; ESI = Logical DOS Drive Description Table address

5710 0000E41C E836E1FFFF <1> call get\_cluster\_by\_index

5711 0000E421 724E <1> jc short mget\_r\_err

5712 <1> ; EAX = Cluster number

5713 <1> mget\_r\_6:

5714 0000E423 A3[7C650100] <1> mov [readi.cluster], eax ; FDT number for Singlix File System

5715 <1> mget\_r\_7:

5716 0000E428 807E0300 <1> cmp byte [esi+LD\_FATType], 0

5717 0000E42C 765F <1> jna short mget\_r\_12

5718 <1>

5719 0000E42E 83E802 <1> sub eax, 2

5720 0000E431 0FB615[72650100] <1> movzx edx, byte [readi.spc]

5721 0000E438 F7E2 <1> mul edx

5722 <1>

5723 0000E43A 034668 <1> add eax, [esi+LD\_DATABegin]

5724 0000E43D 8A15[73650100] <1> mov dl, [readi.s\_index]

5725 0000E443 01D0 <1> add eax, edx

5726 <1> mget\_r\_8:

5727 <1> ; eax = logical sector number

5728 0000E445 803D[70650100]00 <1> cmp byte [readi.valid], 0

5729 0000E44C 7608 <1> jna short mget\_r\_9

5730 0000E44E 3B05[74650100] <1> cmp eax, [readi.sector]

5731 0000E454 7436 <1> je short mget\_r\_11 ; sector is already in 'readi' buffer

5732 <1> mget\_r\_9:

5733 0000E456 A3[74650100] <1> mov [readi.sector], eax

5734 0000E45B BB[8C050300] <1> mov ebx, readi\_buffer ; buffer address

5735 0000E460 B901000000 <1> mov ecx, 1

5736 <1> ; 29/04/2016

5737 <1> ;xor dl, dl

5738 <1>

5739 <1> ; EAX = Logical sector number

5740 <1> ; ECX = Sector count

5741 <1> ; EBX = Buffer address

5742 <1> ; (EDX = 0)

5743 <1> ; ESI = Logical DOS drive description table address

5744 <1>

5745 0000E465 E86E130000 <1> call disk\_read

5746 0000E46A 7314 <1> jnc short mget\_r\_10

5747 <1>

5748 <1> ; 22/10/2016 (15h -> 17)

5749 0000E46C B811000000 <1> mov eax, 17 ; Drive not ready or read error !

5750 <1> mget\_r\_err:

5751 0000E471 A3[C8030300] <1> mov [u.error], eax

5752 <1> ; 12/10/2016

5753 0000E476 A3[64030300] <1> mov [u.r0], eax

5754 0000E47B E93EE2FFFF <1> jmp error

5755 <1> mget\_r\_10:

5756 0000E480 C605[70650100]01 <1> mov byte [readi.valid], 1 ; 24/10/2016

5757 0000E487 A1[74650100] <1> mov eax, [readi.sector]

5758 <1> mget\_r\_11:

5759 0000E48C C3 <1> retn

5760 <1> mget\_r\_12:

5761 <1> ; EAX = FDT number

5762 <1> ; EDX = Sector index from FDT sector (0,1,2,3,4...)

5763 0000E48D 40 <1> inc eax ; the first data sector in FS disk section

5764 0000E48E 8915[88650100] <1> mov [readi.fs\_index], edx

5765 0000E494 01D0 <1> add eax, edx

5766 0000E496 EBAD <1> jmp short mget\_r\_8

5767 <1>

5768 <1> trans\_addr\_r:

5769 <1> ; 12/10/2016

5770 <1> ; 02/05/2016 - TRDOS 386 (TRDOS v2.0)

5771 <1> ; Translate virtual address to physical address

5772 <1> ; for reading from user's memory space

5773 <1> ; 04/06/2015 - 18/10/2015 (Retro UNIX 386 v1)

5774 <1>

5775 0000E498 31D2 <1> xor edx, edx ; 0 (read access sign)

5776 0000E49A EB04 <1> jmp short trans\_addr\_rw

5777 <1>

5778 <1> trans\_addr\_w:

5779 <1> ; 12/10/2016

5780 <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)

5781 <1> ; Translate virtual address to physical address

5782 <1> ; for writing to user's memory space

5783 <1> ; 04/06/2015 - 18/10/2015 (Retro UNIX 386 v1)

5784 <1>

5785 0000E49C 29D2 <1> sub edx, edx

5786 0000E49E FEC2 <1> inc dl ; 1 (write access sign)

5787 <1> trans\_addr\_rw:

5788 0000E4A0 50 <1> push eax

5789 0000E4A1 53 <1> push ebx

5790 0000E4A2 52 <1> push edx ; r/w sign (in DL)

5791 <1> ;

5792 0000E4A3 8B1D[84030300] <1> mov ebx, [u.base]

5793 0000E4A9 E8E16DFFFF <1> call get\_physical\_addr ; get physical address

5794 0000E4AE 730F <1> jnc short passc\_0

5795 0000E4B0 A3[C8030300] <1> mov [u.error], eax

5796 0000E4B5 A3[64030300] <1> mov [u.r0], eax ; 12/10/2016

5797 <1> ;pop edx

5798 <1> ;pop ebx

5799 <1> ;pop eax

5800 0000E4BA E9FFE1FFFF <1> jmp error

5801 <1> passc\_0:

5802 0000E4BF F6C202 <1> test dl, PTE\_A\_WRITE ; writable page

5803 0000E4C2 5A <1> pop edx

5804 0000E4C3 751C <1> jnz short passc\_1

5805 <1>

5806 0000E4C5 20D2 <1> and dl, dl

5807 0000E4C7 7418 <1> jz short passc\_1

5808 <1> ; read only (duplicated) page -must be copied to a new page-

5809 <1> ; EBX = linear address

5810 0000E4C9 51 <1> push ecx

5811 0000E4CA E8596AFFFF <1> call copy\_page

5812 0000E4CF 59 <1> pop ecx

5813 0000E4D0 721E <1> jc short passc\_2

5814 0000E4D2 50 <1> push eax ; physical address of the new/allocated page

5815 0000E4D3 E8E16CFFFF <1> call add\_to\_swap\_queue

5816 0000E4D8 58 <1> pop eax

5817 0000E4D9 81E3FF0F0000 <1> and ebx, PAGE\_OFF ; 0FFFh

5818 <1> ;mov ecx, PAGE\_SIZE

5819 <1> ;sub ecx, ebx

5820 0000E4DF 01D8 <1> add eax, ebx

5821 <1> passc\_1:

5822 0000E4E1 A3[C0030300] <1> mov [u.pbase], eax ; physical address

5823 0000E4E6 66890D[C4030300] <1> mov [u.pcount], cx ; remain byte count in page (1-4096)

5824 0000E4ED 5B <1> pop ebx

5825 0000E4EE 58 <1> pop eax

5826 0000E4EF C3 <1> retn

5827 <1> passc\_2:

5828 0000E4F0 B804000000 <1> mov eax, ERR\_MINOR\_IM ; "Insufficient memory !" error

5829 0000E4F5 A3[64030300] <1> mov [u.r0], eax ; 12/10/2016

5830 0000E4FA A3[C8030300] <1> mov dword [u.error], eax

5831 <1> ;pop ebx

5832 <1> ;pop eax

5833 0000E4FF E9BAE1FFFF <1> jmp error

5834 <1>

5835 <1> sioreg:

5836 <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)

5837 <1> ; 19/05/2015 - 25/07/2015 (Retro UNIX 386 v1)

5838 <1> ; 12/03/2013 - 22/07/2013 (Retro UNIX 8086 v1)

5839 <1> ; INPUTS ->

5840 <1> ; EBX = system buffer (data) address (r5)

5841 <1> ; [u.fofp] = pointer to file offset pointer

5842 <1> ; [u.base] = virtual address of the user buffer

5843 <1> ; [u.pbase] = physical address of the user buffer

5844 <1> ; [u.count] = byte count

5845 <1> ; [u.pcount] = byte count within page frame

5846 <1> ; OUTPUTS ->

5847 <1> ; ESI = user data offset (r1)

5848 <1> ; EDI = system (I/O) buffer offset (r2)

5849 <1> ; ECX = byte count (r3)

5850 <1> ; EAX = remain bytes after byte count within page frame

5851 <1> ; (If EAX > 0, transfer will continue from the next page)

5852 <1> ;

5853 <1> ; ((Modified registers: EDX))

5854 <1>

5855 0000E504 8B35[74030300] <1> mov esi, [u.fofp]

5856 0000E50A 8B3E <1> mov edi, [esi]

5857 0000E50C 89F9 <1> mov ecx, edi

5858 0000E50E 81C900FEFFFF <1> or ecx, 0FFFFFE00h

5859 0000E514 81E7FF010000 <1> and edi, 1FFh

5860 0000E51A 01DF <1> add edi, ebx ; EBX = system buffer (data) address

5861 0000E51C F7D9 <1> neg ecx

5862 0000E51E 3B0D[88030300] <1> cmp ecx, [u.count]

5863 0000E524 7606 <1> jna short sioreg\_0

5864 0000E526 8B0D[88030300] <1> mov ecx, [u.count]

5865 <1> sioreg\_0:

5866 0000E52C 803D[C6030300]00 <1> cmp byte [u.kcall], 0

5867 0000E533 7613 <1> jna short sioreg\_1

5868 <1> ; the caller is 'mkdir' or 'namei'

5869 0000E535 A1[84030300] <1> mov eax, [u.base]

5870 0000E53A A3[C0030300] <1> mov [u.pbase], eax ; physical address = virtual address

5871 0000E53F 66890D[C4030300] <1> mov word [u.pcount], cx ; remain bytes in buffer (1 sector)

5872 0000E546 EB0B <1> jmp short sioreg\_2

5873 <1> sioreg\_1:

5874 0000E548 0FB715[C4030300] <1> movzx edx, word [u.pcount]

5875 0000E54F 39D1 <1> cmp ecx, edx

5876 0000E551 772A <1> ja short sioreg\_4 ; transfer count > [u.pcount]

5877 <1> sioreg\_2: ; 2:

5878 0000E553 31C0 <1> xor eax, eax

5879 <1> sioreg\_3:

5880 0000E555 010D[8C030300] <1> add [u.nread], ecx

5881 0000E55B 290D[88030300] <1> sub [u.count], ecx

5882 0000E561 010D[84030300] <1> add [u.base], ecx

5883 0000E567 010E <1> add [esi], ecx

5884 0000E569 8B35[C0030300] <1> mov esi, [u.pbase]

5885 0000E56F 66290D[C4030300] <1> sub [u.pcount], cx

5886 0000E576 010D[C0030300] <1> add [u.pbase], ecx

5887 0000E57C C3 <1> retn

5888 <1> sioreg\_4:

5889 <1> ; transfer count > [u.pcount]

5890 <1> ; (ecx > edx)

5891 0000E57D 89C8 <1> mov eax, ecx

5892 0000E57F 29D0 <1> sub eax, edx ; remain bytes for 1 sector (block) transfer

5893 0000E581 89D1 <1> mov ecx, edx ; current transfer count = [u.pcount]

5894 0000E583 EBD0 <1> jmp short sioreg\_3

5895 <1>

5896 <1> tswitch: ; Retro UNIX 386 v1

5897 <1> tswap:

5898 <1> ; 16/01/2017

5899 <1> ; 21/05/2016 - TRDOS 386 (TRDOS v2.0)

5900 <1> ; 10/05/2015 - 01/09/2015 (Retro UNIX 386 v1)

5901 <1> ; 14/04/2013 - 14/02/2014 (Retro UNIX 8086 v1)

5902 <1> ; time out swap, called when a user times out.

5903 <1> ; the user is put on the low priority queue.

5904 <1> ; This is done by making a link from the last user

5905 <1> ; on the low priority queue to him via a call to 'putlu'.

5906 <1> ; then he is swapped out.

5907 <1>

5908 <1> ; TRDOS 386 (TRDOS v2.0) modification -> \*\* 21/05/2016 \*\*

5909 <1> ; \* when a high priority (event) process will be stopped

5910 <1> ; (swapped out, swithched out/off), 'tswap/tswitch' will

5911 <1> ; not add it to a run queue.

5912 <1> ; /// What for: Process may be already in a run queue,

5913 <1> ; it is unspeficied state because process might be started

5914 <1> ; by a timer event which does not regard previous priority

5915 <1> ; level and run queue of the process (for fast executing!).

5916 <1> ; After the 'run for event', process will be sequenced

5917 <1> ; to run by it's actual run queue. ///

5918 <1> ;

5919 <1> ; Retro UNIX 386 v1 modification ->

5920 <1> ; swap (software task switch) is performed by changing

5921 <1> ; user's page directory (u.pgdir) instead of segment change

5922 <1> ; as in Retro UNIX 8086 v1.

5923 <1> ;

5924 <1> ; RETRO UNIX 8086 v1 modification ->

5925 <1> ; 'swap to disk' is replaced with 'change running segment'

5926 <1> ; according to 8086 cpu (x86 real mode) architecture.

5927 <1> ; pdp-11 was using 64KB uniform memory while IBM PC

5928 <1> ; compatibles was using 1MB segmented memory

5929 <1> ; in 8086/8088 times.

5930 <1> ;

5931 <1> ; INPUTS ->

5932 <1> ; u.uno - users process number

5933 <1> ; runq+4 - lowest priority queue

5934 <1> ; OUTPUTS ->

5935 <1> ; r0 - users process number

5936 <1> ; r2 - lowest priority queue address

5937 <1> ;

5938 <1> ; ((AX = R0, BX = R2)) output

5939 <1> ; ((Modified registers: EDX, EBX, ECX, ESI, EDI))

5940 <1> ;

5941 <1>

5942 <1> NOTE:

5943 <1> ;\* [u.pri] priority level is specified by run queue which is process

5944 <1> ; comes to run from.

5945 <1> ;\* Initial [u.pri] is 1 ('normal/regular') for programs

5946 <1> ; (which are launched by MainProg or 'sysexec'), it is changed

5947 <1> ; to 2 ('high') by timer event, if program uses 'systimer' system call.

5948 <1> ;\* Program (Process) also can change it's running priority

5949 <1> ; from 1 to 0 or up to 2 by using 'syspri' system call; but,

5950 <1> ; if program selects priority level 2 (high) for running, next time

5951 <1> ; it is reduced to 1 (normal/regular) because 'syspri' adds this

5952 <1> ; program to 'run for normal' queue while running duration is a bit

5953 <1> ; protected from swap/switch out immediate, behalf of other high

5954 <1> ; priority process in sequence. Program (with high priority) will not

5955 <1> ; be swapped/switched out (by timer event) before it's time quantum

5956 <1> ; will be elapsed, but, this will be temporary if program is not using

5957 <1> ; timer event function.

5958 <1>

5959 <1> ;For example:

5960 <1> ;If a process frequently gets a timer event, it runs at high priority

5961 <1> ;level but when it returns from running it returns to actual run queue,

5962 <1> ;not to 'run for event' queue again.

5963 <1> ;'tswap' will not change the sequence at return/stop(swap out) stage.

5964 <1> ;But if priority level not high (=2, 'run for event'), 'tswap/tswitch'

5965 <1> ;will add the stopping process to relevant run queue according to

5966 <1> ;[u.pri] priority level.

5967 <1>

5968 <1> ; 16/01/2017

5969 0000E585 BB[54030300] <1> mov ebx, runq+2 ; 'runq\_normal' ; normal/regular priority

5970 <1> ; 21/05/2016

5971 <1> ;cmp byte [u.pri], 2 ; high priority (run for event) ?

5972 <1> ;jnb short swap

5973 <1> ; 16/01/2017

5974 <1> ; (Normal and also high/event priority processes will be added to

5975 <1> ; normal priority run queue for ensuring circular running sequence!)

5976 <1> ; (Timer interrupt or 'syspri' system call may change priority and run

5977 <1> ; queue to high/event level.)

5978 0000E58A 803D[A9030300]00 <1> cmp byte [u.pri], 0

5979 0000E591 7702 <1> ja short tswap\_1 ; normal priority run queue

5980 <1> ;

5981 0000E593 43 <1> inc ebx

5982 0000E594 43 <1> inc ebx ; runq+4, 'runq\_background', low priority

5983 <1> tswap\_1:

5984 0000E595 A0[B3030300] <1> mov al, [u.uno]

5985 <1> ; movb u.uno,r1 / move users process number to r1

5986 <1> ; mov $runq+4,r2

5987 <1> ; / move lowest priority queue address to r2

5988 <1> ; ebx = run queue

5989 0000E59A E8FE000000 <1> call putlu

5990 <1> ; jsr r0,putlu / create link from last user on Q to

5991 <1> ; / u.uno's user

5992 <1>

5993 <1> switch: ; Retro UNIX 386 v1

5994 <1> swap:

5995 <1> ; 02/01/2017

5996 <1> ; 21/05/2016

5997 <1> ; 20/05/2016

5998 <1> ; 02/05/2016

5999 <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)

6000 <1> ; 10/05/2015 - 02/09/2015 (Retro UNIX 386 v1)

6001 <1> ; 14/04/2013 - 08/03/2014 (Retro UNIX 8086 v1)

6002 <1> ;

6003 <1> ; 'swap' is routine that controls the swapping of processes

6004 <1> ; in and out of core.

6005 <1> ;

6006 <1> ; TRDOS 386 (TRDOS v2.0) modification -> \*\* 20/05/2016 \*\*

6007 <1> ; \* 3 different priority level is applied

6008 <1> ; (just as original unix v1)

6009 <1> ; 1) high priority (event) run queue, 'runq\_event'

6010 <1> ; 2) normal priority (regular) run queue, 'runq\_normal'

6011 <1> ; 3) low priority (background) run queue, 'runq\_backgroud'

6012 <1> ; 'swap' code will run a process which has max. priority

6013 <1> ; (for earliest event at first)

6014 <1> ;

6015 <1> ; Retro UNIX 386 v1 modification ->

6016 <1> ; swap (software task switch) is performed by changing

6017 <1> ; user's page directory (u.pgdir) instead of segment change

6018 <1> ; as in Retro UNIX 8086 v1.

6019 <1> ;

6020 <1> ; RETRO UNIX 8086 v1 modification ->

6021 <1> ; 'swap to disk' is replaced with 'change running segment'

6022 <1> ; according to 8086 cpu (x86 real mode) architecture.

6023 <1> ; pdp-11 was using 64KB uniform memory while IBM PC

6024 <1> ; compatibles was using 1MB segmented memory

6025 <1> ; in 8086/8088 times.

6026 <1> ;

6027 <1> ; INPUTS ->

6028 <1> ; runq table - contains processes to run.

6029 <1> ; p.link - contains next process in line to be run.

6030 <1> ; u.uno - process number of process in core

6031 <1> ; s.stack - swap stack used as an internal stack for swapping.

6032 <1> ; OUTPUTS ->

6033 <1> ; (original unix v1 -> present process to its disk block)

6034 <1> ; (original unix v1 -> new process into core ->

6035 <1> ; Retro Unix 8086 v1 -> segment registers changed

6036 <1> ; for new process)

6037 <1> ; u.quant = 3 (Time quantum for a process)

6038 <1> ; ((INT 1Ch count down speed -> 18.2 times per second)

6039 <1> ; RETRO UNIX 8086 v1 will use INT 1Ch (18.2 times per second)

6040 <1> ; for now, it will swap the process if there is not

6041 <1> ; a keyboard event (keystroke) (Int 15h, function 4Fh)

6042 <1> ; or will count down from 3 to 0 even if there is a

6043 <1> ; keyboard event locking due to repetitive key strokes.

6044 <1> ; u.quant will be reset to 3 for RETRO UNIX 8086 v1.

6045 <1> ;

6046 <1> ; ((Modified registers: EAX, EDX, EBX, ECX, ESI, EDI))

6047 <1>

6048 <1> ;NOTE:

6049 <1> ;High priority queue is the first for selecting a process to run.

6050 <1> ;If there is not a process in high priority level run queue,

6051 <1> ;a process in normal priority run queue will be selected

6052 <1> ;or a proces in low priority run queue will be selected if normal

6053 <1> ;priority level run queue is empty.

6054 <1>

6055 <1> ; 21/05/2016 -(3 priority levels, 3 run queues)

6056 0000E59F BE[52030300] <1> mov esi, runq ; 'runq\_event' ; high priority, 'run for event'

6057 0000E5A4 C605[CC650100]03 <1> mov byte [priority], 3 ; high priority + 1

6058 0000E5AB 31DB <1> xor ebx, ebx ; 02/01/2017

6059 <1> swap\_0: ; 1: / search runq table for highest priority process

6060 0000E5AD 66AD <1> lodsw ; mov ax, [esi], add esi+2

6061 <1> ;xor ebx, ebx ; 02/05/2016

6062 0000E5AF 6621C0 <1> and ax, ax ; are there any processes to run in this Q entry

6063 0000E5B2 750E <1> jnz short swap\_2

6064 <1> ; 21/05/2026

6065 <1> ; runq\_normal = runq+2, runq\_background = runq+4

6066 0000E5B4 FE0D[CC650100] <1> dec byte [priority] ; 3 -> 3, 2 -> 1, 1-> 0

6067 0000E5BA 75F1 <1> jnz short swap\_0

6068 <1> ;cmp esi, runq+6 ; if zero compare address to end of table

6069 <1> ;jb short swap\_0 ; if not at end, go back

6070 <1> swap\_1:

6071 <1> ; 02/05/2016

6072 <1> ; 29/04/2016 (TRDOS 386 = TRDOS v2.0)

6073 <1> ; No user process to run...

6074 <1> ; Run the kernel process... MainProg: Internal Command Interpreter

6075 0000E5BC FEC0 <1> inc al ; mov al, 1 ; process number of MainProg

6076 0000E5BE FEC3 <1> inc bl ; mov bl, al ; 1

6077 0000E5C0 EB1E <1> jmp short swap\_4

6078 <1> swap\_2:

6079 <1> ; 21/05/2016

6080 0000E5C2 FE0D[CC650100] <1> dec byte [priority] ; priority level of present user/process

6081 <1> ; 0, 1, 2

6082 0000E5C8 4E <1> dec esi

6083 0000E5C9 4E <1> dec esi

6084 <1> ;

6085 0000E5CA 88C3 <1> mov bl, al

6086 0000E5CC 38E0 <1> cmp al, ah ; is there only 1 process in the queue to be run

6087 0000E5CE 740A <1> je short swap\_3 ; yes

6088 0000E5D0 8AA3[9F000300] <1> mov ah, [ebx+p.link-1]

6089 0000E5D6 8826 <1> mov [esi], ah ; move next process in line into run queue

6090 0000E5D8 EB06 <1> jmp short swap\_4

6091 <1> swap\_3:

6092 0000E5DA 6631D2 <1> xor dx, dx

6093 0000E5DD 668916 <1> mov [esi], dx ; zero the entry; no processes on the Q

6094 <1> swap\_4:

6095 0000E5E0 8A25[B3030300] <1> mov ah, [u.uno]

6096 0000E5E6 38C4 <1> cmp ah, al ; is this process the same as the process in core?

6097 0000E5E8 743B <1> je short swap\_8 ; yes, don't have to swap

6098 0000E5EA 08E4 <1> or ah, ah ; is the process # = 0

6099 0000E5EC 740D <1> jz short swap\_6 ; 'sysexit'

6100 <1> ;cmp ah, al ;is this process the same as the process in core?

6101 <1> ;je short swap\_8 ; yes, don't have to swap

6102 0000E5EE 8925[60030300] <1> mov [u.usp], esp ; return address for 'syswait' & 'sleep'

6103 0000E5F4 E834000000 <1> call wswap ; write out core to disk

6104 0000E5F9 EB1C <1> jmp short swap\_7

6105 <1> swap\_6:

6106 <1> ; Deallocate memory pages belong to the process

6107 <1> ; which is being terminated.

6108 <1> ; (Retro UNIX 386 v1 modification !)

6109 <1> ;

6110 0000E5FB 53 <1> push ebx

6111 0000E5FC A1[B8030300] <1> mov eax, [u.pgdir] ; page directory of the process

6112 0000E601 8B1D[BC030300] <1> mov ebx, [u.ppgdir] ; page directory of the parent process

6113 0000E607 E8A766FFFF <1> call deallocate\_page\_dir

6114 0000E60C A1[B4030300] <1> mov eax, [u.upage] ; 'user' structure page of the process

6115 0000E611 E84267FFFF <1> call deallocate\_page

6116 0000E616 5B <1> pop ebx

6117 <1> swap\_7:

6118 0000E617 C0E302 <1> shl bl, 2 ; \* 4

6119 0000E61A 8B83[BC000300] <1> mov eax, [ebx+p.upage-4] ; the 'u' page of the new process

6120 0000E620 E840000000 <1> call rswap ; read new process into core

6121 <1> swap\_8:

6122 <1> ; Retro UNIX 8086 v1 modification !

6123 0000E625 C605[A8030300]04 <1> mov byte [u.quant], time\_count

6124 0000E62C C3 <1> retn

6125 <1>

6126 <1> wswap: ; < swap out, swap to disk >

6127 <1> ; 28/02/2017 (fnsave)

6128 <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)

6129 <1> ; 09/05/2015 (Retro UNIX 386 v1)

6130 <1> ; 26/05/2013 - 08/03/2014 (Retro UNIX 8086 v1)

6131 <1> ; 'wswap' writes out the process that is in core onto its

6132 <1> ; appropriate disk area.

6133 <1> ;

6134 <1> ; Retro UNIX 386 v1 modification ->

6135 <1> ; User (u) structure content and the user's register content

6136 <1> ; will be copied to the process's/user's UPAGE (a page for

6137 <1> ; saving 'u' structure and user registers for task switching).

6138 <1> ; u.usp - points to kernel stack address which contains

6139 <1> ; user's registers while entering system call.

6140 <1> ; u.sp - points to kernel stack address

6141 <1> ; to return from system call -for IRET-.

6142 <1> ; [u.usp]+32+16 = [u.sp]

6143 <1> ; [u.usp] -> edi, esi, ebp, esp (= [u.usp]+32), ebx,

6144 <1> ; edx, ecx, eax, gs, fs, es, ds, -> [u.sp].

6145 <1> ;

6146 <1> ; Retro UNIX 8086 v1 modification ->

6147 <1> ; 'swap to disk' is replaced with 'change running segment'

6148 <1> ; according to 8086 cpu (x86 real mode) architecture.

6149 <1> ; pdp-11 was using 64KB uniform memory while IBM PC

6150 <1> ; compatibles was using 1MB segmented memory

6151 <1> ; in 8086/8088 times.

6152 <1> ;

6153 <1> ; INPUTS ->

6154 <1> ; u.break - points to end of program

6155 <1> ; u.usp - stack pointer at the moment of swap

6156 <1> ; core - beginning of process program

6157 <1> ; ecore - end of core

6158 <1> ; user - start of user parameter area

6159 <1> ; u.uno - user process number

6160 <1> ; p.dska - holds block number of process

6161 <1> ; OUTPUTS ->

6162 <1> ; swp I/O queue

6163 <1> ; p.break - negative word count of process

6164 <1> ; r1 - process disk address

6165 <1> ; r2 - negative word count

6166 <1> ;

6167 <1> ; RETRO UNIX 8086 v1 input/output:

6168 <1> ;

6169 <1> ; INPUTS ->

6170 <1> ; u.uno - process number (to be swapped out)

6171 <1> ; OUTPUTS ->

6172 <1> ; none

6173 <1> ;

6174 <1> ; ((Modified registers: ECX, ESI, EDI))

6175 <1> ;

6176 <1>

6177 <1> ; 28/02/2017

6178 <1> ;cmp byte [multi\_tasking], 0 ; Musti tasking mode ?

6179 <1> ;jna short wswp

6180 0000E62D 803D[DA030300]00 <1> cmp byte [u.fpsave], 0 ; 28/02/2017

6181 0000E634 7606 <1> jna short wswp

6182 0000E636 DD35[DC030300] <1> fnsave [u.fpregs] ; save floating point registers (94 bytes)

6183 <1> wswp:

6184 0000E63C 8B3D[B4030300] <1> mov edi, [u.upage] ; process's user (u) structure page addr

6185 0000E642 B938000000 <1> mov ecx, (U\_SIZE + 3) / 4

6186 0000E647 BE[5C030300] <1> mov esi, user ; active user (u) structure

6187 0000E64C F3A5 <1> rep movsd

6188 <1> ;

6189 0000E64E 8B35[60030300] <1> mov esi, [u.usp] ; esp (system stack pointer,

6190 <1> ; points to user registers)

6191 0000E654 8B0D[5C030300] <1> mov ecx, [u.sp] ; return address from the system call

6192 <1> ; (for IRET)

6193 <1> ; [u.sp] -> EIP (user)

6194 <1> ; [u.sp+4]-> CS (user)

6195 <1> ; [u.sp+8] -> EFLAGS (user)

6196 <1> ; [u.sp+12] -> ESP (user)

6197 <1> ; [u.sp+16] -> SS (user)

6198 0000E65A 29F1 <1> sub ecx, esi ; required space for user registers

6199 0000E65C 83C114 <1> add ecx, 20 ; +5 dwords to return from system call

6200 <1> ; (for IRET)

6201 0000E65F C1E902 <1> shr ecx, 2

6202 0000E662 F3A5 <1> rep movsd

6203 0000E664 C3 <1> retn

6204 <1>

6205 <1> rswap: ; < swap in, swap from disk >

6206 <1> ; 28/02/2017 (frstor)

6207 <1> ; 15/01/2017

6208 <1> ; 14/01/2017

6209 <1> ; 21/05/2016

6210 <1> ; 03/05/2016

6211 <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)

6212 <1> ; 09/05/2015 - 15/09/2015 (Retro UNIX 386 v1)

6213 <1> ; 26/05/2013 - 08/03/2014 (Retro UNIX 8086 v1)

6214 <1> ; 'rswap' reads a process whose number is in r1,

6215 <1> ; from disk into core.

6216 <1> ;

6217 <1> ; Retro UNIX 386 v1 modification ->

6218 <1> ; User (u) structure content and the user's register content

6219 <1> ; will be restored from process's/user's UPAGE (a page for

6220 <1> ; saving 'u' structure and user registers for task switching).

6221 <1> ; u.usp - points to kernel stack address which contains

6222 <1> ; user's registers while entering system call.

6223 <1> ; u.sp - points to kernel stack address

6224 <1> ; to return from system call -for IRET-.

6225 <1> ; [u.usp]+32+16 = [u.sp]

6226 <1> ; [u.usp] -> edi, esi, ebp, esp (= [u.usp]+32), ebx,

6227 <1> ; edx, ecx, eax, gs, fs, es, ds, -> [u.sp].

6228 <1> ;

6229 <1> ; RETRO UNIX 8086 v1 modification ->

6230 <1> ; 'swap to disk' is replaced with 'change running segment'

6231 <1> ; according to 8086 cpu (x86 real mode) architecture.

6232 <1> ; pdp-11 was using 64KB uniform memory while IBM PC

6233 <1> ; compatibles was using 1MB segmented memory

6234 <1> ; in 8086/8088 times.

6235 <1> ;

6236 <1> ; INPUTS ->

6237 <1> ; r1 - process number of process to be read in

6238 <1> ; p.break - negative of word count of process

6239 <1> ; p.dska - disk address of the process

6240 <1> ; u.emt - determines handling of emt's

6241 <1> ; u.ilgins - determines handling of illegal instructions

6242 <1> ; OUTPUTS ->

6243 <1> ; 8 = (u.ilgins)

6244 <1> ; 24 = (u.emt)

6245 <1> ; swp - bit 10 is set to indicate read

6246 <1> ; (bit 15=0 when reading is done)

6247 <1> ; swp+2 - disk block address

6248 <1> ; swp+4 - negative word count

6249 <1> ; ((swp+6 - address of user structure))

6250 <1> ;

6251 <1> ; RETRO UNIX 8086 v1 input/output:

6252 <1> ;

6253 <1> ; INPUTS ->

6254 <1> ; AL - new process number (to be swapped in)

6255 <1> ; OUTPUTS ->

6256 <1> ; none

6257 <1> ;

6258 <1> ; ((Modified registers: EAX, ECX, ESI, EDI, ESP))

6259 <1> ;

6260 <1> ; Retro UNIX 386 v1 - modification ! 14/05/2015

6261 0000E665 89C6 <1> mov esi, eax ; process's user (u) structure page addr

6262 0000E667 B938000000 <1> mov ecx, (U\_SIZE + 3) / 4

6263 0000E66C BF[5C030300] <1> mov edi, user ; active user (u) structure

6264 0000E671 F3A5 <1> rep movsd

6265 0000E673 58 <1> pop eax ; 'rswap' return address

6266 <1> ;

6267 <1> ;cli

6268 0000E674 8B3D[60030300] <1> mov edi, [u.usp] ; esp (system stack pointer,

6269 <1> ; points to user registers)

6270 0000E67A 89FC <1> mov esp, edi ; 14/01/2017

6271 0000E67C 8B0D[5C030300] <1> mov ecx, [u.sp] ; return address from the system call

6272 <1> ; (for IRET)

6273 <1> ; [u.sp] -> EIP (user)

6274 <1> ; [u.sp+4]-> CS (user)

6275 <1> ; [u.sp+8] -> EFLAGS (user)

6276 <1> ; [u.sp+12] -> ESP (user)

6277 <1> ; [u.sp+16] -> SS (user)

6278 0000E682 29F9 <1> sub ecx, edi ; required space for user registers

6279 0000E684 83C114 <1> add ecx, 20 ; +5 dwords to return from system call

6280 <1> ; (for IRET)

6281 0000E687 C1E902 <1> shr ecx, 2

6282 0000E68A F3A5 <1> rep movsd

6283 <1> ;mov esp, [u.usp] ; 15/09/2015

6284 <1> ;sti

6285 <1> ; 28/02/2017

6286 <1> ;cmp byte [multi\_tasking], 0 ; Musti tasking mode ?

6287 <1> ;jna short rswp\_retn

6288 0000E68C 803D[DA030300]00 <1> cmp byte [u.fpsave], 0

6289 0000E693 7606 <1> jna short rswp\_retn

6290 0000E695 DD25[DC030300] <1> frstor [u.fpregs] ; restore floating point regs (94 bytes)

6291 <1> rswp\_retn:

6292 0000E69B 50 <1> push eax ; 'rswap' return address

6293 0000E69C C3 <1> retn

6294 <1>

6295 <1> putlu:

6296 <1> ; 20/05/2016

6297 <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)

6298 <1> ; 10/05/2015 - 12/09/2015 (Retro UNIX 386 v1)

6299 <1> ; 15/04/2013 - 23/02/2014 (Retro UNIX 8086 v1)

6300 <1> ; 'putlu' is called with a process number in r1 and a pointer

6301 <1> ; to lowest priority Q (runq+4) in r2. A link is created from

6302 <1> ; the last process on the queue to process in r1 by putting

6303 <1> ; the process number in r1 into the last process's link.

6304 <1> ;

6305 <1> ; INPUTS ->

6306 <1> ; r1 - user process number

6307 <1> ; r2 - points to lowest priority queue

6308 <1> ; p.dska - disk address of the process

6309 <1> ; u.emt - determines handling of emt's

6310 <1> ; u.ilgins - determines handling of illegal instructions

6311 <1> ; OUTPUTS ->

6312 <1> ; r3 - process number of last process on the queue upon

6313 <1> ; entering putlu

6314 <1> ; p.link-1 + r3 - process number in r1

6315 <1> ; r2 - points to lowest priority queue

6316 <1> ;

6317 <1> ; ((Modified registers: EDX, EBX))

6318 <1> ;

6319 <1> ; / r1 = user process no.; r2 points to lowest priority queue

6320 <1>

6321 <1> ; EBX = r2

6322 <1> ; EAX = r1 (AL=r1b)

6323 <1>

6324 <1> ; 20/05/2016

6325 <1> ; AL = process number (1 to 16) // Retro UNIX 8086, 386 v1 //

6326 <1> ; (max. 16 processes available for current kernel version)

6327 <1> ; EBX = run queue address ; 20/05/2016 (TRDOS 386)

6328 <1> ; which is one of following addresses:

6329 <1> ; 1) 'runq\_event' high priority run queue

6330 <1> ; 2) 'runq\_normal' normal/regular priority run queue

6331 <1> ; 3) 'runq\_background' low priority run queue

6332 <1>

6333 <1> ;mov ebx, runq

6334 0000E69D 0FB613 <1> movzx edx, byte [ebx]

6335 0000E6A0 43 <1> inc ebx

6336 0000E6A1 20D2 <1> and dl, dl

6337 <1> ; tstb (r2)+ / is queue empty?

6338 0000E6A3 740A <1> jz short putlu\_1

6339 <1> ; beq 1f / yes, branch

6340 0000E6A5 8A13 <1> mov dl, [ebx] ; 12/09/2015

6341 <1> ; movb (r2),r3 / no, save the "last user" process number

6342 <1> ; / in r3

6343 0000E6A7 8882[9F000300] <1> mov [edx+p.link-1], al

6344 <1> ; movb r1,p.link-1(r3) / put pointer to user on

6345 <1> ; / "last users" link

6346 0000E6AD EB03 <1> jmp short putlu\_2

6347 <1> ; br 2f /

6348 <1> putlu\_1: ; 1:

6349 0000E6AF 8843FF <1> mov [ebx-1], al

6350 <1> ; movb r1,-1(r2) / user is only user;

6351 <1> ; / put process no. at beginning and at end

6352 <1> putlu\_2: ; 2:

6353 0000E6B2 8803 <1> mov [ebx], al

6354 <1> ; movb r1,(r2) / user process in r1 is now the last entry

6355 <1> ; / on the queue

6356 0000E6B4 88C2 <1> mov dl, al

6357 0000E6B6 88B2[9F000300] <1> mov [edx+p.link-1], dh ; 0

6358 <1> ; dec r2 / restore r2

6359 0000E6BC C3 <1> retn

6360 <1> ; rts r0

6361 <1>

6362 <1> sysver:

6363 <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)

6364 0000E6BD C705[64030300]0002- <1> mov dword [u.r0], 200h ; AH = major version, AL = minor version

6364 0000E6C5 0000 <1>

6365 0000E6C7 E912E0FFFF <1> jmp sysret

6366 <1>

6367 <1>

6368 <1> syspri: ; change running priority (of the process)

6369 <1> ; 21/05/2016

6370 <1> ; 20/05/2026 - TRDOS 386 (TRDOS v2.0)

6371 <1> ; INPUT ->

6372 <1> ; BL = priority level

6373 <1> ; 0 = low running priority (running on background)

6374 <1> ; 1 = normal/regular priority (running as regular)

6375 <1> ; 2 = high/event priority (running for event)

6376 <1> ; >2 = invalid, it will accepted as 2 (event)

6377 <1> ; 0FFh = get/return current running priority only

6378 <1> ; OUTPUT ->

6379 <1> ; \* if current [u.pri] < 2

6380 <1> ; if BL input < 0FFh ->

6381 <1> ; [u.pri] is updated as in BL input (0,1,2)

6382 <1> ; if BL input = 0FFh -> AL = [u.pri] (current)

6383 <1> ;

6384 <1> ; \* if current [u.pri] = 2

6385 <1> ; if BL input < 0FFh -> cf = 1 & AL = 2

6386 <1> ; if BL input = 0FFh -> cf = 0 & AL = 2

6387 <1> ;

6388 <1> ; NOTE:

6389 <1> ; If [u.pri] = 2, it can not be changed to 1 or 0;

6390 <1> ; because, run queue of the running process is unspecified

6391 <1> ; at this stage. Process might be started by a timer event

6392 <1> ; or priority might be changed to high by previous

6393 <1> ; 'syspri' system call. In both cases, the process is in

6394 <1> ; 'runq\_normal' or 'runq\_background' queue.

6395 <1> ; As result of this fact, when the [u.quant] time quantum

6396 <1> ; of the process is elapsed or 'sysrele' system call is

6397 <1> ; instructed by the process, 'tswap' ('tswitch') procedure

6398 <1> ; will be called (to 'swap' or 'switch' out the procedure)

6399 <1> ; and it will not call 'putlu' to add the (stopping)

6400 <1> ; process to relevant run queue when [u.pri] = 2.

6401 <1> ; (Otherwise, it would be possible to add process to

6402 <1> ; a run queue while it is already in a run queue, wrongly.)

6403 <1> ;

6404 <1> ; If [u.pri]< 2, 'tswap/tswitch' procedure will call

6405 <1> ; 'putlu' to add process to relevant run queue

6406 <1> ; according to [u.pri] value. ('runq\_normal' for 1,

6407 <1> ; 'runq\_background' for 0).

6408 <1> ;

6409 <1> ; If BL input >= 2 and < 0FFh while [u.pri] < 2,

6410 <1> ; process will be added to 'runq\_normal' queue and

6411 <1> ; [u.pri] will be set to 2. (in 'syspri' system call)

6412 <1> ;

6413 <1>

6414 0000E6CC 29C0 <1> sub eax, eax ; 0

6415 0000E6CE A3[C8030300] <1> mov [u.error], eax

6416 <1>

6417 0000E6D3 A0[A9030300] <1> mov al, [u.pri]

6418 0000E6D8 A3[64030300] <1> mov [u.r0], eax

6419 <1>

6420 0000E6DD FEC3 <1> inc bl

6421 0000E6DF 0F84F9DFFFFF <1> jz sysret ; 0FFh -> 0, get priority level

6422 <1>

6423 0000E6E5 3C02 <1> cmp al, 2

6424 0000E6E7 0F83D1DFFFFF <1> jnb error ; CF = 1 & AL = 2 (& last error = 0)

6425 <1>

6426 0000E6ED FECB <1> dec bl

6427 0000E6EF 80FB02 <1> cmp bl, 2

6428 0000E6F2 7602 <1> jna short syspri\_1

6429 0000E6F4 B302 <1> mov bl, 2

6430 <1> syspri\_1:

6431 0000E6F6 881D[A9030300] <1> mov [u.pri], bl

6432 0000E6FC 80FB02 <1> cmp bl, 2

6433 0000E6FF 0F82D9DFFFFF <1> jb sysret

6434 <1>

6435 <1> ; here...

6436 <1> ; Priority of current process has been changed to high

6437 <1> ; ('run for event') but current process will be added to

6438 <1> ; 'run as normal' queue. ('run for event' high priority

6439 <1> ; queue is under control of timer -& RTC- interrupt only!)

6440 <1> ;

6441 <1> ; (Otherwise, process can fall into black hole!

6442 <1> ; e.g. if it is not in waiting list and it has not got

6443 <1> ; a timer event and it is not in a run queue!

6444 <1> ; Because, when [u.pri] is 2, 'tswap/tswitch' will not

6445 <1> ; add the stopping process to a run queue.)

6446 <1>

6447 0000E705 A0[B3030300] <1> mov al, [u.uno]

6448 0000E70A BB[54030300] <1> mov ebx, runq\_normal ; normal priority !

6449 <1> ; [u.pri] is set to high

6450 <1> ; but 'runq\_event' queue is set

6451 <1> ; only by the kernel's timer

6452 <1> ; event function (timer interrupt).

6453 0000E70F E889FFFFFF <1> call putlu

6454 0000E714 E9C5DFFFFF <1> jmp sysret

6455 <1>

6456 <1> cpass: ; / get next character from user area of core and put it in AL (r1)

6457 <1> ; 02/05/2016 - TRDOS 386 (TRDOS v2.0)

6458 <1> ; 19/05/2015 - 18/10/2015 (Retro UNIX 386 v1)

6459 <1> ; 14/08/2013 - 20/09/2013 (Retro UNIX 8086 v1)

6460 <1> ; INPUTS ->

6461 <1> ; [u.base] = virtual address in user area

6462 <1> ; [u.count] = byte count (max.)

6463 <1> ; [u.pcount] = byte count in page (0 = reset)

6464 <1> ; OUTPUTS ->

6465 <1> ; AL = the character which is pointed by [u.base]

6466 <1> ; zf = 1 -> transfer count has been completed

6467 <1> ;

6468 <1> ; ((Modified registers: EAX, EDX, ECX))

6469 <1> ;

6470 0000E719 833D[88030300]00 <1> cmp dword [u.count], 0 ; have all the characters been transferred

6471 <1> ; i.e., u.count, # of chars. left

6472 0000E720 763F <1> jna short cpass\_3 ; to be transferred = 0?) yes, branch

6473 0000E722 FF0D[88030300] <1> dec dword [u.count] ; no, decrement u.count

6474 <1> ; 19/05/2015

6475 <1> ;(Retro UNIX 386 v1 - translation from user's virtual address

6476 <1> ; to physical address

6477 0000E728 66833D[C4030300]00 <1> cmp word [u.pcount], 0 ; byte count in page = 0 (initial value)

6478 <1> ; 1-4095 --> use previous physical base address

6479 <1> ; in [u.pbase]

6480 0000E730 770E <1> ja short cpass\_1

6481 0000E732 833D[BC030300]00 <1> cmp dword [u.ppgdir], 0 ; is the caller os kernel

6482 0000E739 7427 <1> je short cpass\_k ; (sysexec, '/etc/init') ? (MainProg)

6483 0000E73B E858FDFFFF <1> call trans\_addr\_r

6484 <1> cpass\_1:

6485 0000E740 66FF0D[C4030300] <1> dec word [u.pcount]

6486 <1> cpass\_2:

6487 0000E747 8B15[C0030300] <1> mov edx, [u.pbase]

6488 0000E74D 8A02 <1> mov al, [edx] ; take the character pointed to

6489 <1> ; by u.base and put it in r1

6490 0000E74F FF05[8C030300] <1> inc dword [u.nread] ; increment no. of bytes transferred

6491 0000E755 FF05[84030300] <1> inc dword [u.base] ; increment the buffer address to point to the

6492 <1> ; next byte

6493 0000E75B FF05[C0030300] <1> inc dword [u.pbase]

6494 <1> cpass\_3:

6495 0000E761 C3 <1> retn

6496 <1> cpass\_k:

6497 <1> ; 02/07/2015

6498 <1> ; The caller is os kernel

6499 <1> ; (get sysexec arguments from kernel's memory space)

6500 0000E762 8B1D[84030300] <1> mov ebx, [u.base]

6501 0000E768 66C705[C4030300]00- <1> mov word [u.pcount], PAGE\_SIZE ; 4096

6501 0000E770 10 <1>

6502 0000E771 891D[C0030300] <1> mov [u.pbase], ebx

6503 0000E777 EBCE <1> jmp short cpass\_2

6504 <1>

6505 <1> transfer\_to\_user\_buffer: ; fast transfer

6506 <1> ; 27/05/2016

6507 <1> ; 16/05/2016 - TRDOS 386 (TRDOS v2.0)

6508 <1> ;

6509 <1> ; INPUT ->

6510 <1> ; ESI = source address in system space

6511 <1> ; EDI = user's buffer address

6512 <1> ; ECX = transfer (byte) count

6513 <1> ; [u.pgdir] = user's page directory

6514 <1> ; OUTPUT ->

6515 <1> ; ECX = actual transfer count

6516 <1> ; cf = 1 -> error

6517 <1> ; [u.count] = remain byte count

6518 <1> ;

6519 <1> ; Modified registers: eax, ecx

6520 <1> ;

6521 <1>

6522 0000E779 21C9 <1> and ecx, ecx

6523 0000E77B 743B <1> jz short ttub\_4

6524 <1>

6525 0000E77D 890D[88030300] <1> mov [u.count], ecx

6526 <1>

6527 0000E783 57 <1> push edi

6528 0000E784 56 <1> push esi

6529 0000E785 53 <1> push ebx

6530 0000E786 52 <1> push edx

6531 0000E787 51 <1> push ecx

6532 <1>

6533 0000E788 89FB <1> mov ebx, edi

6534 0000E78A 81C300004000 <1> add ebx, CORE ; 27/05/2016

6535 <1> ttub\_1:

6536 <1> ; ebx = virtual (linear) address

6537 <1> ; [u.pgdir] = user's page directory

6538 0000E790 E8006BFFFF <1> call get\_physical\_addr\_x ; get physical address

6539 0000E795 7222 <1> jc short ttub\_5

6540 <1> ; eax = physical address

6541 <1> ; ecx = remain byte count in page (1-4096)

6542 0000E797 89C7 <1> mov edi, eax

6543 0000E799 A1[88030300] <1> mov eax, [u.count]

6544 0000E79E 39C1 <1> cmp ecx, eax

6545 0000E7A0 7602 <1> jna short ttub\_2

6546 0000E7A2 89C1 <1> mov ecx, eax

6547 <1> ttub\_2:

6548 0000E7A4 29C8 <1> sub eax, ecx

6549 0000E7A6 01CB <1> add ebx, ecx

6550 0000E7A8 F3A4 <1> rep movsb

6551 0000E7AA A3[88030300] <1> mov [u.count], eax

6552 0000E7AF 09C0 <1> or eax, eax

6553 0000E7B1 75DD <1> jnz short ttub\_1

6554 <1> ttub\_retn:

6555 <1> tfub\_retn:

6556 0000E7B3 59 <1> pop ecx ; transfer count = actual transfer count

6557 <1> ttub\_3:

6558 0000E7B4 5A <1> pop edx

6559 0000E7B5 5B <1> pop ebx

6560 0000E7B6 5E <1> pop esi

6561 0000E7B7 5F <1> pop edi

6562 <1> ttub\_4:

6563 0000E7B8 C3 <1> retn

6564 <1> ttub\_5:

6565 0000E7B9 59 <1> pop ecx

6566 0000E7BA 2B0D[88030300] <1> sub ecx, [u.count] ; actual transfer count

6567 0000E7C0 F9 <1> stc

6568 0000E7C1 EBF1 <1> jmp short ttub\_3

6569 <1>

6570 <1> transfer\_from\_user\_buffer: ; fast transfer

6571 <1> ; 27/05/2016

6572 <1> ; 16/05/2016 - TRDOS 386 (TRDOS v2.0)

6573 <1> ;

6574 <1> ; INPUT ->

6575 <1> ; ESI = user's buffer address

6576 <1> ; EDI = destination address in system space

6577 <1> ; ECX = transfer (byte) count

6578 <1> ; [u.pgdir] = user's page directory

6579 <1> ; OUTPUT ->

6580 <1> ; ecx = actual transfer count

6581 <1> ; cf = 1 -> error

6582 <1> ; [u.count] = remain byte count

6583 <1> ;

6584 <1> ; Modified registers: eax, ecx

6585 <1> ;

6586 <1>

6587 0000E7C3 21C9 <1> and ecx, ecx

6588 <1> ;jz short tfub\_4

6589 0000E7C5 74F1 <1> jz short ttub\_4

6590 <1>

6591 0000E7C7 890D[88030300] <1> mov [u.count], ecx

6592 <1>

6593 0000E7CD 57 <1> push edi

6594 0000E7CE 56 <1> push esi

6595 0000E7CF 53 <1> push ebx

6596 0000E7D0 52 <1> push edx

6597 0000E7D1 51 <1> push ecx

6598 <1>

6599 0000E7D2 89F3 <1> mov ebx, esi

6600 0000E7D4 81C300004000 <1> add ebx, CORE ; 27/05/2016

6601 <1> tfub\_1:

6602 <1> ; ebx = virtual (linear) address

6603 <1> ; [u.pgdir] = user's page directory

6604 0000E7DA E8B66AFFFF <1> call get\_physical\_addr\_x ; get physical address

6605 <1> ;jc short tfub\_5

6606 0000E7DF 72D8 <1> jc short ttub\_5

6607 <1> ; eax = physical address

6608 <1> ; ecx = remain byte count in page (1-4096)

6609 0000E7E1 89C6 <1> mov esi, eax

6610 0000E7E3 A1[88030300] <1> mov eax, [u.count]

6611 0000E7E8 39C1 <1> cmp ecx, eax

6612 0000E7EA 7602 <1> jna short tfub\_2

6613 0000E7EC 89C1 <1> mov ecx, eax

6614 <1> tfub\_2:

6615 0000E7EE 29C8 <1> sub eax, ecx

6616 0000E7F0 01CB <1> add ebx, ecx

6617 0000E7F2 F3A4 <1> rep movsb

6618 0000E7F4 A3[88030300] <1> mov [u.count], eax

6619 0000E7F9 09C0 <1> or eax, eax

6620 0000E7FB 75DD <1> jnz short tfub\_1

6621 <1>

6622 0000E7FD EBB4 <1> jmp short tfub\_retn

6623 <1>

6624 <1> ;tfub\_retn:

6625 <1> ; pop ecx ; transfer count = actual transfer count

6626 <1> ;tfub\_3:

6627 <1> ; pop edx

6628 <1> ; pop ebx

6629 <1> ; pop esi

6630 <1> ; pop edi

6631 <1> ;tfub\_4:

6632 <1> ; retn

6633 <1> ;tfub\_5:

6634 <1> ; pop ecx

6635 <1> ; sub ecx, [u.count] ; actual transfer count

6636 <1> ; stc

6637 <1> ; jmp short tfub\_3

6638 <1>

6639 <1> sysfff: ; <Find First File>

6640 <1> ; 17/10/2016

6641 <1> ; 16/10/2016

6642 <1> ; 15/10/2016 TRDOS 386 (TRDOS v2.0) feature only !

6643 <1> ; -derived from TRDOS v1.0, INT\_21H.ASM-

6644 <1> ; ("loc\_INT21h\_find\_first\_file")

6645 <1> ; TRDOS 8086 (v1.0)

6646 <1> ; 07/08/2011

6647 <1> ; Find First File

6648 <1> ; INPUT:

6649 <1> ; CX= Attributes

6650 <1> ; DS:DX= Pointer to filename

6651 <1> ; MSDOS OUTPUT:

6652 <1> ; DTA: (Default address: PSP offset 80h)

6653 <1> ; Offset Descrription

6654 <1> ; 0 Reserved for use find next file

6655 <1> ; 21 Attribute of file found

6656 <1> ; 22 Time stamp of file

6657 <1> ; 24 Date stamp of file

6658 <1> ; 26 File size in bytes

6659 <1> ; 30 Filename and extension (zero terminated)

6660 <1> ; If cf = 1:

6661 <1> ; Error Codes: (in AX)

6662 <1> ; 2 - File not found

6663 <1> ; 18 - No more files

6664 <1> ;

6665 <1> ; TRDOS 386 (v2.0)

6666 <1> ; 15/10/2016

6667 <1> ;

6668 <1> ; INPUT ->

6669 <1> ; CL = File attributes

6670 <1> ; bit 0 (1) - Read only file (R)

6671 <1> ; bit 1 (1) - Hidden file (H)

6672 <1> ; bit 2 (1) - System file (R)

6673 <1> ; bit 3 (1) - Volume label/name (V)

6674 <1> ; bit 4 (1) - Subdirectory (D)

6675 <1> ; bit 5 (1) - File has been archived (A)

6676 <1> ; CH = 0 -> Return basic parameters (24 bytes)

6677 <1> ; CH > 0 -> Return FindFile structure/table (128 bytes)

6678 <1> ; EBX = Pointer to filename (ASCIIZ) -path-

6679 <1> ; EDX = File parameters buffer address

6680 <1> ; (buffer size = 24 bytes if CH input = 0)

6681 <1> ; (buffer size = 128 bytes if CH input > 0)

6682 <1> ;

6683 <1> ; OUTPUT ->

6684 <1> ; EAX = 0 if CH input > 0

6685 <1> ; EAX = First cluster number of file if CH input = 0

6686 <1> ; EDX = File parameters table/structure address

6687 <1> ; Basic Parameters:

6688 <1> ; Offset Description

6689 <1> ; ------ ---------------

6690 <1> ; 0 File Attributes

6691 <1> ; 1 Ambiguous filename chars are used sign

6692 <1> ; (0 = filename fits exactly with request)

6693 <1> ; (>0 = ambiguous filename chars are used)

6694 <1> ; 2 Time stamp of file

6695 <1> ; 4 Date stamp of file

6696 <1> ; 6 File size in bytes

6697 <1> ; 10 Short Filename (ASCIIZ, max. 13 bytes)

6698 <1> ; 23 Longname Length (1-255) if existing

6699 <1> ;

6700 <1> ; cf = 1 -> Error code in AL

6701 <1> ;

6702 <1> ; Modified Registers: EAX (at the return of system call)

6703 <1> ;

6704 <1> ; TR-DOS FindFile (FFF) Structure (128 bytes):

6705 <1> ; 09/10/2011 (DIR.ASM) - 10/02/2016 (trdoskx.s)

6706 <1> ;

6707 <1> ; Offset Parameter Size

6708 <1> ; ------ ------------------ --------

6709 <1> ; 0 FindFile\_Drv 1 byte

6710 <1> ; 1 FindFile\_Directory 65 bytes

6711 <1> ; 66 FindFile\_Name 13 bytes

6712 <1> ; 79 FindFile\_LongNameEntryLength 1 byte

6713 <1> ;Above 80 bytes form

6714 <1> ;TR-DOS Source/Destination File FullName Format/Structure

6715 <1> ; 80 FindFile\_AttributesMask 1 word

6716 <1> ; 82 FindFile\_DirEntry 32 bytes (\*)

6717 <1> ; 114 FindFile\_DirFirstCluster 1 double word

6718 <1> ; 118 FindFile\_DirCluster 1 double word

6719 <1> ; 122 FindFile\_DirEntryNumber 1 word

6720 <1> ; 124 FindFile\_MatchCounter 1 word

6721 <1> ; 126 FindFile\_Reserved 1 word

6722 <1> ; (\*) MS-DOS, FAT 12-16-32 classic directory entry (32 bytes)

6723 <1>

6724 <1> ;mov [u.namep], ebx

6725 <1> ; 16/10/2016

6726 0000E7FF 8915[EC650100] <1> mov [FFF\_UBuffer], edx

6727 0000E805 66890D[F1650100] <1> mov [FFF\_Attrib], cx ; [FFF\_RType] = ch

6728 <1> ; Attributes in CL, return data type in CH

6729 0000E80C 89DE <1> mov esi, ebx

6730 <1> ; file name is forced, change directory as temporary

6731 <1> ;mov ax, 1

6732 <1> ;mov [FFF\_Valid], ah ; 0 ; reset ; 17/10/2016

6733 <1> ;call set\_working\_path

6734 0000E80E E8E8130000 <1> call set\_working\_path\_x ; 17/10/2016

6735 0000E813 731D <1> jnc short sysfff\_0

6736 <1>

6737 0000E815 21C0 <1> and eax, eax ; 0 -> Bad Path!

6738 0000E817 7505 <1> jnz short sysfff\_err

6739 <1>

6740 <1> ; eax = 0

6741 0000E819 B80C000000 <1> mov eax, ERR\_DIR\_NOT\_FOUND ; Directory not found !

6742 <1> sysfff\_err:

6743 0000E81E A3[64030300] <1> mov [u.r0], eax

6744 0000E823 A3[C8030300] <1> mov [u.error], eax

6745 0000E828 E8A3140000 <1> call reset\_working\_path

6746 0000E82D E98CDEFFFF <1> jmp error

6747 <1>

6748 <1> sysfff\_0:

6749 <1> ;sub ah, ah ; ah = 0

6750 0000E832 8A0424 <1> mov al, [esp]

6751 0000E835 08C0 <1> or al, al

6752 0000E837 7412 <1> jz short sysfff\_2

6753 0000E839 B410 <1> mov ah, 10h

6754 0000E83B A808 <1> test al, 08h

6755 0000E83D 7503 <1> jnz short sysfff\_1

6756 0000E83F 80CC08 <1> or ah, 08h

6757 <1> sysfff\_1:

6758 0000E842 2410 <1> and al, 10h ; Directory

6759 0000E844 7405 <1> jz short sysfff\_2

6760 0000E846 80E408 <1> and ah, 08h

6761 0000E849 30C0 <1> xor al, al ; When a directory is searched,

6762 <1> ; filename will be returned even if

6763 <1> ; it is not a directory!

6764 <1> ; Because: (in order to prevent

6765 <1> ; creating a dir with existing file name)

6766 <1> ; Dir and file names must not be same!

6767 <1> ; (return attribute must be checked)

6768 <1> sysfff\_2:

6769 <1> ; AX = Attributes mask

6770 <1> ; AL = AND mask (result must be equal to AL)

6771 <1> ; AH = Negative AND mask (result must be ZERO)

6772 <1> ; ESI = FindFile\_Name address

6773 <1>

6774 0000E84B E8139AFFFF <1> call find\_first\_file

6775 0000E850 72CC <1> jc short sysfff\_err ; eax = 2 (File not found !)

6776 <1>

6777 <1> ; ESI = Directory Entry (FindFile\_DirEntry) Location

6778 <1> ; EDI = Directory Buffer Directory Entry Location

6779 <1> ; EAX = File Size

6780 <1> ; BL = Attributes of The File/Directory

6781 <1> ; BH = Long Name Yes/No Status (>0 is YES)

6782 <1> ; DX > 0 : Ambiguous filename chars are used

6783 <1>

6784 <1> sysfff\_3:

6785 <1> ; 16/10/2016

6786 0000E852 668B0D[F1650100] <1> mov cx, [FFF\_Attrib]

6787 <1> ; Attribs in CL, return data type in CH

6788 <1>

6789 <1> ;or cl, cl

6790 <1> ;jz short sysfff\_4 ; 0 = No filter

6791 0000E859 80F1FF <1> xor cl, 0FFh

6792 0000E85C 20D9 <1> and cl, bl

6793 0000E85E 7409 <1> jz short sysfff\_4

6794 <1>

6795 <1> ;mov eax, 2 ; 'file not found !' error

6796 <1> ;jmp short sysfff\_err\_1

6797 <1>

6798 <1> ; 16/10/2016

6799 0000E860 E8AD9AFFFF <1> call find\_next\_file

6800 0000E865 72B7 <1> jc short sysfff\_err ; eax = 12 (no more files !)

6801 0000E867 EBE9 <1> jmp short sysfff\_3

6802 <1>

6803 <1> sysfff\_4:

6804 0000E869 20ED <1> and ch, ch ; [FFF\_RType]

6805 0000E86B 7412 <1> jz short sysfff\_5

6806 0000E86D B980000000 <1> mov ecx, 128 ; ; transfer length

6807 0000E872 880D[F0650100] <1> mov [FFF\_Valid], cl

6808 <1> sysfnf\_11:

6809 0000E878 BE[A2620100] <1> mov esi, FindFile\_Drv

6810 0000E87D EB44 <1> jmp short sysfff\_6

6811 <1> sysfff\_5:

6812 <1> ;mov esi, FindFile\_DirEntry

6813 0000E87F B918000000 <1> mov ecx, 24 ; transfer length

6814 0000E884 880D[F0650100] <1> mov [FFF\_Valid], cl

6815 <1> sysfnf\_12:

6816 0000E88A BF[AC6A0100] <1> mov edi, DTA ; FFF data transfer address

6817 <1> ;mov al, [esi+DirEntry\_Attr] ; 11

6818 0000E88F 88D8 <1> mov al, bl ; File/Dir Attributes

6819 0000E891 887F17 <1> mov [edi+23], bh ; Longname length (0= none)

6820 0000E894 AA <1> stosb

6821 0000E895 88D0 <1> mov al, dl ; DL is for '?'

6822 0000E897 00F0 <1> add al, dh ; DH is for '\*'

6823 <1> ; AL > 0 if ambiguous file name wildcards are used

6824 0000E899 AA <1> stosb

6825 0000E89A 8B4616 <1> mov eax, [esi+DirEntry\_WrtTime] ; 22

6826 0000E89D AB <1> stosd ; DirEntry\_WrtTime & DirEntry\_WrtDate

6827 0000E89E 8B461C <1> mov eax, [esi+DirEntry\_FileSize] ; 28

6828 0000E8A1 AB <1> stosd

6829 0000E8A2 668B4614 <1> mov ax, [esi+DirEntry\_FstClusHI] ; 20

6830 0000E8A6 66C1E010 <1> shl ax, 16

6831 0000E8AA 668B461A <1> mov ax, [esi+DirEntry\_FstClusLO] ; 26

6832 0000E8AE A3[64030300] <1> mov [u.r0], eax ; First Cluster

6833 <1>

6834 <1> ;mov esi, FindFile\_DirEntry

6835 0000E8B3 E855140000 <1> call get\_file\_name

6836 <1>

6837 0000E8B8 8A0D[F0650100] <1> mov cl, [FFF\_Valid]

6838 0000E8BE BE[AC6A0100] <1> mov esi, DTA ; FFF data transfer address

6839 <1> sysfff\_6:

6840 0000E8C3 8B3D[EC650100] <1> mov edi, [FFF\_UBuffer] ; user's buffer address (edx)

6841 0000E8C9 E8ABFEFFFF <1> call transfer\_to\_user\_buffer

6842 <1>

6843 0000E8CE 890D[64030300] <1> mov [u.r0], ecx ; actual transfer count

6844 0000E8D4 E8F7130000 <1> call reset\_working\_path

6845 0000E8D9 E900DEFFFF <1> jmp sysret

6846 <1>

6847 <1> sysfnf: ; <Find Next File>

6848 <1> ; 16/10/2016 TRDOS 386 (TRDOS v2.0) feature only !

6849 <1> ; -derived from TRDOS v1.0, INT\_21H.ASM-

6850 <1> ; ("loc\_INT21h\_find\_next\_file")

6851 <1> ; TRDOS 8086 (v1.0)

6852 <1> ; 07/08/2011

6853 <1> ; Find First File

6854 <1> ; INPUT:

6855 <1> ; none

6856 <1> ; MSDOS OUTPUT:

6857 <1> ; DTA: (Default address: PSP offset 80h)

6858 <1> ; Offset Descrription

6859 <1> ; 0 Reserved for use find next file

6860 <1> ; 21 Attribute of file found

6861 <1> ; 22 Time stamp of file

6862 <1> ; 24 Date stamp of file

6863 <1> ; 26 File size in bytes

6864 <1> ; 30 Filename and extension (zero terminated)

6865 <1> ; If cf = 1:

6866 <1> ; Error Codes: (in AX)

6867 <1> ; 18 - No more files

6868 <1> ;

6869 <1> ; TRDOS 386 (v2.0)

6870 <1> ; 16/10/2016

6871 <1> ;

6872 <1> ; INPUT ->

6873 <1> ; none

6874 <1> ; OUTPUT ->

6875 <1> ; EAX = 0 if CH input of 'Find First File' > 0

6876 <1> ; EAX = First cluster number of file

6877 <1> ; if CH input of 'Find First File' = 0

6878 <1> ; EDX = File parameters table/structure address

6879 <1> ;

6880 <1> ; cf = 1 -> Error code in AL

6881 <1> ;

6882 <1> ; Modified Registers: EAX (at the return of system call)

6883 <1>

6884 <1> ;

6885 <1> ; Note: If byte [FFF\_Valid] = 0

6886 <1> ; 'sysfnf' will return with 'no more files' error.

6887 <1> ; If byte [FFF\_Valid] = 24

6888 <1> ; 'sysfnf' will return with 32 bytes basic parameters

6889 <1> ; at the address which is in EDX.

6890 <1> ; If byte [FFF\_Valid] = 128

6891 <1> ; 'sysfnf' will return with 128 bytes Find File

6892 <1> ; Structure/Table at the address which is in EDX.

6893 <1>

6894 0000E8DE 803D[F0650100]00 <1> cmp byte [FFF\_Valid], 0

6895 0000E8E5 7714 <1> ja short stsfnf\_0

6896 <1> ; 'no more files !' error

6897 0000E8E7 B80C000000 <1> mov eax, ERR\_NO\_MORE\_FILES ; 12

6898 0000E8EC A3[64030300] <1> mov [u.r0], eax

6899 0000E8F1 A3[C8030300] <1> mov [u.error], eax

6900 0000E8F6 E9C3DDFFFF <1> jmp error

6901 <1> stsfnf\_0:

6902 <1> ;cmp byte [FFF\_Valid], 128

6903 <1> ;je short stsfnf\_1

6904 <1> ;cmp byte [FFF\_Valid], 24

6905 <1> ;je short stsfnf\_1

6906 <1> ;mov [FFF\_Valid], 24 ; Default

6907 <1> stsfnf\_1:

6908 0000E8FB 0FB61D[FE580100] <1> movzx ebx, byte [Current\_Drv]

6909 0000E902 66891D[F6650100] <1> mov [SWP\_DRV], bx

6910 0000E909 8A15[A2620100] <1> mov dl, [FindFile\_Drv]

6911 0000E90F 38DA <1> cmp dl, bl

6912 0000E911 750B <1> jne short stsfnf\_2

6913 0000E913 86FB <1> xchg bh, bl

6914 0000E915 BE00010900 <1> mov esi, Logical\_DOSDisks

6915 0000E91A 01DE <1> add esi, ebx

6916 0000E91C EB0D <1> jmp short sysfnf\_3

6917 <1>

6918 <1> stsfnf\_2:

6919 0000E91E FE05[F7650100] <1> inc byte [SWP\_DRV\_chg]

6920 <1>

6921 0000E924 E89785FFFF <1> call change\_current\_drive

6922 0000E929 7245 <1> jc short sysfnf\_err\_1 ; read error !

6923 <1> ; (do not stop, because

6924 <1> ; we don't have a

6925 <1> ; 'no more files'

6926 <1> ; -file not found- error,

6927 <1> ; next sysfnf system call

6928 <1> ; may solve the problem,

6929 <1> ; after re-placing the disk)

6930 <1> sysfnf\_3:

6931 0000E92B A1[18630100] <1> mov eax, [FindFile\_DirCluster]

6932 0000E930 21C0 <1> and eax, eax

6933 0000E932 7550 <1> jnz short sysfnf\_6

6934 <1>

6935 0000E934 803D[FD580100]02 <1> cmp byte [Current\_FATType], 2

6936 0000E93B 772C <1> ja short sysfnf\_err\_0 ; invalid, we neeed to stop !?

6937 0000E93D 803D[FD580100]01 <1> cmp byte [Current\_FATType], 1

6938 0000E944 7223 <1> jb short sysfnf\_err\_0 ; invalid, we neeed to stop !?

6939 <1>

6940 0000E946 3805[28610100] <1> cmp byte [DirBuff\_ValidData], al ; 0

6941 0000E94C 7608 <1> jna short sysfnf\_4

6942 <1>

6943 0000E94E 3B05[2D610100] <1> cmp eax, [DirBuff\_Cluster] ; 0 ?

6944 0000E954 745E <1> je short sysfnf\_9

6945 <1>

6946 <1> ;cmp byte [Current\_Dir\_Level], 0

6947 <1> ;ja short sysfnf\_4

6948 <1> ;jna short sysfnf\_9

6949 <1>

6950 <1> sysfnf\_4:

6951 0000E956 FE05[F7650100] <1> inc byte [SWP\_DRV\_chg]

6952 0000E95C E842D3FFFF <1> call load\_FAT\_root\_directory

6953 0000E961 7351 <1> jnc short sysfnf\_9

6954 <1> ; eax = error code (17, 'drv not ready or read error')

6955 0000E963 EB0B <1> jmp short sysfnf\_err\_1 ; read error ! (no FNF stop)

6956 <1> ; (if you want, try again,

6957 <1> ; after re-placing the disk)

6958 <1> sysfnf\_5:

6959 0000E965 3C0C <1> cmp al, 12 ; 'no more files' error

6960 0000E967 7507 <1> jne short sysfnf\_err\_1 ; (no FNF stop -sysfnf will try

6961 <1> ; to read the directory again,

6962 <1> ; if the user calls sysfnf

6963 <1> ; just after this error return-)

6964 <1> ; (FNF stop -sysfnf will not try

6965 <1> ; to read the directory again-)

6966 <1>

6967 <1> sysfnf\_err\_0:

6968 0000E969 C605[F0650100]00 <1> mov byte [FFF\_Valid], 0 ; FNF stop sign

6969 <1> sysfnf\_err\_1:

6970 0000E970 A3[64030300] <1> mov [u.r0], eax

6971 0000E975 A3[C8030300] <1> mov [u.error], eax

6972 0000E97A E851130000 <1> call reset\_working\_path

6973 0000E97F E93ADDFFFF <1> jmp error

6974 <1>

6975 <1> sysfnf\_6:

6976 0000E984 803D[28610100]00 <1> cmp byte [DirBuff\_ValidData], 0

6977 0000E98B 7608 <1> jna short sysfnf\_7

6978 <1>

6979 0000E98D 3B05[2D610100] <1> cmp eax, [DirBuff\_Cluster]

6980 0000E993 741F <1> je short sysfnf\_9

6981 <1>

6982 <1> sysfnf\_7:

6983 0000E995 FE05[F7650100] <1> inc byte [SWP\_DRV\_chg]

6984 0000E99B 803D[FD580100]01 <1> cmp byte [Current\_FATType], 1

6985 0000E9A2 7309 <1> jnb short sysfnf\_8

6986 <1>

6987 <1> ; Singlix (TRFS) File System

6988 <1> ; (access via compatibility buffer)

6989 0000E9A4 E8C2D3FFFF <1> call load\_FS\_sub\_directory

6990 0000E9A9 7309 <1> jnc short sysfnf\_9

6991 <1>

6992 0000E9AB EBC3 <1> jmp short sysfnf\_err\_1 ; read error (no FNF stop)

6993 <1>

6994 <1> sysfnf\_8:

6995 0000E9AD E87CD3FFFF <1> call load\_FAT\_sub\_directory

6996 0000E9B2 72BC <1> jc short sysfnf\_err\_1 ; read error (no FNF stop)

6997 <1>

6998 <1> sysfnf\_9:

6999 0000E9B4 E85999FFFF <1> call find\_next\_file

7000 0000E9B9 72AA <1> jc short sysfnf\_5

7001 <1>

7002 0000E9BB A0[F1650100] <1> mov al, [FFF\_Attrib]

7003 <1> ;or al, al

7004 <1> ;jz short sysfnf\_10 ; 0 = No filter

7005 0000E9C0 34FF <1> xor al, 0FFh

7006 0000E9C2 20D8 <1> and al, bl

7007 0000E9C4 75EE <1> jnz short sysfnf\_9 ; search for next file until

7008 <1> ; an error return from

7009 <1> ; find\_next\_file procedure

7010 <1> sysfnf\_10:

7011 0000E9C6 0FB60D[F0650100] <1> movzx ecx, byte [FFF\_Valid]

7012 0000E9CD 80F980 <1> cmp cl, 128 ; complete FindFile structure/table

7013 0000E9D0 0F84A2FEFFFF <1> je sysfnf\_11

7014 <1> ;cmp cl, 24 ; basic parameters

7015 <1> ;je sysfnf\_12

7016 0000E9D6 E9AFFEFFFF <1> jmp sysfnf\_12

7017 <1>

7018 <1> writei:

7019 <1> ; 26/10/2016

7020 <1> ; 25/10/2016

7021 <1> ; 23/10/2016

7022 <1> ; 22/10/2016

7023 <1> ; 19/10/2016 - TRDOS 386 (TRDOS v2.0)

7024 <1> ; 19/05/2015 - 20/05/2015 (Retro UNIX 386 v1)

7025 <1> ; 12/03/2013 - 31/07/2013 (Retro UNIX 8086 v1)

7026 <1> ;

7027 <1> ; Write data to file with first cluster number in EAX

7028 <1> ;

7029 <1> ; INPUTS ->

7030 <1> ; EAX - First cluster number of the file

7031 <1> ; EBX - File number (Open file index number)

7032 <1> ; u.count - byte count to be written

7033 <1> ; u.base - points to user buffer

7034 <1> ; u.fofp - points to dword with current file offset

7035 <1> ; i.size - file size

7036 <1> ; cdev - logical dos drive number of the file

7037 <1> ; OUTPUTS ->

7038 <1> ; u.count - cleared

7039 <1> ; u.nread - accumulates total bytes passed back

7040 <1> ; i.size - new file size (if file byte offset overs file size)

7041 <1> ; u.fofp - points to u.off (with new offset value)

7042 <1> ;

7043 <1> ; (Retro UNIX Prototype : 11/11/2012 - 18/11/2012, UNIXCOPY.ASM)

7044 <1> ; ((Modified registers: eax, edx, ebx, ecx, esi, edi, ebp))

7045 <1>

7046 0000E9DB 31C9 <1> xor ecx, ecx

7047 0000E9DD 890D[8C030300] <1> mov [u.nread], ecx ; 0

7048 0000E9E3 66890D[C4030300] <1> mov [u.pcount], cx ; 19/05/2015

7049 0000E9EA 390D[88030300] <1> cmp [u.count], ecx

7050 0000E9F0 7701 <1> ja short writei\_1

7051 0000E9F2 C3 <1> retn

7052 <1> writei\_1:

7053 0000E9F3 881D[B0650100] <1> mov [writei.ofn], bl ; Open file number

7054 0000E9F9 880D[EB650100] <1> mov [setfmod], cl ; 0 ; reset 'update lm date&time' sign

7055 <1> dskw\_0:

7056 <1> ; 26/10/2016

7057 <1> ; 22/10/2016, 23/10/2016, 25/10/2016

7058 <1> ; 19/10/2016 - TRDOS 386 (TRDOS v2.0)

7059 <1> ; 31/05/2015 - 25/07/2015 (Retro UNIX 386 v1)

7060 <1> ; 26/04/2013 - 20/09/2013 (Retro UNIX 8086 v1)

7061 <1> ;

7062 <1> ; 01/08/2013 (mkdir\_w check)

7063 0000E9FF E8D7000000 <1> call mget\_w

7064 <1> ; eax = sector/block number

7065 <1>

7066 0000EA04 8B1D[74030300] <1> mov ebx, [u.fofp]

7067 0000EA0A 8B13 <1> mov edx, [ebx]

7068 0000EA0C 81E2FF010000 <1> and edx, 1FFh ; / test the lower 9 bits of the file offset

7069 0000EA12 750C <1> jnz short dskw\_1 ; / if its non-zero, branch

7070 <1> ; if zero, file offset = 0,

7071 <1> ; / 512, 1024,...(i.e., start of new block)

7072 0000EA14 813D[88030300]0002- <1> cmp dword [u.count], 512

7072 0000EA1C 0000 <1>

7073 <1> ; / if zero, is there enough data to fill

7074 <1> ; / an entire block? (i.e., no. of

7075 0000EA1E 7337 <1> jnb short dskw\_2 ; / bytes to be written greater than 512.?

7076 <1> ; / Yes, branch. Don't have to read block

7077 <1> dskw\_1: ; in as no past info. is to be saved

7078 <1> ; (the entire block will be overwritten).

7079 <1> ; 23/10/2016

7080 <1>

7081 0000EA20 BB[94070300] <1> mov ebx, writei\_buffer

7082 <1> ; esi = logical dos drive description table address

7083 <1> ; eax = sector number

7084 <1> ; ebx = buffer address (in kernel's memory space)

7085 <1> ; ecx = sector count

7086 0000EA25 B901000000 <1> mov ecx, 1

7087 0000EA2A E8A90D0000 <1> call disk\_read

7088 <1> ;call dskrd ; / no, must retain old info..

7089 <1> ; / Hence, read block 'r1' into an I/O buffer

7090 0000EA2F 7326 <1> jnc short dskw\_2

7091 <1>

7092 <1> ; disk read error

7093 0000EA31 B811000000 <1> mov eax, 17 ; drive not ready or READ ERROR !

7094 <1> dskw\_err: ; jump from disk write error

7095 0000EA36 A3[64030300] <1> mov [u.r0], eax

7096 0000EA3B A3[C8030300] <1> mov [u.error], eax

7097 <1>

7098 0000EA40 803D[EB650100]00 <1> cmp byte [setfmod], 0

7099 0000EA47 0F8671DCFFFF <1> jna error

7100 <1>

7101 0000EA4D E8AF030000 <1> call update\_file\_lmdt ; update last modif. date&time of the file

7102 <1> ;mov byte [setfmod], 0

7103 <1>

7104 0000EA52 E967DCFFFF <1> jmp error

7105 <1>

7106 <1> dskw\_2: ; 3:

7107 <1> ; 23/10/2016

7108 0000EA57 C605[8C650100]01 <1> mov byte [writei.valid], 1 ; writei buffer contains valid data

7109 0000EA5E 56 <1> push esi ; logical dos drive description table address

7110 <1> ; EAX (r1) = block/sector number

7111 <1> ;call wslot

7112 <1> ; jsr r0,wslot / set write and inhibit bits in I/O queue,

7113 <1> ; / proc. status=0, r5 points to 1st word of data

7114 0000EA5F 803D[C6030300]00 <1> cmp byte [u.kcall], 0

7115 0000EA66 770F <1> ja short dskw\_4 ; zf=0 -> the caller is 'mkdir'

7116 <1> ;

7117 0000EA68 66833D[C4030300]00 <1> cmp word [u.pcount], 0

7118 0000EA70 7705 <1> ja short dskw\_4

7119 <1> dskw\_3:

7120 <1> ; [u.base] = virtual address to transfer (as source address)

7121 0000EA72 E821FAFFFF <1> call trans\_addr\_r ; translate virtual address to physical (r)

7122 <1> dskw\_4:

7123 0000EA77 BB[94070300] <1> mov ebx, writei\_buffer

7124 <1> ; EBX (r5) = system (I/O) buffer address

7125 0000EA7C E883FAFFFF <1> call sioreg

7126 <1> ; ESI = file (user data) offset

7127 <1> ; EDI = sector (I/O) buffer offset

7128 <1> ; ECX = byte count

7129 <1> ;

7130 0000EA81 F3A4 <1> rep movsb

7131 <1> ; 25/07/2015

7132 <1> ; eax = remain bytes in buffer

7133 <1> ; (check if remain bytes in the buffer > [u.pcount])

7134 0000EA83 09C0 <1> or eax, eax

7135 0000EA85 75EB <1> jnz short dskw\_3 ; (page end before system buffer end!)

7136 <1>

7137 <1> ; 23/10/2016

7138 0000EA87 B101 <1> mov cl, 1

7139 0000EA89 5E <1> pop esi

7140 0000EA8A A1[90650100] <1> mov eax, [writei.sector]

7141 <1> ; esi = logical dos drive description table address

7142 <1> ; eax = sector number

7143 <1> ; ebx = writei buffer address

7144 <1> ; ecx = sector count

7145 0000EA8F E8350D0000 <1> call disk\_write ; / yes, write the block

7146 0000EA94 7307 <1> jnc short dskw\_5

7147 <1>

7148 0000EA96 B812000000 <1> mov eax, 18 ; drive not ready or WRITE ERROR !

7149 0000EA9B EB99 <1> jmp short dskw\_err

7150 <1>

7151 <1> dskw\_5:

7152 <1> ; 26/10/2016

7153 0000EA9D 0FB61D[B0650100] <1> movzx ebx, byte [writei.ofn] ; open file number

7154 0000EAA4 C0E302 <1> shl bl, 2 ; \*4

7155 0000EAA7 8B83[80690100] <1> mov eax, [ebx+OF\_POINTER]

7156 0000EAAD 3B83[A8690100] <1> cmp eax, [ebx+OF\_SIZE]

7157 0000EAB3 7606 <1> jna short dskw\_6

7158 0000EAB5 8983[A8690100] <1> mov [ebx+OF\_SIZE], eax

7159 <1> dskw\_6:

7160 <1> ;shr bl, 2

7161 0000EABB 833D[88030300]00 <1> cmp dword [u.count], 0 ; / any more data to write?

7162 0000EAC2 760A <1> jna short dskw\_7

7163 0000EAC4 A1[A0650100] <1> mov eax, [writei.fclust]

7164 0000EAC9 E931FFFFFF <1> jmp dskw\_0 ; / yes, branch

7165 <1> dskw\_7:

7166 <1> ; update last modif. date&time of the file

7167 <1> ; (also updates file size as OF\_SIZE)

7168 0000EACE E82E030000 <1> call update\_file\_lmdt

7169 <1> ;mov byte [setfmod], 0

7170 <1>

7171 <1> ; 03/08/2013

7172 0000EAD3 C605[C6030300]00 <1> mov byte [u.kcall], 0

7173 <1> ; 23/10/2016

7174 <1> ;mov eax, [writei.fclust]

7175 0000EADA C3 <1> retn

7176 <1>

7177 <1> mget\_w:

7178 <1> ; 02/11/2016

7179 <1> ; 01/11/2016

7180 <1> ; 23/10/2016, 31/10/2016

7181 <1> ; 22/10/2016 - TRDOS 386 (TRDOS v2.0)

7182 <1> ; 03/06/2015 (Retro UNIX 386 v1, 'mget', u.5s)

7183 <1> ; 22/03/2013 - 31/07/2013 (Retro UNIX 8086 v1)

7184 <1> ;

7185 <1> ; Get existing or (allocate) a new disk block for file

7186 <1> ;

7187 <1> ; INPUTS ->

7188 <1> ; [u.fofp] = file offset pointer

7189 <1> ; [i.size] = file size

7190 <1> ; [u.count] = byte count

7191 <1> ; EAX = First cluster

7192 <1> ; [cdev] = Logical dos drive number

7193 <1> ; [writei.ofn] = File Number

7194 <1> ; (Open file index, 0 based)

7195 <1> ; ([u.off] = file offset)

7196 <1> ; OUTPUTS ->

7197 <1> ; EAX = logical sector number

7198 <1> ; ESI = Logical Dos Drive Description Table address

7199 <1> ;

7200 <1> ; Modified registers: EDX, EBX, ECX, ESI, EDI, EBP

7201 <1>

7202 0000EADB 8B35[74030300] <1> mov esi, [u.fofp]

7203 0000EAE1 8B2E <1> mov ebp, [esi] ; u.off (or EBX\*4+OF\_POINTER)

7204 <1>

7205 0000EAE3 29C9 <1> sub ecx, ecx

7206 0000EAE5 8A2D[46030300] <1> mov ch, [cdev]

7207 <1>

7208 0000EAEB BE00010900 <1> mov esi, Logical\_DOSDisks

7209 0000EAF0 01CE <1> add esi, ecx

7210 <1>

7211 <1> ; 31/10/2016

7212 0000EAF2 89C3 <1> mov ebx, eax ; First Cluster or FDT address

7213 <1>

7214 0000EAF4 807E0300 <1> cmp byte [esi+LD\_FATType], 0

7215 0000EAF8 0F86DD010000 <1> jna mget\_w\_14 ; Singlix FS

7216 <1>

7217 0000EAFE 0FB74611 <1> movzx eax, word [esi+LD\_BPB+BytesPerSec]

7218 0000EB02 0FB65613 <1> movzx edx, byte [esi+LD\_BPB+SecPerClust]

7219 0000EB06 8815[8E650100] <1> mov [writei.spc], dl ; sectors per cluster

7220 0000EB0C F7E2 <1> mul edx

7221 <1> ; edx = 0

7222 <1> ; eax = bytes per cluster (<= 65536)

7223 <1>

7224 <1> ; 02/11/2016

7225 0000EB0E 89C1 <1> mov ecx, eax

7226 0000EB10 48 <1> dec eax

7227 0000EB11 66A3[94650100] <1> mov [writei.bpc], ax

7228 <1>

7229 0000EB17 89E8 <1> mov eax, ebp

7230 0000EB19 0305[88030300] <1> add eax, [u.count] ; next file position

7231 0000EB1F 3B05[55040300] <1> cmp eax, [i.size] ; <= file size ?

7232 0000EB25 0F86FC000000 <1> jna mget\_w\_4 ; no

7233 <1>

7234 0000EB2B F7F1 <1> div ecx

7235 0000EB2D A3[9C650100] <1> mov [writei.c\_index], eax ; cluster index

7236 <1> ; edx = byte offset in cluster (<= 65535)

7237 <1> ;mov [writei.offset], dx

7238 <1> ;shr dx, 9 ; / 512

7239 <1> ;mov [writei.s\_index], dl ; sector index in cluster (0 to spc -1)

7240 <1>

7241 0000EB32 29D2 <1> sub edx, edx ; 01/11/2016

7242 0000EB34 8915[90650100] <1> mov [writei.sector], edx ; 0

7243 0000EB3A 668915[96650100] <1> mov [writei.offset], dx ; byte offset in cluster

7244 0000EB41 8815[8F650100] <1> mov [writei.s\_index], dl ; sector index in cluster (0 to spc -1)

7245 <1>

7246 0000EB47 89D8 <1> mov eax, ebx ; First Cluster

7247 <1>

7248 <1> ; is this the 1st mget\_w or a next mget\_w call ? (by 'writei')

7249 0000EB49 3815[8C650100] <1> cmp byte [writei.valid], dl ; 0

7250 0000EB4F 7624 <1> jna short mget\_w\_0

7251 <1>

7252 0000EB51 8815[8C650100] <1> mov byte [writei.valid], dl ; 0 ; reset ('writei' will set it)

7253 <1>

7254 0000EB57 3B05[A0650100] <1> cmp eax, [writei.fclust]

7255 0000EB5D 7516 <1> jne short mget\_w\_0

7256 <1>

7257 0000EB5F 8A0D[46030300] <1> mov cl, [cdev]

7258 0000EB65 3A0D[8D650100] <1> cmp cl, [writei.drv]

7259 0000EB6B 7508 <1> jne short mget\_w\_0

7260 <1> ; [writei.l\_clust] & [writei.l\_index] are valid,

7261 <1> ; we don't need to get last cluster & last cluster index

7262 0000EB6D 8B0D[AC650100] <1> mov ecx, [writei.l\_index]

7263 0000EB73 EB64 <1> jmp short mget\_w\_2

7264 <1> mget\_w\_0:

7265 0000EB75 A3[A0650100] <1> mov [writei.fclust], eax ; first cluster

7266 <1> ; edx = 0

7267 0000EB7A A3[98650100] <1> mov [writei.cluster], eax ; first cluster ; 01/11/2016

7268 0000EB7F 8915[A4650100] <1> mov [writei.fs\_index], edx ; 0 ; curret cluster index

7269 <1>

7270 <1> ; FAT file system (FAT12, FAT16, FAT32)

7271 0000EB85 E8B9D7FFFF <1> call get\_last\_cluster

7272 0000EB8A 0F822B010000 <1> jc mget\_w\_err ; eax = error code

7273 <1>

7274 0000EB90 A3[A8650100] <1> mov [writei.lclust], eax ; last cluster

7275 <1>

7276 0000EB95 8B0D[CC630100] <1> mov ecx, [glc\_index] ; last cluster index

7277 0000EB9B 890D[AC650100] <1> mov [writei.l\_index], ecx

7278 <1>

7279 0000EBA1 A0[B0650100] <1> mov al, [writei.ofn]

7280 0000EBA6 FEC0 <1> inc al

7281 0000EBA8 A2[EB650100] <1> mov [setfmod], al ; update lm date&time sign

7282 <1>

7283 <1> mget\_w\_1:

7284 0000EBAD 3B0D[9C650100] <1> cmp ecx, [writei.c\_index] ; last cluster index

7285 0000EBB3 7324 <1> jnb short mget\_w\_2 ; 01/11/2016

7286 <1>

7287 0000EBB5 A1[A8650100] <1> mov eax, [writei.lclust]

7288 <1> ; EAX = Last cluster

7289 0000EBBA E892D8FFFF <1> call add\_new\_cluster

7290 0000EBBF 0F82F6000000 <1> jc mget\_w\_err ; eax = error code

7291 <1> ; edx = 0

7292 0000EBC5 A3[A8650100] <1> mov [writei.lclust], eax ; (new) last cluster

7293 0000EBCA 8B0D[AC650100] <1> mov ecx, [writei.l\_index]

7294 0000EBD0 41 <1> inc ecx ; add 1 to last cluster index

7295 0000EBD1 890D[AC650100] <1> mov [writei.l\_index], ecx ; current last cluster index

7296 <1>

7297 0000EBD7 EBD4 <1> jmp short mget\_w\_1

7298 <1>

7299 <1> mget\_w\_2:

7300 0000EBD9 89E9 <1> mov ecx, ebp

7301 0000EBDB 030D[88030300] <1> add ecx, [u.count]

7302 0000EBE1 890D[55040300] <1> mov [i.size], ecx ; save new file size

7303 <1> ;sub edx, edx ; 0

7304 <1>

7305 0000EBE7 A0[46030300] <1> mov al, [cdev]

7306 0000EBEC A2[8D650100] <1> mov [writei.drv], al ; physical drive number

7307 <1> ; edx = 0

7308 0000EBF1 89E8 <1> mov eax, ebp ; file offset

7309 0000EBF3 0FB70D[94650100] <1> movzx ecx, word [writei.bpc] ; bytes per cluster - 1

7310 0000EBFA 41 <1> inc ecx ; bytes per cluster

7311 0000EBFB F7F1 <1> div ecx

7312 <1> ; edx = byte offset in cluster (<= 65535)

7313 <1> ; eax = cluster index

7314 0000EBFD A3[9C650100] <1> mov [writei.c\_index], eax

7315 0000EC02 668915[96650100] <1> mov [writei.offset], dx

7316 0000EC09 66C1EA09 <1> shr dx, 9 ; / 512

7317 0000EC0D 8815[8F650100] <1> mov [writei.s\_index], dl ; sector index in cluster (0 to spc -1)

7318 <1>

7319 <1> mget\_w\_3:

7320 0000EC13 3B05[AC650100] <1> cmp eax, [writei.l\_index] ; last cluster index

7321 0000EC19 752A <1> jne short mget\_w\_5

7322 <1>

7323 0000EC1B A3[A4650100] <1> mov [writei.fs\_index], eax ; cluster index (for next check)

7324 0000EC20 A1[A8650100] <1> mov eax, [writei.lclust] ; last cluster

7325 0000EC25 EB60 <1> jmp short mget\_w\_10

7326 <1>

7327 <1> mget\_w\_4: ; 02/11/2016

7328 <1> ; eax = next file position

7329 0000EC27 2B05[88030300] <1> sub eax, [u.count] ; current file position

7330 <1> ; edx = 0

7331 <1> ; ecx = bytes per cluster

7332 0000EC2D F7F1 <1> div ecx

7333 0000EC2F A3[9C650100] <1> mov [writei.c\_index], eax ; cluster index

7334 0000EC34 668915[96650100] <1> mov [writei.offset], dx

7335 0000EC3B 66C1EA09 <1> shr dx, 9 ; / 512

7336 0000EC3F 8815[8F650100] <1> mov [writei.s\_index], dl ; sector index in cluster (0 to spc -1)

7337 <1>

7338 <1> mget\_w\_5:

7339 0000EC45 21C0 <1> and eax, eax ; 0 = First Cluster's index number

7340 0000EC47 750C <1> jnz short mget\_w\_6

7341 <1>

7342 0000EC49 A3[A4650100] <1> mov [writei.fs\_index], eax ; cluster index (for next check)

7343 0000EC4E A1[A0650100] <1> mov eax, [writei.fclust] ; first cluster

7344 0000EC53 EB32 <1> jmp short mget\_w\_10

7345 <1>

7346 <1> mget\_w\_6:

7347 0000EC55 3B05[A4650100] <1> cmp eax, [writei.fs\_index] ; current cluster index (>0)

7348 0000EC5B 7507 <1> jne short mget\_w\_7

7349 0000EC5D A1[98650100] <1> mov eax, [writei.cluster] ; current cluster

7350 0000EC62 EB3A <1> jmp short mget\_w\_11

7351 <1>

7352 <1> mget\_w\_7:

7353 0000EC64 89C1 <1> mov ecx, eax

7354 0000EC66 2B0D[A4650100] <1> sub ecx, [writei.fs\_index]

7355 0000EC6C 730D <1> jnc short mget\_w\_8

7356 <1> ; get cluster by index from the first cluster

7357 0000EC6E A1[A0650100] <1> mov eax, [writei.fclust]

7358 0000EC73 8B0D[9C650100] <1> mov ecx, [writei.c\_index]

7359 0000EC79 EB05 <1> jmp short mget\_w\_9

7360 <1>

7361 <1> mget\_w\_8:

7362 0000EC7B A1[98650100] <1> mov eax, [writei.cluster] ; beginning cluster

7363 <1> ; ecx = cluster sequence number after the beginning cluster

7364 <1> ; sub edx, edx ; 0

7365 <1>

7366 <1> mget\_w\_9:

7367 <1> ; EAX = Beginning cluster

7368 <1> ; EDX = Sector index in disk/file section

7369 <1> ; (Only for SINGLIX file system!)

7370 <1> ; ECX = Cluster sequence number after the beginning cluster

7371 <1> ; ESI = Logical DOS Drive Description Table address

7372 0000EC80 E8D2D8FFFF <1> call get\_cluster\_by\_index

7373 0000EC85 7234 <1> jc short mget\_w\_err ; error code in EAX

7374 <1> ; EAX = Cluster number

7375 <1> mget\_w\_10:

7376 0000EC87 A3[98650100] <1> mov [writei.cluster], eax ; FDT number for Singlix File System

7377 <1>

7378 0000EC8C 807E0300 <1> cmp byte [esi+LD\_FATType], 0

7379 0000EC90 7638 <1> jna short mget\_w\_13

7380 <1> ; 01/11/2016

7381 0000EC92 8B15[9C650100] <1> mov edx, [writei.c\_index]

7382 0000EC98 8915[A4650100] <1> mov [writei.fs\_index], edx

7383 <1> mget\_w\_11:

7384 0000EC9E 83E802 <1> sub eax, 2

7385 0000ECA1 0FB615[8E650100] <1> movzx edx, byte [writei.spc]

7386 0000ECA8 F7E2 <1> mul edx

7387 <1>

7388 0000ECAA 034668 <1> add eax, [esi+LD\_DATABegin]

7389 0000ECAD 8A15[8F650100] <1> mov dl, [writei.s\_index]

7390 0000ECB3 01D0 <1> add eax, edx

7391 <1> mget\_w\_12:

7392 0000ECB5 A3[90650100] <1> mov [writei.sector], eax

7393 <1> ;; buffer validation must be done in writei

7394 <1> ;;mov byte [writei.valid], 1

7395 0000ECBA C3 <1> retn

7396 <1>

7397 <1> mget\_w\_err:

7398 0000ECBB A3[C8030300] <1> mov [u.error], eax

7399 0000ECC0 A3[64030300] <1> mov [u.r0], eax

7400 0000ECC5 E9F4D9FFFF <1> jmp error

7401 <1>

7402 <1> mget\_w\_13:

7403 <1> ; EAX = FDT number (Current Section)

7404 <1> ; EDX = Sector index from the first section (0,1,2,3,4...)

7405 0000ECCA 2B15[A4650100] <1> sub edx, [writei.fs\_index]

7406 <1> ; EDX = Sector index from current section

7407 0000ECD0 8915[A4650100] <1> mov [writei.fs\_index], edx

7408 0000ECD6 40 <1> inc eax ; the first data sector in FS disk section

7409 0000ECD7 01D0 <1> add eax, edx

7410 0000ECD9 EBDA <1> jmp short mget\_w\_12

7411 <1>

7412 <1> mget\_w\_14:

7413 0000ECDB 8A4E12 <1> mov cl, [esi+LD\_FS\_BytesPerSec+1]

7414 0000ECDE D0E9 <1> shr cl, 1 ; ; 1 for 512 bytes, 4 for 2048 bytes

7415 0000ECE0 880D[8E650100] <1> mov [writei.spc], cl ; sectors per cluster

7416 <1> ; NOTE: writei bytes per sector value is always 512 !

7417 0000ECE6 66C705[94650100]00- <1> mov word [writei.bpc], 512

7417 0000ECEE 02 <1>

7418 <1>

7419 0000ECEF 89E9 <1> mov ecx, ebp

7420 0000ECF1 030D[88030300] <1> add ecx, [u.count] ; next file position

7421 0000ECF7 3B0D[55040300] <1> cmp ecx, [i.size] ; <= file size ?

7422 0000ECFD 0F86C8000000 <1> jna mget\_w\_19 ; no

7423 <1>

7424 0000ED03 29D2 <1> sub edx, edx ; 0

7425 0000ED05 8915[90650100] <1> mov [writei.sector], edx ; 0

7426 0000ED0B 668915[96650100] <1> mov [writei.offset], dx ; byte offset in cluster

7427 0000ED12 8815[8F650100] <1> mov [writei.s\_index], dl ; sector index in cluster (0 to spc -1)

7428 <1>

7429 0000ED18 C1E909 <1> shr ecx, 9 ; 1 cluster = 512 bytes

7430 0000ED1B 890D[9C650100] <1> mov [writei.c\_index], ecx ; section/cluster index

7431 <1>

7432 0000ED21 89D8 <1> mov eax, ebx ; FDT number (First FDT address)

7433 <1>

7434 <1> ; is this the 1st mget\_w or a next mget\_w call ? (by 'writei')

7435 0000ED23 3815[8C650100] <1> cmp byte [writei.valid], dl ; 0

7436 0000ED29 7624 <1> jna short mget\_w\_15

7437 <1>

7438 0000ED2B 8815[8C650100] <1> mov byte [writei.valid], dl ; 0 ; reset ('writei' will set it)

7439 <1>

7440 0000ED31 3B05[A0650100] <1> cmp eax, [writei.fclust]

7441 0000ED37 7516 <1> jne short mget\_w\_15

7442 <1>

7443 0000ED39 8A0D[46030300] <1> mov cl, [cdev]

7444 0000ED3F 3A0D[8D650100] <1> cmp cl, [writei.drv]

7445 0000ED45 7508 <1> jne short mget\_w\_15

7446 <1> ; [writei.l\_clust] & [writei.l\_index] are valid,

7447 <1> ; we don't need to get last cluster & last cluster index

7448 0000ED47 8B0D[AC650100] <1> mov ecx, [writei.l\_index]

7449 0000ED4D EB49 <1> jmp short mget\_w\_17

7450 <1> mget\_w\_15:

7451 0000ED4F A3[A0650100] <1> mov [writei.fclust], eax ; first section (FDT number)

7452 <1> ; edx = 0

7453 0000ED54 8915[98650100] <1> mov [writei.cluster], edx ; 0 ; current section

7454 0000ED5A 8915[A4650100] <1> mov [writei.fs\_index], edx ; 0 ; curret section index

7455 <1>

7456 <1> ; eax = FDT number (section 0 header address)

7457 0000ED60 E81CD8FFFF <1> call get\_last\_section

7458 0000ED65 0F8250FFFFFF <1> jc mget\_w\_err ; eax = error code

7459 <1>

7460 0000ED6B 8915[A4650100] <1> mov [writei.fs\_index], edx ; sector index in last section

7461 <1>

7462 0000ED71 A3[A8650100] <1> mov [writei.lclust], eax ; last section address

7463 <1>

7464 0000ED76 8B0D[CC630100] <1> mov ecx, [glc\_index] ; last section index

7465 0000ED7C 890D[AC650100] <1> mov [writei.l\_index], ecx

7466 <1>

7467 0000ED82 A0[B0650100] <1> mov al, [writei.ofn]

7468 0000ED87 FEC0 <1> inc al

7469 0000ED89 A2[EB650100] <1> mov [setfmod], al ; update lm date&time sign

7470 <1>

7471 <1> mget\_w\_16:

7472 <1> ; edx = (existing) last section (sector) index

7473 0000ED8E 8B0D[9C650100] <1> mov ecx, [writei.c\_index] ; final section (sector) index

7474 0000ED94 29D1 <1> sub ecx, edx

7475 0000ED96 7633 <1> jna short mget\_w\_19

7476 <1> ; ecx = sector count

7477 <1> mget\_w\_17:

7478 0000ED98 A1[A8650100] <1> mov eax, [writei.lclust]

7479 <1> ; ESI = Logical dos drv desc. table address

7480 <1> ; EAX = Last section

7481 <1> ; (ECX = 0 for directory)

7482 <1> ; ECX = sector count (except FDT)

7483 0000ED9D E8A2CDFFFF <1> call add\_new\_fs\_section

7484 0000EDA2 7312 <1> jnc short mget\_w\_18

7485 <1>

7486 <1> ; If error number = 27h (insufficient disk space)

7487 <1> ; it is needed to check free consequent sectors

7488 <1> ; (1 data sector at least and +1 section header sector)

7489 <1>

7490 0000EDA4 83F827 <1> cmp eax, 27h

7491 0000EDA7 0F850EFFFFFF <1> jne mget\_w\_err ; eax = error code

7492 <1>

7493 <1> ; ecx = count of free consequent sectors

7494 <1> ; ecx must be > 1 (1 data + 1 header sector)

7495 0000EDAD 49 <1> dec ecx

7496 0000EDAE 0F8407FFFFFF <1> jz mget\_w\_err

7497 0000EDB4 EBE2 <1> jmp short mget\_w\_17

7498 <1>

7499 <1> mget\_w\_18:

7500 0000EDB6 A3[A8650100] <1> mov [writei.lclust], eax ; (new) last section

7501 <1> ; ecx = sector count (except section header)

7502 0000EDBB 8B15[AC650100] <1> mov edx, [writei.l\_index]

7503 0000EDC1 01CA <1> add edx, ecx ; add sector count to index

7504 0000EDC3 8915[AC650100] <1> mov [writei.l\_index], edx

7505 0000EDC9 EBC3 <1> jmp short mget\_w\_16

7506 <1>

7507 <1> mget\_w\_19:

7508 0000EDCB 89E9 <1> mov ecx, ebp

7509 0000EDCD 030D[88030300] <1> add ecx, [u.count]

7510 0000EDD3 890D[55040300] <1> mov [i.size], ecx ; save new file size

7511 <1> ;sub edx, edx ; 0

7512 <1>

7513 0000EDD9 A0[46030300] <1> mov al, [cdev]

7514 0000EDDE A2[8D650100] <1> mov [writei.drv], al ; physical drive number

7515 <1> ; edx = 0

7516 0000EDE3 89E8 <1> mov eax, ebp ; file offset

7517 0000EDE5 89C2 <1> mov edx, eax

7518 <1> ; 1 cluster = 512 bytes (for Singlix FS)

7519 0000EDE7 C1E809 <1> shr eax, 9 ; / 512

7520 0000EDEA 81E2FF010000 <1> and edx, 1FFh

7521 <1> ; edx = byte offset in cluster/sector (<= 511)

7522 <1> ; eax = section (sector/cluster) index

7523 0000EDF0 A3[9C650100] <1> mov [writei.c\_index], eax

7524 0000EDF5 668915[96650100] <1> mov [writei.offset], dx

7525 <1> ;mov byte [writei.s\_index], 0 ; sector index in cluster

7526 0000EDFC E912FEFFFF <1> jmp mget\_w\_3

7527 <1>

7528 <1> update\_file\_lmdt: ; & update file size

7529 <1> ; 26/10/2016

7530 <1> ; 24/10/2016

7531 <1> ; 23/10/2016

7532 <1> ; 22/10/2016 - TRDOS 386 (TRDOS v2.0)

7533 <1> ;

7534 <1> ; Update last modification date&time of file

7535 <1> ; (call from syswrite -> writei)

7536 <1> ; ((also updates file size)) // 26/10/2016

7537 <1> ;

7538 <1> ; INPUT:

7539 <1> ; byte [setfmod] = open file number

7540 <1> ; OUTPUT:

7541 <1> ; cf = 0 -> success !

7542 <1> ; cf = 1 -> lmdt update has been failed!

7543 <1> ;

7544 <1> ; Modified registers: eax, ebx, ecx, edx, esi, edi

7545 <1> ;

7546 <1>

7547 <1> ;cmp byte [setfmod], 0

7548 <1> ;jna short uflmdt\_2 ; nothing to do

7549 <1>

7550 0000EE01 31C0 <1> xor eax, eax

7551 <1>

7552 0000EE03 0FB61D[EB650100] <1> movzx ebx, byte [setfmod]

7553 0000EE0A FECB <1> dec bl ; open file index number (0 based)

7554 <1>

7555 0000EE0C 8AA3[58690100] <1> mov ah, [ebx+OF\_DRIVE]

7556 0000EE12 BE00010900 <1> mov esi, Logical\_DOSDisks

7557 0000EE17 01C6 <1> add esi, eax

7558 0000EE19 C0E302 <1> shl bl, 2 ; \*4

7559 0000EE1C 8B8B[30690100] <1> mov ecx, [ebx+OF\_FCLUSTER] ; first cluster

7560 0000EE22 8B93[F8690100] <1> mov edx, [ebx+OF\_DIRCLUSTER] ; dir cluster

7561 <1>

7562 0000EE28 D0EB <1> shr bl, 1 ; /2

7563 0000EE2A 0FB7BB[986A0100] <1> movzx edi, word [ebx+OF\_DIRENTRY]

7564 <1>

7565 0000EE31 803D[28610100]01 <1> cmp byte [DirBuff\_ValidData], 1

7566 0000EE38 726E <1> jb short uflmdt\_4

7567 <1>

7568 0000EE3A A0[26610100] <1> mov al, [DirBuff\_DRV]

7569 0000EE3F 2C41 <1> sub al, 'A'

7570 0000EE41 38E0 <1> cmp al, ah

7571 0000EE43 7563 <1> jne short uflmdt\_4 ; different drive

7572 0000EE45 8A4603 <1> mov al, [esi+LD\_FATType]

7573 0000EE48 3A05[27610100] <1> cmp al, [DirBuff\_FATType]

7574 0000EE4E 755B <1> jne short uflmdt\_5 ; different FS type

7575 0000EE50 3B15[2D610100] <1> cmp edx, [DirBuff\_Cluster]

7576 0000EE56 7553 <1> jne short uflmdt\_5 ; different cluster

7577 <1>

7578 <1> uflmdt\_1:

7579 <1> ; Directory buffer is ready here!

7580 <1> ; OF\_FCLUSTER must be compared/verified

7581 0000EE58 BE00000800 <1> mov esi, Directory\_Buffer

7582 0000EE5D 66C1E705 <1> shl di, 5 ; dir entry index \* 32

7583 0000EE61 01FE <1> add esi, edi ; offset

7584 <1> ;

7585 0000EE63 F6460B18 <1> test byte [esi+DirEntry\_Attr], 18h ; Vol & Dir

7586 0000EE67 750F <1> jnz short uflmdt\_2 ; not a valid file !

7587 0000EE69 668B4614 <1> mov ax, [esi+DirEntry\_FstClusHI]

7588 0000EE6D C1E010 <1> shl eax, 16

7589 0000EE70 668B461A <1> mov ax, [esi+DirEntry\_FstClusLO]

7590 0000EE74 39C8 <1> cmp eax, ecx ; same first cluster ?

7591 0000EE76 7407 <1> je short uflmdt\_3 ; yes, it is OK !!!

7592 <1>

7593 <1> uflmdt\_2:

7594 <1> ; save directory buffer if has modified/changed sign

7595 <1> ; (It is good to save dir buff even if the searched

7596 <1> ; directory entry is not found !?)

7597 0000EE78 E81BBAFFFF <1> call save\_directory\_buffer

7598 0000EE7D F9 <1> stc ; update failed

7599 0000EE7E C3 <1> retn

7600 <1>

7601 <1> uflmdt\_3:

7602 <1> ; Update directory entry

7603 <1> ; 26/10/2016

7604 0000EE7F D0E3 <1> shl bl, 1 ; \*2

7605 0000EE81 8B83[A8690100] <1> mov eax, [ebx+OF\_SIZE] ; file size

7606 0000EE87 89461C <1> mov [esi+DirEntry\_FileSize], eax

7607 <1> ;

7608 0000EE8A E86BB9FFFF <1> call convert\_current\_date\_time

7609 <1> ; OUTPUT -> DX = Date in dos dir entry format

7610 <1> ; AX = Time in dos dir entry format

7611 0000EE8F 66894616 <1> mov [esi+DirEntry\_WrtTime], ax

7612 0000EE93 66895618 <1> mov [esi+DirEntry\_WrtDate], dx

7613 0000EE97 66895612 <1> mov [esi+DirEntry\_LastAccDate], dx

7614 0000EE9B C605[28610100]02 <1> mov byte [DirBuff\_ValidData], 2

7615 0000EEA2 E8F1B9FFFF <1> call save\_directory\_buffer

7616 0000EEA7 C3 <1> retn

7617 <1>

7618 <1> uflmdt\_4:

7619 <1> ; Directory buffer sector read&write

7620 <1> ; 23/10/2016

7621 <1> ;

7622 0000EEA8 8A4603 <1> mov al, [esi+LD\_FATType]

7623 <1> uflmdt\_5:

7624 0000EEAB BB[9C090300] <1> mov ebx, rw\_buffer ; Common r/w sector buffer addr

7625 <1>

7626 0000EEB0 20C0 <1> and al, al ; 0 = Singlix FS

7627 0000EEB2 0F8492000000 <1> jz uflmdt\_11

7628 <1>

7629 0000EEB8 21D2 <1> and edx, edx

7630 0000EEBA 7521 <1> jnz short uflmdt\_9

7631 <1>

7632 0000EEBC 3C02 <1> cmp al, 2 ; 3 = FAT32

7633 0000EEBE 771A <1> ja short uflmdt\_8

7634 <1>

7635 0000EEC0 89F8 <1> mov eax, edi ; directory entry index number

7636 0000EEC2 66C1E804 <1> shr ax, 4 ; 16 entries per sector

7637 0000EEC6 034664 <1> add eax, [esi+LD\_ROOTBegin]

7638 <1> ; eax = root directory sector

7639 <1> uflmdt\_6:

7640 0000EEC9 50 <1> push eax ; \* ; disk sector address

7641 0000EECA 51 <1> push ecx ; first cluster

7642 0000EECB B901000000 <1> mov ecx, 1

7643 <1> ; ecx = sector count

7644 0000EED0 E803090000 <1> call disk\_read

7645 0000EED5 59 <1> pop ecx

7646 0000EED6 731A <1> jnc short uflmdt\_10

7647 0000EED8 58 <1> pop eax ; \*

7648 <1> uflmdt\_7:

7649 0000EED9 C3 <1> retn

7650 <1>

7651 <1> uflmdt\_8:

7652 0000EEDA 8B5632 <1> mov edx, [esi+LD\_BPB+FAT32\_RootFClust]

7653 <1> uflmdt\_9:

7654 0000EEDD 83FA02 <1> cmp edx, 2

7655 0000EEE0 72F7 <1> jb short uflmdt\_7 ; invalid, nothing to do

7656 <1>

7657 0000EEE2 83EA02 <1> sub edx, 2

7658 0000EEE5 89D0 <1> mov eax, edx

7659 0000EEE7 0FB65613 <1> movzx edx, byte [esi+LD\_BPB+SecPerClust]

7660 0000EEEB F7E2 <1> mul edx

7661 0000EEED 034668 <1> add eax, [esi+LD\_DATABegin]

7662 <1> ; eax = sub directory (data) sector

7663 0000EEF0 EBD7 <1> jmp short uflmdt\_6

7664 <1>

7665 <1> uflmdt\_10:

7666 <1> ; Directory sector buffer is ready here!

7667 <1> ; OF\_FCLUSTER must be compared/verified

7668 <1> ; edi = dir entry index number (<= 2047)

7669 0000EEF2 6683E70F <1> and di, 0Fh ; 16 entries per sector

7670 0000EEF6 66C1E705 <1> shl di, 5 ; dir entry index \* 32

7671 0000EEFA 81C7[9C090300] <1> add edi, rw\_buffer

7672 <1> ;

7673 0000EF00 F6470B18 <1> test byte [edi+DirEntry\_Attr], 18h ; Vol & Dir

7674 0000EF04 0F856EFFFFFF <1> jnz uflmdt\_2 ; not a valid file !

7675 0000EF0A 668B5714 <1> mov dx, [edi+DirEntry\_FstClusHI]

7676 0000EF0E C1E210 <1> shl edx, 16

7677 0000EF11 668B571A <1> mov dx, [edi+DirEntry\_FstClusLO]

7678 0000EF15 39CA <1> cmp edx, ecx ; same first cluster ?

7679 0000EF17 0F855BFFFFFF <1> jne uflmdt\_2 ; no !?

7680 <1>

7681 <1> ; Update directory entry

7682 0000EF1D E8D8B8FFFF <1> call convert\_current\_date\_time

7683 <1> ; OUTPUT -> DX = Date in dos dir entry format

7684 <1> ; AX = Time in dos dir entry format

7685 0000EF22 66894716 <1> mov [edi+DirEntry\_WrtTime], ax

7686 0000EF26 66895718 <1> mov [edi+DirEntry\_WrtDate], dx

7687 0000EF2A 66895712 <1> mov [edi+DirEntry\_LastAccDate], dx

7688 <1>

7689 0000EF2E 58 <1> pop eax ; \*

7690 <1>

7691 0000EF2F BB[9C090300] <1> mov ebx, rw\_buffer ; Common r/w sector buffer addr

7692 0000EF34 B901000000 <1> mov ecx, 1

7693 <1> ; esi = logical dos description table address

7694 <1> ; eax = disk sector number/address (LBA)

7695 <1> ; ecx = sector count

7696 <1> ; ebx = buffer address

7697 0000EF39 E88B080000 <1> call disk\_write

7698 0000EF3E 0F8234FFFFFF <1> jc uflmdt\_2

7699 <1>

7700 <1> ; save directory buffer if has modified/changed sign

7701 0000EF44 E84FB9FFFF <1> call save\_directory\_buffer

7702 0000EF49 C3 <1> retn

7703 <1>

7704 <1> uflmdt\_11:

7705 <1> ; 24/10/2016

7706 <1> ; Update last modification date & time of a file

7707 <1> ; on a disk with Singlix File System.

7708 <1> ;

7709 <1> ; (Method: Read the FDT -File Description Table-

7710 <1> ; sector of the file and update the lmdt data fields,

7711 <1> ; then write FDT sector to the disk.

7712 <1> ; /// It is easy but there is compatibility buffer

7713 <1> ; method also for changing directory entry data and

7714 <1> ; also there are some programming issues for Singlix

7715 <1> ; file system (TRFS), which are not completed yet!)

7716 <1> ;

7717 <1> ; Not ready yet ! (24/10/2016)

7718 <1> ; /// Temporary code for error return ! ///

7719 0000EF4A 31C0 <1> xor eax, eax

7720 0000EF4C F9 <1> stc

7721 0000EF4D C3 <1> retn

7722 <1>

7723 <1> sysalloc:

7724 <1> ; 14/10/2017

7725 <1> ; 20/08/2017, 01/09/2017

7726 <1> ; 20/02/2017, 04/03/2017, 15/05/2017

7727 <1> ; 19/02/2017 - TRDOS 386 (TRDOS v2.0)

7728 <1> ; (TRDOS 386 feature only!)

7729 <1> ;

7730 <1> ; Allocate Contiguous Memory Block/Pages (for user)

7731 <1> ; (System call for DMA Buffer allocation etc.)

7732 <1> ;

7733 <1> ; INPUT ->

7734 <1> ; EBX = Virtual address (for user)

7735 <1> ; (Physical memory block/aperture

7736 <1> ; will be mapped to this virtual address)

7737 <1> ; ECX = Byte Count

7738 <1> ; (will be rounded up to page border)

7739 <1> ; If ECX = 0

7740 <1> ; System call will return with an error (cf=1)

7741 <1> ; but ECX will contain maximum size of

7742 <1> ; available memory aperture and physical

7743 <1> ; (beginning) address of that aperture

7744 <1> ; (which have maximum size) will be in EAX.

7745 <1> ; EDX = Upper limit of the requested physical memory

7746 <1> ; block/pages.

7747 <1> ; (The last byte address of the memory aperture

7748 <1> ; must not be equal to or above this limit.)

7749 <1> ; If EDX = 0

7750 <1> ; there is NOLIMIT !

7751 <1> ; If EDX = 0FFFFFFFFh (-1)

7752 <1> ; ESI = Lower Limit !

7753 <1> ; (Beginning of the block must not be 'less'

7754 <1> ; than this.) (Must be equal to or above...)

7755 <1> ; EDI = Upper Limit !

7756 <1> ; (End of the block must be !less! than this)

7757 <1> ; (The last byte addr of the memory aperture

7758 <1> ; must not be equal to or above this limit.)

7759 <1> ;

7760 <1> ; OUTPUT ->

7761 <1> ; If CF = 0

7762 <1> ; EAX = Physical address of the allocated memory block

7763 <1> ; ECX = Allocated bytes (as rounded up to page borders)

7764 <1> ; EBX = Virtual address (as rounded up)

7765 <1> ; IF CF = 1

7766 <1> ; Requested (size of) Memory block could not be

7767 <1> ; allocated to the user!

7768 <1> ; IF CF = 1 & EAX = 0 (Insufficient memory error!)

7769 <1> ; ECX = Total number of free bytes

7770 <1> ; (not size of available contiguous bytes!)

7771 <1> ; If CF = 1 & EAX > 0

7772 <1> ; there is not a memory aperture with requested size

7773 <1> ; but total free mem is not less than requested size.

7774 <1> ; EAX = Physical addr of available memory aperture

7775 <1> ; with max size

7776 <1> ; (but it doesn't fit to the conditions!)

7777 <1> ; ECX = Size of available memory aperture in bytes.

7778 <1> ; If CF = 1 -> EAX = 0FFFFFFFFh

7779 <1> ; Conditions/Parameters are wrong !

7780 <1> ; ECX is same with input value.

7781 <1> ;

7782 <1> ; Note: Previously allocated pages will be deallocated if

7783 <1> ; new allocation conditions are met.

7784 <1> ;

7785 <1> ; Note: u.break control may be included in future versions

7786 <1> ;

7787 <1>

7788 0000EF4E 31C0 <1> xor eax, eax ; 0

7789 <1> ; 14/10/2017

7790 0000EF50 4A <1> dec edx ; is there a limit ?

7791 0000EF51 7810 <1> js short sysalloc\_1 ; 0 -> 0FFFFFFFFh -> NO LIMIT

7792 0000EF53 42 <1> inc edx ; > 0

7793 <1> ; Check upper address limit

7794 <1> ;(round up to page borders)

7795 0000EF54 81C1FF0F0000 <1> add ecx, PAGE\_SIZE-1 ; 4095

7796 0000EF5A 6681E100F0 <1> and cx, ~PAGE\_OFF ; not 4095

7797 0000EF5F 39CA <1> cmp edx, ecx ; upper limit - block size

7798 0000EF61 7224 <1> jb short sysalloc\_err

7799 <1> sysalloc\_1:

7800 <1> ; EAX = Beginning address (physical)

7801 <1> ; EAX = 0 -> Allocate mem block from the 1st proper aperture

7802 <1> ; ECX = Number of bytes to be allocated

7803 0000EF63 E8BC64FFFF <1> call allocate\_memory\_block

7804 0000EF68 721D <1> jc short sysalloc\_err

7805 <1> ; 01/09/2017

7806 0000EF6A 29C2 <1> sub edx, eax ; upper limit address - beginning address

7807 0000EF6C 760F <1> jna short sysalloc\_3 ; begin addr not less than the limit

7808 0000EF6E 39CA <1> cmp edx, ecx

7809 0000EF70 720B <1> jb short sysalloc\_3 ; end address overs the limit

7810 <1> sysalloc\_2:

7811 <1> ; EAX = Beginning (physical) addr of the allocated mem block

7812 <1> ; ECX = Num of allocated bytes (rounded up to page borders)

7813 0000EF72 50 <1> push eax ; \* ; 04/03/2017

7814 <1> ; Here, requested contiquous memory pages have been allocated

7815 <1> ; on Memory Allocation Table but user's page directory

7816 <1> ; and page tables have not been updated yet!

7817 0000EF73 51 <1> push ecx ; \*\*

7818 <1> ; ebx = virtual address (will be rounded up to page border)

7819 <1> ; ecx = number of bytes to be deallocated

7820 <1> ; will be adjusted to ebx+ecx round down - ebx round up

7821 0000EF74 E80668FFFF <1> call deallocate\_user\_pages

7822 0000EF79 731F <1> jnc short sysalloc\_4 ; EAX = Deallocated memory bytes

7823 0000EF7B 59 <1> pop ecx ; \*\*

7824 0000EF7C 58 <1> pop eax ; \*

7825 <1> sysalloc\_3:

7826 <1> ; error !

7827 <1> ; restore Memory Allocation Table Content

7828 0000EF7D E8AF66FFFF <1> call deallocate\_memory\_block

7829 0000EF82 31C0 <1> xor eax, eax ; 0

7830 0000EF84 48 <1> dec eax ; 0FFFFFFFFh ; 15/05/2017

7831 0000EF85 EB09 <1> jmp short sysalloc\_wrong

7832 <1> sysalloc\_err:

7833 0000EF87 8B2D[60030300] <1> mov ebp, [u.usp] ; ebp points to user's registers

7834 0000EF8D 894D18 <1> mov [ebp+24], ecx ; return to user with ecx value

7835 <1> sysalloc\_wrong:

7836 <1> ; eax = 0FFFFFFFFh

7837 0000EF90 A3[64030300] <1> mov [u.r0], eax

7838 0000EF95 E924D7FFFF <1> jmp error

7839 <1> sysalloc\_4:

7840 0000EF9A 8B2D[60030300] <1> mov ebp, [u.usp] ; ebp points to user's registers

7841 0000EFA0 894518 <1> mov [ebp+24], eax ; return to user with ecx value

7842 0000EFA3 895D10 <1> mov [ebp+16], ebx ; new value of ebx (rounded up)

7843 0000EFA6 89C1 <1> mov ecx, eax ; byte count (from 'deallocate\_user\_pages')

7844 0000EFA8 5A <1> pop edx ; \*\* ; discard (another) byte count

7845 0000EFA9 58 <1> pop eax ; \*

7846 0000EFAA A3[64030300] <1> mov [u.r0], eax ; physical address

7847 <1>

7848 0000EFAF 51 <1> push ecx ; 20/08/2017

7849 <1> ;

7850 <1> ; Write newly allocated contiguous (physical) pages

7851 <1> ; on page dir and page tables of current user/process

7852 <1> ; as PRESENT, USER, WRITABLE

7853 <1> ; (then clear allocated pages)

7854 0000EFB0 E8BF68FFFF <1> call allocate\_user\_pages

7855 <1> ;jnc sysret ; OK! return to process with success...

7856 <1>

7857 <1> ; 20/08/2017 ('sysdma' modification)

7858 0000EFB5 59 <1> pop ecx

7859 0000EFB6 A1[64030300] <1> mov eax, [u.r0] ; physical address (of the block)

7860 <1>

7861 0000EFBB 721D <1> jc short sysalloc\_6

7862 <1>

7863 0000EFBD 833D[00700100]FF <1> cmp dword [dma\_addr], 0FFFFFFFFh ; -1

7864 0000EFC4 0F8214D7FFFF <1> jb sysret

7865 <1>

7866 0000EFCA A3[00700100] <1> mov [dma\_addr], eax ; save dma address for sysdma

7867 0000EFCF 890D[04700100] <1> mov [dma\_size], ecx ; save dma buff size for sysdma

7868 <1>

7869 0000EFD5 E904D7FFFF <1> jmp sysret

7870 <1>

7871 <1> sysalloc\_6:

7872 <1> ;

7873 <1> ; unexpected error ! insufficient memory !? conflict !?

7874 <1> ; (!!?there is not a free page for a new page table?!!)

7875 <1> ; We need to terminate process with error message !!!

7876 <1> ;

7877 0000EFDA 8B2D[60030300] <1> mov ebp, [u.usp] ; ebp points to user's registers

7878 0000EFE0 8B4D18 <1> mov ecx, [ebp+24] ; byte count

7879 <1>

7880 <1> ; 20/08/2017

7881 <1> ;mov eax, [u.r0] ; physical address (of the block)

7882 <1>

7883 <1> ;

7884 <1> ; restore Memory Allocation Table Content

7885 0000EFE3 E84966FFFF <1> call deallocate\_memory\_block

7886 <1> ;

7887 0000EFE8 803D[C25E0000]03 <1> cmp byte [CRT\_MODE], 3 ; 80x25 text mode?

7888 0000EFEF 7407 <1> je short sysalloc\_7 ; yes

7889 <1> ; Current mode is VGA (or CGA graphics) mode,

7890 <1> ; We need to return to text mode for displaying

7891 <1> ; error message just before 'sysexit'.

7892 0000EFF1 B003 <1> mov al, 3

7893 0000EFF3 E86D25FFFF <1> call \_set\_mode

7894 <1> sysalloc\_7:

7895 0000EFF8 BE[34100100] <1> mov esi, beep\_Insufficient\_Memory ; error message

7896 0000EFFD E85B73FFFF <1> call print\_msg ; print/display the message

7897 0000F002 B801000000 <1> mov eax, 1 ; ax=1 is needed for 'sysexit' procedure

7898 0000F007 E959D8FFFF <1> jmp sysexit ; and terminate the process !

7899 <1>

7900 <1> sysdalloc:

7901 <1> ; 19/02/2017 - TRDOS 386 (TRDOS v2.0)

7902 <1> ; (TRDOS 386 feature only!)

7903 <1> ;

7904 <1> ; Deallocate Memory Block/Pages (for user)

7905 <1> ; (Complementary call for sysalloc.)

7906 <1> ;

7907 <1> ; INPUT ->

7908 <1> ; EBX = Virtual address (for user)

7909 <1> ; (will be rounded up to page border)

7910 <1> ; ECX = Byte Count

7911 <1> ; (will be adjusted to page borders)

7912 <1> ; If ICX = 0

7913 <1> ; nothing to do

7914 <1> ; If EBX + ECX > User's ESP

7915 <1> ; nothing to do

7916 <1> ;

7917 <1> ; Note: u.break control may be included in future versions

7918 <1> ;

7919 <1> ; OUTPUT ->

7920 <1> ; If CF = 0

7921 <1> ; EAX = Deallocated memory bytes

7922 <1> ; EBX = Virtual address (as rounded up)

7923 <1> ; IF CF = 1

7924 <1> ; EAX = 0

7925 <1> ;

7926 <1> ; Note: Main purpose of this call is to deallocate/release

7927 <1> ; previously allocated (physically) contiguous memory

7928 <1> ; pages but beginning (virtual) address may not be

7929 <1> ; followed by physically contiguous pages. So, this

7930 <1> ; system call will deallocate user's virtually

7931 <1> ; contiguous memory pages. Also, there is not any

7932 <1> ; objections to use this system call without sysalloc

7933 <1> ; system call; only possible objection is to lost data

7934 <1> ; within user's memory space, if the beginning address

7935 <1> ; and size is not proper.

7936 <1> ;

7937 <1> ; Note: Empty page tables will not be deallocated!!!

7938 <1> ; (they will be deallocated at process termination)

7939 <1> ;

7940 <1> ; Note: When the program terminates itself or when it is

7941 <1> ; terminated by operating system kernel, all allocated

7942 <1> ; memory pages will be deallocated during termination

7943 <1> ; stage. So, 'sysdalloc' is not necessary except

7944 <1> ; forgiving memory block to other programs/processes.

7945 <1> ;

7946 0000F00C 8B15[5C030300] <1> mov edx, [u.sp]

7947 0000F012 8B420C <1> mov eax, [edx+12] ; user's stack pointer

7948 0000F015 29C8 <1> sub eax, ecx ; esp - byte count

7949 0000F017 24FC <1> and al, 0FCh ; dword alignment

7950 0000F019 39D8 <1> cmp eax, ebx

7951 0000F01B 7220 <1> jb short sysdalloc\_err ; deallocation overlaps with stack

7952 <1>

7953 0000F01D 31C0 <1> xor eax, eax

7954 0000F01F 21C9 <1> and ecx, ecx

7955 0000F021 7407 <1> jz short sysdalloc\_2

7956 <1>

7957 0000F023 E85767FFFF <1> call deallocate\_user\_pages

7958 0000F028 7213 <1> jc short sysdalloc\_err

7959 <1>

7960 <1> sysdalloc\_2:

7961 0000F02A A3[64030300] <1> mov [u.r0], eax

7962 0000F02F 8B2D[60030300] <1> mov ebp, [u.usp]

7963 0000F035 895D10 <1> mov [ebp+16], ebx ; new value of ebx

7964 0000F038 E9A1D6FFFF <1> jmp sysret

7965 <1>

7966 <1> sysdalloc\_err:

7967 0000F03D A3[64030300] <1> mov [u.r0], eax ; 0

7968 0000F042 E977D6FFFF <1> jmp error

7969 <1>

7970 <1> syscalbac:

7971 <1> ; SYS CALLBACK

7972 <1> ; 16/04/2017

7973 <1> ; 14/04/2017

7974 <1> ; 13/04/2017

7975 <1> ; 28/02/2017

7976 <1> ; 26/02/2017

7977 <1> ; 24/02/2017

7978 <1> ; 21/02/2017 - TRDOS 386 (TRDOS v2.0)

7979 <1> ; (TRDOS 386 feature only!)

7980 <1> ;

7981 <1> ; Link or unlink IRQ callback service to/from user (ring 3)

7982 <1> ;

7983 <1> ; INPUT ->

7984 <1> ; BL = IRQ number (Hardware interrupt request number)

7985 <1> ; (0 t0 15 but IRQ 0,1,2,6,8,14,15 are prohibited)

7986 <1> ; IRQ numbers 3,4,5,7,9,10,11,12,13 are valid

7987 <1> ; (numbers >15 are invalid)

7988 <1> ;

7989 <1> ; BH = 0 = Unlink IRQ (in BL) from user (ring 3) service

7990 <1> ; 1 = Link IRQ by using Signal Response Byte method

7991 <1> ; 2 = Link IRQ by using Callback service method

7992 <1> ; 3 = Link IRQ by using Auto Increment S.R.B. method

7993 <1> ; >3 = invalid

7994 <1> ;

7995 <1> ; CL = Signal Return/Response Byte value

7996 <1> ;

7997 <1> ; If BH = 2, kernel will put a counter value

7998 <1> ; (into the S.R.B. addr)

7999 <1> ; between 0 to 255. (start value = CL+1)

8000 <1> ;

8001 <1> ; NOTE: counter value, for example: even and odd numbers

8002 <1> ; may be used for -audio- DMA buffer switch

8003 <1> ; within double buffer method, etc.

8004 <1> ;

8005 <1> ; EDX = Signal return (Response) byte address

8006 <1> ; - or -

8007 <1> ; Interrupt/Callback service/routine address

8008 <1> ;

8009 <1> ; (virtual address in user's memory space)

8010 <1> ;

8011 <1> ; OUTPUT ->

8012 <1> ; CF = 0 & EAX = 0 -> Successful setting

8013 <1> ; CF = 1 & EAX > 0 -> IRQ is prohibited or locked

8014 <1> ; by another process

8015 <1> ; eax = ERR\_PERM\_DENIED -> prohibited or locked

8016 <1> ; eax = ERR\_INV\_PARAMETER ->

8017 <1> ; invalid parameter/option or bad address

8018 <1> ;

8019 <1> ; NOTE: Timer callbacks are set by using 'systimer'

8020 <1> ; system call (IRQ 0, PIT and IRQ 8, RTC)

8021 <1> ;

8022 <1> ; Direct keyboard access is performed by using

8023 <1> ; Keyboard Interrupt (INT 32h)

8024 <1> ;

8025 <1> ; It is prohibited here because:

8026 <1> ; 1) Signal Response Byte method has not advantage

8027 <1> ; against INT 32h, function AH = 1. Also,

8028 <1> ; keyboard service interrupt will return with

8029 <1> ; ascii and scan codes (AL, AH) while

8030 <1> ; SRB method has only 1 byte space for ascii code

8031 <1> ; or scan code. One byte signal response is used

8032 <1> ; for ensuring very simple and very fast

8033 <1> ; virtual to physical memory address conversion

8034 <1> ; without any memory page crossover risk.

8035 <1> ; (Otherwise double page conversion or word

8036 <1> ; alignment would be needed.)

8037 <1> ; 2) Badly written user code (callback code)

8038 <1> ; can prevent keyboard and timesharing functions

8039 <1> ; of the operating system via continuous and long

8040 <1> ; keyboard event handling by callback service.

8041 <1> ; (It can cause to lose immediate keystroke

8042 <1> ; response from hardware to user.)

8043 <1> ; 3) If user will check any keyboard events, 'getkey'

8044 <1> ; (or 'getchar') must have more priority than other

8045 <1> ; (video etc.) events because only control ability

8046 <1> ; on a procedural infinite loop is a keyboard or

8047 <1> ; mouse event. So user can use keyboard function

8048 <1> ; at the end or at the beginning of a loop.

8049 <1> ; In this case, INT 32h is used for that purpose

8050 <1> ; and timer interrupt etc. callbacks can be used

8051 <1> ; for dynamic and synchronized data refresh/transfer

8052 <1> ; while cpu is in a static loop (without polling).

8053 <1> ; Keyboard Int callback is not more useful because

8054 <1> ; already a manual check (a key is pressed or not)

8055 <1> ; can be performed (via INT 32h, AH = 1) efficiently

8056 <1> ; in a loop to prevent a locked infinitive loop.

8057 <1> ;

8058 <1> ; Disk IRQs (6,14,15) have been phohibited from ring 3

8059 <1> ; callback because, disk operations (file system services

8060 <1> ; etc.) are independent from user program, for fast disk r/w.

8061 <1> ; They are not more useful at ring 3 while they are in use

8062 <1> ; by standard diskio functions which are mandatory part of

8063 <1> ; (monolithic) OS kernel and mainprog command interpreter.

8064 <1> ; INT 33h diskio functions are enough for user level disk

8065 <1> ; r/w.

8066 <1> ;

8067 <1> ; TRDOS 386 - IRQ CALLBACK structures (parameters):

8068 <1> ;

8069 <1> ; [u.irqlock] = 1 word, IRQ flags (0-15) that indicates

8070 <1> ; which IRQs are locked by (that) user.

8071 <1> ; Lock and unlock (by user) will change

8072 <1> ; these flags or 'terminate process' (sysexit)

8073 <1> ; will clear these flags and unlock those IRQs.

8074 <1> ;

8075 <1> ; Bit 0 is for IRQ 0 and Bit 15 is for IRQ 15

8076 <1> ;

8077 <1> ; IRQ(x).owner : 1 byte, user, [u.uno], 0 = free (unlocked)

8078 <1> ;

8079 <1> ; IRQ(x).method : 1 byte for callback method & status

8080 <1> ; 0 = Signal Response Byte method

8081 <1> ; 1 = Callback service method

8082 <1> ; >1 = invalid for current 'syscalback'.

8083 <1> ; or(+) 80h = IRQ is in use by system (ring 0)

8084 <1> ; function (audio etc.) or

8085 <1> ; a device driver.

8086 <1> ; (system function will ignore the lock/owner)

8087 <1> ;

8088 <1> ; IRQ(x).srb : 1 byte, Signal Return/Response byte value

8089 <1> ; (a fixed value by user or a counter value

8090 <1> ; from 0 to 255, which is increased by every

8091 <1> ; interrupt just before putting it into

8092 <1> ; the Signal Response byte address

8093 <1> ; (This is not used in callback serv method)

8094 <1> ;

8095 <1> ; IRQ(x).addr : 1 dword

8096 <1> ; Signal Response Byte address (physical)

8097 <1> ; -or-

8098 <1> ; Callback service address (virtual)

8099 <1> ;

8100 <1> ; IRQ(x).dev : 1 byte

8101 <1> ; 0 = Default device or kernel function

8102 <1> ; -or-

8103 <1> ; 1-255 = Assigned device driver number

8104 <1> ;

8105 <1> ; (x) = 3,4,5,7,9,10,11,12,13

8106 <1> ;

8107 <1> ;

8108 <1> ; NOTE: If user's process/program calls the kernel (INT 40h)

8109 <1> ; while it is already running in a (ring 3) callback

8110 <1> ; service, kernel will force (convert) system call to

8111 <1> ; 'sysrele' (sys release). So, this feature provides

8112 <1> ; easy and simple usage of callback services without

8113 <1> ; falling into deepless <please 'callback me' then

8114 <1> ; let me 'callback you'> cycles! (User must return

8115 <1> ; from callback service by using 'sysrele' system

8116 <1> ; call, without a significant delay. Otherwise user

8117 <1> ; process/program may be late to catch the next event

8118 <1> ; within same callback purpose.

8119 <1> ;

8120 <1>

8121 0000F047 30C0 <1> xor al, al ; the caller is 'syscalbac' sign/flag

8122 0000F049 E85A180000 <1> call set\_irq\_callback\_service

8123 <1> ; 16/04/2017

8124 0000F04E A3[64030300] <1> mov [u.r0], eax

8125 0000F053 0F8385D6FFFF <1> jnc sysret

8126 0000F059 A3[C8030300] <1> mov dword [u.error], eax

8127 0000F05E E95BD6FFFF <1> jmp error

8128 <1>

8129 <1> sysfpstat:

8130 <1> ; 28/02/2017 - TRDOS 386 (TRDOS v2.0)

8131 <1> ; (TRDOS 386 feature only!)

8132 <1> ;

8133 <1> ; Set or reset FPU registers save/restore option (for user)

8134 <1> ; (during software task switching, wswap-rswap)

8135 <1> ;

8136 <1> ; INPUT ->

8137 <1> ; BL = 0 -> reset

8138 <1> ; BL = 1 -> set (FPU register will be saved and restored)

8139 <1> ;

8140 <1> ; OUTPUT ->

8141 <1> ; cf = 0 -> no error, FPU is ready...

8142 <1> ; (EAX = 0)

8143 <1> ; Cf = 1 -> error, 80387 FPU is not ready !

8144 <1> ; (EAX = 0FFFFFFFFh)

8145 <1>

8146 0000F063 31C0 <1> xor eax, eax

8147 0000F065 803D[F8650100]00 <1> cmp byte [fpready], 0

8148 0000F06C 7613 <1> jna short sysfpstat\_err

8149 <1>

8150 0000F06E 80E301 <1> and bl, 1 ; use BIT 0 only !

8151 0000F071 881D[DA030300] <1> mov [u.fpsave], bl

8152 0000F077 A3[64030300] <1> mov [u.r0], eax ; 0

8153 0000F07C E95DD6FFFF <1> jmp sysret

8154 <1>

8155 <1> sysfpstat\_err:

8156 0000F081 48 <1> dec eax ; 0FFFFFFFFh

8157 0000F082 A3[64030300] <1> mov [u.r0], eax ; -1

8158 0000F087 E932D6FFFF <1> jmp error

8159 <1>

8160 <1> sysdelete: ; Delete (Remove, Unlink) File

8161 <1> ; 29/12/2017 (TRDOS 386 = TRDOS v2.0)

8162 <1> ;

8163 <1> ; INPUT ->

8164 <1> ; EBX = File name (ASCIIZ string) address

8165 <1> ; OUTPUT ->

8166 <1> ; cf = 0 -> eax = 0

8167 <1> ; cf = 1 -> Error code in AL

8168 <1> ;

8169 <1> ; Modified Registers: EAX (at the return of system call)

8170 <1> ;

8171 <1>

8172 0000F08C 89DE <1> mov esi, ebx

8173 <1> ; file name is forced, change directory as temporary

8174 <1> ;mov ax, 1

8175 <1> ;mov [FFF\_Valid], ah ; 0 ; reset

8176 <1> ;call set\_working\_path

8177 0000F08E E8680B0000 <1> call set\_working\_path\_x

8178 0000F093 731D <1> jnc short sysdelete\_1

8179 <1>

8180 0000F095 21C0 <1> and eax, eax ; 0 -> Bad Path!

8181 0000F097 7505 <1> jnz short sysdelete\_err

8182 <1> ; eax = 0

8183 <1> sysdelete\_path\_err:

8184 0000F099 B813000000 <1> mov eax, ERR\_INV\_PATH\_NAME ; 'bad path name !'

8185 <1> sysdelete\_err:

8186 0000F09E A3[64030300] <1> mov [u.r0], eax

8187 0000F0A3 A3[C8030300] <1> mov [u.error], eax

8188 0000F0A8 E8230C0000 <1> call reset\_working\_path

8189 0000F0AD E90CD6FFFF <1> jmp error

8190 <1> sysdelete\_1:

8191 <1> ;mov esi, FindFile\_Name

8192 0000F0B2 66B80018 <1> mov ax, 1800h ; Only files

8193 0000F0B6 E8A891FFFF <1> call find\_first\_file

8194 0000F0BB 72E1 <1> jc short sysdelete\_err

8195 <1> sysdelete\_2:

8196 <1> ; check file attributes

8197 <1>

8198 <1> ;test bl, 17 ; system, hidden, readonly, directory

8199 0000F0BD F6C307 <1> test bl, 7 ; system, hidden, readonly

8200 0000F0C0 7407 <1> jz short sysdelete\_3

8201 <1>

8202 0000F0C2 B80B000000 <1> mov eax, ERR\_FILE\_ACCESS ; 11 = 'permission denied !'

8203 0000F0C7 EBD5 <1> jmp short sysdelete\_err

8204 <1> sysdelete\_3:

8205 0000F0C9 6621D2 <1> and dx, dx ; Ambiguous filename chars used sign (DX>0)

8206 0000F0CC 7407 <1> jz short sysdelete\_4

8207 0000F0CE B81A000000 <1> mov eax, ERR\_INV\_FILE\_NAME ; 26 = 'invalid file name !'

8208 0000F0D3 EBC9 <1> jmp short sysdelete\_err

8209 <1> sysdelete\_4:

8210 <1> ;mov bh, [LongName\_EntryLength]

8211 0000F0D5 883D[6A630100] <1> mov [DelFile\_LNEL], bh ; Long name entry length (if > 0)

8212 <1> ; edi = Directory Entry Offset (DirBuff)

8213 <1> ; esi = Directory Entry (FFF Structure)

8214 0000F0DB E800BBFFFF <1> call remove\_file

8215 0000F0E0 72BC <1> jc short sysdelete\_err

8216 <1> sysrmdir\_5:

8217 0000F0E2 31C0 <1> xor eax, eax ; 0

8218 0000F0E4 A3[64030300] <1> mov [u.r0], eax

8219 <1> ;mov [u.error], eax

8220 0000F0E9 E8E20B0000 <1> call reset\_working\_path

8221 0000F0EE E9EBD5FFFF <1> jmp sysret

8222 <1>

8223 <1>

8224 <1> sysrmdir: ; Remove (Unlink) Directory

8225 <1> ; 29/12/2017 (TRDOS 386 = TRDOS v2.0)

8226 <1> ;

8227 <1> ; INPUT ->

8228 <1> ; EBX = Pointer to directory name

8229 <1> ; OUTPUT ->

8230 <1> ; cf = 0 -> eax = 0

8231 <1> ; cf = 1 -> Error code in AL

8232 <1> ;

8233 <1> ; Modified Registers: EAX (at the return of system call)

8234 <1> ;

8235 <1>

8236 0000F0F3 803D[B3030300]00 <1> cmp byte [u.uno], 0 ; root (super user) ?

8237 0000F0FA 7614 <1> jna short sysrmdir\_0

8238 <1>

8239 <1> ;mov dword [u.r0], ERR\_PERM\_DENIED

8240 0000F0FC B80B000000 <1> mov eax, ERR\_PERM\_DENIED ; ERR\_NOT\_SUPERUSER

8241 0000F101 A3[64030300] <1> mov [u.r0], eax

8242 0000F106 A3[C8030300] <1> mov [u.error], eax

8243 0000F10B E9AED5FFFF <1> jmp error

8244 <1>

8245 <1> sysrmdir\_0:

8246 0000F110 89DE <1> mov esi, ebx

8247 <1> ; file name is forced, change directory as temporary

8248 <1> ;mov ax, 1

8249 <1> ;mov [FFF\_Valid], ah ; 0 ; reset

8250 <1> ;call set\_working\_path

8251 0000F112 E8E40A0000 <1> call set\_working\_path\_x

8252 0000F117 731D <1> jnc short sysrmdir\_1

8253 <1>

8254 0000F119 21C0 <1> and eax, eax ; 0 -> Bad Path!

8255 0000F11B 7505 <1> jnz short sysrmdir\_err

8256 <1> ; eax = 0

8257 <1> sysrmdir\_not\_found:

8258 0000F11D B80C000000 <1> mov eax, ERR\_DIR\_NOT\_FOUND ; Directory not found !

8259 <1> sysrmdir\_err:

8260 0000F122 A3[64030300] <1> mov [u.r0], eax

8261 0000F127 A3[C8030300] <1> mov [u.error], eax

8262 0000F12C E89F0B0000 <1> call reset\_working\_path

8263 0000F131 E988D5FFFF <1> jmp error

8264 <1> sysrmdir\_1:

8265 <1> ;mov esi, FindFile\_Name

8266 0000F136 66B81008 <1> mov ax, 0810h ; Only directories

8267 0000F13A E82491FFFF <1> call find\_first\_file

8268 0000F13F 7306 <1> jnc short sysrmdir\_2

8269 <1>

8270 <1> ; eax = 2 (File not found !)

8271 0000F141 3C02 <1> cmp al, 2 ; ERR\_NOT\_FOUND

8272 0000F143 74D8 <1> je short sysrmdir\_not\_found

8273 0000F145 EBDB <1> jmp short sysrmdir\_err

8274 <1> sysrmdir\_2:

8275 <1> ; check directory attributes

8276 <1>

8277 0000F147 F6C307 <1> test bl, 7 ; system, hidden, readonly

8278 0000F14A 7407 <1> jz short sysrmdir\_3

8279 <1>

8280 0000F14C B80B000000 <1> mov eax, ERR\_DIR\_ACCESS ; 11 = 'permission denied !'

8281 0000F151 EBCF <1> jmp short sysrmdir\_err

8282 <1> sysrmdir\_3:

8283 0000F153 6621D2 <1> and dx, dx ; Ambiguous filename chars used sign (DX>0)

8284 0000F156 7407 <1> jz short sysrmdir\_4

8285 <1> ;mov eax, ERR\_NOT\_DIR ; 'not a valid directory !'

8286 0000F158 B813000000 <1> mov eax, ERR\_INV\_PATH\_NAME ; 'bad path name !'

8287 0000F15D EBC3 <1> jmp short sysrmdir\_err

8288 <1> sysrmdir\_4:

8289 <1> ;mov bh, [LongName\_EntryLength]

8290 0000F15F 883D[6A630100] <1> mov [DelFile\_LNEL], bh ; Long name entry length (if > 0)

8291 <1> ; edi = Directory Entry Offset (DirBuff)

8292 <1> ; esi = Directory Entry (FFF Structure)

8293 0000F165 E8CE97FFFF <1> call delete\_sub\_directory

8294 0000F16A 0F8372FFFFFF <1> jnc sysrmdir\_5

8295 <1> ; jc short sysrmdir\_6

8296 <1> ;

8297 <1> ; xor eax, eax ; 0

8298 <1> ;sysrmdir\_5:

8299 <1> ; mov [u.r0], eax

8300 <1> ; ;mov [u.error], eax

8301 <1> ; call reset\_working\_path

8302 <1> ; jmp sysret

8303 <1> sysrmdir\_6:

8304 0000F170 A3[64030300] <1> mov [u.r0], eax

8305 0000F175 A3[C8030300] <1> mov [u.error], eax

8306 <1>

8307 0000F17A 09C0 <1> or eax, eax ; EAX = 0 -> Directory not empty!

8308 0000F17C 741C <1> jz short sysrmdir\_9

8309 <1>

8310 <1> ; EAX > 0 -> Error code in AL (or AX or EAX)

8311 <1>

8312 0000F17E 833D[1E610100]01 <1> cmp dword [FAT\_ClusterCounter], 1

8313 0000F185 7209 <1> jb short sysrmdir\_8

8314 <1> sysrmdir\_7:

8315 <1> ; ESI = Logical DOS Drive Description Table address

8316 0000F187 66BB00FF <1> mov bx, 0FF00h ; BH = FFh -> use ESI for Drive parameters

8317 <1> ; BL = 0 -> Recalculate free cluster count

8318 0000F18B E834D0FFFF <1> call calculate\_fat\_freespace

8319 <1> sysrmdir\_8:

8320 0000F190 E83B0B0000 <1> call reset\_working\_path

8321 0000F195 E924D5FFFF <1> jmp error

8322 <1>

8323 <1> sysrmdir\_9:

8324 0000F19A A1[1E610100] <1> mov eax, [FAT\_ClusterCounter]

8325 0000F19F 09C0 <1> or eax, eax ; 0 ?

8326 0000F1A1 0F847BFFFFFF <1> jz sysrmdir\_err

8327 <1> ; ESI = Logical DOS Drive Description Table address

8328 0000F1A7 66BB01FF <1> mov bx, 0FF01h ; BH = FFh -> use ESI for Drive parameters

8329 <1> ; BL = 1 -> add free clusters

8330 0000F1AB E814D0FFFF <1> call calculate\_fat\_freespace

8331 0000F1B0 09C9 <1> or ecx, ecx

8332 0000F1B2 74DC <1> jz short sysrmdir\_8 ; ecx = 0 -> OK

8333 <1> ; ecx > 0 -> Error (Recalculation is needed)

8334 0000F1B4 EBD1 <1> jmp short sysrmdir\_7

8335 <1>

8336 <1>

8337 <1> syschdir: ; Change Current (Working) Drive & Directory (for user)

8338 <1> ; 30/12/2017 (TRDOS 386 = TRDOS v2.0)

8339 <1> ;

8340 <1> ; INPUT ->

8341 <1> ; EBX = Directory name (ASCIIZ string) address

8342 <1> ; OUTPUT ->

8343 <1> ; cf = 0 -> eax = 0

8344 <1> ; cf = 1 -> Error code in AL

8345 <1> ;

8346 <1> ; Modified Registers: EAX (at the return of system call)

8347 <1> ;

8348 <1> ; NOTE: If drive name is not included, only the working

8349 <1> ; directory (for user, not for drive/OS) will be chanded.

8350 <1> ; If there is a drive name (as A:, B:, C:, D: etc.)

8351 <1> ; at the beginning of the ASCIIZ (directory) string,

8352 <1> ; working drive and working directory (for user)

8353 <1> ; will be changed together.

8354 <1> ; (When the program is terminated, MainProg -internal

8355 <1> ; shell- will reset working directory to the previous

8356 <1> ; -current- logical drive's current directory again.)

8357 <1>

8358 0000F1B6 89DE <1> mov esi, ebx

8359 <1> ; file name is not forced, change directory as temporary

8360 0000F1B8 31C0 <1> xor eax, eax

8361 <1> ;mov [FFF\_Valid], ah ; 0 ; reset

8362 <1> ;call set\_working\_path

8363 0000F1BA E8400A0000 <1> call set\_working\_path\_xx

8364 0000F1BF 731D <1> jnc short syschdir\_ok

8365 0000F1C1 21C0 <1> and eax, eax ; 0 -> Bad Path!

8366 0000F1C3 7505 <1> jnz short syschdir\_err

8367 <1> ; eax = 0

8368 <1> syschdir\_not\_found:

8369 0000F1C5 B80C000000 <1> mov eax, ERR\_DIR\_NOT\_FOUND ; Directory not found !

8370 <1> syschdir\_err:

8371 0000F1CA A3[64030300] <1> mov [u.r0], eax

8372 0000F1CF A3[C8030300] <1> mov [u.error], eax

8373 0000F1D4 E8F70A0000 <1> call reset\_working\_path

8374 0000F1D9 E9E0D4FFFF <1> jmp error

8375 <1> syschdir\_ok:

8376 0000F1DE 31C0 <1> xor eax, eax ; 0

8377 0000F1E0 A3[64030300] <1> mov [u.r0], eax

8378 <1> ;mov [u.error], eax

8379 0000F1E5 E9F4D4FFFF <1> jmp sysret

8380 <1>

8381 <1>

8382 <1> syschmod: ; Get & Change File (or Directory) Attributes

8383 <1> ; 30/12/2017 (TRDOS 386 = TRDOS v2.0)

8384 <1> ;

8385 <1> ; INPUT ->

8386 <1> ; EBX = File/Directory (ASCIIZ) name address

8387 <1> ; CL = New attributes (if CL < 40h)

8388 <1> ; CL >= 40h -> Get File Attributes

8389 <1> ; OUTPUT ->

8390 <1> ; cf = 0 -> EAX = File attributes (in AL)

8391 <1> ; cf = 1 -> Error code in AL

8392 <1> ;

8393 <1> ; Modified Registers: EAX (at the return of system call)

8394 <1> ;

8395 <1> ; MSDOS File Attributes: (bit value of attrib byte)

8396 <1> ; ATTR\_READ\_ONLY = 01h (bit 0, 'R')

8397 <1> ; ATTR\_HIDDEN = 02h (bit 1, 'H')

8398 <1> ; ATTR\_SYSTEM = 04h (bit 2, 'S')

8399 <1> ; ATTR\_VOLUME\_ID = 08h (bit 3)

8400 <1> ; ATTR\_DIRECTORY = 10h (bit 4)

8401 <1> ; ATTR\_ARCHIVE = 20h (bit 5, 'A')

8402 <1> ; ATTR\_LONG\_NAME = ATTR\_READONLY |

8403 <1> ; ATTR\_HIDDEN |

8404 <1> ; ATTR\_SYSTEM |

8405 <1> ; ATTR\_VOLUME\_ID

8406 <1> ; The upper two bits of attributes must be 0.

8407 <1>

8408 <1> ; Note: \* If ATTR\_DIRECTORY is set, only directory names

8409 <1> ; will be searched (and S,H,R,A attributeds of

8410 <1> ; the directory will be changed.)

8411 <1> ; \* If ATTR\_VOLUME\_ID is set, 'syschmod' system call

8412 <1> ; will return with 'permission denied' error.

8413 <1> ; \* If ATTR\_DIRECTORY is not set, only file names

8414 <1> ; will be searched (and S,H,R,A attributes of the

8415 <1> ; file will be changed.)

8416 <1> ;

8417 <1> ; (Ony Super User can change S,H,R attributes.)

8418 <1>

8419 0000F1EA 80F940 <1> cmp cl, 40h

8420 0000F1ED 7327 <1> jnb short syschmod\_0

8421 <1>

8422 0000F1EF F6C108 <1> test cl, 08h ; ATTR\_VOLUME\_ID

8423 0000F1F2 750E <1> jnz short syschmod\_perm\_err

8424 <1>

8425 0000F1F4 803D[B3030300]00 <1> cmp byte [u.uno], 0 ; root (super user) ?

8426 0000F1FB 7619 <1> jna short syschmod\_0

8427 <1>

8428 <1> ; Not super user..

8429 0000F1FD F6C107 <1> test cl, 07h ; S,H,R attributes

8430 0000F200 7414 <1> jz short syschmod\_0

8431 <1>

8432 <1> syschmod\_perm\_err:

8433 <1> ;mov dword [u.r0], ERR\_PERM\_DENIED

8434 0000F202 B80B000000 <1> mov eax, ERR\_PERM\_DENIED ; 'permission denied !'

8435 0000F207 A3[64030300] <1> mov [u.r0], eax

8436 0000F20C A3[C8030300] <1> mov [u.error], eax

8437 0000F211 E9A8D4FFFF <1> jmp error

8438 <1>

8439 <1> syschmod\_0:

8440 0000F216 880D[B8630100] <1> mov [Attributes], cl

8441 0000F21C 89DE <1> mov esi, ebx

8442 <1> ; file name is forced, change directory as temporary

8443 <1> ;mov ax, 1

8444 <1> ;mov [FFF\_Valid], ah ; 0 ; reset

8445 <1> ;call set\_working\_path

8446 0000F21E E8D8090000 <1> call set\_working\_path\_x

8447 0000F223 731D <1> jnc short syschmod\_1

8448 0000F225 21C0 <1> and eax, eax ; 0 -> Bad Path!

8449 0000F227 7505 <1> jnz short syschmod\_err

8450 <1> ; eax = 0

8451 <1> syschmod\_path\_not\_found:

8452 0000F229 B813000000 <1> mov eax, ERR\_INV\_PATH\_NAME ; 'Bad path name !'

8453 <1> syschmod\_err:

8454 0000F22E A3[64030300] <1> mov [u.r0], eax

8455 0000F233 A3[C8030300] <1> mov [u.error], eax

8456 0000F238 E8930A0000 <1> call reset\_working\_path

8457 0000F23D E97CD4FFFF <1> jmp error

8458 <1> syschmod\_1:

8459 0000F242 B008 <1> mov al, 08h ; Except volume labels (& long names)

8460 0000F244 A0[B8630100] <1> mov al, [Attributes]

8461 0000F249 2410 <1> and al, 10h ;

8462 <1> ;mov esi, FindFile\_Name

8463 <1> ;mov ax, 1800h ; Only files

8464 <1> ;mov ax, 0810h ; Only directories

8465 0000F24B E81390FFFF <1> call find\_first\_file

8466 <1> ;jnc short syschmod\_2

8467 0000F250 72DC <1> jc short syschmod\_err

8468 <1>

8469 <1> ;; eax = 2 (File not found !)

8470 <1> ;cmp al, 2 ; ERR\_NOT\_FOUND

8471 <1> ;jne short syschmod\_err

8472 <1>

8473 <1> ;and byte [Attributes], 10h

8474 <1> ;jz short syschmod\_err

8475 <1>

8476 <1> ;; Directory not found !

8477 <1> ;mov al, 3 ; ERR\_PATH\_NOT\_FOUND

8478 <1> ;jmp short syschmod\_err

8479 <1>

8480 <1> syschmod\_2:

8481 0000F252 6621D2 <1> and dx, dx ; Ambiguous filename chars used sign (DX>0)

8482 0000F255 7407 <1> jz short syschmod\_3

8483 0000F257 B81A000000 <1> mov eax, ERR\_INV\_FILE\_NAME ; 'invalid file name !'

8484 0000F25C EBD0 <1> jmp short syschmod\_err

8485 <1> syschmod\_3:

8486 <1> ; EDI = Directory buffer entry offset/address

8487 <1> ; BL = File (or Directory) Attributes

8488 <1> ; mov bl, [EDI+0Bh]

8489 <1>

8490 <1> ; check directory attributes

8491 0000F25E 8A3D[B8630100] <1> mov bh, [Attributes] ; new attributes

8492 0000F264 80FF40 <1> cmp bh, 40h ;>=40 -> get file/directory attributes

8493 0000F267 732D <1> jnb short syschmod\_6

8494 <1>

8495 <1> ; set file/directory attributes

8496 0000F269 F6C307 <1> test bl, 7 ; system, hidden, readonly

8497 0000F26C 7409 <1> jz short syschmod\_4

8498 <1>

8499 0000F26E 803D[B3030300]00 <1> cmp byte [u.uno], 0 ; root (super user) ?

8500 0000F275 778B <1> ja short syschmod\_perm\_err

8501 <1> syschmod\_4:

8502 0000F277 66817F0CA101 <1> cmp word [edi+DirEntry\_NTRes], 01A1h ; Singlix FS

8503 0000F27D 7424 <1> je short syschmod\_7

8504 <1>

8505 0000F27F 887F0B <1> mov [edi+0Bh], bh ; Attributes (New!)

8506 <1>

8507 0000F282 C605[28610100]02 <1> mov byte [DirBuff\_ValidData], 2 ; modified sign

8508 <1> ; to force write

8509 0000F289 E80AB6FFFF <1> call save\_directory\_buffer

8510 0000F28E 729E <1> jc short syschmod\_err

8511 <1>

8512 <1> syschmod\_5:

8513 0000F290 8A1D[B8630100] <1> mov bl, [Attributes]

8514 <1> syschmod\_6:

8515 0000F296 0FB6C3 <1> movzx eax, bl

8516 0000F299 A3[64030300] <1> mov [u.r0], eax

8517 <1> ;mov dword [u.error], 0

8518 0000F29E E93BD4FFFF <1> jmp sysret

8519 <1>

8520 <1> syschmod\_7:

8521 0000F2A3 29C0 <1> sub eax, eax

8522 0000F2A5 8A25[26610100] <1> mov ah, [DirBuff\_DRV]

8523 0000F2AB BE00010900 <1> mov esi, Logical\_DOSDisks

8524 0000F2B0 01C6 <1> add esi, eax

8525 0000F2B2 807E04A1 <1> cmp byte [esi+LD\_FSType], 0A1h

8526 0000F2B6 7307 <1> jnc short syschmod\_8

8527 0000F2B8 B01D <1> mov al, ERR\_INV\_DATA ; 29 = Invalid Data

8528 0000F2BA E96FFFFFFF <1> jmp syschmod\_err

8529 <1>

8530 <1> syschmod\_8:

8531 <1> ; BH = New MS-DOS File Attributes

8532 0000F2BF 88F8 <1> mov al, bh ; File/Directory Attributes

8533 0000F2C1 30E4 <1> xor ah, ah ; Attributes in MS-DOS format sign

8534 0000F2C3 E8F9A0FFFF <1> call change\_fs\_file\_attributes

8535 0000F2C8 0F8260FFFFFF <1> jc syschmod\_err

8536 0000F2CE EBC0 <1> jmp short syschmod\_5

8537 <1>

8538 <1>

8539 <1> sysdrive: ; Get/Set Current (Working) Drive (for user)

8540 <1> ; 30/12/2017 (TRDOS 386 = TRDOS v2.0)

8541 <1> ;

8542 <1> ; INPUT ->

8543 <1> ; BL = Logical DOS Drive number (0=A: ... 2=C:)

8544 <1> ; If BL = 0FFh -> Get Current Drive

8545 <1> ; OUTPUT ->

8546 <1> ; cf = 0 ->

8547 <1> ; AL = Current Drive number

8548 <1> ; AH = The Last Logical DOS Drive no.

8549 <1> ; cf = 1 -> Error code in AL

8550 <1> ;

8551 <1> ; Modified Registers: EAX (at the return of system call)

8552 <1> ;

8553 <1> ; NOTE: If the requested logical dos drive is ready,

8554 <1> ; it's current current directory will be the user's

8555 <1> ; (program's) current directory.

8556 <1> ; (When the program is terminated, MainProg -internal

8557 <1> ; shell- will reset the previous -current- logical drive

8558 <1> ; as current drive again).

8559 <1>

8560 0000F2D0 80FBFF <1> cmp bl, 0FFh

8561 0000F2D3 7435 <1> je short sysdrive\_ok

8562 0000F2D5 3A1D[D20C0100] <1> cmp bl, [Last\_DOS\_DiskNo]

8563 0000F2DB 771E <1> ja short sysdrive\_err

8564 <1>

8565 <1> ; Save current drive and reset mode

8566 <1> ; for 'reset\_working\_path' procedure (for MainProg)

8567 0000F2DD 30C0 <1> xor al, al

8568 0000F2DF 66A3[F4650100] <1> mov [SWP\_Mode], ax ; ah = 0

8569 0000F2E5 A0[FE580100] <1> mov al, [Current\_Drv]

8570 0000F2EA FEC4 <1> inc ah ; mov ah, 1

8571 0000F2EC 66A3[F6650100] <1> mov [SWP\_DRV], ax

8572 <1>

8573 0000F2F2 88DA <1> mov dl, bl

8574 0000F2F4 E8C77BFFFF <1> call change\_current\_drive

8575 0000F2F9 730F <1> jnc short sysdrive\_ok

8576 <1> sysdrive\_err:

8577 0000F2FB C705[64030300]0F00- <1> mov dword [u.r0], ERR\_DRV\_NOT\_RDY ; 'drive not ready !'

8577 0000F303 0000 <1>

8578 0000F305 E9B4D3FFFF <1> jmp error

8579 <1> sysdrive\_ok:

8580 0000F30A A0[FE580100] <1> mov al, [Current\_Drv]

8581 0000F30F 8A25[D20C0100] <1> mov ah, [Last\_DOS\_DiskNo]

8582 0000F315 A3[64030300] <1> mov [u.r0], eax

8583 0000F31A E9BFD3FFFF <1> jmp sysret

8584 <1>

8585 <1>

8586 <1> sysdir: ; Get Current (Working) Drive & Directory (for user)

8587 <1> ; 30/12/2017 (TRDOS 386 = TRDOS v2.0)

8588 <1> ;

8589 <1> ; INPUT ->

8590 <1> ; EBX = Current directory name buffer address

8591 <1> ; (Buffer length = 92 bytes)

8592 <1> ; OUTPUT ->

8593 <1> ; AL = Current drive (0=A: .. 2=C:)

8594 <1> ; If CF = 1 -> AL = error code

8595 <1> ;

8596 <1> ; Modified Registers: EAX (at the return of system call)

8597 <1> ;

8598 <1> ; Note: Required directory name buffer length may be

8599 <1> ; <= 92 bytes for current TRDOS 386 version.

8600 <1> ; (7\*12 name chars + 7 slash + 0)

8601 <1>

8602 0000F31F 89E5 <1> mov ebp, esp

8603 0000F321 83EC60 <1> sub esp, 96

8604 0000F324 53 <1> push ebx ; User's buffer address

8605 0000F325 30D2 <1> xor dl, dl ; 0 = current drive

8606 0000F327 E890AAFFFF <1> call get\_current\_directory

8607 0000F32C 72CD <1> jc short sysdrive\_err ; 'drive not ready !' error

8608 0000F32E 89E6 <1> mov esi, esp ; System's buffer address

8609 0000F330 5F <1> pop edi ; User's buffer address

8610 <1> ; ecx = transfer (byte) count (<=92)

8611 0000F331 E843F4FFFF <1> call transfer\_to\_user\_buffer

8612 0000F336 89EC <1> mov esp, ebp

8613 0000F338 730F <1> jnc short sysdir\_ok

8614 <1> sysdir\_err:

8615 0000F33A C705[64030300]2E00- <1> mov dword [u.r0], ERR\_BUFFER ; 'buffer error !'

8615 0000F342 0000 <1>

8616 0000F344 E975D3FFFF <1> jmp error

8617 <1> sysdir\_ok:

8618 0000F349 8A0D[FE580100] <1> mov cl, [Current\_Drv]

8619 0000F34F 890D[64030300] <1> mov [u.r0], ecx

8620 0000F355 E984D3FFFF <1> jmp sysret

8621 <1>

8622 <1>

8623 <1> sysldrvt: ; Get copy of Logical DOS Drive Description Table

8624 <1> ; 30/12/2017 (TRDOS 386 = TRDOS v2.0)

8625 <1> ;

8626 <1> ; INPUT ->

8627 <1> ; BL = Logical DOS drive number (zero based)

8628 <1> ; ECX = Logical DOS drv desc table buffer addr

8629 <1> ; (Buffer length = 256 bytes)

8630 <1> ; OUTPUT ->

8631 <1> ; cf = 0 ->

8632 <1> ; AL = Current Drive number

8633 <1> ; AH = The Last Logical DOS Drive no.

8634 <1> ; cf = 1 -> Error code in AL

8635 <1> ; AH = The Last Logical DOS Drive no.

8636 <1> ;

8637 <1> ; Modified Registers: EAX (at the return of system call)

8638 <1> ;

8639 <1> ; Note: Required description table buffer length is

8640 <1> ; 256 bytes for current TRDOS 386 version.

8641 <1>

8642 0000F35A 89CF <1> mov edi, ecx ; Destination address (user space)

8643 0000F35C 88DC <1> mov ah, bl

8644 0000F35E 30C0 <1> xor al, al

8645 0000F360 BE00010900 <1> mov esi, Logical\_DOSDisks

8646 0000F365 01C6 <1> add esi, eax ; Source address (system space)

8647 0000F367 B900010000 <1> mov ecx, 256 ; Byte count

8648 <1> ; Logical Dos Drv Desc Table size

8649 0000F36C E808F4FFFF <1> call transfer\_to\_user\_buffer

8650 0000F371 72C7 <1> jc short sysdir\_err

8651 0000F373 8A2D[D20C0100] <1> mov ch, [Last\_DOS\_DiskNo]

8652 0000F379 EBCE <1> jmp short sysdir\_ok

8653 <1>

8654 <1>

8655 <1> systime: ; Get System Date&Time

8656 <1> ; 30/12/2017 (TRDOS 386 = TRDOS v2.0)

8657 <1> ;

8658 <1> ; INPUT -> BL =

8659 <1> ; 0 = Get Date&Time in Unix/Epoch format

8660 <1> ; 1 = Get Time in MSDOS format

8661 <1> ; 2 = Get Date in MSDOS format

8662 <1> ; 3 = Get Date&Time in MSDOS format

8663 <1> ; 4 & other values =

8664 <1> ; System timer ticks will be returned

8665 <1> ; in EAX and Carry Flag will be set.

8666 <1> ; (CF will not be set if BL = 4)

8667 <1> ; OUTPUT ->

8668 <1> ; For BL input = 3

8669 <1> ; EAX = Current Time (RTC)

8670 <1> ; AL = Second (DL in MSDOS)

8671 <1> ; AH = Minute (CL in MSDOS)

8672 <1> ; HW of EAX = Hour (CH in MSDOS)

8673 <1> ; EDX = Current System Date (RTC)

8674 <1> ; DL = Day (DL in MSDOS)

8675 <1> ; DH = Month (DH in MSDOS)

8676 <1> ; HW of EDX = Year (CX in MSDOS)

8677 <1> ;

8678 <1> ; For BL input = 2

8679 <1> ; EAX = Current System Date (RTC)

8680 <1> ; DL = Day (DL in MSDOS)

8681 <1> ; DH = Month (DH in MSDOS)

8682 <1> ; HW of EDX = Year (CX in MSDOS)

8683 <1> ;

8684 <1> ; For BL input = 1

8685 <1> ; EAX = Current Time (RTC)

8686 <1> ; AL = Second (DL in MSDOS)

8687 <1> ; AH = Minute (CL in MSDOS)

8688 <1> ; HW of EAX = Hour (CH in MSDOS)

8689 <1> ;

8690 <1> ; For BL input = 0

8691 <1> ; EAX = Unix (Epoch) Time Ticks/Seconds

8692 <1> ;

8693 <1> ; For BL input = 4

8694 <1> ; EAX = System timer ticks

8695 <1> ;

8696 <1> ; If CF = 1 (for other values of BL input)

8697 <1> ; EAX = System timer ticks (no error code!)

8698 <1> ;

8699 <1> ; Modified Registers: EAX, (EDX)

8700 <1> ; (at the return of system call)

8701 <1> ;

8702 <1>

8703 0000F37B 20DB <1> and bl, bl

8704 0000F37D 750F <1> jnz short systime\_1

8705 0000F37F E84071FFFF <1> call epoch

8706 <1> systime\_0:

8707 0000F384 A3[64030300] <1> mov [u.r0], eax

8708 0000F389 E950D3FFFF <1> jmp sysret

8709 <1> systime\_1:

8710 0000F38E 80FB04 <1> cmp bl, 4

8711 0000F391 7211 <1> jb short systime\_2

8712 0000F393 A1[B8580100] <1> mov eax, [TIMER\_LH] ; 18.2 Hz timer ticks

8713 <1> ; Note: [TIMER\_LH] may be set

8714 <1> ; to wrong timer value due to

8715 <1> ; program functions.

8716 <1> ; (This value must not be

8717 <1> ; accepted as [TIMER\_LH]/18.2

8718 <1> ; seconds since the midnight.)

8719 0000F398 76EA <1> jna short systime\_0

8720 0000F39A A3[64030300] <1> mov [u.r0], eax

8721 0000F39F E91AD3FFFF <1> jmp error ; cf = 1 & [u.r0] = eax = timer ticks

8722 <1>

8723 <1> systime\_2:

8724 <1> ;push ebx

8725 0000F3A4 E87D70FFFF <1> call get\_rtc\_date\_time

8726 <1> ;pop ebx

8727 0000F3A9 F6C301 <1> test bl, 1

8728 0000F3AC 7429 <1> jz short systime\_4

8729 0000F3AE 30E4 <1> xor ah, ah

8730 0000F3B0 A0[30550100] <1> mov al, [hour]

8731 0000F3B5 88C2 <1> mov dl, al

8732 0000F3B7 C1E010 <1> shl eax, 16

8733 0000F3BA A0[34550100] <1> mov al, [second]

8734 0000F3BF 8A25[32550100] <1> mov ah, [minute]

8735 0000F3C5 F6C302 <1> test bl, 2

8736 0000F3C8 74BA <1> jz short systime\_0

8737 <1> ; Check time & date match risk

8738 <1> ; (23:59:59 may cause to wrong

8739 <1> ; date -new day with previous date-...)

8740 0000F3CA 80FA17 <1> cmp dl, 23

8741 0000F3CD 7206 <1> jb short systime\_3

8742 0000F3CF 663D3B3B <1> cmp ax, (59\*256)+59 ; if hour is 23:59:59

8743 0000F3D3 73CF <1> jnb short systime\_2 ; wait for 1 second

8744 <1> systime\_3:

8745 <1> ; eax = time

8746 0000F3D5 89C6 <1> mov esi, eax

8747 <1> systime\_4:

8748 0000F3D7 66A1[2A550100] <1> mov ax, [year]

8749 0000F3DD C1E010 <1> shl eax, 16

8750 0000F3E0 A0[2E550100] <1> mov al, [day]

8751 0000F3E5 8A25[2C550100] <1> mov ah, [month]

8752 <1> ; eax = date

8753 0000F3EB 80E301 <1> and bl, 1

8754 0000F3EE 7494 <1> jz short systime\_0

8755 0000F3F0 96 <1> xchg esi, eax

8756 <1> ; eax = time, esi = date

8757 0000F3F1 8B2D[60030300] <1> mov ebp, [u.usp] ; EBP points to user's registers

8758 <1> ; (user) edx <-- (system) esi

8759 0000F3F7 897514 <1> mov [ebp+20], esi ; return to user with EDX value

8760 0000F3FA EB88 <1> jmp short systime\_0

8761 <1>

8762 <1>

8763 <1> sysstime: ; Set System Date&Time

8764 <1> ; 31/12/2017

8765 <1> ; 30/12/2017 (TRDOS 386 = TRDOS v2.0)

8766 <1> ;

8767 <1> ; INPUT -> BL =

8768 <1> ; 0 = Set Date&Time in Unix/Epoch format

8769 <1> ; 1 = Set Time in MSDOS format

8770 <1> ; 2 = Set Date in MSDOS format

8771 <1> ; 3 = Set Date&Time in MSDOS format

8772 <1> ; 4 = Set System Timer (Ticks)

8773 <1> ; 5 = Convert/Save current time to/as

8774 <1> ; 18.2 Hz system timer ticks

8775 <1> ; 6 = Convert MSDOS Date&Time to UNIX format

8776 <1> ; without setting system date&time ; (test)

8777 <1> ; 7 = Convert UNIX Date&Time to MSDOS format

8778 <1> ; without setting system date&time ; (test)

8779 <1> ; 8-0FFh = invalid !

8780 <1> ; ECX = Time (or Timer) value in selected format

8781 <1> ; EDX = Date value in MSDOS format if BL=2,3,6

8782 <1> ;

8783 <1> ; OUTPUT ->

8784 <1> ; If CF = 0 ->

8785 <1> ; EAX = Set value

8786 <1> ; If CF = 1 -> (invalid BL input)

8787 <1> ; EAX = Ticks count [TIMER\_LH]

8788 <1> ;

8789 <1>

8790 0000F3FC 20DB <1> and bl, bl ; 0

8791 0000F3FE 7511 <1> jnz short sysstime\_0

8792 0000F400 89C8 <1> mov eax, ecx

8793 0000F402 E84771FFFF <1> call convert\_from\_epoch

8794 0000F407 E8F371FFFF <1> call set\_rtc\_date\_time

8795 0000F40C E9CDD2FFFF <1> jmp sysret

8796 <1> sysstime\_0:

8797 0000F411 80FB08 <1> cmp bl, 8

8798 0000F414 722D <1> jb short sysstime\_1

8799 <1> ; invalid input (>7)

8800 0000F416 A1[B8580100] <1> mov eax, [TIMER\_LH] ; 18.2 Hz timer ticks

8801 <1> ; Note: [TIMER\_LH] may be set

8802 <1> ; to wrong timer value due to

8803 <1> ; program functions.

8804 <1> ; (This value must not be

8805 <1> ; accepted as [TIMER\_LH]/18.2

8806 <1> ; seconds since the midnight.)

8807 0000F41B A3[64030300] <1> mov [u.r0], eax

8808 0000F420 E999D2FFFF <1> jmp error ; cf = 1 & [u.r0] = eax = timer ticks

8809 <1>

8810 <1> sysstime\_8:

8811 <1> ; BL = 7

8812 0000F425 89C8 <1> mov eax, ecx ; seconds since 1/1/1970 00:00:00

8813 0000F427 E82271FFFF <1> call convert\_from\_epoch

8814 0000F42C 30E4 <1> xor ah, ah

8815 0000F42E A0[30550100] <1> mov al, [hour]

8816 0000F433 C1E010 <1> shl eax, 16

8817 0000F436 A0[34550100] <1> mov al, [second]

8818 0000F43B 8A25[32550100] <1> mov ah, [minute]

8819 0000F441 EB92 <1> jmp short systime\_3

8820 <1>

8821 <1> sysstime\_1:

8822 0000F443 80FB04 <1> cmp bl, 4

8823 0000F446 743F <1> je short sysstime\_2 ; set system timer ticks

8824 0000F448 80FB05 <1> cmp bl, 5

8825 0000F44B 754B <1> jne short sysstime\_4

8826 <1> ; convert current time to system timer ticks (18.2Hz)

8827 0000F44D E8D46FFFFF <1> call get\_rtc\_date\_time

8828 0000F452 0FB60D[30550100] <1> movzx ecx, byte [hour]

8829 0000F459 B8100E0000 <1> mov eax, 60\*60 ; 1 hour = 3600 seconds

8830 0000F45E F7E1 <1> mul ecx

8831 0000F460 89C3 <1> mov ebx, eax

8832 0000F462 B13C <1> mov cl, 60 ; 1 minute = 60 seconds

8833 0000F464 0FB605[32550100] <1> movzx eax, byte [minute]

8834 0000F46B F7E1 <1> mul ecx

8835 0000F46D 01D8 <1> add eax, ebx

8836 0000F46F 8A0D[34550100] <1> mov cl, [second]

8837 0000F475 01C8 <1> add eax, ecx

8838 0000F477 B1B6 <1> mov cl, 182

8839 0000F479 F7E1 <1> mul ecx

8840 0000F47B 83C009 <1> add eax, 9

8841 0000F47E 83D200 <1> adc edx, 0

8842 0000F481 B10A <1> mov cl, 10

8843 0000F483 F7F1 <1> div ecx

8844 <1> ; eax = ((182\*seconds)+9)/10

8845 0000F485 89C1 <1> mov ecx, eax

8846 <1> sysstime\_2:

8847 0000F487 890D[B8580100] <1> mov [TIMER\_LH], ecx ; 18.2 \* seconds

8848 <1> sysstime\_3:

8849 0000F48D 890D[64030300] <1> mov [u.r0], ecx

8850 0000F493 E946D2FFFF <1> jmp sysret

8851 <1> sysstime\_4:

8852 0000F498 80FB06 <1> cmp bl, 6

8853 0000F49B 7788 <1> ja short sysstime\_8

8854 <1>

8855 0000F49D 890D[64030300] <1> mov [u.r0], ecx

8856 <1>

8857 0000F4A3 880D[34550100] <1> mov [second], cl

8858 0000F4A9 882D[32550100] <1> mov [minute], ch

8859 0000F4AF C1E910 <1> shr ecx, 16

8860 0000F4B2 880D[30550100] <1> mov [hour], cl

8861 <1> ; BL = 1,2,3,6

8862 0000F4B8 80FB01 <1> cmp bl, 1

8863 0000F4BB 762A <1> jna short sysstime\_5

8864 <1> ; BL = 2,3,6

8865 0000F4BD 8815[2E550100] <1> mov [day], dl

8866 0000F4C3 8835[2C550100] <1> mov [month], dh

8867 0000F4C9 C1EA10 <1> shr edx, 16

8868 0000F4CC 668915[2A550100] <1> mov [year], dx

8869 0000F4D3 80E303 <1> and bl, 3

8870 0000F4D6 742D <1> jz short sysstime\_7 ; 6

8871 <1> ; BL = 2,3

8872 0000F4D8 F6C301 <1> test bl, 1

8873 0000F4DB 7419 <1> jz short sysstime\_6 ; 2

8874 <1> ; BL = 3

8875 0000F4DD E81D71FFFF <1> call set\_rtc\_date\_time

8876 0000F4E2 E9F7D1FFFF <1> jmp sysret

8877 <1> sysstime\_5:

8878 <1> ; BL = 1

8879 0000F4E7 E85471FFFF <1> call set\_time\_bcd

8880 0000F4EC E82C65FFFF <1> call set\_rtc\_time

8881 0000F4F1 E9E8D1FFFF <1> jmp sysret

8882 <1> sysstime\_6:

8883 <1> ; BL = 2

8884 0000F4F6 E81871FFFF <1> call set\_date\_bcd

8885 0000F4FB E88C65FFFF <1> call set\_rtc\_date

8886 0000F500 E9D9D1FFFF <1> jmp sysret

8887 <1> sysstime\_7:

8888 <1> ; BL = 6

8889 <1> ; [year], [month], [day],

8890 <1> ; [hour], [minute], [second]

8891 0000F505 E8BF6FFFFF <1> call convert\_to\_epoch

8892 0000F50A 89C1 <1> mov ecx, eax ; seconds since 1/1/1970 00:00:00

8893 0000F50C E97CFFFFFF <1> jmp sysstime\_3

8894 <1>

8895 <1> sysrename: ; Rename File (or Directory)

8896 <1> ; 31/12/2017 (TRDOS 386 = TRDOS v2.0)

8897 <1> ;

8898 <1> ; INPUT ->

8899 <1> ; EBX = File/Directory (ASCIIZ) name address

8900 <1> ; ECX = New name (in same dir, no path name)

8901 <1> ; OUTPUT ->

8902 <1> ; cf = 0 -> EAX = 0

8903 <1> ; cf = 1 -> Error code in AL

8904 <1>

8905 0000F511 803D[B3030300]00 <1> cmp byte [u.uno], 0 ; root (super user) ?

8906 0000F518 7614 <1> jna short sysrename\_0

8907 <1>

8908 <1> sysrename\_perm\_err:

8909 <1> ;mov dword [u.r0], ERR\_PERM\_DENIED

8910 0000F51A B80B000000 <1> mov eax, ERR\_PERM\_DENIED ; 'permission denied !'

8911 0000F51F A3[64030300] <1> mov [u.r0], eax

8912 0000F524 A3[C8030300] <1> mov [u.error], eax

8913 0000F529 E990D1FFFF <1> jmp error

8914 <1>

8915 <1> sysrename\_0:

8916 0000F52E 51 <1> push ecx ; new file name address (in user space)

8917 0000F52F 89DE <1> mov esi, ebx

8918 <1> ; file name is forced, change directory as temporary

8919 <1> ;mov ax, 1

8920 <1> ;mov [FFF\_Valid], ah ; 0 ; reset

8921 <1> ;call set\_working\_path

8922 0000F531 E8C5060000 <1> call set\_working\_path\_x

8923 0000F536 731E <1> jnc short sysrename\_1

8924 0000F538 21C0 <1> and eax, eax ; 0 -> Bad Path!

8925 0000F53A 7505 <1> jnz short sysrename\_err

8926 <1> ; eax = 0

8927 <1> sysrename\_path\_not\_found:

8928 0000F53C B813000000 <1> mov eax, ERR\_INV\_PATH\_NAME ; 'Bad path name !'

8929 <1> sysrename\_err:

8930 0000F541 59 <1> pop ecx ; new file name address (in user space)

8931 <1> sysrename\_error:

8932 0000F542 A3[64030300] <1> mov [u.r0], eax

8933 0000F547 A3[C8030300] <1> mov [u.error], eax

8934 0000F54C E87F070000 <1> call reset\_working\_path

8935 0000F551 E968D1FFFF <1> jmp error

8936 <1> sysrename\_1:

8937 0000F556 B008 <1> mov al, 08h ; Except volume labels (& long names)

8938 0000F558 A0[B8630100] <1> mov al, [Attributes]

8939 0000F55D 2410 <1> and al, 10h ;

8940 <1> ;mov esi, FindFile\_Name

8941 <1> ;mov ax, 1800h ; Only files

8942 <1> ;mov ax, 0810h ; Only directories

8943 0000F55F 66B80008 <1> mov ax, 0800h ; Find File or Directory

8944 0000F563 E8FB8CFFFF <1> call find\_first\_file

8945 <1> ;jnc short sysrename\_2

8946 0000F568 72D7 <1> jc short sysrename\_err

8947 <1> sysrename\_2:

8948 <1> ; ESI = Directory Entry (FindFile\_DirEntry) Location

8949 <1> ; EDI = Directory Buffer Directory Entry Location

8950 <1> ; EAX = File Size

8951 <1> ; BL = Attributes of The File/Directory

8952 <1> ; BH = Long Name Yes/No Status (>0 is YES)

8953 <1> ; DX > 0 : Ambiguous filename chars are used

8954 <1>

8955 0000F56A 6621D2 <1> and dx, dx ; Ambiguous filename chars used sign (DX>0)

8956 0000F56D 7407 <1> jz short sysrename\_3

8957 0000F56F B81A000000 <1> mov eax, ERR\_INV\_FILE\_NAME ; 'invalid file name !'

8958 0000F574 EBCB <1> jmp short sysrename\_err

8959 <1> sysrename\_3:

8960 <1> ; EDI = Directory buffer entry offset/address

8961 <1> ; BL = File (or Directory) Attributes

8962 <1> ; mov bl, [EDI+0Bh]

8963 <1>

8964 0000F576 5A <1> pop edx ; new file name address (in user space)

8965 <1>

8966 <1> ; check file/directory attributes

8967 0000F577 F6C307 <1> test bl, 7 ; system, hidden, readonly

8968 0000F57A 759E <1> jnz short sysrename\_perm\_err

8969 <1> sysrename\_4:

8970 0000F57C 66817F0CA101 <1> cmp word [edi+DirEntry\_NTRes], 01A1h ; Singlix FS

8971 0000F582 7496 <1> je short sysrename\_perm\_err ; -temporary!-

8972 <1>

8973 <1> ; save old file name & file info (FFF structure)

8974 0000F584 BE[A2620100] <1> mov esi, FindFile\_Drv

8975 0000F589 BF[E8630100] <1> mov edi, SourceFile\_Drv

8976 0000F58E B920000000 <1> mov ecx, 128/4

8977 0000F593 F3A5 <1> rep movsd

8978 <1>

8979 0000F595 89D6 <1> mov esi, edx ; new file name address (in user space)

8980 0000F597 BF[68640100] <1> mov edi, DestinationFile\_Drv

8981 0000F59C E893AEFFFF <1> call parse\_path\_name

8982 0000F5A1 729F <1> jc short sysrename\_error ; eax = 1 (Bad file name)

8983 <1>

8984 <1> ; same drive ?

8985 0000F5A3 A0[A2620100] <1> mov al, [FindFile\_Drv]

8986 0000F5A8 3A05[68640100] <1> cmp al, [DestinationFile\_Drv]

8987 <1> ;jne short sysrename\_perm\_err ; Permission denied

8988 0000F5AE 7509 <1> jne short sysrename\_5 ; Bad file name

8989 <1>

8990 <1> ; no path name !? (rename file in same directory)

8991 0000F5B0 803D[69640100]20 <1> cmp byte [DestinationFile\_Directory], 20h

8992 0000F5B7 7607 <1> jna short sysrename\_6

8993 <1> sysrename\_5:

8994 0000F5B9 B801000000 <1> mov eax, ERR\_BAD\_CMD\_ARG ; 1 = Bad file name

8995 <1> ; (Bad argument)

8996 0000F5BE EB82 <1> jmp short sysrename\_error

8997 <1> sysrename\_6:

8998 0000F5C0 803D[AA640100]20 <1> cmp byte [DestinationFile\_Name], 20h

8999 0000F5C7 76F0 <1> jna short sysrename\_5

9000 <1>

9001 0000F5C9 BE[AA640100] <1> mov esi, DestinationFile\_Name

9002 0000F5CE E84E90FFFF <1> call check\_filename ; is it a valid msdos file name?

9003 0000F5D3 0F8269FFFFFF <1> jc sysrename\_error ; 26 = ERR\_INV\_FILE\_NAME

9004 <1>

9005 <1> ;mov esi, DestinationFile\_Name

9006 0000F5D9 66B80008 <1> mov ax, 0800h ; Find File or Directory

9007 0000F5DD E8818CFFFF <1> call find\_first\_file

9008 0000F5E2 720A <1> jc short sysrename\_7

9009 <1>

9010 0000F5E4 B80E000000 <1> mov eax, ERR\_FILE\_EXISTS ; file already exists !

9011 0000F5E9 E954FFFFFF <1> jmp sysrename\_error

9012 <1> sysrename\_7:

9013 <1> ; eax = 2 (File not found !)

9014 0000F5EE 3C02 <1> cmp al, 2 ; ERR\_NOT\_FOUND

9015 0000F5F0 0F854CFFFFFF <1> jne sysrename\_error

9016 <1>

9017 <1> ; 31/12/2017

9018 <1> ; Following code is also part of 'rename\_file' in

9019 <1> ; 'trdosk3.s' (MainProg's 'rename' command) ; 13/11/2017

9020 0000F5F6 BE[AA640100] <1> mov esi, DestinationFile\_Name ; (Rename\_NewName)

9021 0000F5FB 668B0D[62640100] <1> mov cx, [SourceFile\_DirEntryNumber]

9022 0000F602 66A1[4E640100] <1> mov ax, [SourceFile\_DirEntry+20] ; First Cluster, HW

9023 0000F608 C1E010 <1> shl eax, 16

9024 0000F60B 66A1[54640100] <1> mov ax, [SourceFile\_DirEntry+26] ; First Cluster, LW

9025 0000F611 0FB61D[37640100] <1> movzx ebx, byte [SourceFile\_LongNameEntryLength]

9026 0000F618 E85FB6FFFF <1> call rename\_directory\_entry

9027 0000F61D 0F821FFFFFFF <1> jc sysrename\_error

9028 <1> ;xor eax, eax

9029 0000F623 A3[64030300] <1> mov [u.r0], eax ; 0

9030 <1> ;mov [u.error], eax

9031 0000F628 E8A3060000 <1> call reset\_working\_path

9032 0000F62D E9ACD0FFFF <1> jmp sysret

9033 <1>

9034 <1> sysmem: ; Get Total&Free Memory amount

9035 <1> ; 31/12/2017 (TRDOS 386 = TRDOS v2.0)

9036 <1> ;

9037 <1> ; INPUT ->

9038 <1> ; none

9039 <1> ; OUTPUT ->

9040 <1> ; EAX = Total memory count (in bytes)

9041 <1> ; EBX = Virtually available memory amount (in bytes)

9042 <1> ; = 4GB - CORE (4MB)

9043 <1> ; ECX = Free memory count (in bytes)

9044 <1> ; EDX = Calculated free memory count (in bytes)

9045 <1>

9046 0000F632 A1[3C580100] <1> mov eax, [memory\_size] ; in pages

9047 0000F637 C1E00C <1> shl eax, 12 ; in bytes

9048 0000F63A A3[64030300] <1> mov [u.r0], eax

9049 0000F63F E8043DFFFF <1> call calc\_free\_mem

9050 <1> ; edx = calculated free pages

9051 <1> ; ecx = 0

9052 0000F644 8B2D[60030300] <1> mov ebp, [u.usp] ; EBP points to user's registers

9053 0000F64A C745100000C0FF <1> mov dword [ebp+16], ECORE ; EBX (for user)

9054 <1> ; 0FFC00000h ; 4GB - 4MB

9055 0000F651 C1E20C <1> shl edx, 12

9056 0000F654 895514 <1> mov [ebp+20], edx ; EDX (for user)

9057 0000F657 8B0D[40580100] <1> mov ecx, [free\_pages]

9058 0000F65D C1E10C <1> shl ecx, 12 ; free bytes

9059 0000F660 894D18 <1> mov [ebp+24], ecx ; ECX (for user)

9060 <1> ;mov [free\_pages], edx

9061 0000F663 E976D0FFFF <1> jmp sysret

9062 <1>

9063 <1> sysprompt:

9064 <1> ; Set TRDOS 386 Command Interpreter (MainProg) prompt

9065 <1> ; 31/12/2017 (TRDOS 386 = TRDOS v2.0)

9066 <1> ;

9067 <1> ; INPUT ->

9068 <1> ; EBX = 0 -> use default prompt

9069 <1> ; EBX > 0 -> prompt string (ASCIIZ) address

9070 <1> ; (Max. 11 characters except ZERO tail)

9071 <1> ; OUTPUT ->

9072 <1> ; (EAX = 0)

9073 <1> ; CF = 0 -> Successful

9074 <1> ; CF = 1 -> Failed

9075 <1>

9076 0000F668 21DB <1> and ebx, ebx

9077 0000F66A 750A <1> jnz short sysprompt\_0

9078 <1>

9079 0000F66C E8F685FFFF <1> call default\_command\_prompt ; '['+'TRDOS'+']'

9080 0000F671 E968D0FFFF <1> jmp sysret

9081 <1>

9082 <1> sysprompt\_0:

9083 0000F676 31C0 <1> xor eax, eax

9084 0000F678 A3[64030300] <1> mov [u.r0], eax

9085 0000F67D 89DE <1> mov esi, ebx

9086 0000F67F B90C000000 <1> mov ecx, 12

9087 0000F684 89E5 <1> mov ebp, esp

9088 0000F686 29CC <1> sub esp, ecx

9089 0000F688 49 <1> dec ecx ; 11

9090 0000F689 89E7 <1> mov edi, esp

9091 0000F68B E833F1FFFF <1> call transfer\_from\_user\_buffer

9092 0000F690 7211 <1> jc short sysprompt\_err

9093 0000F692 803E20 <1> cmp byte [esi], 20h

9094 0000F695 760C <1> jna short sysprompt\_err

9095 0000F697 E8DD85FFFF <1> call set\_command\_prompt

9096 0000F69C 89EC <1> mov esp, ebp

9097 0000F69E E93BD0FFFF <1> jmp sysret

9098 <1> sysprompt\_err:

9099 <1> syspath\_err:

9100 0000F6A3 89EC <1> mov esp, ebp

9101 0000F6A5 E914D0FFFF <1> jmp error

9102 <1>

9103 <1> syspath:

9104 <1> ; Get/Set Run Path

9105 <1> ; 31/12/2017 (TRDOS 386 = TRDOS v2.0)

9106 <1> ;

9107 <1> ; INPUT ->

9108 <1> ; EBX = 0 -> get path (to buffer address in ECX)

9109 <1> ; EBX > 0 -> set path

9110 <1> ; EBX = Path string buffer address (ASCIIZ)

9111 <1> ; (Path description except 'PATH=')

9112 <1> ; ECX = Buffer address (if EBX = 0)

9113 <1> ; (ECX will not be used if EBX > 0)

9114 <1> ; DL = Buffer size (0 = 256 byte)

9115 <1> ;

9116 <1> ; OUTPUT ->

9117 <1> ; CF = 0 -> Successful (EAX = String length)

9118 <1> ; CF = 1 -> Failed (EAX = 0)

9119 <1> ;

9120 <1> ; NOTE: 'PATH=' or 'PATH' must be excluded

9121 <1> ; (It must not be at the beginning of the string.)

9122 <1>

9123 0000F6AA 89E5 <1> mov ebp, esp

9124 0000F6AC 81EC00010000 <1> sub esp, 256

9125 0000F6B2 89E7 <1> mov edi, esp

9126 <1>

9127 0000F6B4 31C0 <1> xor eax, eax

9128 0000F6B6 A3[64030300] <1> mov [u.r0], eax

9129 <1>

9130 0000F6BB 21DB <1> and ebx, ebx

9131 0000F6BD 752E <1> jnz short syspath\_0

9132 <1>

9133 <1> ; EBX = 0 -> get run path

9134 0000F6BF 89CB <1> mov ebx, ecx ; buffer addr (in user's mem space)

9135 0000F6C1 BE[9F0D0100] <1> mov esi, Cmd\_Path ; 'PATH' address

9136 0000F6C6 0FB6CA <1> movzx ecx, dl

9137 0000F6C9 80E901 <1> sub cl, 1 ; 0 -> 255, 1 -> 0

9138 0000F6CC 6683D101 <1> adc cx, 1 ; 255 -> 256, 0 -> 1

9139 <1> ; EDI = Output buffer

9140 <1> ; CX = Buffer length

9141 <1> ; AL = 0 -> use ASCIIZ word in [ESI]

9142 <1> ; ESI = 'PATH' address (with zero tail)

9143 0000F6D0 E8D89DFFFF <1> call get\_environment\_string

9144 0000F6D5 72CC <1> jc short syspath\_err

9145 0000F6D7 89DF <1> mov edi, ebx ; User's buffer address

9146 0000F6D9 89E6 <1> mov esi, esp

9147 <1> ; EDI = User's buffer address

9148 <1> ; ECX = transfer (byte) count

9149 0000F6DB E899F0FFFF <1> call transfer\_to\_user\_buffer

9150 0000F6E0 72C1 <1> jc short syspath\_err

9151 0000F6E2 890D[64030300] <1> mov [u.r0], ecx

9152 0000F6E8 E9F1CFFFFF <1> jmp sysret

9153 <1>

9154 <1> syspath\_0:

9155 0000F6ED 89DE <1> mov esi, ebx

9156 0000F6EF 0FB6CA <1> movzx ecx, dl

9157 0000F6F2 80E901 <1> sub cl, 1 ; 0 -> 255, 1 -> 0

9158 0000F6F5 6683D101 <1> adc cx, 1 ; 255 -> 256, 0 -> 1

9159 0000F6F9 E8C5F0FFFF <1> call transfer\_from\_user\_buffer

9160 0000F6FE 72A3 <1> jc short syspath\_err

9161 <1> ;(\*) 'PATH=' will be added to

9162 <1> ; the head of the string

9163 0000F700 83EC08 <1> sub esp, 8 ;(\*)

9164 0000F703 89FE <1> mov esi, edi ;(\*)

9165 0000F705 E8879DFFFF <1> call set\_path\_x ;(\*)

9166 0000F70A 7297 <1> jc short syspath\_err

9167 0000F70C 8915[64030300] <1> mov [u.r0], edx ; run path string length

9168 0000F712 E9C7CFFFFF <1> jmp sysret

9169 <1>

9170 <1> sysenv:

9171 <1> ; Get/Set Environment Variables

9172 <1> ; 31/12/2017 (TRDOS 386 = TRDOS v2.0)

9173 <1> ;

9174 <1> ; INPUT ->

9175 <1> ; EBX = 0 -> get (all) environment variables

9176 <1> ; (Required Buffer length = 512 bytes)

9177 <1> ; EBX > 0 -> set (one) environment variable

9178 <1> ; (If there is not a '=' after

9179 <1> ; the environment variable name, it will

9180 <1> ; accepted as 'get environment variable'.)

9181 <1> ; EBX = Buffer address

9182 <1> ; ECX = Buffer address (if EBX = 0)

9183 <1> ; (ECX will not be used if EBX > 0)

9184 <1> ; (Note: Buffer size is 512 bytes.)

9185 <1> ; DL = Buffer size (0 = 256 byte)

9186 <1> ; (For one envrionment variable)

9187 <1> ;

9188 <1> ; OUTPUT ->

9189 <1> ; (EAX = 0)

9190 <1> ; CF = 0 -> Successful (EAX = String length)

9191 <1> ; CF = 1 -> Failed (EAX = 0)

9192 <1> ;

9193 <1> ; Note: Environment variable name, for example,

9194 <1> ; 'PATH=' must be included at the beginning

9195 <1> ; of the environment string. If the variable

9196 <1> ; name is as 'PATH' but it is not as 'PATH='

9197 <1> ; the variable string (row) will be returned.

9198 <1> ; If variable name is as 'PATH=' but there is

9199 <1> ; not a following text after the variable name,

9200 <1> ; the environment variable will be reset/deleted.

9201 <1>

9202 0000F717 89E5 <1> mov ebp, esp

9203 0000F719 81EC00020000 <1> sub esp, 512

9204 0000F71F 89E7 <1> mov edi, esp

9205 <1>

9206 0000F721 31C0 <1> xor eax, eax

9207 0000F723 A3[64030300] <1> mov [u.r0], eax

9208 <1>

9209 0000F728 21DB <1> and ebx, ebx

9210 0000F72A 7524 <1> jnz short sysenv\_0

9211 <1>

9212 <1> ; EBX = 0 -> get (all) environment variables

9213 0000F72C 89EC <1> mov esp, ebp

9214 0000F72E BE00300900 <1> mov esi, Env\_Page ; Environment page

9215 0000F733 89CF <1> mov edi, ecx ; buffer addr (in user's mem space)

9216 0000F735 B900020000 <1> mov ecx, 512

9217 0000F73A E83AF0FFFF <1> call transfer\_to\_user\_buffer

9218 0000F73F 0F8279CFFFFF <1> jc error

9219 0000F745 890D[64030300] <1> mov [u.r0], ecx

9220 0000F74B E98ECFFFFF <1> jmp sysret

9221 <1>

9222 <1> sysenv\_0:

9223 0000F750 89DE <1> mov esi, ebx ; \* ; user's buffer address

9224 0000F752 0FB6CA <1> movzx ecx, dl

9225 0000F755 80E901 <1> sub cl, 1 ; 0 -> 255, 1 -> 0

9226 0000F758 6683D101 <1> adc cx, 1 ; 255 -> 256, 0 -> 1

9227 0000F75C E862F0FFFF <1> call transfer\_from\_user\_buffer

9228 0000F761 723F <1> jc short sysenv\_err

9229 0000F763 89FE <1> mov esi, edi

9230 0000F765 8A06 <1> mov al, [esi]

9231 0000F767 3C20 <1> cmp al, 20h

9232 0000F769 7637 <1> jna short sysenv\_err

9233 0000F76B 3C3D <1> cmp al, '='

9234 0000F76D 7433 <1> je short sysenv\_err

9235 0000F76F 56 <1> push esi

9236 <1> sysenv\_1:

9237 0000F770 46 <1> inc esi

9238 0000F771 803E3D <1> cmp byte [esi], '='

9239 0000F774 7433 <1> je short sysenv\_3

9240 0000F776 803E20 <1> cmp byte [esi], 20h

9241 0000F779 73F5 <1> jnb short sysenv\_1

9242 0000F77B C60600 <1> mov byte [esi], 0

9243 0000F77E 5E <1> pop esi

9244 <1> ; EDI = Output buffer

9245 <1> ; CX = Buffer length

9246 0000F77F 30C0 <1> xor al, al

9247 <1> ; AL = 0 -> use ASCIIZ word in [ESI]

9248 <1> ; ESI = Environment variable name address

9249 0000F781 E8279DFFFF <1> call get\_environment\_string

9250 0000F786 721A <1> jc short sysenv\_err

9251 0000F788 89DF <1> mov edi, ebx ; \* ; user's buffer address

9252 0000F78A 89C1 <1> mov ecx, eax ; String length

9253 0000F78C 89E6 <1> mov esi, esp

9254 <1> ; ESI = system buffer address

9255 <1> ; EDI = User's buffer address

9256 <1> ; ECX = transfer (byte) count

9257 0000F78E E8E6EFFFFF <1> call transfer\_to\_user\_buffer

9258 0000F793 720D <1> jc short sysenv\_err

9259 0000F795 890D[64030300] <1> mov [u.r0], ecx ; transfer (byte) count

9260 <1> sysenv\_2:

9261 0000F79B 89EC <1> mov esp, ebp

9262 0000F79D E93CCFFFFF <1> jmp sysret

9263 <1> sysenv\_err:

9264 0000F7A2 89EC <1> mov esp, ebp

9265 0000F7A4 E915CFFFFF <1> jmp error

9266 <1> sysenv\_3:

9267 0000F7A9 46 <1> inc esi

9268 0000F7AA 803E20 <1> cmp byte [esi], 20h

9269 0000F7AD 73FA <1> jnb short sysenv\_3

9270 0000F7AF C60600 <1> mov byte [esi], 0

9271 0000F7B2 5E <1> pop esi

9272 0000F7B3 E8B89DFFFF <1> call set\_environment\_string

9273 0000F7B8 72E8 <1> jc short sysenv\_err

9274 0000F7BA 8915[64030300] <1> mov [u.r0], edx

9275 0000F7C0 EBD9 <1> jmp short sysenv\_2

9276 <1>

9277 <1>

9278 <1> ; temporary - 24/01/2016

9279 <1>

9280 <1> iget:

9281 0000F7C2 C3 <1> retn

9282 <1> isintr:

9283 0000F7C3 C3 <1> retn

9284 <1> iopen:

9285 0000F7C4 C3 <1> retn

9286 <1> iclose:

9287 0000F7C5 C3 <1> retn

9288 <1> sndc:

9289 0000F7C6 C3 <1> retn

9290 <1> access:

9291 0000F7C7 C3 <1> retn

9292 <1> sleep:

9293 0000F7C8 C3 <1> retn

2311 %include 'trdosk7.s' ; 24/01/2016

1 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2 <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.0) - DISK READ&WRITE : trdosk7.s

3 <1> ; ----------------------------------------------------------------------------

4 <1> ; Last Update: 25/02/2016

5 <1> ; ----------------------------------------------------------------------------

6 <1> ; Beginning: 24/01/2016

7 <1> ; ----------------------------------------------------------------------------

8 <1> ; Assembler: NASM version 2.11 (trdos386.s)

9 <1> ; ----------------------------------------------------------------------------

10 <1> ; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan

11 <1> ; DISK\_IO.ASM (20/07/2011)

12 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

13 <1> ; DISK\_IO.ASM (c) 2009-2011 Erdogan TAN [ 04/07/2009 ] Last Update: 20/07/2011

14 <1>

15 <1> disk\_write:

16 <1> ; 25/02/2016

17 <1> ; 24/02/2016

18 <1> ; 23/02/2016

19 0000F7C9 807E0500 <1> cmp byte [esi+LD\_LBAYes], 0

20 0000F7CD 777B <1> ja short lba\_write

21 <1>

22 <1> chs\_write:

23 <1> ; 25/02/2016

24 <1> ; 23/02/2016

25 0000F7CF C605[F1610100]03 <1> mov byte [disk\_rw\_op], 3 ; CHS write

26 0000F7D6 EB0D <1> jmp short chs\_rw

27 <1>

28 <1> disk\_read:

29 <1> ; 25/02/2016

30 <1> ; 24/02/2016

31 <1> ; 23/02/2016

32 <1> ; 17/02/2016

33 <1> ; 14/02/2016

34 <1> ; 31/01/2016 (TRDOS 386 = TRDOS v2.0)

35 <1> ; 17/10/2010

36 <1> ; 18/04/2010

37 <1> ;

38 <1> ; INPUT -> EAX = Logical Block Address

39 <1> ; ESI = Logical Dos Disk Table Offset (DRV)

40 <1> ; ECX = Sector Count

41 <1> ; EBX = Destination Buffer

42 <1> ; OUTPUT ->

43 <1> ; cf = 0 or cf = 1

44 <1> ; (Modified registers: EAX, EBX, ECX, EDX)

45 <1>

46 0000F7D8 807E0500 <1> cmp byte [esi+LD\_LBAYes], 0

47 0000F7DC 7775 <1> ja short lba\_read

48 <1>

49 <1> chs\_read:

50 <1> ; 25/02/2016

51 <1> ; 24/02/2016

52 <1> ; 23/02/2016

53 <1> ; 31/01/2016 (TRDOS 386 = TRDOS v2.0)

54 <1> ; 20/07/2011

55 <1> ; 04/07/2009

56 <1> ;

57 <1> ; INPUT -> EAX = Logical Block Address

58 <1> ; ECX = Number of sectors to read

59 <1> ; ESI = Logical Dos Disk Table Offset (DRV)

60 <1> ; EBX = Destination Buffer

61 <1> ; OUTPUT ->

62 <1> ; cf = 0 or cf = 1

63 <1> ; (Modified registers: EAX; EBX, ECX, EDX)

64 <1>

65 <1> ; 23/02/2016

66 0000F7DE C605[F1610100]02 <1> mov byte [disk\_rw\_op], 2 ; CHS read

67 <1>

68 <1> chs\_rw:

69 <1> ;;movzx edx, word [esi+LD\_BPB+SecPerTrack]

70 <1> ;movzx edx, byte [esi+LD\_BPB+SecPerTrack] ; <= 63

71 <1> ;mov [disk\_rw\_spt], dl

72 <1>

73 <1> chs\_read\_next\_sector:

74 0000F7E5 C605[F2610100]04 <1> mov byte [retry\_count], 4

75 <1>

76 <1> chs\_read\_retry:

77 <1> ;mov [sector\_count], ecx ; 23/02/2016

78 <1>

79 0000F7EC 50 <1> push eax ; Linear sector #

80 0000F7ED 51 <1> push ecx ; # of FAT/FILE/DIR sectors

81 <1>

82 0000F7EE 0FB74E1E <1> movzx ecx, word [esi+LD\_BPB+SecPerTrack]

83 <1> ;movzx ecx, byte [disk\_rw\_spt] ; 23/02/2016

84 0000F7F2 29D2 <1> sub edx, edx

85 0000F7F4 F7F1 <1> div ecx

86 <1> ; eax = track, dx (dl ) = sector (on track)

87 <1> ;sub cl, dl ; 24/02/2016 (spt - sec)

88 <1> ;push ecx ; \*

89 0000F7F6 6689D1 <1> mov cx, dx ; Sector (zero based)

90 0000F7F9 6641 <1> inc cx ; To make it 1 based

91 0000F7FB 6651 <1> push cx

92 0000F7FD 668B4E20 <1> mov cx, [esi+LD\_BPB+Heads]

93 0000F801 6629D2 <1> sub dx, dx

94 0000F804 F7F1 <1> div ecx ; Convert track to head & cyl

95 <1> ; eax (ax) = cylinder, dx (dl) = head (max. FFh)

96 0000F806 88D6 <1> mov dh, dl

97 0000F808 6659 <1> pop cx ; AX=Cyl, DH=Head, CX=Sector

98 0000F80A 8A5602 <1> mov dl, [esi+LD\_PhyDrvNo]

99 <1>

100 0000F80D 88C5 <1> mov ch, al ; NOTE: max. 1023 cylinders !

101 0000F80F C0CC02 <1> ror ah, 2 ; Rotate 2 bits right

102 0000F812 08E1 <1> or cl, ah

103 <1>

104 <1> ; 24/02/2016

105 <1> ;pop eax ; \* (spt - sec) (example: 63 - 0 = 63)

106 <1> ;cmp eax, [sector\_count]

107 <1> ;jb short chs\_write\_sectors

108 <1> ;je short chs\_read\_sectors

109 <1> ;; (# of sectors to read is more than remaining sectors on the track)

110 <1> ;mov al, [sector\_count]

111 <1> ;chs\_read\_sectors: ; read or write !

112 0000F814 B001 <1> mov al, 1 ; 25/02/2016

113 0000F816 8A25[F1610100] <1> mov ah, [disk\_rw\_op] ; 02h = chs read, 03h = chs write

114 <1> ;

115 0000F81C E8E549FFFF <1> call int13h ; BIOS Service func ( ah ) = 2

116 <1> ; Read disk sectors

117 <1> ; AL-sec num CH-track CL-sec

118 <1> ; DH-head DL-drive ES:BX-buffer

119 <1> ; CF-flag AH-stat AL-sec read

120 <1> ; If CF = 1 then (If AH > 0)

121 0000F821 8825[F3610100] <1> mov [disk\_rw\_err], ah

122 <1>

123 0000F827 59 <1> pop ecx

124 0000F828 58 <1> pop eax

125 0000F829 7314 <1> jnc short chs\_read\_ok

126 <1>

127 0000F82B 803D[F3610100]09 <1> cmp byte [disk\_rw\_err], 09h ; DMA crossed 64K segment boundary

128 0000F832 7408 <1> je short chs\_read\_error\_retn

129 <1>

130 0000F834 FE0D[F2610100] <1> dec byte [retry\_count]

131 0000F83A 75B0 <1> jnz short chs\_read\_retry

132 <1>

133 <1> chs\_read\_error\_retn:

134 0000F83C F9 <1> stc

135 <1> ;retn

136 0000F83D EB69 <1> jmp short update\_drv\_error\_byte

137 <1>

138 <1> ;chs\_write\_sectors: ; read or write

139 <1> ;; (# of sectors to read is less than remaining sectors on the track)

140 <1> ;mov [sector\_count], al

141 <1> ;jmp short chs\_read\_sectors

142 <1>

143 <1> chs\_read\_ok:

144 <1> ;; 23/02/2016

145 <1> ;movzx edx, byte [sector\_count] ; sector count (<= spt)

146 <1> ;sub ecx, edx ; remaining sector count

147 <1> ;jna short update\_drv\_error\_byte

148 <1> ;add eax, edx ; next disk sector

149 <1> ;shl edx, 9 ; 512 \* sector count

150 <1> ;add ebx, edx ; next buffer byte address

151 <1> ;jmp chs\_read\_next\_sector

152 <1> ; 25/02/2016

153 0000F83F 40 <1> inc eax ; next sector

154 0000F840 81C300020000 <1> add ebx, 512

155 0000F846 E29D <1> loop chs\_read\_next\_sector

156 0000F848 EB5E <1> jmp short update\_drv\_error\_byte

157 <1>

158 <1> lba\_write:

159 <1> ; 23/02/2016

160 0000F84A C605[F1610100]1C <1> mov byte [disk\_rw\_op], 1Ch ; LBA write

161 0000F851 EB07 <1> jmp short lba\_rw

162 <1>

163 <1> lba\_read:

164 <1> ; 23/02/2016

165 <1> ; 17/02/2016

166 <1> ; 14/02/2016

167 <1> ; 13/02/2016

168 <1> ; 31/01/2016 (TRDOS 386 = TRDOS v2.0)

169 <1> ; 10/07/2015 (Retro UNIX 386 v1)

170 <1> ;

171 <1> ; INPUT -> EAX = Logical Block Address

172 <1> ; ESI = Logical Dos Disk Table Offset (DRV)

173 <1> ; ECX = Sector Count

174 <1> ; EBX = Destination Buffer

175 <1> ; OUTPUT ->

176 <1> ; cf = 0 or cf = 1

177 <1> ; (Modified registers: EAX, EBX, ECX, EDX)

178 <1>

179 <1> ; LBA read/write (with private LBA function)

180 <1> ;((Retro UNIX 386 v1 - DISK I/O code by Erdogan Tan))

181 <1>

182 <1>

183 <1> ; 23/02/2016

184 0000F853 C605[F1610100]1B <1> mov byte [disk\_rw\_op], 1Bh ; LBA read

185 <1>

186 <1> lba\_rw:

187 <1> ; 17/02/2016

188 0000F85A 57 <1> push edi

189 <1>

190 0000F85B 890D[F4610100] <1> mov [sector\_count], ecx ; total sector (read) count

191 <1>

192 0000F861 8A5602 <1> mov dl, [esi+LD\_PhyDrvNo]

193 <1> ; dl = physical drive number (0,1, 80h, 81h, 82h, 83h)

194 <1>

195 <1> lba\_read\_next:

196 0000F864 81F900010000 <1> cmp ecx, 256

197 0000F86A 7605 <1> jna short lba\_read\_rsc

198 0000F86C B900010000 <1> mov ecx, 256 ; 17/02/2016

199 <1> lba\_read\_rsc:

200 0000F871 290D[F4610100] <1> sub [sector\_count], ecx ; remain sectors

201 <1>

202 0000F877 89CF <1> mov edi, ecx

203 0000F879 89C1 <1> mov ecx, eax ; sector number/address

204 <1>

205 0000F87B C605[F2610100]04 <1> mov byte [retry\_count], 4

206 <1> lba\_read\_retry:

207 0000F882 89F8 <1> mov eax, edi

208 <1> ;

209 <1> ; ecx = sector number

210 <1> ; al = sector count (0 - 255) /// (0 = 256)

211 <1> ; dl = drive number

212 <1> ; ebx = buffer offset

213 <1> ;

214 <1> ; Function 1Bh = LBA read, 1Ch = LBA write

215 <1> ; 23/02/2016

216 0000F884 8A25[F1610100] <1> mov ah, [disk\_rw\_op] ; 1Bh = LBA read, 1Ch = LBA write

217 0000F88A E87749FFFF <1> call int13h

218 <1> ; al = ? (changed)

219 <1> ; ah = error code

220 0000F88F 8825[F3610100] <1> mov [disk\_rw\_err], ah

221 0000F895 7334 <1> jnc short lba\_read\_ok

222 0000F897 80FC80 <1> cmp ah, 80h ; time out?

223 0000F89A 740A <1> je short lba\_read\_stc\_retn

224 0000F89C FE0D[F2610100] <1> dec byte [retry\_count]

225 0000F8A2 7FDE <1> jg short lba\_read\_retry

226 0000F8A4 743A <1> jz short lba\_read\_reset

227 <1> ; sf = 1

228 <1>

229 <1> lba\_read\_stc\_retn:

230 0000F8A6 F9 <1> stc

231 <1> lba\_read\_retn:

232 0000F8A7 5F <1> pop edi

233 <1>

234 <1> update\_drv\_error\_byte:

235 0000F8A8 9C <1> pushf

236 0000F8A9 53 <1> push ebx

237 0000F8AA 6651 <1> push cx

238 <1> ;or ecx, ecx

239 <1> ;jz short udrv\_errb0

240 0000F8AC 8A0D[F3610100] <1> mov cl, [disk\_rw\_err]

241 <1> udrv\_errb0:

242 0000F8B2 0FB65E02 <1> movzx ebx, byte [esi+LD\_PhyDrvNo]

243 0000F8B6 80FB02 <1> cmp bl, 2

244 0000F8B9 7203 <1> jb short udrv\_errb1

245 0000F8BB 80EB7E <1> sub bl, 7Eh

246 <1> ;cmp bl, 5

247 <1> ;ja short udrv\_errb2

248 <1> udrv\_errb1:

249 0000F8BE 81C3[495D0000] <1> add ebx, drv.error ; 13/02/2016

250 0000F8C4 880B <1> mov [ebx], cl ; error code

251 <1> udrv\_errb2:

252 0000F8C6 6659 <1> pop cx

253 0000F8C8 5B <1> pop ebx

254 0000F8C9 9D <1> popf

255 0000F8CA C3 <1> retn

256 <1>

257 <1> lba\_read\_ok:

258 0000F8CB 89C8 <1> mov eax, ecx ; sector number

259 0000F8CD 01F8 <1> add eax, edi ; sector number (next)

260 0000F8CF C1E709 <1> shl edi, 9 ; sector count \* 512

261 0000F8D2 01FB <1> add ebx, edi ; next buffer offset

262 <1>

263 0000F8D4 8B0D[F4610100] <1> mov ecx, [sector\_count] ; remaining sectors

264 0000F8DA 09C9 <1> or ecx, ecx

265 0000F8DC 7586 <1> jnz short lba\_read\_next

266 0000F8DE EBC7 <1> jmp short lba\_read\_retn

267 <1>

268 <1> lba\_read\_reset:

269 0000F8E0 B40D <1> mov ah, 0Dh ; Alternate reset

270 0000F8E2 E81F49FFFF <1> call int13h

271 <1> ; al = ? (changed)

272 <1> ; ah = error code

273 0000F8E7 7399 <1> jnc short lba\_read\_retry

274 0000F8E9 EBBC <1> jmp short lba\_read\_retn

2312 %include 'trdosk8.s' ; 24/01/2016

1 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2 <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.0) - MAIN PROGRAM : trdosk8.s

3 <1> ; ----------------------------------------------------------------------------

4 <1> ; Last Update: 30/12/2017

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> ; u0.s (20/11/2015), u4.s (14/10/2015)

12 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

13 <1> ; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan

14 <1> ; TRDOS2.ASM (09/11/2011)

15 <1> ; ----------------------------------------------------------------------------

16 <1> ; DIR.ASM (c) 2004-2011 Erdogan TAN [07/01/2004] Last Update: 09/10/2011

17 <1>

18 <1> set\_run\_sequence:

19 <1> ; 23/12/2016

20 <1> ; 10/06/2016

21 <1> ; 22/05/2016

22 <1> ; 20/05/2016

23 <1> ; 19/05/2016 - TRDOS 386 (TRDOS v2.0)

24 <1> ; TRDOS 386 feature only !

25 <1> ;

26 <1> ; INPUT ->

27 <1> ; AL = process number (next process)

28 <1> ;

29 <1> ; this process must be added to run sequence

30 <1> ;

31 <1> ; [u.pri] = priority of present process

32 <1> ;

33 <1> ; DL = priority (queue)

34 <1> ; 0 = background (low) ; run on background

35 <1> ; 1 = regular (normal) ; run as regular

36 <1> ; 2 = event (high) ; run for event

37 <1> ;

38 <1> ; 1) If the requested process is already running:

39 <1> ; a) If present priority is high ([u.pri]=2)

40 <1> ; and requested priority is also high,

41 <1> ; there is nothing to do! Because it has been

42 <1> ; done already (before this attempt).

43 <1> ; b) If present priority is high ([u.pri]=2)

44 <1> ; and requested priority is not high, there is

45 <1> ; nothing to do! Because, it's current

46 <1> ; run queue is unspecified, here. (It may be in

47 <1> ; a waiting list or in a run queue; if the new

48 <1> ; priority would be used to add it to relavant

49 <1> ; run queue, this would be wrong, unnecessary

50 <1> ; and destabilizing duplication!)

51 <1> ; c) If present priority is not high ([u.pri]<2)

52 <1> ; and requested priority is high (event),

53 <1> ; process will be added to present priority's

54 <1> ; run queue and then, priority will be changed

55 <1> ; to high ([u.pri]=2).

56 <1> ; d) If present priority is not high ([u.pri]<2)

57 <1> ; and requested priority is not high, [u.pri]

58 <1> ; value will be changed. There is nothing to do

59 <1> ; in addition. (The new priority value will be

60 <1> ; used by 'tswap/tswitch' procedure at 'sysret'

61 <1> ; or 'sysrele' stage.)

62 <1> ;

63 <1> ; 2) If the requested process is not running:

64 <1> ; a) If requested priority of the requested

65 <1> ; (next) process is high (event) and priority

66 <1> ; of present process is not high, the requested

67 <1> ; process will be added to ('runq\_event') high

68 <1> ; priority run queue and then present (running)

69 <1> ; process will be stopped (swapped/switched out)

70 <1> ; immediately if it is in user mode, or it's

71 <1> ; [u.quant] value will be reset to 0 and (then)

72 <1> ; it will be stopped at 'sysret' stage.

73 <1> ; b) If requested priority of the requested

74 <1> ; (next) process is high (event) and priority

75 <1> ; of present process is also high, the requested

76 <1> ; process will be added to ('runq\_event') high

77 <1> ; priority run queue and present (running)

78 <1> ; process will be allowed to run until it's

79 <1> ; time quantum will be elapsed ([u.quant]=0).

80 <1> ; c) If requested priority of the requested

81 <1> ; (next) process is not high ('run for event'),

82 <1> ; there is nothing to do. Because, it's current

83 <1> ; run queue is unspecified, here. (It may be in

84 <1> ; a waiting list or in a run queue; if the new

85 <1> ; priority would be used to add it to relavant

86 <1> ; run queue, this would be wrong, unnecessary

87 <1> ; and destabilizing duplication!)

88 <1> ;

89 <1> ; OUTPUT ->

90 <1> ; none

91 <1> ;

92 <1> ; [u.pri] = priority of present process

93 <1> ;

94 <1> ; cf = 1, if the request could not be fulfilled.

95 <1> ;

96 <1> ; NOTE:

97 <1> ; \* Processes in 'run as regular' queue can run

98 <1> ; if there is no process in 'run for event' queue

99 <1> ; ('run for event' processes have higher priority)

100 <1> ; \* When [u.quant] time quantum of a process is

101 <1> ; elapsed, it's high priority ('run for event')

102 <1> ; status will be disabled, it can be run in sequence

103 <1> ; of it's actual run queue.

104 <1> ; \* A 'run on background' process will always be

105 <1> ; sequenced in 'run on background' (low priority)

106 <1> ; queue, it can run only when other priority queues

107 <1> ; are empty. (idle time processes, e.g. printing)

108 <1> ;

109 <1> ; Modified registers: eax, ebx, edx

110 <1> ;

111 <1>

112 <1> srunseq\_0:

113 0000F8EB 3A05[B3030300] <1> cmp al, [u.uno] ; same process ?

114 0000F8F1 750C <1> jne short srunseq\_2 ; no

115 <1>

116 0000F8F3 8A25[A9030300] <1> mov ah, [u.pri] ; present/current priority

117 0000F8F9 80FC02 <1> cmp ah, 2 ; 'run for event' priority level

118 0000F8FC 7221 <1> jb short srunseq\_6 ; no

119 <1>

120 <1> srunseq\_1:

121 <1> ; there is nothing to do!

122 0000F8FE C3 <1> retn

123 <1>

124 <1> srunseq\_2:

125 <1> ;;this not necessary ! 23/12/2016

126 <1> ;;cmp al, nproc ; number of processes = 16

127 <1> ;;jnb short srunseq\_5 ; error ! invalid process number

128 <1>

129 <1> ; dl = priority

130 0000F8FF 80FA02 <1> cmp dl, 2 ; event queue

131 0000F902 72FA <1> jb short srunseq\_1 ; requested process is not present

132 <1> ; process and priority of requested

133 <1> ; process is not high (event),

134 <1> ; there is nothing to do!

135 <1>

136 <1> ; requested process is not present process

137 <1> ; & priority of requested process is high

138 0000F904 3A15[A9030300] <1> cmp dl, [u.pri] ; priority of present process

139 0000F90A 7606 <1> jna short srunseq\_3 ; is high, also

140 <1> ;

141 <1> ; present process will be swapped/switched out

142 0000F90C FE05[CD650100] <1> inc byte [p\_change] ; 1

143 <1>

144 <1> srunseq\_3:

145 <1> ; add process to 'runq\_event' queue for new event

146 0000F912 BB[52030300] <1> mov ebx, runq\_event ; high priority run queue

147 <1>

148 <1> srunseq\_4:

149 <1> ; al = process number

150 <1> ; ebx = run queue

151 0000F917 E881EDFFFF <1> call putlu

152 0000F91C C3 <1> retn

153 <1>

154 <1> srunseq\_5:

155 0000F91D F5 <1> cmc

156 0000F91E C3 <1> retn

157 <1>

158 <1> srunseq\_6:

159 <1> ; present priority of the process is not high

160 <1>

161 0000F91F 8815[A9030300] <1> mov [u.pri], dl ; new priority

162 <1> ; (will be used by 'tswap')

163 <1>

164 0000F925 80FA02 <1> cmp dl, 2 ; high priority ?

165 0000F928 72F3 <1> jb short srunseq\_5 ; no, there is nothing to do

166 <1> ; in addition

167 <1>

168 <1> ; process must be added to relevant run queue, here!

169 <1> ; (new priority is high/event priority and process

170 <1> ; will not be added to a run queue by 'tswap')

171 <1>

172 0000F92A BB[54030300] <1> mov ebx, runq\_normal ; 'run as regular' queue

173 <1>

174 0000F92F 20E4 <1> and ah, ah ; previous value of [u.pri]

175 0000F931 75E4 <1> jnz short srunseq\_4

176 <1>

177 0000F933 43 <1> inc ebx

178 0000F934 43 <1> inc ebx

179 <1> ; ebx = runq\_background ; 'run on backgroud' queue

180 <1>

181 0000F935 EBE0 <1> jmp short srunseq\_4

182 <1> clock:

183 <1> ; 23/05/2016

184 <1> ; 22/05/2016

185 <1> ; 20/05/2016

186 <1> ; 19/05/2016 - TRDOS 386 (TRDOS v2.0)

187 <1> ; 14/05/2015 - 14/10/2015 (Retro UNIX 386 v1)

188 <1> ; 07/12/2013 - 10/04/2014 (Retro UNIX 8086 v1)

189 <1>

190 0000F937 803D[A8030300]00 <1> cmp byte [u.quant], 0

191 0000F93E 772C <1> ja short clk\_1

192 <1> ;

193 0000F940 803D[B3030300]01 <1> cmp byte [u.uno], 1 ; /etc/init ? (for Retro UNIX 8086 & 386 v1)

194 <1> ; MainProg (Kernel's Command Interpreter)

195 <1> ; for TRDOS 386.

196 0000F947 7623 <1> jna short clk\_1 ; yes, do not swap out

197 <1> ;

198 0000F949 803D[5B030300]FF <1> cmp byte [sysflg], 0FFh ; user or system space ?

199 0000F950 7520 <1> jne short clk\_2 ; system space (sysflg <> 0FFh)

200 <1> ;

201 0000F952 66833D[AA030300]00 <1> cmp word [u.intr], 0

202 0000F95A 7616 <1> jna short clk\_2

203 <1> ;

204 <1> ; 23/05/2016

205 0000F95C 803D[CE650100]00 <1> cmp byte [multi\_tasking], 0

206 0000F963 760D <1> jna short clk\_2

207 <1> ;

208 0000F965 FE05[CD650100] <1> inc byte [p\_change] ; it is time to change running process

209 0000F96B C3 <1> retn

210 <1> clk\_1:

211 0000F96C FE0D[A8030300] <1> dec byte [u.quant]

212 <1> clk\_2:

213 0000F972 C3 <1> retn ; return to (hardware) timer interrupt routine

214 <1>

215 <1> ; 12/10/2017

216 <1> ; 15/01/2017

217 <1> ; 14/01/2017

218 <1> ; 07/01/2017

219 <1> ; 02/01/2017

220 <1> ; 17/08/2016

221 <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)

222 <1> int34h: ; #IOCTL# (I/O port access support for ring 3)

223 <1> ; 23/05/2016

224 <1> ; 20/06/2016

225 <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)

226 <1> ;

227 <1> ; INPUT ->

228 <1> ; AH = 0 -> read port (physical IO port) -byte-

229 <1> ; AH = 1 -> write port (physical IO port) -byte-

230 <1> ; AL = data byte

231 <1> ; AH = 2 -> read port (physical IO port) -word-

232 <1> ; AH = 3 -> write port (physical IO port) -word-

233 <1> ; BX = data word

234 <1> ; AH = 4 -> read port (physical IO port) -dword-

235 <1> ; AH = 5 -> write port (physical IO port) -dword-

236 <1> ; EBX = data dword

237 <1> ; ; 12/10/2017

238 <1> ; AH = 6 -> read port (physical IO port) twice -byte-

239 <1> ; AH = 7 -> write port (physical IO port) twice -byte-

240 <1> ; BX = data word

241 <1> ;

242 <1> ; DX = Port number (<= 0FFFFh)

243 <1> ;

244 <1> ; OUTPUT ->

245 <1> ; AL = data byte (in al, dx)

246 <1> ; AX = data word (in ax, dx)

247 <1> ; EAX = data dword (in eax, dx)

248 <1> ;

249 <1> ; (ECX = actual TRANSFER COUNT for string functions)

250 <1> ;

251 <1> ;

252 <1> ; Modified registers: EAX

253 <1> ;

254 <1>

255 <1> ;cmp ah, 5

256 <1> ;ja short int34h\_5 ; invalid function !

257 <1>

258 <1> ; 12/10/2017

259 0000F973 80FC07 <1> cmp ah, 7

260 0000F976 7743 <1> ja short int34h\_5 ; invalid function !

261 <1>

262 <1> ;; 15/01/2017

263 <1> ; 14/01/2017

264 <1> ; 02/01/2017

265 <1> ;;mov byte [ss:intflg], 34h ; IOCTL interrupt

266 0000F978 FB <1> sti

267 <1>

268 <1> ;sti ; enable interrupts

269 0000F979 80642408FE <1> and byte [esp+8], 11111110b ; clear carry bit of eflags register

270 <1>

271 0000F97E 80FC01 <1> cmp ah, 1

272 0000F981 7205 <1> jb short int34h\_0

273 0000F983 7705 <1> ja short int34h\_1

274 <1>

275 0000F985 EE <1> out dx, al

276 <1> ;iretd

277 0000F986 EB01 <1> jmp short int34h\_iret

278 <1>

279 <1> int34h\_0:

280 0000F988 EC <1> in al, dx

281 <1> ;iretd

282 <1> int34h\_iret:

283 <1> ;cli ; 07/01/2017

284 <1> ;; 15/01/2017

285 <1> ;;mov byte [ss:intflg], 0 ; reset

286 0000F989 CF <1> iretd

287 <1>

288 <1> int34h\_1:

289 0000F98A F6C401 <1> test ah, 1

290 0000F98D 7516 <1> jnz short int34h\_3 ; out

291 <1>

292 <1> ; in

293 0000F98F 80FC02 <1> cmp ah, 2

294 0000F992 7707 <1> ja short int34h\_2

295 <1>

296 0000F994 6689D8 <1> mov ax, bx

297 0000F997 66ED <1> in ax, dx

298 <1> ;iretd

299 0000F999 EBEE <1> jmp short int34h\_iret

300 <1>

301 <1> int34h\_2:

302 0000F99B 80FC04 <1> cmp ah, 4

303 0000F99E 772C <1> ja short int34h\_7 ; 12/10/2017

304 <1> ; ah = 4

305 0000F9A0 89D8 <1> mov eax, ebx

306 0000F9A2 ED <1> in eax, dx

307 <1> ;iretd

308 0000F9A3 EBE4 <1> jmp short int34h\_iret

309 <1>

310 <1> int34h\_3:

311 0000F9A5 80FC03 <1> cmp ah, 3

312 0000F9A8 7707 <1> ja short int34h\_4

313 <1>

314 0000F9AA 6689D8 <1> mov ax, bx

315 0000F9AD 66EF <1> out dx, ax

316 <1> ;iretd

317 0000F9AF EBD8 <1> jmp short int34h\_iret

318 <1>

319 <1> int34h\_4:

320 0000F9B1 80FC05 <1> cmp ah, 5

321 0000F9B4 770B <1> ja short int34h\_6 ; 12/10/2017

322 <1> ; ah = 5

323 0000F9B6 89D8 <1> mov eax, ebx

324 0000F9B8 EF <1> out dx, eax

325 <1> ;iretd

326 0000F9B9 EBCE <1> jmp short int34h\_iret

327 <1>

328 <1> int34h\_5:

329 0000F9BB 804C240801 <1> or byte [esp+8], 1 ; set carry bit of eflags register

330 0000F9C0 CF <1> iretd

331 <1>

332 <1> ; 12/10/2017

333 <1> int34h\_6:

334 0000F9C1 6689D8 <1> mov ax, bx

335 0000F9C4 EE <1> out dx, al

336 0000F9C5 EB00 <1> jmp short $+2

337 0000F9C7 86E0 <1> xchg ah, al

338 0000F9C9 EE <1> out dx, al

339 <1> ;xchg al, ah

340 <1> ;iretd

341 0000F9CA EB06 <1> jmp short int34h\_8

342 <1> int34h\_7:

343 0000F9CC EC <1> in al, dx

344 0000F9CD EB00 <1> jmp short $+2

345 0000F9CF 88C4 <1> mov ah, al

346 0000F9D1 EC <1> in al, dx

347 <1> int34h\_8:

348 0000F9D2 86C4 <1> xchg al, ah

349 0000F9D4 CF <1> iretd

350 <1>

351 <1>

352 <1> INT4Ah:

353 <1> ; 24/01/2016

354 <1> ; this procedure will be called by 'RTC\_INT' (in 'timer.s')

355 0000F9D5 C3 <1> retn

356 <1>

357 <1> ; u0.s

358 <1> ; Retro UNIX 386 v1 Kernel (v0.2) - SYS0.INC

359 <1> ; Last Modification: 20/11/2015

360 <1>

361 <1> com2\_int:

362 <1> ; 07/11/2015

363 <1> ; 24/10/2015

364 <1> ; 23/10/2015

365 <1> ; 14/03/2015 (Retro UNIX 386 v1 - Beginning)

366 <1> ; 28/07/2014 (Retro UNIX 8086 v1)

367 <1> ; < serial port 2 interrupt handler >

368 <1> ;

369 0000F9D6 890424 <1> mov [esp], eax ; overwrite call return address

370 <1> ;push eax

371 0000F9D9 66B80900 <1> mov ax, 9

372 0000F9DD EB07 <1> jmp short comm\_int

373 <1> com1\_int:

374 <1> ; 07/11/2015

375 <1> ; 24/10/2015

376 0000F9DF 890424 <1> mov [esp], eax ; overwrite call return address

377 <1> ; 23/10/2015

378 <1> ;push eax

379 0000F9E2 66B80800 <1> mov ax, 8

380 <1> comm\_int:

381 <1> ; 20/11/2015

382 <1> ; 18/11/2015

383 <1> ; 17/11/2015

384 <1> ; 16/11/2015

385 <1> ; 09/11/2015

386 <1> ; 08/11/2015

387 <1> ; 07/11/2015

388 <1> ; 06/11/2015 (serial4.asm, 'serial')

389 <1> ; 01/11/2015

390 <1> ; 26/10/2015

391 <1> ; 23/10/2015

392 0000F9E6 53 <1> push ebx

393 0000F9E7 56 <1> push esi

394 0000F9E8 57 <1> push edi

395 0000F9E9 1E <1> push ds

396 0000F9EA 06 <1> push es

397 <1> ; 18/11/2015

398 0000F9EB 0F20DB <1> mov ebx, cr3

399 0000F9EE 53 <1> push ebx ; \*\*\*\*

400 <1> ;

401 0000F9EF 51 <1> push ecx ; \*\*\*

402 0000F9F0 52 <1> push edx ; \*\*

403 <1> ;

404 0000F9F1 BB10000000 <1> mov ebx, KDATA

405 0000F9F6 8EDB <1> mov ds, bx

406 0000F9F8 8EC3 <1> mov es, bx

407 <1> ;

408 0000F9FA 8B0D[38580100] <1> mov ecx, [k\_page\_dir]

409 0000FA00 0F22D9 <1> mov cr3, ecx

410 <1> ; 20/11/2015

411 <1> ; Interrupt identification register

412 0000FA03 66BAFA02 <1> mov dx, 2FAh ; COM2

413 <1> ;

414 0000FA07 3C08 <1> cmp al, 8

415 0000FA09 7702 <1> ja short com\_i0

416 <1> ;

417 <1> ; 20/11/2015

418 <1> ; 17/11/2015

419 <1> ; 16/11/2015

420 <1> ; 15/11/2015

421 <1> ; 24/10/2015

422 <1> ; 14/03/2015 (Retro UNIX 386 v1 - Beginning)

423 <1> ; 28/07/2014 (Retro UNIX 8086 v1)

424 <1> ; < serial port 1 interrupt handler >

425 <1> ;

426 0000FA0B FEC6 <1> inc dh ; 3FAh ; COM1 Interrupt id. register

427 <1> com\_i0:

428 <1> ;push eax ; \*

429 <1> ; 07/11/2015

430 0000FA0D A2[A2580100] <1> mov byte [ccomport], al

431 <1> ; 09/11/2015

432 0000FA12 0FB7D8 <1> movzx ebx, ax ; 8 or 9

433 <1> ; 17/11/2015

434 <1> ; reset request for response status

435 0000FA15 88A3[98580100] <1> mov [ebx+req\_resp-8], ah ; 0

436 <1> ;

437 <1> ; 20/11/2015

438 0000FA1B EC <1> in al, dx ; read interrupt id. register

439 0000FA1C EB00 <1> JMP $+2 ; I/O DELAY

440 0000FA1E 2404 <1> and al, 4 ; received data available?

441 0000FA20 7470 <1> jz short com\_eoi ; (transmit. holding reg. empty)

442 <1> ;

443 <1> ; 20/11/2015

444 0000FA22 80EA02 <1> sub dl, 3FAh-3F8h ; data register (3F8h, 2F8h)

445 0000FA25 EC <1> in al, dx ; read character

446 <1> ;JMP $+2 ; I/O DELAY

447 <1> ; 08/11/2015

448 <1> ; 07/11/2015

449 0000FA26 89DE <1> mov esi, ebx

450 0000FA28 89DF <1> mov edi, ebx

451 0000FA2A 81C6[9C580100] <1> add esi, rchar - 8 ; points to last received char

452 0000FA30 81C7[9E580100] <1> add edi, schar - 8 ; points to last sent char

453 0000FA36 8806 <1> mov [esi], al ; received char (current char)

454 <1> ; query

455 0000FA38 20C0 <1> and al, al

456 0000FA3A 7527 <1> jnz short com\_i2

457 <1> ; response

458 <1> ; 17/11/2015

459 <1> ; set request for response status

460 0000FA3C FE83[98580100] <1> inc byte [ebx+req\_resp-8] ; 1

461 <1> ;

462 0000FA42 6683C205 <1> add dx, 3FDh-3F8h ; (3FDh, 2FDh)

463 0000FA46 EC <1> in al, dx ; read line status register

464 0000FA47 EB00 <1> JMP $+2 ; I/O DELAY

465 0000FA49 2420 <1> and al, 20h ; transmitter holding reg. empty?

466 0000FA4B 7445 <1> jz short com\_eoi ; no

467 0000FA4D B0FF <1> mov al, 0FFh ; response

468 0000FA4F 6683EA05 <1> sub dx, 3FDh-3F8h ; data port (3F8h, 2F8h)

469 0000FA53 EE <1> out dx, al ; send on serial port

470 <1> ; 17/11/2015

471 0000FA54 803F00 <1> cmp byte [edi], 0 ; query ? (schar)

472 0000FA57 7502 <1> jne short com\_i1 ; no

473 0000FA59 8807 <1> mov [edi], al ; 0FFh (responded)

474 <1> com\_i1:

475 <1> ; 17/11/2015

476 <1> ; reset request for response status (again)

477 0000FA5B FE8B[98580100] <1> dec byte [ebx+req\_resp-8] ; 0

478 0000FA61 EB2F <1> jmp short com\_eoi

479 <1> com\_i2:

480 <1> ; 08/11/2015

481 0000FA63 3CFF <1> cmp al, 0FFh ; (response ?)

482 0000FA65 7417 <1> je short com\_i3 ; (check for response signal)

483 <1> ; 07/11/2015

484 0000FA67 3C04 <1> cmp al, 04h ; EOT

485 0000FA69 751C <1> jne short com\_i4

486 <1> ; EOT = 04h (End of Transmit) - 'CTRL + D'

487 <1> ;(an EOT char is supposed as a ctrl+brk from the terminal)

488 <1> ; 08/11/2015

489 <1> ; ptty -> tty 0 to 7 (pseudo screens)

490 0000FA6B 861D[66580100] <1> xchg bl, [ptty] ; tty number (8 or 9)

491 0000FA71 E86069FFFF <1> call ctrlbrk

492 0000FA76 861D[66580100] <1> xchg [ptty], bl ; (restore ptty value and BL value)

493 <1> ;mov al, 04h ; EOT

494 <1> ; 08/11/2015

495 0000FA7C EB09 <1> jmp short com\_i4

496 <1> com\_i3:

497 <1> ; 08/11/2015

498 <1> ; If 0FFh has been received just after a query

499 <1> ; (schar, ZERO), it is a response signal.

500 <1> ; 17/11/2015

501 0000FA7E 803F00 <1> cmp byte [edi], 0 ; query ? (schar)

502 0000FA81 7704 <1> ja short com\_i4 ; no

503 <1> ; reset query status (schar)

504 0000FA83 8807 <1> mov [edi], al ; 0FFh

505 0000FA85 FEC0 <1> inc al ; 0

506 <1> com\_i4:

507 <1> ; 27/07/2014

508 <1> ; 09/07/2014

509 0000FA87 D0E3 <1> shl bl, 1

510 0000FA89 81C3[68580100] <1> add ebx, ttychr

511 <1> ; 23/07/2014 (always overwrite)

512 <1> ;;cmp word [ebx], 0

513 <1> ;;ja short com\_eoi

514 <1> ;

515 0000FA8F 668903 <1> mov [ebx], ax ; Save ascii code

516 <1> ; scan code = 0

517 <1> com\_eoi:

518 <1> ;mov al, 20h

519 <1> ;out 20h, al ; end of interrupt

520 <1> ;

521 <1> ; 07/11/2015

522 <1> ;pop eax ; \*

523 0000FA92 A0[A2580100] <1> mov al, byte [ccomport] ; current COM port

524 <1> ; al = tty number (8 or 9)

525 0000FA97 E85E010000 <1> call wakeup

526 <1> com\_iret:

527 <1> ; 23/10/2015

528 0000FA9C 5A <1> pop edx ; \*\*

529 0000FA9D 59 <1> pop ecx ; \*\*\*

530 <1> ; 18/11/2015

531 <1> ;pop eax ; \*\*\*\*

532 <1> ;mov cr3, eax

533 <1> ;jmp iiret

534 0000FA9E E93D10FFFF <1> jmp iiretp

535 <1>

536 <1> ;iiretp: ; 01/09/2015

537 <1> ; ; 28/08/2015

538 <1> ; pop eax ; (\*) page directory

539 <1> ; mov cr3, eax

540 <1> ;iiret:

541 <1> ; ; 22/08/2014

542 <1> ; mov al, 20h ; END OF INTERRUPT COMMAND TO 8259

543 <1> ; out 20h, al ; 8259 PORT

544 <1> ; ;

545 <1> ; pop es

546 <1> ; pop ds

547 <1> ; pop edi

548 <1> ; pop esi

549 <1> ; pop ebx ; 29/08/2014

550 <1> ; pop eax

551 <1> ; iretd

552 <1>

553 <1> sp\_init:

554 <1> ; 07/11/2015

555 <1> ; 29/10/2015

556 <1> ; 26/10/2015

557 <1> ; 23/10/2015

558 <1> ; 29/06/2015

559 <1> ; 14/03/2015 (Retro UNIX 386 v1 - 115200 baud)

560 <1> ; 28/07/2014 (Retro UNIX 8086 v1 - 9600 baud)

561 <1> ; Initialization of Serial Port Communication Parameters

562 <1> ; (COM1 base port address = 3F8h, COM1 Interrupt = IRQ 4)

563 <1> ; (COM2 base port address = 2F8h, COM1 Interrupt = IRQ 3)

564 <1> ;

565 <1> ; ((Modified registers: EAX, ECX, EDX, EBX))

566 <1> ;

567 <1> ; INPUT: (29/06/2015)

568 <1> ; AL = 0 for COM1

569 <1> ; 1 for COM2

570 <1> ; AH = Communication parameters

571 <1> ;

572 <1> ; (\*) Communication parameters (except BAUD RATE):

573 <1> ; Bit 4 3 2 1 0

574 <1> ; -PARITY-- STOP BIT -WORD LENGTH-

575 <1> ; this one --> 00 = none 0 = 1 bit 11 = 8 bits

576 <1> ; 01 = odd 1 = 2 bits 10 = 7 bits

577 <1> ; 11 = even

578 <1> ; Baud rate setting bits: (29/06/2015)

579 <1> ; Retro UNIX 386 v1 feature only !

580 <1> ; Bit 7 6 5 | Baud rate

581 <1> ; ------------------------

582 <1> ; value 0 0 0 | Default (Divisor = 1)

583 <1> ; 0 0 1 | 9600 (12)

584 <1> ; 0 1 0 | 19200 (6)

585 <1> ; 0 1 1 | 38400 (3)

586 <1> ; 1 0 0 | 14400 (8)

587 <1> ; 1 0 1 | 28800 (4)

588 <1> ; 1 1 0 | 57600 (2)

589 <1> ; 1 1 1 | 115200 (1)

590 <1>

591 <1> ; References:

592 <1> ; (1) IBM PC-XT Model 286 BIOS Source Code

593 <1> ; RS232.ASM --- 10/06/1985 COMMUNICATIONS BIOS (RS232)

594 <1> ; (2) Award BIOS 1999 - ATORGS.ASM

595 <1> ; (3) http://wiki.osdev.org/Serial\_Ports

596 <1> ;

597 <1> ; Set communication parameters for COM1 (= 03h)

598 <1> ;

599 0000FAA3 BB[9E580100] <1> mov ebx, com1p ; COM1 parameters

600 0000FAA8 66BAF803 <1> mov dx, 3F8h ; COM1

601 <1> ; 29/10/2015

602 0000FAAC 66B90103 <1> mov cx, 301h ; divisor = 1 (115200 baud)

603 0000FAB0 E86F000000 <1> call sp\_i3 ; call A4

604 0000FAB5 A880 <1> test al, 80h

605 0000FAB7 7410 <1> jz short sp\_i0 ; OK..

606 <1> ; Error !

607 <1> ;mov dx, 3F8h

608 0000FAB9 80EA05 <1> sub dl, 5 ; 3FDh -> 3F8h

609 0000FABC 66B90E03 <1> mov cx, 30Eh ; divisor = 12 (9600 baud)

610 0000FAC0 E85F000000 <1> call sp\_i3 ; call A4

611 0000FAC5 A880 <1> test al, 80h

612 0000FAC7 7508 <1> jnz short sp\_i1

613 <1> sp\_i0:

614 <1> ; (Note: Serial port interrupts will be disabled here...)

615 <1> ; (INT 14h initialization code disables interrupts.)

616 <1> ;

617 0000FAC9 C603E3 <1> mov byte [ebx], 0E3h ; 11100011b

618 0000FACC E8DC000000 <1> call sp\_i5 ; 29/06/2015

619 <1> sp\_i1:

620 0000FAD1 43 <1> inc ebx

621 0000FAD2 66BAF802 <1> mov dx, 2F8h ; COM2

622 <1> ; 29/10/2015

623 0000FAD6 66B90103 <1> mov cx, 301h ; divisor = 1 (115200 baud)

624 0000FADA E845000000 <1> call sp\_i3 ; call A4

625 0000FADF A880 <1> test al, 80h

626 0000FAE1 7410 <1> jz short sp\_i2 ; OK..

627 <1> ; Error !

628 <1> ;mov dx, 2F8h

629 0000FAE3 80EA05 <1> sub dl, 5 ; 2FDh -> 2F8h

630 0000FAE6 66B90E03 <1> mov cx, 30Eh ; divisor = 12 (9600 baud)

631 0000FAEA E835000000 <1> call sp\_i3 ; call A4

632 0000FAEF A880 <1> test al, 80h

633 0000FAF1 7530 <1> jnz short sp\_i7

634 <1> sp\_i2:

635 0000FAF3 C603E3 <1> mov byte [ebx], 0E3h ; 11100011b

636 <1> sp\_i6:

637 <1> ;; COM2 - enabling IRQ 3

638 <1> ; 07/11/2015

639 <1> ; 26/10/2015

640 0000FAF6 9C <1> pushf

641 0000FAF7 FA <1> cli

642 <1> ;

643 0000FAF8 66BAFC02 <1> mov dx, 2FCh ; modem control register

644 0000FAFC EC <1> in al, dx ; read register

645 0000FAFD EB00 <1> JMP $+2 ; I/O DELAY

646 0000FAFF 0C08 <1> or al, 8 ; enable bit 3 (OUT2)

647 0000FB01 EE <1> out dx, al ; write back to register

648 0000FB02 EB00 <1> JMP $+2 ; I/O DELAY

649 0000FB04 66BAF902 <1> mov dx, 2F9h ; interrupt enable register

650 0000FB08 EC <1> in al, dx ; read register

651 0000FB09 EB00 <1> JMP $+2 ; I/O DELAY

652 <1> ;or al, 1 ; receiver data interrupt enable and

653 0000FB0B 0C03 <1> or al, 3 ; transmitter empty interrupt enable

654 0000FB0D EE <1> out dx, al ; write back to register

655 0000FB0E EB00 <1> JMP $+2 ; I/O DELAY

656 0000FB10 E421 <1> in al, 21h ; read interrupt mask register

657 0000FB12 EB00 <1> JMP $+2 ; I/O DELAY

658 0000FB14 24F7 <1> and al, 0F7h ; enable IRQ 3 (COM2)

659 0000FB16 E621 <1> out 21h, al ; write back to register

660 <1> ;

661 <1> ; 23/10/2015

662 0000FB18 B8[D6F90000] <1> mov eax, com2\_int

663 0000FB1D A3[F5FB0000] <1> mov [com2\_irq3], eax

664 <1> ; 26/10/2015

665 0000FB22 9D <1> popf

666 <1> sp\_i7:

667 0000FB23 C3 <1> retn

668 <1>

669 <1> sp\_i3:

670 <1> ;A4: ;----- INITIALIZE THE COMMUNICATIONS PORT

671 <1> ; 28/10/2015

672 0000FB24 FEC2 <1> inc dl ; 3F9h (2F9h) ; 3F9h, COM1 Interrupt enable register

673 0000FB26 B000 <1> mov al, 0

674 0000FB28 EE <1> out dx, al ; disable serial port interrupt

675 0000FB29 EB00 <1> JMP $+2 ; I/O DELAY

676 0000FB2B 80C202 <1> add dl, 2 ; 3FBh (2FBh) ; COM1 Line control register (3FBh)

677 0000FB2E B080 <1> mov al, 80h

678 0000FB30 EE <1> out dx, al ; SET DLAB=1 ; divisor latch access bit

679 <1> ;----- SET BAUD RATE DIVISOR

680 <1> ; 26/10/2015

681 0000FB31 80EA03 <1> sub dl, 3 ; 3F8h (2F8h) ; register for least significant byte

682 <1> ; of the divisor value

683 0000FB34 88C8 <1> mov al, cl ; 1

684 0000FB36 EE <1> out dx, al ; 1 = 115200 baud (Retro UNIX 386 v1)

685 <1> ; 2 = 57600 baud

686 <1> ; 3 = 38400 baud

687 <1> ; 6 = 19200 baud

688 <1> ; 12 = 9600 baud (Retro UNIX 8086 v1)

689 0000FB37 EB00 <1> JMP $+2 ; I/O DELAY

690 0000FB39 28C0 <1> sub al, al

691 0000FB3B FEC2 <1> inc dl ; 3F9h (2F9h) ; register for most significant byte

692 <1> ; of the divisor value

693 0000FB3D EE <1> out dx, al ; 0

694 0000FB3E EB00 <1> JMP $+2 ; I/O DELAY

695 <1> ;

696 0000FB40 88E8 <1> mov al, ch ; 3 ; 8 data bits, 1 stop bit, no parity

697 <1> ;and al, 1Fh ; Bits 0,1,2,3,4

698 0000FB42 80C202 <1> add dl, 2 ; 3FBh (2FBh) ; Line control register

699 0000FB45 EE <1> out dx, al

700 0000FB46 EB00 <1> JMP $+2 ; I/O DELAY

701 <1> ; 29/10/2015

702 0000FB48 FECA <1> dec dl ; 3FAh (2FAh) ; FIFO Control register (16550/16750)

703 0000FB4A 30C0 <1> xor al, al ; 0

704 0000FB4C EE <1> out dx, al ; Disable FIFOs (reset to 8250 mode)

705 0000FB4D EB00 <1> JMP $+2

706 <1> sp\_i4:

707 <1> ;A18: ;----- COMM PORT STATUS ROUTINE

708 <1> ; 29/06/2015 (line status after modem status)

709 0000FB4F 80C204 <1> add dl, 4 ; 3FEh (2FEh) ; Modem status register

710 <1> sp\_i4s:

711 0000FB52 EC <1> in al, dx ; GET MODEM CONTROL STATUS

712 0000FB53 EB00 <1> JMP $+2 ; I/O DELAY

713 0000FB55 88C4 <1> mov ah, al ; PUT IN (AH) FOR RETURN

714 0000FB57 FECA <1> dec dl ; 3FDh (2FDh) ; POINT TO LINE STATUS REGISTER

715 <1> ; dx = 3FDh for COM1, 2FDh for COM2

716 0000FB59 EC <1> in al, dx ; GET LINE CONTROL STATUS

717 <1> ; AL = Line status, AH = Modem status

718 0000FB5A C3 <1> retn

719 <1>

720 <1> sp\_status:

721 <1> ; 29/06/2015

722 <1> ; 27/06/2015 (Retro UNIX 386 v1)

723 <1> ; Get serial port status

724 0000FB5B 66BAFE03 <1> mov dx, 3FEh ; Modem status register (COM1)

725 0000FB5F 28C6 <1> sub dh, al ; dh = 2 for COM2 (al = 1)

726 <1> ; dx = 2FEh for COM2

727 0000FB61 EBEF <1> jmp short sp\_i4s

728 <1>

729 <1> sp\_setp: ; Set serial port communication parameters

730 <1> ; 07/11/2015

731 <1> ; 29/10/2015

732 <1> ; 29/06/2015

733 <1> ; Retro UNIX 386 v1 feature only !

734 <1> ;

735 <1> ; INPUT:

736 <1> ; AL = 0 for COM1

737 <1> ; 1 for COM2

738 <1> ; AH = Communication parameters (\*)

739 <1> ; OUTPUT:

740 <1> ; CL = Line status

741 <1> ; CH = Modem status

742 <1> ; If cf = 1 -> Error code in [u.error]

743 <1> ; 'invalid parameter !'

744 <1> ; or

745 <1> ; 'device not ready !' error

746 <1> ;

747 <1> ; (\*) Communication parameters (except BAUD RATE):

748 <1> ; Bit 4 3 2 1 0

749 <1> ; -PARITY-- STOP BIT -WORD LENGTH-

750 <1> ; this one --> 00 = none 0 = 1 bit 11 = 8 bits

751 <1> ; 01 = odd 1 = 2 bits 10 = 7 bits

752 <1> ; 11 = even

753 <1> ; Baud rate setting bits: (29/06/2015)

754 <1> ; Retro UNIX 386 v1 feature only !

755 <1> ; Bit 7 6 5 | Baud rate

756 <1> ; ------------------------

757 <1> ; value 0 0 0 | Default (Divisor = 1)

758 <1> ; 0 0 1 | 9600 (12)

759 <1> ; 0 1 0 | 19200 (6)

760 <1> ; 0 1 1 | 38400 (3)

761 <1> ; 1 0 0 | 14400 (8)

762 <1> ; 1 0 1 | 28800 (4)

763 <1> ; 1 1 0 | 57600 (2)

764 <1> ; 1 1 1 | 115200 (1)

765 <1> ;

766 <1> ; (COM1 base port address = 3F8h, COM1 Interrupt = IRQ 4)

767 <1> ; (COM2 base port address = 2F8h, COM1 Interrupt = IRQ 3)

768 <1> ;

769 <1> ; ((Modified registers: EAX, ECX, EDX, EBX))

770 <1> ;

771 0000FB63 66BAF803 <1> mov dx, 3F8h

772 0000FB67 BB[9E580100] <1> mov ebx, com1p ; COM1 control byte offset

773 0000FB6C 3C01 <1> cmp al, 1

774 0000FB6E 776B <1> ja short sp\_invp\_err

775 0000FB70 7203 <1> jb short sp\_setp1 ; COM1 (AL = 0)

776 0000FB72 FECE <1> dec dh ; 2F8h

777 0000FB74 43 <1> inc ebx ; COM2 control byte offset

778 <1> sp\_setp1:

779 <1> ; 29/10/2015

780 0000FB75 8823 <1> mov [ebx], ah

781 0000FB77 0FB6CC <1> movzx ecx, ah

782 0000FB7A C0E905 <1> shr cl, 5 ; -> baud rate index

783 0000FB7D 80E41F <1> and ah, 1Fh ; communication parameters except baud rate

784 0000FB80 8A81[EAFB0000] <1> mov al, [ecx+b\_div\_tbl]

785 0000FB86 6689C1 <1> mov cx, ax

786 0000FB89 E896FFFFFF <1> call sp\_i3

787 0000FB8E 6689C1 <1> mov cx, ax ; CL = Line status, CH = Modem status

788 0000FB91 A880 <1> test al, 80h

789 0000FB93 740F <1> jz short sp\_setp2

790 0000FB95 C603E3 <1> mov byte [ebx], 0E3h ; Reset to initial value (11100011b)

791 <1> stp\_dnr\_err:

792 0000FB98 C705[C8030300]0F00- <1> mov dword [u.error], ERR\_DEV\_NOT\_RDY ; 'device not ready !'

792 0000FBA0 0000 <1>

793 <1> ; CL = Line status, CH = Modem status

794 0000FBA2 F9 <1> stc

795 0000FBA3 C3 <1> retn

796 <1> sp\_setp2:

797 0000FBA4 80FE02 <1> cmp dh, 2 ; COM2 (2F?h)

798 0000FBA7 0F8649FFFFFF <1> jna sp\_i6

799 <1> ; COM1 (3F?h)

800 <1> sp\_i5:

801 <1> ; 07/11/2015

802 <1> ; 26/10/2015

803 <1> ; 29/06/2015

804 <1> ;

805 <1> ;; COM1 - enabling IRQ 4

806 0000FBAD 9C <1> pushf

807 0000FBAE FA <1> cli

808 0000FBAF 66BAFC03 <1> mov dx, 3FCh ; modem control register

809 0000FBB3 EC <1> in al, dx ; read register

810 0000FBB4 EB00 <1> JMP $+2 ; I/O DELAY

811 0000FBB6 0C08 <1> or al, 8 ; enable bit 3 (OUT2)

812 0000FBB8 EE <1> out dx, al ; write back to register

813 0000FBB9 EB00 <1> JMP $+2 ; I/O DELAY

814 0000FBBB 66BAF903 <1> mov dx, 3F9h ; interrupt enable register

815 0000FBBF EC <1> in al, dx ; read register

816 0000FBC0 EB00 <1> JMP $+2 ; I/O DELAY

817 <1> ;or al, 1 ; receiver data interrupt enable and

818 0000FBC2 0C03 <1> or al, 3 ; transmitter empty interrupt enable

819 0000FBC4 EE <1> out dx, al ; write back to register

820 0000FBC5 EB00 <1> JMP $+2 ; I/O DELAY

821 0000FBC7 E421 <1> in al, 21h ; read interrupt mask register

822 0000FBC9 EB00 <1> JMP $+2 ; I/O DELAY

823 0000FBCB 24EF <1> and al, 0EFh ; enable IRQ 4 (COM1)

824 0000FBCD E621 <1> out 21h, al ; write back to register

825 <1> ;

826 <1> ; 23/10/2015

827 0000FBCF B8[DFF90000] <1> mov eax, com1\_int

828 0000FBD4 A3[F1FB0000] <1> mov [com1\_irq4], eax

829 <1> ; 26/10/2015

830 0000FBD9 9D <1> popf

831 0000FBDA C3 <1> retn

832 <1>

833 <1> sp\_invp\_err:

834 0000FBDB C705[C8030300]1700- <1> mov dword [u.error], ERR\_INV\_PARAMETER ; 'invalid parameter !'

834 0000FBE3 0000 <1>

835 0000FBE5 31C9 <1> xor ecx, ecx

836 0000FBE7 49 <1> dec ecx ; 0FFFFh

837 0000FBE8 F9 <1> stc

838 0000FBE9 C3 <1> retn

839 <1>

840 <1> ; 29/10/2015

841 <1> b\_div\_tbl: ; Baud rate divisor table (115200/divisor)

842 0000FBEA 010C0603080401 <1> db 1, 12, 6, 3, 8, 4, 1

843 <1>

844 <1>

845 <1> ; 23/10/2015

846 <1> com1\_irq4:

847 0000FBF1 [F9FB0000] <1> dd dummy\_retn

848 <1> com2\_irq3:

849 0000FBF5 [F9FB0000] <1> dd dummy\_retn

850 <1>

851 <1> dummy\_retn:

852 0000FBF9 C3 <1> retn

853 <1>

854 <1> wakeup:

855 <1> ; 24/01/2016

856 0000FBFA C3 <1> retn

857 <1>

858 <1> set\_working\_path\_x:

859 <1> ; 17/10/2016 (TRDOS 386 - FFF & FNF)

860 0000FBFB 66B80100 <1> mov ax, 1

861 <1> ; File name is needed/forced (AL=1)

862 <1> ; Change directory as temporary (AH=0)

863 <1>

864 <1> set\_working\_path\_xx: ; 30/12/2017 (syschdir)

865 <1> ; This is needed for preventing wrong Find Next File

866 <1> ; system call after sysopen, syscreate, sysmkdir etc.

867 <1> ; Find Next File must immediate follow Find First File)

868 <1>

869 0000FBFF 8825[F0650100] <1> mov [FFF\_Valid], ah ; 0 ; reset ; 17/10/2016

870 <1>

871 <1> set\_working\_path:

872 <1> ; 16/10/2016

873 <1> ; 12/10/2016

874 <1> ; 10/10/2016

875 <1> ; 05/10/2016 - TRDOS 386 (TRDOS v2.0)

876 <1> ;

877 <1> ; TRDOS v1.0 (DIR.ASM, "proc\_set\_working\_path")

878 <1> ; 27/01/2011 - 08/02/2011

879 <1> ; Set/Changes current drive, directory and file

880 <1> ; depending on command tail

881 <1> ; (procedure is derivated from CMD\_INTR.ASM

882 <1> ; file or dir locating code of internal commands)

883 <1> ; (This procedure is prepared for INT 21H file/dir

884 <1> ; functions and also to get compact code for

885 <1> ; internal mainprog -command interpreter- commands)

886 <1> ;

887 <1> ; INPUT: DS:SI -> Command tail (ASCIIZ string)

888 <1> ; AL = 0 -> any, AL > 0 -> file name is forced

889 <1> ; AH = CD -> Change directory permanently

890 <1> ; AH <> CD -> Change directory as temporary

891 <1> ;

892 <1> ; OUTPUT: ES=DS, FindFile structure has been set

893 <1> ; RUN\_CDRV points previous current drive

894 <1> ; DS:SI = FindFile structure address

895 <1> ; (DS=CS)

896 <1> ; AX, BX, CX, DX, DI will be changed

897 <1> ; cf = 1 -> Error code in AX (AL)

898 <1> ; stc & AX = 0 -> Bad command or path name

899 <1> ; -----------------------------------------------

900 <1> ;

901 <1> ; TRDOS 386 (05/10/2016)

902 <1> ; INPUT:

903 <1> ; ESI = File/Directory Path (ASCIIZ string)

904 <1> ; address in user's memory space

905 <1> ; AL = 0 -> any

906 <1> ; AL > 0 -> file name is forced

907 <1> ; AH = CD -> change directory as permanent

908 <1> ; AH <> CD -> change directory as temporary

909 <1> ;

910 <1> ; OUTPUT:

911 <1> ; FindFile structure has been set

912 <1> ; RUN\_CDRV points previous current drive

913 <1> ; ESI = FindFile\_Name address ; 12/10/2016

914 <1> ;

915 <1> ; cf = 1 -> Error code in EAX (AL)

916 <1> ; stc & EAX = 0 -> Bad command or path name

917 <1> ;

918 <1> ; Modified registers: EAX, EBX, ECX, EDX, ESI, EDI

919 <1>

920 0000FC05 66A3[F4650100] <1> mov [SWP\_Mode], ax

921 0000FC0B A0[FE580100] <1> mov al, [Current\_Drv]

922 0000FC10 30E4 <1> xor ah, ah

923 0000FC12 66A3[F6650100] <1> mov [SWP\_DRV], ax

924 <1>

925 <1> ; TRDOS 386 ring 3 (user's page directory)

926 <1> ; to ring 0 (kernel's page directory)

927 <1> ; transfer modifications (05/10/2016).

928 <1>

929 0000FC18 55 <1> push ebp

930 0000FC19 89E5 <1> mov ebp, esp

931 <1>

932 0000FC1B B980000000 <1> mov ecx, 128 ; maximum path length = 128 bytes

933 0000FC20 29CC <1> sub esp, ecx ; reserve 128 bytes (buffer) on stack

934 0000FC22 89E7 <1> mov edi, esp ; destination address (kernel space)

935 <1> ; esi = source address (virtual, in user's memory space)

936 0000FC24 E89AEBFFFF <1> call transfer\_from\_user\_buffer

937 0000FC29 720A <1> jc short loc\_swp\_xor\_retn

938 <1>

939 0000FC2B 89E6 <1> mov esi, esp ; temporary buffer (the path) on stack

940 <1> loc\_swp\_fchar:

941 0000FC2D 8A06 <1> mov al, [esi]

942 0000FC2F 3C20 <1> cmp al, 20h

943 0000FC31 7711 <1> ja short loc\_swp\_parse\_path\_name

944 0000FC33 740C <1> je short loc\_swp\_fchar\_next

945 <1>

946 <1> loc\_swp\_xor\_retn:

947 0000FC35 31C0 <1> xor eax, eax

948 0000FC37 F9 <1> stc

949 <1> loc\_swp\_retn:

950 0000FC38 89EC <1> mov esp, ebp

951 0000FC3A 5D <1> pop ebp

952 <1>

953 <1> ;mov esi, FindFile\_Drv

954 0000FC3B BE[E4620100] <1> mov esi, FindFile\_Name ; 12/10/2016

955 0000FC40 C3 <1> retn

956 <1>

957 <1> loc\_swp\_fchar\_next:

958 0000FC41 46 <1> inc esi

959 0000FC42 EBE9 <1> jmp short loc\_swp\_fchar

960 <1>

961 <1> loc\_swp\_parse\_path\_name:

962 0000FC44 BF[A2620100] <1> mov edi, FindFile\_Drv

963 0000FC49 E8E6A7FFFF <1> call parse\_path\_name

964 0000FC4E 72E8 <1> jc short loc\_swp\_retn

965 <1>

966 <1> loc\_swp\_checkfile\_name:

967 0000FC50 803D[F4650100]00 <1> cmp byte [SWP\_Mode], 0

968 0000FC57 761E <1> jna short loc\_swp\_drv

969 <1>

970 <1> ; 10/10/2016 (valid file name checking)

971 0000FC59 BE[E4620100] <1> mov esi, FindFile\_Name

972 0000FC5E 803E20 <1> cmp byte [esi], 20h

973 0000FC61 76D2 <1> jna short loc\_swp\_xor\_retn

974 <1>

975 <1> ; 16/10/2016

976 0000FC63 C605[F3650100]00 <1> mov byte [SWP\_inv\_fname], 0 ; reset

977 <1> ; esi = file name address (ASCIIZ)

978 0000FC6A E8B289FFFF <1> call check\_filename

979 0000FC6F 7306 <1> jnc short loc\_swp\_drv

980 <1>

981 0000FC71 FE05[F3650100] <1> inc byte [SWP\_inv\_fname] ; set

982 <1> loc\_swp\_drv:

983 0000FC77 8A35[FE580100] <1> mov dh, [Current\_Drv]

984 <1> ;mov [RUN\_CDRV], dh

985 <1>

986 0000FC7D 8A15[A2620100] <1> mov dl, [FindFile\_Drv]

987 <1> ;cmp dl, dh

988 0000FC83 3A15[FE580100] <1> cmp dl, [Current\_Drv]

989 0000FC89 740D <1> je short loc\_swp\_change\_directory

990 <1>

991 0000FC8B FE05[F7650100] <1> inc byte [SWP\_DRV\_chg]

992 0000FC91 E82A72FFFF <1> call change\_current\_drive

993 0000FC96 72A0 <1> jc short loc\_swp\_retn ; eax = error code

994 <1> ; eax = 0

995 <1>

996 <1> loc\_swp\_change\_directory:

997 0000FC98 803D[A3620100]21 <1> cmp byte [FindFile\_Directory], 21h

998 0000FC9F F5 <1> cmc

999 0000FCA0 7396 <1> jnc short loc\_swp\_retn

1000 <1>

1001 0000FCA2 FE05[F7650100] <1> inc byte [SWP\_DRV\_chg]

1002 0000FCA8 FE05[D30C0100] <1> inc byte [Restore\_CDIR]

1003 0000FCAE BE[A3620100] <1> mov esi, FindFile\_Directory

1004 0000FCB3 8A25[F5650100] <1> mov ah, [SWP\_Mode+1]

1005 0000FCB9 E860A1FFFF <1> call change\_current\_directory

1006 0000FCBE 0F8274FFFFFF <1> jc loc\_swp\_retn ; eax = error code

1007 <1>

1008 <1> loc\_swp\_change\_prompt\_dir\_string:

1009 <1> ; esi = PATH\_Array

1010 <1> ; eax = Current Directory First Cluster

1011 <1> ; edi = Logical DOS Drive Description Table

1012 0000FCC4 E87AA0FFFF <1> call change\_prompt\_dir\_str

1013 0000FCC9 29C0 <1> sub eax, eax ; 0

1014 0000FCCB E968FFFFFF <1> jmp loc\_swp\_retn

1015 <1>

1016 <1> reset\_working\_path:

1017 <1> ; 06/10/2016 - TRDOS 386 (TRDOS v2.0)

1018 <1> ;

1019 <1> ; TRDOS v1.0 (DIR.ASM, "proc\_reset\_working\_path")

1020 <1> ; 05/02/2011 - 08/02/2011

1021 <1> ;

1022 <1> ; Restores current drive and directory

1023 <1> ;

1024 <1> ; INPUT: none

1025 <1> ; OUTPUT: DL = SWP\_DRV, EAX = 0 -> OK

1026 <1> ;

1027 <1> ; AX = 0 -> ESI = Logical Dos Drv Desc. Table

1028 <1> ;

1029 <1> ; EAX, EBX, ECX, EDX, ESI, EDI will be changed

1030 <1> ;

1031 <1>

1032 <1>

1033 0000FCD0 31C0 <1> xor eax, eax

1034 0000FCD2 48 <1> dec eax

1035 <1>

1036 0000FCD3 668B15[F6650100] <1> mov dx, [SWP\_DRV]

1037 0000FCDA 08F6 <1> or dh, dh

1038 0000FCDC 742E <1> jz short loc\_rwp\_return

1039 <1>

1040 0000FCDE 3A15[FE580100] <1> cmp dl, [Current\_Drv]

1041 0000FCE4 7407 <1> je short loc\_rwp\_restore\_cdir

1042 <1> loc\_rwp\_restore\_cdrv:

1043 0000FCE6 E8D571FFFF <1> call change\_current\_drive

1044 0000FCEB EB10 <1> jmp short loc\_rwp\_restore\_ok

1045 <1> loc\_rwp\_restore\_cdir:

1046 0000FCED 31DB <1> xor ebx, ebx

1047 0000FCEF 88D7 <1> mov bh, dl

1048 0000FCF1 BE00010900 <1> mov esi, Logical\_DOSDisks

1049 0000FCF6 01DE <1> add esi, ebx

1050 <1>

1051 0000FCF8 E87A72FFFF <1> call restore\_current\_directory

1052 <1>

1053 <1> loc\_rwp\_restore\_ok:

1054 0000FCFD 668B15[F6650100] <1> mov dx, [SWP\_DRV]

1055 0000FD04 31C0 <1> xor eax, eax

1056 0000FD06 66A3[F7650100] <1> mov [SWP\_DRV\_chg], ax

1057 <1> loc\_rwp\_return:

1058 0000FD0C C3 <1> retn

1059 <1>

1060 <1> get\_file\_name:

1061 <1> ; 15/10/2016 - TRDOS 386 (TRDOS v2.0)

1062 <1> ; Convert file name

1063 <1> ; from directory entry format

1064 <1> ; to (8.3) dot file name format

1065 <1> ;

1066 <1> ; TRDOS v1.0 (DIR.ASM, "get\_file\_name")

1067 <1> ; 2005 - 09/10/2011

1068 <1> ; INPUT:

1069 <1> ; DS:SI -> Directory Entry Format File Name

1070 <1> ; ES:DI -> DOS Dot File Name Address

1071 <1> ; OUTPUT:

1072 <1> ; DS:SI -> DOS Dot File Name Address

1073 <1> ; ES:DI -> Directory Entry Format File Name

1074 <1> ;

1075 <1> ; TRDOS 386 (15/10/2016)

1076 <1> ; INPUT:

1077 <1> ; ESI = File name addr in dir entry format

1078 <1> ; EDI = Dot file name address (destination)

1079 <1> ; OUTPUT:

1080 <1> ; File name is converted and moved

1081 <1> ; to destination (as 8.3 dot filename)

1082 <1> ;

1083 <1> ; Modified registers: EAX, ECX

1084 <1>

1085 <1> ; 2005 (TRDOS 8086) - 2016 (TRDOS 386)

1086 <1>

1087 0000FD0D 57 <1> push edi

1088 0000FD0E 56 <1> push esi

1089 0000FD0F AC <1> lodsb

1090 0000FD10 3C20 <1> cmp al, 20h

1091 0000FD12 762A <1> jna short pass\_gfn\_ext

1092 0000FD14 56 <1> push esi

1093 0000FD15 AA <1> stosb

1094 0000FD16 B907000000 <1> mov ecx, 7

1095 <1> loc\_gfn\_next\_char:

1096 0000FD1B AC <1> lodsb

1097 0000FD1C 3C20 <1> cmp al, 20h

1098 0000FD1E 7603 <1> jna short pass\_gfn\_fn

1099 0000FD20 AA <1> stosb

1100 0000FD21 E2F8 <1> loop loc\_gfn\_next\_char

1101 <1> pass\_gfn\_fn:

1102 0000FD23 5E <1> pop esi

1103 0000FD24 83C607 <1> add esi, 7

1104 0000FD27 AC <1> lodsb

1105 0000FD28 3C20 <1> cmp al, 20h

1106 0000FD2A 7612 <1> jna short pass\_gfn\_ext

1107 0000FD2C B42E <1> mov ah, '.'

1108 0000FD2E 86E0 <1> xchg ah, al

1109 0000FD30 66AB <1> stosw

1110 0000FD32 AC <1> lodsb

1111 0000FD33 3C20 <1> cmp al, 20h

1112 0000FD35 7607 <1> jna short pass\_gfn\_ext

1113 0000FD37 AA <1> stosb

1114 0000FD38 AC <1> lodsb

1115 0000FD39 3C20 <1> cmp al, 20h

1116 0000FD3B 7601 <1> jna short pass\_gfn\_ext

1117 0000FD3D AA <1> stosb

1118 <1> pass\_gfn\_ext:

1119 0000FD3E 30C0 <1> xor al, al

1120 0000FD40 AA <1> stosb

1121 0000FD41 5E <1> pop esi

1122 0000FD42 5F <1> pop edi

1123 0000FD43 C3 <1> retn

1124 <1>

1125 <1> set\_hardware\_int\_vector:

1126 <1> ; 18/03/2017

1127 <1> ; 03/03/2017

1128 <1> ; 28/02/2017 - TRDOS 386 (TRDOS v2.0)

1129 <1> ;

1130 <1> ; SET/RESET HARDWARE INTERRUPT GATE

1131 <1> ;

1132 <1> ; Changes interrupt gate descriptor table

1133 <1> ; (without changing default interrupt list)

1134 <1> ;

1135 <1> ; INPUT:

1136 <1> ; AL = IRQ number (0 to 15)

1137 <1> ; AH > 0 -> set

1138 <1> ; AH = 0 -> reset

1139 <1> ;

1140 <1> ; Modified registers: eax, ebx, edx, edi

1141 <1> ;

1142 <1>

1143 0000FD44 C0E002 <1> shl al, 2 ; IRQ number \* 4

1144 0000FD47 0FB6D8 <1> movzx ebx, al

1145 <1>

1146 0000FD4A 08E4 <1> or ah, ah

1147 0000FD4C 7508 <1> jnz short shintv\_1 ; set (for user call service)

1148 <1>

1149 <1> ; 18/03/2017

1150 0000FD4E 81C3[D0160100] <1> add ebx, IRQ\_list ; reset to default interrupt list

1151 0000FD54 EB06 <1> jmp short shintv\_2

1152 <1> shintv\_1:

1153 0000FD56 81C3[7DFD0000] <1> add ebx, IRQ\_u\_list

1154 <1> shintv\_2:

1155 0000FD5C 8B13 <1> mov edx, [ebx] ; IRQ handler address

1156 <1>

1157 <1> ; 03/03/2017

1158 0000FD5E D0E0 <1> shl al, 1 ; IRQ number \* 8

1159 <1> ; 18/03/2017

1160 0000FD60 0FB6F8 <1> movzx edi, al

1161 0000FD63 81C7[50560100] <1> add edi, idt + (8\*32) ; IRQ 0 offset = idt + 256

1162 <1>

1163 0000FD69 89D0 <1> mov eax, edx ; IRQ handler address

1164 0000FD6B BB00000800 <1> mov ebx, 80000h

1165 <1>

1166 <1> ;mov edx, eax

1167 0000FD70 66BA008E <1> mov dx, 8E00h

1168 0000FD74 6689C3 <1> mov bx, ax

1169 0000FD77 89D8 <1> mov eax, ebx ; /\* selector = 0x0008 = cs \*/

1170 <1> ; /\* interrupt gate - dpl=0, present \*/

1171 0000FD79 AB <1> stosd ; selector & offset bits 0-15

1172 0000FD7A 8917 <1> mov [edi], edx ; attributes & offset bits 16-23

1173 <1>

1174 0000FD7C C3 <1> retn

1175 <1> IRQ\_u\_list:

1176 <1> ; 28/02/2017

1177 0000FD7D [8B060000] <1> dd timer\_int

1178 0000FD81 [FF0D0000] <1> dd kb\_int

1179 0000FD85 [6D080000] <1> dd irq2

1180 0000FD89 [BDFD0000] <1> dd IRQ\_service3

1181 0000FD8D [C7FD0000] <1> dd IRQ\_service4

1182 0000FD91 [D1FD0000] <1> dd IRQ\_service5

1183 0000FD95 [B0410000] <1> dd fdc\_int

1184 0000FD99 [DBFD0000] <1> dd IRQ\_service7

1185 0000FD9D [F6070000] <1> dd rtc\_int

1186 0000FDA1 [E5FD0000] <1> dd IRQ\_service9

1187 0000FDA5 [EFFD0000] <1> dd IRQ\_service10

1188 0000FDA9 [F9FD0000] <1> dd IRQ\_service11

1189 0000FDAD [03FE0000] <1> dd IRQ\_service12

1190 0000FDB1 [0DFE0000] <1> dd IRQ\_service13

1191 0000FDB5 [2D4B0000] <1> dd hdc1\_int

1192 0000FDB9 [544B0000] <1> dd hdc2\_int

1193 <1>

1194 <1> ; 03/03/2017

1195 <1> ; 27/02/2017

1196 <1> IRQ\_service3:

1197 0000FDBD 36C605[BA6B0100]03 <1> mov byte [ss:IRQnum], 3

1198 0000FDC5 EB4E <1> jmp short IRQ\_service

1199 <1> IRQ\_service4:

1200 0000FDC7 36C605[BA6B0100]04 <1> mov byte [ss:IRQnum], 4

1201 0000FDCF EB44 <1> jmp short IRQ\_service

1202 <1> IRQ\_service5:

1203 0000FDD1 36C605[BA6B0100]05 <1> mov byte [ss:IRQnum], 5

1204 0000FDD9 EB3A <1> jmp short IRQ\_service

1205 <1> IRQ\_service7:

1206 0000FDDB 36C605[BA6B0100]07 <1> mov byte [ss:IRQnum], 7

1207 0000FDE3 EB30 <1> jmp short IRQ\_service

1208 <1> IRQ\_service9:

1209 0000FDE5 36C605[BA6B0100]09 <1> mov byte [ss:IRQnum], 9

1210 0000FDED EB26 <1> jmp short IRQ\_service

1211 <1> IRQ\_service10:

1212 0000FDEF 36C605[BA6B0100]0A <1> mov byte [ss:IRQnum], 10

1213 0000FDF7 EB1C <1> jmp short IRQ\_service

1214 <1> IRQ\_service11:

1215 0000FDF9 36C605[BA6B0100]0B <1> mov byte [ss:IRQnum], 11

1216 0000FE01 EB12 <1> jmp short IRQ\_service

1217 <1> IRQ\_service12:

1218 0000FE03 36C605[BA6B0100]0C <1> mov byte [ss:IRQnum], 12

1219 0000FE0B EB08 <1> jmp short IRQ\_service

1220 <1> IRQ\_service13:

1221 0000FE0D 36C605[BA6B0100]0D <1> mov byte [ss:IRQnum], 13

1222 <1> ;jmp short IRQ\_service

1223 <1> IRQ\_service:

1224 <1> ; 13/06/2017

1225 <1> ; 11/06/2017

1226 <1> ; 10/06/2017

1227 <1> ; 01/03/2017, 04/03/2017

1228 <1> ; 27/02/2017, 28/02/2017

1229 0000FE15 1E <1> push ds

1230 0000FE16 06 <1> push es

1231 0000FE17 0FA0 <1> push fs

1232 0000FE19 0FA8 <1> push gs

1233 <1>

1234 0000FE1B 60 <1> pushad ; eax,ecx,edx,ebx,esp,ebp,esi,edi

1235 0000FE1C 66B91000 <1> mov cx, KDATA

1236 0000FE20 8ED9 <1> mov ds, cx

1237 0000FE22 8EC1 <1> mov es, cx

1238 0000FE24 8EE1 <1> mov fs, cx

1239 0000FE26 8EE9 <1> mov gs, cx

1240 <1>

1241 0000FE28 0F20D8 <1> mov eax, cr3

1242 0000FE2B A3[B66B0100] <1> mov [IRQ\_cr3], eax

1243 <1>

1244 0000FE30 A1[38580100] <1> mov eax, [k\_page\_dir]

1245 0000FE35 0F22D8 <1> mov cr3, eax

1246 <1>

1247 0000FE38 A0[BA6B0100] <1> mov al, [IRQnum]

1248 <1>

1249 <1> ;mov cl, [sysflg]

1250 <1> ;mov [u.r\_mode], cl ; system (0) or user mode (FFh)

1251 <1> IRQsrv\_0:

1252 0000FE3D 0FB6D8 <1> movzx ebx, al

1253 0000FE40 8A9B[08160100] <1> mov bl, [ebx+IRQenum] ; IRQ (available) index number + 1

1254 <1> ; 01/03/2017

1255 0000FE46 FECB <1> dec bl ; IRQ index number, 0 to 8

1256 0000FE48 0F8807010000 <1> js IRQsrv\_5 ; not available to use here!?

1257 <1> ;

1258 0000FE4E 80BB[806B0100]80 <1> cmp byte [ebx+IRQ.method], 80h ; using by a dev or kernel?

1259 0000FE55 7205 <1> jb short IRQsrv\_1 ; no

1260 <1>

1261 <1> ; If the IRQ service is already owned by TRDOS 386 kernel

1262 <1> ; or a Device driver

1263 <1> ; we need to call 'dev\_IRQ\_service'

1264 <1>

1265 <1> ; IRQ number in AL

1266 0000FE57 E868020000 <1> call dev\_IRQ\_service ; IRQ service for device drivers

1267 <1> ; IRQ number in AL

1268 <1> IRQsrv\_1:

1269 <1> ; check user callback service status

1270 <1> ; AL = IRQ number

1271 <1> ; EBX = IRQ (Available) Index number

1272 <1>

1273 0000FE5C A2[D7030300] <1> mov [u.irqwait], al ; set waiting IRQ flag

1274 <1>

1275 0000FE61 8A83[6E6B0100] <1> mov al, [ebx+IRQ.owner]

1276 0000FE67 20C0 <1> and al, al

1277 0000FE69 0F84E6000000 <1> jz IRQsrv\_5 ; it is not owned by a user/proc

1278 <1>

1279 <1> ; 03/03/2017

1280 0000FE6F 89DA <1> mov edx, ebx

1281 0000FE71 C0E202 <1> shl dl, 2

1282 0000FE74 8B92[926B0100] <1> mov edx, [edx+IRQ.addr] ; S.R.B. or Callback service addr

1283 <1>

1284 0000FE7A 8AA3[806B0100] <1> mov ah, [ebx+IRQ.method]

1285 0000FE80 F6C401 <1> test ah, 1

1286 0000FE83 7534 <1> jnz short IRQsrv\_4 ; Callback service method

1287 <1>

1288 <1> ; Signal Response Byte method

1289 <1> ;mov edx, [edx+IRQ.addr] ; Signal Response Byte address

1290 <1> ; ; (Physical address, non-swappable)

1291 0000FE85 80E402 <1> and ah, 2 ; bit 1, (S.R.B.) counter (auto increment) method

1292 0000FE88 8AA3[896B0100] <1> mov ah, [ebx+IRQ.srb] ; Signal Response Byte value

1293 0000FE8E 7408 <1> jz short IRQsrv\_2 ; fixed S.R.B. value

1294 <1> ; counter method (auto increment)

1295 0000FE90 FEC4 <1> inc ah

1296 0000FE92 88A3[896B0100] <1> mov [ebx+IRQ.srb], ah ; next (count) number

1297 <1> IRQsrv\_2:

1298 0000FE98 8822 <1> mov [edx], ah ; put S.R.B. val to the user's S.R.B. addr

1299 0000FE9A C605[D7030300]00 <1> mov byte [u.irqwait], 0 ; clear waiting IRQ flag

1300 <1>

1301 0000FEA1 3A05[B3030300] <1> cmp al, [u.uno]

1302 0000FEA7 0F84A8000000 <1> je IRQsrv\_5 ; the owner is current user/process

1303 <1> IRQsrv\_3:

1304 <1> ; the owner is not current user/process

1305 <1> ; AL = process number

1306 0000FEAD B202 <1> mov dl, 2 ; priority, 2 = event (high)

1307 0000FEAF E837FAFFFF <1> call set\_run\_sequence

1308 <1>

1309 <1> ; [u.irqwait] = waiting IRQ number for callback service

1310 <1>

1311 0000FEB4 E99C000000 <1> jmp IRQsrv\_5

1312 <1> IRQsrv\_4:

1313 0000FEB9 3A05[B3030300] <1> cmp al, [u.uno] ; is the owner is current user/process?

1314 0000FEBF 75EC <1> jne short IRQsrv\_3 ; no !

1315 <1>

1316 <1> ; Check if an IRQ callback service already in progress

1317 0000FEC1 803D[D8030300]00 <1> cmp byte [u.r\_lock], 0

1318 0000FEC8 0F8787000000 <1> ja IRQsrv\_5 ; nothing to do !

1319 <1> ; (we need to complete prev callback)

1320 0000FECE 803D[D4030300]00 <1> cmp byte [u.t\_lock], 0

1321 0000FED5 777E <1> ja short IRQsrv\_5 ; nothing to do !

1322 <1> ; (we need to complete timer callback)

1323 <1>

1324 <1> ; 04/03/2017

1325 0000FED7 C605[D7030300]00 <1> mov byte [u.irqwait], 0 ; reset/clear waiting IRQ flag

1326 <1>

1327 0000FEDE FE05[D8030300] <1> inc byte [u.r\_lock] ; 'IRQ callback service in progress' flag

1328 <1>

1329 0000FEE4 8A0D[5B030300] <1> mov cl, [sysflg] ; (system call) mode flag (kernel/user)

1330 0000FEEA 880D[D9030300] <1> mov [u.r\_mode], cl ; system mode (0) or user mode (FFh)

1331 <1>

1332 <1> ;

1333 0000FEF0 8B2D[D4570100] <1> mov ebp, [tss.esp0] ; kernel stack address (for ring 0)

1334 0000FEF6 83ED14 <1> sub ebp, 20 ; eip, cs, eflags, esp, ss

1335 0000FEF9 892D[5C030300] <1> mov [u.sp], ebp

1336 0000FEFF 8925[60030300] <1> mov [u.usp], esp

1337 <1>

1338 <1> ;or word [ebp+8], 200h ; 22/01/2017, force enabling interrupts

1339 <1>

1340 0000FF05 8B44241C <1> mov eax, [esp+28] ; pushed eax

1341 0000FF09 A3[64030300] <1> mov [u.r0], eax

1342 <1>

1343 0000FF0E E81AE7FFFF <1> call wswap ; save user's registers & status

1344 <1>

1345 <1> ; software int is in ring 0 but IRQ handler must return to ring 3

1346 <1> ; so, ring 3 return address and stack registers

1347 <1> ; (eip, cs, eflags, esp, ss)

1348 <1> ; must be copied to IRQ handler return

1349 <1> ; eip will be replaced by callback service routine address

1350 <1>

1351 0000FF13 C605[5B030300]FF <1> mov byte [sysflg], 0FFh ; user mode

1352 <1>

1353 <1> ; system mode (system call)

1354 <1> ;mov ebp, [u.sp] ; EIP (u), CS (UCODE), EFLAGS (u),

1355 <1> ; ESP (u), SS (UDATA)

1356 <1>

1357 0000FF1A 8B4510 <1> mov eax, [ebp+16] ; SS (UDATA)

1358 0000FF1D 89E6 <1> mov esi, esp

1359 0000FF1F 50 <1> push eax

1360 0000FF20 50 <1> push eax

1361 0000FF21 89E7 <1> mov edi, esp

1362 0000FF23 893D[60030300] <1> mov [u.usp], edi

1363 0000FF29 B908000000 <1> mov ecx, ((ESPACE/4) - 4) ; except DS, ES, FS, GS

1364 0000FF2E F3A5 <1> rep movsd

1365 0000FF30 B104 <1> mov cl, 4

1366 0000FF32 F3AB <1> rep stosd

1367 0000FF34 893D[5C030300] <1> mov [u.sp], edi

1368 0000FF3A 89EE <1> mov esi, ebp

1369 0000FF3C B105 <1> mov cl, 5 ; EIP (u), CS (UCODE), EFLAGS (u), ESP (u), SS (UDATA)

1370 0000FF3E F3A5 <1> rep movsd

1371 <1> ;

1372 <1>

1373 0000FF40 8B0D[B8030300] <1> mov ecx, [u.pgdir]

1374 0000FF46 890D[B66B0100] <1> mov [IRQ\_cr3], ecx

1375 <1>

1376 <1> set\_IRQ\_callback\_addr:

1377 <1> ;

1378 <1> ; This routine sets return address

1379 <1> ; to start of user's interrupt

1380 <1> ; service (callback) address

1381 <1> ;

1382 <1> ; INPUT:

1383 <1> ; EDX = callback routine/service address

1384 <1> ; (virtual, not physical address!)

1385 <1> ; [u.sp] = kernel stack, points to

1386 <1> ; user's EIP,CS,EFLAGS,ESP,SS

1387 <1> ; registers.

1388 <1> ; OUTPUT:

1389 <1> ; EIP (user) = callback (service) address

1390 <1> ; CS (user) = UCODE

1391 <1> ; EFLAGS (user) = flags before callback

1392 <1> ; ESP (user) = ESP-4 (user, before callback)

1393 <1> ; [ESP](user) = EIP (user) before callback

1394 <1> ;

1395 <1> ; Note: If CPU was in user mode while entering

1396 <1> ; the timer interrupt service routine,

1397 <1> ; 'IRET' will get return to callback routine

1398 <1> ; immediately. If CPU was in system/kernel mode

1399 <1> ; 'iret' will get return to system call and

1400 <1> ; then, callback routine will be return address

1401 <1> ; from system call. (User's callback/service code

1402 <1> ; will be able to return to normal return address

1403 <1> ; via a 'sysrele' system call at the end.)

1404 <1> ;

1405 <1> ; Note: User's IRQ callback service code must be ended

1406 <1> ; with a 'sysrele' system call !

1407 <1> ;

1408 <1> ; For example:

1409 <1> ;

1410 <1> ; audio\_IRQ\_callback:

1411 <1> ; ...

1412 <1> ; <load DMA buffer with audio data>

1413 <1> ; ...

1414 <1> ; mov eax, 39 ; 'sysrele'

1415 <1> ; int 40h ; TRDOS 386 system call (interrupt)

1416 <1> ;

1417 <1>

1418 <1> ;mov edx, [edx+IRQ.addr] ; Callback service address

1419 <1> ; ; (Virtual address)

1420 <1>

1421 0000FF4C 8B2D[5C030300] <1> mov ebp, [u.sp]; kernel's stack, points to EIP (user)

1422 0000FF52 895500 <1> mov [ebp], edx

1423 <1> IRQsrv\_5:

1424 <1> ; EOI & return

1425 <1> ; 11/06/2017

1426 <1> ; 10/06/2017

1427 0000FF55 A0[BA6B0100] <1> mov al, [IRQnum]

1428 0000FF5A FA <1> cli

1429 0000FF5B 3C07 <1> cmp al, 7

1430 0000FF5D 7604 <1> jna short IRQsrv\_6

1431 <1> ;

1432 <1> ;mov al, EOI ; end of interrupt

1433 0000FF5F B020 <1> mov al, 20h

1434 <1> ;cli ; disable interrupts till stack cleared

1435 <1> ;out INTB00, al ; For controll2 #2

1436 0000FF61 E6A0 <1> out 0A0h, al

1437 <1> IRQsrv\_6:

1438 <1> ;mov byte [IRQnum], 0 ; reset

1439 <1> ;mov al, EOI ; end of interrupt

1440 0000FF63 B020 <1> mov al, 20h

1441 <1> ;cli ; disable interrupts till stack cleared

1442 <1> ;out INTA00, al ; end of interrupt to 8259 - 1

1443 0000FF65 E620 <1> out 20h, al

1444 <1> IRQsrv\_7:

1445 <1> ;; 13/06/2017

1446 <1> ;or word [ebp+8], 200h ; force enabling interrupts

1447 <1> ;

1448 0000FF67 8B0D[B66B0100] <1> mov ecx, [IRQ\_cr3] ; previous content of cr3 register

1449 0000FF6D 0F22D9 <1> mov cr3, ecx ; restore cr3 register content

1450 <1> ;

1451 0000FF70 61 <1> popad ; edi,esi,ebp,(icrement esp by 4), ebx,edx,ecx,eax

1452 <1> ;

1453 0000FF71 0FA9 <1> pop gs

1454 0000FF73 0FA1 <1> pop fs

1455 0000FF75 07 <1> pop es

1456 0000FF76 1F <1> pop ds

1457 <1> ;

1458 0000FF77 CF <1> iretd ; return from interrupt

1459 <1>

1460 <1> get\_device\_number:

1461 <1> ; 08/10/2016

1462 <1> ; 07/10/2016 - TRDOS 386 (TRDOS v2.0)

1463 <1> ;

1464 <1> ; This procedure compares name of requested

1465 <1> ; device with kernel device names and

1466 <1> ; installable device names. If names match,

1467 <1> ; the relevant device index (entry) number

1468 <1> ; will be returned the caller (sysopen)

1469 <1> ; for the requested device.

1470 <1> ;

1471 <1> ; NOTE: Installable device drivers must

1472 <1> ; be loaded before using 'sysopen'

1473 <1> ; (opendev) system call.

1474 <1> ;

1475 <1> ; INPUT:

1476 <1> ; ESI = device name address (ASCIIZ)

1477 <1> ; (in kernel's memory space)

1478 <1> ; max name length = 8 without '/dev/')

1479 <1> ; Device name will be capitalized

1480 <1> ; and if there is, '/dev/' will be

1481 <1> ; removed from name before comparising)

1482 <1> ;

1483 <1> ; OUTPUT:

1484 <1> ; cf = 0 ->

1485 <1> ; EAX (AL) = device entry/index number

1486 <1> ; cf = 1 -> device not found (installed)

1487 <1> ; or invalid device name

1488 <1> ; (AL=0)

1489 <1> ; device\_name = device name address (asciiz)

1490 <1> ;

1491 <1> ; Modified registers: EAX, EBX, ESI, EDI

1492 <1>

1493 0000FF78 BF[F9650100] <1> mov edi, device\_name

1494 0000FF7D E805010000 <1> call lodsb\_capitalize

1495 0000FF82 88C4 <1> mov ah, al

1496 0000FF84 3C2F <1> cmp al, '/'

1497 0000FF86 750E <1> jne short gdn\_1

1498 0000FF88 BF[F9650100] <1> mov edi, device\_name

1499 0000FF8D E8F5000000 <1> call lodsb\_capitalize

1500 <1> gdn\_0:

1501 0000FF92 20C0 <1> and al, al ; 0 ?

1502 0000FF94 7420 <1> jz short gdn\_err ; null name after '/'

1503 <1> gdn\_1:

1504 0000FF96 3C44 <1> cmp al, 'D'

1505 0000FF98 7517 <1> jne short gdn\_2

1506 0000FF9A E8E8000000 <1> call lodsb\_capitalize

1507 0000FF9F 3C45 <1> cmp al, 'E'

1508 0000FFA1 750E <1> jne short gdn\_2

1509 0000FFA3 E8DF000000 <1> call lodsb\_capitalize

1510 0000FFA8 3C56 <1> cmp al, 'V'

1511 0000FFAA 7505 <1> jne short gdn\_2

1512 0000FFAC AC <1> lodsb

1513 0000FFAD 3C2F <1> cmp al, '/'

1514 0000FFAF 740D <1> je short gdn\_4

1515 <1> gdn\_2:

1516 0000FFB1 80FC2F <1> cmp ah, '/'

1517 0000FFB4 750F <1> jne short gdn\_5

1518 <1> gdn\_err:

1519 <1> ; invalid device name or device not found

1520 0000FFB6 31C0 <1> xor eax, eax ; 0

1521 0000FFB8 F9 <1> stc

1522 0000FFB9 C3 <1> retn

1523 <1> gdn\_3:

1524 0000FFBA 3C2F <1> cmp al, '/'

1525 0000FFBC 7507 <1> jne short gdn\_5

1526 <1> gdn\_4:

1527 0000FFBE BF[F9650100] <1> mov edi, device\_name

1528 0000FFC3 EB04 <1> jmp short gdn\_6

1529 <1> gdn\_5:

1530 0000FFC5 3C00 <1> cmp al, 0

1531 0000FFC7 7419 <1> je short gdn\_7

1532 <1> gdn\_6:

1533 0000FFC9 E8B9000000 <1> call lodsb\_capitalize

1534 0000FFCE 81FF[01660100] <1> cmp edi, device\_name + 8

1535 0000FFD4 72E4 <1> jb short gdn\_3

1536 0000FFD6 3C00 <1> cmp al, 0

1537 0000FFD8 75DC <1> jne short gdn\_err

1538 0000FFDA 81FF[FA650100] <1> cmp edi, device\_name + 1

1539 0000FFE0 76D4 <1> jna short gdn\_err ; null name after '/'

1540 <1> gdn\_7:

1541 0000FFE2 AA <1> stosb

1542 <1> ; zero padding ("NAME",0,0,0,0)

1543 0000FFE3 81FF[01660100] <1> cmp edi, device\_name + 8

1544 0000FFE9 72F7 <1> jb short gdn\_7

1545 <1> gdn\_8:

1546 <1> ; search for kernel device names

1547 0000FFEB BE[F9650100] <1> mov esi, device\_name

1548 0000FFF0 BF[EE130100] <1> mov edi, KDEV\_NAME

1549 0000FFF5 31C0 <1> xor eax, eax

1550 <1> gdn\_9:

1551 0000FFF7 A7 <1> cmpsd

1552 0000FFF8 7505 <1> jne short gdn\_10

1553 0000FFFA A7 <1> cmpsd

1554 0000FFFB 7503 <1> jne short gdn\_11

1555 0000FFFD EB2B <1> jmp short gdn\_17 ; match

1556 <1> gdn\_10:

1557 0000FFFF A7 <1> cmpsd ; add esi, 4 & add edi, 4

1558 <1> gdn\_11:

1559 00010000 BE[F9650100] <1> mov esi, device\_name

1560 00010005 FEC0 <1> inc al

1561 00010007 3C16 <1> cmp al, NumOfKernelDevNames

1562 00010009 72EC <1> jb short gdn\_9

1563 <1> gdn\_12:

1564 <1> ; search for installable device names

1565 <1> ; esi = offset device\_name

1566 0001000B BF[24660100] <1> mov edi, IDEV\_NAME

1567 00010010 28C0 <1> sub al, al ; 0

1568 <1> gdn\_13:

1569 00010012 A7 <1> cmpsd

1570 00010013 7505 <1> jne short gdn\_14

1571 00010015 A7 <1> cmpsd

1572 00010016 7503 <1> jne short gdn\_15

1573 00010018 EB3F <1> jmp short gdn\_19 ; match

1574 <1> gdn\_14:

1575 0001001A A7 <1> cmpsd ; add esi, 4 & add edi, 4

1576 <1> gdn\_15:

1577 0001001B BE[F9650100] <1> mov esi, device\_name

1578 00010020 FEC0 <1> inc al

1579 00010022 3C08 <1> cmp al, NumOfInstallableDevices

1580 00010024 72EC <1> jb short gdn\_13

1581 <1>

1582 <1> gdn\_16: ; error: invalid device name (not found) !

1583 00010026 30C0 <1> xor al, al

1584 00010028 F9 <1> stc

1585 00010029 C3 <1> retn

1586 <1>

1587 <1> gdn\_17: ; name match (with one of kernel device names)

1588 <1> ;

1589 <1> ; convert KDEV\_NAME index to

1590 <1> ; KDEV\_NUMBER index

1591 <1> ; (different names are used for same devices)

1592 <1> ; (example: "COM1" & "TTY8" = device number 18)

1593 0001002A 89C3 <1> mov ebx, eax ; < 256

1594 0001002C 8A83[9E140100] <1> mov al, [KDEV\_NUMBER+ebx]

1595 <1>

1596 <1> ; check if empty dev entry in the list

1597 00010032 80B8[A8670100]00 <1> cmp byte [DEV\_OPENMODE+eax], 0

1598 00010039 771B <1> ja short gdn\_18 ; it must be already set

1599 <1>

1600 <1> ; (re)set device name and access flags

1601 <1> ; (remain open work will be easy after that)

1602 <1> ; (NOTE: here, data will be copied to bss section)

1603 0001003B 88C3 <1> mov bl, al

1604 0001003D 83EF08 <1> sub edi, 8 ; kernel device name address (data)

1605 00010040 66C1E302 <1> shl bx, 2

1606 00010044 89BB[C6670100] <1> mov [DEV\_NAME\_PTR+ebx], edi ; (all) device names

1607 0001004A 8A98[F4150100] <1> mov bl, [KDEV\_ACCESS+eax] ; kernel dev list (data)

1608 00010050 8898[F4660100] <1> mov [DEV\_ACCESS+eax], bl ; (all) device list (bss)

1609 <1> gdn\_18:

1610 00010056 FEC0 <1> inc al ; 1 to NumOfKernelDevNames (<=7Fh)

1611 <1> ; eax = device index/entry number

1612 00010058 C3 <1> retn

1613 <1>

1614 <1> gdn\_19: ; name match (with one of installable device names)

1615 <1> ;

1616 <1> ; al = 0 to NumOfInstallableDevices - 1 (<=7Fh)

1617 <1>

1618 00010059 89C3 <1> mov ebx, eax

1619 0001005B 80C316 <1> add bl, NumOfKernelDevices ; < NUMOFDEVICES

1620 <1>

1621 <1> ; check if empty dev entry in the list

1622 0001005E 80BB[A8670100]00 <1> cmp byte [DEV\_OPENMODE+ebx], 0

1623 00010065 771D <1> ja short gdn\_20 ; it must be already set

1624 <1>

1625 <1> ; (re)set device name and access flags

1626 <1> ; (remain open work will be easy after that)

1627 00010067 83EF08 <1> sub edi, 8 ; installable device name address

1628 0001006A 66C1E302 <1> shl bx, 2 ;\*4

1629 0001006E 89BB[C6670100] <1> mov [DEV\_NAME\_PTR+ebx], edi ; (all) device names

1630 00010074 66C1EB02 <1> shr bx, 2

1631 00010078 8A80[6C660100] <1> mov al, [IDEV\_FLAGS+eax] ; installable dev list

1632 0001007E 8883[F4660100] <1> mov [DEV\_ACCESS+ebx], al ; (all) device list

1633 <1> gdn\_20:

1634 00010084 88D8 <1> mov al, bl

1635 <1> ; eax = device index/entry number ; < NUMOFDEVICES

1636 00010086 C3 <1> retn

1637 <1>

1638 <1> lodsb\_capitalize:

1639 <1> ; 07/10/2016 - TRDOS 386 (TRDOS v2.0)

1640 <1> ; INPUT -> [esi] = character

1641 <1> ; edi = destination

1642 <1> ; OUTPUT -> AL contains capitalized character

1643 <1> ; esi = esi+1

1644 <1> ; edi = edi+1

1645 <1> ;

1646 00010087 AC <1> lodsb

1647 00010088 3C61 <1> cmp al, 61h

1648 0001008A 7206 <1> jb short lodsb\_cap\_retn

1649 0001008C 3C7A <1> cmp al, 7Ah

1650 0001008E 7702 <1> ja short lodsb\_cap\_retn

1651 00010090 24DF <1> and al, 0DFh

1652 <1> lodsb\_cap\_retn:

1653 00010092 AA <1> stosb

1654 00010093 C3 <1> retn

1655 <1>

1656 <1> device\_open:

1657 <1> ; 08/10/2016 - TRDOS 386 (TRDOS v2.0)

1658 <1> ; Complete device opening work for sysopen (device)

1659 <1> ;

1660 <1> ; INPUT ->

1661 <1> ; EAX = Device Number (AL)

1662 <1> ; CL = Open mode (1 = read, 2 = write)

1663 <1> ; CH = Device access byte (bit 0 = 0)

1664 <1> ; OUTPUT ->

1665 <1> ; EAX = Device Number

1666 <1> ; CF = 0 -> device has been opened

1667 <1> ; CF = 1 -> device could not be opened

1668 <1> ;

1669 <1> ; Modified registers: ebx, (edx, ecx, esi, edi, ebp)

1670 <1> ;

1671 <1>

1672 00010094 89C3 <1> mov ebx, eax

1673 00010096 66C1E302 <1> shl bx, 2 ; \*4

1674 <1>

1675 0001009A F6C580 <1> test ch, 80h ; bit 7, installable device driver flag

1676 0001009D 7406 <1> jz short d\_open\_2 ; Kernel device

1677 <1> ; installable device

1678 <1> d\_open\_1:

1679 0001009F FFA3[70660100] <1> jmp dword [ebx+IDEV\_OADDR-4]

1680 <1> d\_open\_2:

1681 000100A5 FFA3[B0140100] <1> jmp dword [ebx+KDEV\_OADDR-4]

1682 <1>

1683 <1> device\_close:

1684 <1> ; 08/10/2016 - TRDOS 386 (TRDOS v2.0)

1685 <1> ; Complete device closing work for sysclose (device)

1686 <1> ;

1687 <1> ; INPUT ->

1688 <1> ; EAX = Device Number (AL)

1689 <1> ; CL = Open mode (1 = read, 2 = write)

1690 <1> ; CH = Device access byte (bit 0 = 0)

1691 <1> ; OUTPUT ->

1692 <1> ; EAX = Device Number

1693 <1> ; CF = 0 -> device has been closed

1694 <1> ; CF = 1 -> device could not be closed

1695 <1> ;

1696 <1> ; Modified registers: ebx, (edx, ecx, esi, edi, ebp)

1697 <1> ;

1698 <1>

1699 000100AB 89C3 <1> mov ebx, eax

1700 000100AD 66C1E302 <1> shl bx, 2 ; \*4

1701 <1>

1702 000100B1 F6C580 <1> test ch, 80h ; bit 7, installable device driver flag

1703 000100B4 7406 <1> jz short d\_close\_2 ; Kernel device

1704 <1> ; installable device

1705 <1> d\_close\_1:

1706 000100B6 FFA3[90660100] <1> jmp dword [ebx+IDEV\_CADDR-4]

1707 <1> d\_close\_2:

1708 000100BC FFA3[00150100] <1> jmp dword [ebx+KDEV\_CADDR-4]

1709 <1>

1710 <1> rnull:

1711 <1> ; 07/10/2016 - TRDOS 386 (TRDOS v2.0)

1712 <1> ; read null (read from null device)

1713 000100C2 C3 <1> retn

1714 <1>

1715 <1> wnull:

1716 <1> ; 07/10/2016 - TRDOS 386 (TRDOS v2.0)

1717 <1> ; write null (write to null device)

1718 000100C3 C3 <1> retn

1719 <1>

1720 <1> dev\_IRQ\_service:

1721 <1> ; 12/05/2017

1722 <1> ; 13/04/2017

1723 <1> ; 27/02/2017 - TRDOS 386 (TRDOS v2.0)

1724 <1> ; INPUT ->

1725 <1> ; AL = IRQ Number (0 to 15)

1726 <1> ;

1727 000100C4 53 <1> push ebx

1728 000100C5 0FB6D8 <1> movzx ebx, al

1729 000100C8 C0E302 <1> shl bl, 2 ; \* 4

1730 000100CB 8B9B[2E6B0100] <1> mov ebx, [ebx+DEV\_INT\_HNDLR]

1731 000100D1 21DB <1> and ebx, ebx

1732 000100D3 7404 <1> jz short dIRQ\_s\_retn

1733 000100D5 50 <1> push eax

1734 <1>

1735 000100D6 FFD3 <1> call ebx

1736 <1>

1737 000100D8 58 <1> pop eax

1738 <1> dIRQ\_s\_retn:

1739 000100D9 5B <1> pop ebx

1740 000100DA C3 <1> retn

1741 <1>

1742 <1>

1743 <1> set\_dev\_IRQ\_service:

1744 <1> ; 13/04/2017 - TRDOS 386 (TRDOS v2.0)

1745 <1> ;

1746 <1> ; Set Device Interrupt Service

1747 <1> ;

1748 <1> ; INPUT ->

1749 <1> ; AL = IRQ Number

1750 <1> ; EBX = Hardware Interrupt Service Address

1751 <1> ;

1752 <1> ; Note: There is not a validation check here

1753 <1> ; because this procedure is called by

1754 <1> ; TRDOS 386 kernel !

1755 <1> ; (Even if a device driver does not exist

1756 <1> ; this setting may be used by sysaudio

1757 <1> ; and other system calls for hardware

1758 <1> ; components which use IRQ method for I/O.)

1759 <1> ;

1760 <1> ;push esi

1761 000100DB 0FB6F0 <1> movzx esi, al

1762 000100DE 66C1E602 <1> shl si, 2 ; \* 4

1763 000100E2 899E[2E6B0100] <1> mov [esi+DEV\_INT\_HNDLR], ebx

1764 <1> ;pop esi

1765 000100E8 C3 <1> retn

1766 <1>

1767 <1>

1768 <1> sysaudio: ; AUDIO FUNCTIONS

1769 <1> ; 10/10/2017

1770 <1> ; 22/06/2017

1771 <1> ; 28/05/2017, 04/06/2017, 05/06/2017, 10/06/2017

1772 <1> ; 01/05/2017, 12/05/2017, 15/05/2017, 20/05/2017

1773 <1> ; 21/04/2017, 22/04/2017, 23/04/2017, 24/04/2017

1774 <1> ; 10/04/2017, 13/04/2017, 14/04/2017, 16/04/2017

1775 <1> ; 03/04/2017 (VIA VT8237R)

1776 <1> ; 01/04/2016 (trdosk6.s -> tdosk8.s)

1777 <1> ; 16/05/2016 - TRDOS 386 (TRDOS v2.0)

1778 <1> ;

1779 <1> ; Inputs:

1780 <1> ;

1781 <1> ; BH = 0 -> Beep (PC Speaker)

1782 <1> ; BL = Duration Counter (1 for 1/64 second)

1783 <1> ; CX = Frequency Divisor (1193180/Frequency)

1784 <1> ; (1331 for 886 Hz)

1785 <1> ;

1786 <1> ; 01/04/2017

1787 <1> ;

1788 <1> ; BH = 1 -> DETECT (& ENABLE) AUDIO DEVICE

1789 <1> ; BL = 0 : PC SPEAKER

1790 <1> ; 1 : SOUND BLASTER 16

1791 <1> ; 2 : INTEL AC'97

1792 <1> ; 3 : VIA VT8237R (VT8233)

1793 <1> ; 4 : INTEL HDA

1794 <1> ; 5-FEh : unknown/invalid

1795 <1> ; ; 04/06/2017

1796 <1> ; FFh : Get current audio device id

1797 <1> ;

1798 <1> ; BH = 2 -> ALLOCATE AUDIO BUFFER (for user)

1799 <1> ; ECX = Audio Buffer Size (must be equal to

1800 <1> ; the half of DMA buffer size)

1801 <1> ; EDX = Virtual Address of the buffer

1802 <1> ; (This is not DMA buffer!)

1803 <1> ;

1804 <1> ; BH = 3 -> INITIALIZE AUDIO DEVICE

1805 <1> ; BL = 0,2 -> for Signal Response Byte

1806 <1> ; CL = Signal Response Byte Value (fixed)

1807 <1> ; if BL = 0

1808 <1> ; auto increment of S.R.B. value

1809 <1> ; if BL = 2

1810 <1> ; EDX = Signal Response (Return) Byte Address

1811 <1> ;

1812 <1> ; BL = 1 for CallBack Method

1813 <1> ; EDX = CallBack Service Address (Virtual)

1814 <1> ;

1815 <1> ; BL > 2 -> invalid function

1816 <1> ;

1817 <1> ; (Audio buffer must be allocated before

1818 <1> ; initialization.)

1819 <1> ;

1820 <1> ; BH = 4 -> START TO PLAY

1821 <1> ; BL = Mode

1822 <1> ; Bit 0 = mono/stereo (1 = stereo)

1823 <1> ; Bit 1 = 8 bit / 16 bit (1 = 16 bit)

1824 <1> ; CX = Sampling Rate (Hz)

1825 <1> ;

1826 <1> ; BH = 5 -> PAUSE

1827 <1> ; BL = Any

1828 <1> ;

1829 <1> ; BH = 6 -> CONTINUE TO PLAY

1830 <1> ; BL = Any

1831 <1> ;

1832 <1> ; BH = 7 -> STOP

1833 <1> ; BL = Any

1834 <1> ;

1835 <1> ; BH = 8 -> RESET

1836 <1> ; BL = Any

1837 <1> ;

1838 <1> ; BH = 9 -> CANCEL (CALLBACK or S.R.B. SERVICE)

1839 <1> ; BL = Any

1840 <1> ;

1841 <1> ; BH = 10 -> DEALLOCATE AUDIO BUFFER (for user)

1842 <1> ; BL = Any

1843 <1> ;

1844 <1> ; BH = 11 -> SET VOLUME LEVEL

1845 <1> ; BL: (Bit 0 to 6)

1846 <1> ; 0 = Master (Playback, Lineout) volume

1847 <1> ; CL = Left Channel Volume

1848 <1> ; CH = Right Channel Volume

1849 <1> ;

1850 <1> ; Note: If BL >= 80h (Bit 7 of BL is set),

1851 <1> ; volume level will be set for next playing

1852 <1> ; (actual volume level will not be changed

1853 <1> ; immediately)

1854 <1> ;

1855 <1> ; BH = 12 -> DISABLE AUDIO DEVICE

1856 <1> ; (reset audio device and unlink dma buffer)

1857 <1> ; BL = Any

1858 <1> ;

1859 <1> ; 12/05/2017

1860 <1> ; BH = 13 -> MAP DMA BUFFER TO USER

1861 <1> ; (for direct access to system's dma buffer)

1862 <1> ;

1863 <1> ; ECX = map size in bytes

1864 <1> ; (will be rounded up to page borders)

1865 <1> ; EDX = Virtual Address of the buffer

1866 <1> ; (Will be rounded up to page borders)

1867 <1> ;

1868 <1> ; 05/06/2017

1869 <1> ; 04/06/2017

1870 <1> ; BH = 14 -> GET AUDIO DEVICE INFO

1871 <1> ; BL: 0 = Audio Controller Info

1872 <1> ; > 0 = Invalid for now!

1873 <1> ;

1874 <1> ; 22/06/2017

1875 <1> ; BH = 15 -> GET CURRENT SOUND DATA (for graphics)

1876 <1> ; BL: 0 -> PCM OUT data

1877 <1> ; > 0 -> Invalid for now!

1878 <1> ; ECX = 0 -> Get DMA Buffer Pointer

1879 <1> ; EDX = Not Used

1880 <1> ; ECX > 0 -> Byte count for buffer (EDX)

1881 <1> ; EDX = Buffer Address (Virtual)

1882 <1> ;

1883 <1> ; 10/10/2017

1884 <1> ; BH = 16 -> UPDATE DMA BUFFER DATA

1885 <1> ; (by using the Audio Buffer content)

1886 <1> ; BL = 0 : Update dma half buffer in sequence

1887 <1> ; (automatic destination)

1888 <1> ; 1 : Update 1st half of the dma buffer

1889 <1> ; 2 : Update 2nd half of the dma buffer

1890 <1> ; 3-FEh: Invalid!

1891 <1> ; FFh = Get current flag value

1892 <1> ; (Half buffer number -1)

1893 <1> ;

1894 <1> ;

1895 <1> ; Outputs:

1896 <1> ;

1897 <1> ; For BH = 0 -> Beep

1898 <1> ; None

1899 <1> ;

1900 <1> ; 01/04/2017

1901 <1> ;

1902 <1> ; For BH = 1 -> DETECT (& ENABLE) AUDIO DEVICE

1903 <1> ; AH = 0 : PC SPEAKER

1904 <1> ; 1 : SOUND BLASTER 16

1905 <1> ; 2 : INTEL AC'97

1906 <1> ; 3 : VIA VT8237R (VT8233)

1907 <1> ; 4 : INTEL HDA

1908 <1> ; 5-FFh : unknown/invalid

1909 <1> ; AL = mode status

1910 <1> ; bit 0 = mono /stereo (1 = stereo)

1911 <1> ; bit 1 = 8 bit / 16 bit ( 1 = 16 bit)

1912 <1> ; 04/06/2017

1913 <1> ; EBX = PCI DEVICE/VENDOR ID (if >0)

1914 <1> ; (BX = VENDOR ID)

1915 <1> ; (if CF = 1 -> Error code in EAX)

1916 <1> ;

1917 <1> ; For BH = 2 -> ALLOCATE AUDIO BUFFER (for user)

1918 <1> ; EAX = Physical Address of the buffer

1919 <1> ; (if CF = 1 -> Error code in EAX)

1920 <1> ;

1921 <1> ; For BH = 3 -> INITIALIZE AUDIO DEVICE

1922 <1> ; (if CF = 1 -> Error code in EAX)

1923 <1> ;

1924 <1> ; For BH = 4 -> START TO PLAY

1925 <1> ; none (if CF = 1 -> Error code in EAX)

1926 <1> ;

1927 <1> ; For BH = 5 -> PAUSE

1928 <1> ; none (if CF = 1 -> Error code in EAX)

1929 <1> ;

1930 <1> ; For BH = 6 -> CONTINUE TO PLAY

1931 <1> ; none (if CF = 1 -> Error code in EAX)

1932 <1> ;

1933 <1> ; For BH = 7 -> STOP

1934 <1> ; none (if CF = 1 -> Error code in EAX)

1935 <1> ;

1936 <1> ; For BH = 8 -> RESET

1937 <1> ; none (if CF = 1 -> Error code in EAX)

1938 <1> ;

1939 <1> ; For BH = 9 -> CANCEL (CALLBACK or S.R.B. SERVICE)

1940 <1> ; none (if CF = 1 -> Error code in EAX)

1941 <1> ;

1942 <1> ; For BH = 10 -> DEALLOCATE AUDIO BUFFER (for user)

1943 <1> ; none (if CF = 1 -> Error code in EAX)

1944 <1> ;

1945 <1> ; For BH = 11 -> SET VOLUME LEVEL

1946 <1> ; none (if CF = 1 -> Error code in EAX)

1947 <1> ;

1948 <1> ; For BH = 12 -> DISABLE AUDIO DEVICE

1949 <1> ; none (if CF = 1 -> Error code in EAX)

1950 <1> ;

1951 <1> ; 12/05/2017

1952 <1> ; For BH = 13 -> MAP DMA BUFFER TO USER

1953 <1> ; EAX = Physical Address of the buffer

1954 <1> ; (if CF = 1 -> Error code in EAX)

1955 <1> ;

1956 <1> ; 04/06/2017

1957 <1> ; For BH = 14 -> GET AUDIO DEVICE INFO

1958 <1> ; (for BL = 0) ; 05/06/2017

1959 <1> ; EAX = IRQ Number in AL

1960 <1> ; Audio Device Number in AH

1961 <1> ; EBX = DEV/VENDOR ID

1962 <1> ; (DDDDDDDDDDDDDDDDVVVVVVVVVVVVVVVV)

1963 <1> ; ECX = BUS/DEV/FN

1964 <1> ; (00000000BBBBBBBBDDDDDFFF00000000)

1965 <1> ; EDX = NABMBAR/NAMBAR (for AC97)

1966 <1> ; (Low word, DX = NAMBAR address)

1967 <1> ; EDX = Base IO Addr (DX) for SB16 & VT8233

1968 <1> ; (if CF = 1 -> Error code in EAX)

1969 <1> ; (ERR\_DEV\_NOT\_RDY = 15)

1970 <1> ;

1971 <1> ; 22/06/2017

1972 <1> ; For BH = 15 -> GET CURRENT SOUND DATA

1973 <1> ; (for graphics)

1974 <1> ; (for BL = 0)

1975 <1> ; If ECX input is 0

1976 <1> ; EAX = DMA Buffer Current Position (Offset)

1977 <1> ; If ECX input > 0

1978 <1> ; EAX = Actual transfer count

1979 <1> ; (Sound samples will be copied from

1980 <1> ; Current DMA Buffer Position to EDX

1981 <1> ; virtual address as EAX bytes.)

1982 <1> ; ((If CF = 1 -> Error code in EAX))

1983 <1> ;

1984 <1> ;

1985 <1> ; 10/10/2017

1986 <1> ; For BH = 16 -> UPDATE DMA BUFFER DATA

1987 <1> ; EAX = 0, if the updated (or current)

1988 <1> ; half buffer is DMA half buffer 1

1989 <1> ; EAX = 1, if the updated (or current)

1990 <1> ; half buffer is DMA half buffer 2

1991 <1> ; (If CF = 1 -> Error code in EAX)

1992 <1> ;

1993 <1>

1994 000100E9 80FF11 <1> cmp bh, AUDIO1L/4

1995 000100EC 0F83ECC5FFFF <1> jnb sysret

1996 <1>

1997 000100F2 C0E702 <1> shl bh, 2 ; \*4

1998 000100F5 0FB6F7 <1> movzx esi, bh

1999 <1>

2000 <1> ; 22/04/2017

2001 000100F8 31C0 <1> xor eax, eax

2002 000100FA A3[64030300] <1> mov [u.r0], eax ; 0

2003 <1>

2004 000100FF FF96[0A010100] <1> call dword [esi+AUDIO1]

2005 <1> ;jc error

2006 00010105 E9D4C5FFFF <1> jmp sysret

2007 <1>

2008 0001010A [A11D0000] <1> AUDIO1: dd beep ; FUNCTION = 0 (bl = Duration Counter

2009 <1> ; cx = Frequency Divisor

2010 0001010E [4E010100] <1> dd soundc\_detect

2011 00010112 [EA010100] <1> dd sound\_alloc

2012 00010116 [A1020100] <1> dd soundc\_init

2013 0001011A [59040100] <1> dd sound\_play

2014 0001011E [EF040100] <1> dd sound\_pause

2015 00010122 [19050100] <1> dd sound\_continue

2016 00010126 [43050100] <1> dd sound\_stop

2017 0001012A [6C050100] <1> dd soundc\_reset

2018 0001012E [9D050100] <1> dd soundc\_cancel

2019 00010132 [C3050100] <1> dd sound\_dalloc

2020 00010136 [EE050100] <1> dd sound\_volume

2021 0001013A [40060100] <1> dd soundc\_disable

2022 0001013E [B2060100] <1> dd sound\_dma\_map

2023 00010142 [21070100] <1> dd soundc\_info

2024 00010146 [80070100] <1> dd sound\_data

2025 0001014A [2D080100] <1> dd sound\_update

2026 <1>

2027 <1> AUDIO1L EQU $ - AUDIO1

2028 <1>

2029 <1> soundc\_detect:

2030 <1> ; FUNCTION = 1

2031 <1> ; bl = Audio device type number

2032 <1> ; (0= pc speaker, 1 = sound blaster 16, 2 = intel ac97

2033 <1> ; 3= via vt823x, 4 = intel HDA, 0FFh= any)

2034 <1>

2035 <1> ; 04/06/2017

2036 0001014E 8A25[BD6B0100] <1> mov ah, [audio\_device]

2037 00010154 80FBFF <1> cmp bl, 0FFh ; get current audio device id

2038 00010157 7408 <1> je short sysaudio0

2039 <1>

2040 00010159 20E4 <1> and ah, ah

2041 0001015B 741E <1> jz short soundc\_get\_dev

2042 <1>

2043 0001015D 38DC <1> cmp ah, bl

2044 0001015F 7567 <1> jne short soundc\_dev\_err

2045 <1>

2046 <1> sysaudio0:

2047 00010161 A0[BE6B0100] <1> mov al, [audio\_mode]

2048 <1> sysaudio1:

2049 00010166 A3[64030300] <1> mov [u.r0], eax

2050 0001016B 8B1D[C86B0100] <1> mov ebx, [audio\_vendor] ; (DEVICE/VENDOR ID)

2051 00010171 8B2D[60030300] <1> mov ebp, [u.usp]

2052 00010177 895D10 <1> mov [ebp+16], ebx ; ebx

2053 0001017A C3 <1> retn

2054 <1>

2055 <1> soundc\_get\_dev:

2056 <1> ; 28/05/2017

2057 <1> ; 03/04/2017, 24/04/2017

2058 0001017B C605[BC6B0100]00 <1> mov byte [audio\_pci], 0

2059 00010182 80FB03 <1> cmp bl, 3 ; VIA VT8233 (VT8237R) Audio Controller & AC97 Codec

2060 <1> ;jne short soundc\_get\_dev\_sb

2061 <1> ; 28/05/2017

2062 00010185 7220 <1> jb short soundc\_get\_dev\_sb

2063 00010187 773F <1> ja short soundc\_dev\_err ; temporary (28/05/2017)

2064 <1> ;

2065 00010189 E83A180000 <1> call DetectVT8233

2066 0001018E 7238 <1> jc short soundc\_dev\_err

2067 <1> ; eax = 0

2068 <1>

2069 <1> ;mov ebx, [audio\_vendor]

2070 <1> ; ebx = DEVICE/VENDOR ID

2071 <1> ; DDDDDDDDDDDDDDDDVVVVVVVVVVVVVVVV

2072 <1>

2073 00010190 B003 <1> mov al, 3 ; VIA VT8237R (VT3233) Audio Controller

2074 00010192 88C4 <1> mov ah, al

2075 <1>

2076 <1> soundc\_get\_pci\_dev\_ok: ; 28/05/2017

2077 00010194 FE05[BC6B0100] <1> inc byte [audio\_pci] ; = 1

2078 <1> soundc\_get\_dev\_ok:

2079 <1>

2080 <1> soundc\_get\_dev\_sb16\_ok:

2081 0001019A A2[BD6B0100] <1> mov [audio\_device], al

2082 0001019F 8825[BE6B0100] <1> mov [audio\_mode], ah ; stereo (bit0), 16 bit (bit1) capability

2083 000101A5 EBBF <1> jmp short sysaudio1

2084 <1>

2085 <1> soundc\_get\_dev\_sb:

2086 <1> ; 24/04/2017

2087 000101A7 80FB01 <1> cmp bl, 1 ; Sound Blaster 16

2088 000101AA 750E <1> jne short soundc\_get\_dev\_ich ; 28/05/2017

2089 <1> ;

2090 000101AC E8451D0000 <1> call DetectSB

2091 000101B1 7215 <1> jc short soundc\_dev\_err

2092 000101B3 B801030000 <1> mov eax, 0301h ; Sound Blaster 16

2093 000101B8 EBE0 <1> jmp short soundc\_get\_dev\_sb16\_ok

2094 <1>

2095 <1> soundc\_get\_dev\_ich:

2096 <1> ; 28/05/2017

2097 <1> ;cmp bl, 2 ; Intel AC'97 Audio Controller (ICH)

2098 <1> ;jne short soundc\_dev\_err ; Temporary (28/05/2017)

2099 <1> ; ; (Here will be modified just after

2100 <1> ; ; new sound card code will be ready!)

2101 000101BA E8FC170000 <1> call DetectICH

2102 000101BF 7207 <1> jc short soundc\_dev\_err

2103 <1> ;

2104 000101C1 B802030000 <1> mov eax, 0302h ; AC'97 (ICH)

2105 000101C6 EBCC <1> jmp short soundc\_get\_pci\_dev\_ok

2106 <1>

2107 <1> soundc\_dev\_err:

2108 000101C8 B80F000000 <1> mov eax, ERR\_DEV\_NOT\_RDY ; Device not ready !

2109 000101CD EB0C <1> jmp short sysaudio\_err

2110 <1>

2111 <1> sound\_buff\_error:

2112 000101CF B82E000000 <1> mov eax, ERR\_BUFFER ; Buffer error !

2113 000101D4 EB05 <1> jmp short sysaudio\_err

2114 <1>

2115 <1> soundc\_respond\_err:

2116 <1> ; ERR\_TIME\_OUT ; 'time out !' error

2117 000101D6 B819000000 <1> mov eax, ERR\_DEV\_NOT\_RESP ; 'device not responding !' error

2118 <1> sysaudio\_err:

2119 000101DB A3[64030300] <1> mov [u.r0], eax

2120 000101E0 A3[C8030300] <1> mov [u.error], eax

2121 000101E5 E9D4C4FFFF <1> jmp error

2122 <1>

2123 <1> sound\_alloc:

2124 <1> ; FUNCTION = 2

2125 <1> ; ecx = audio buffer size (in bytes)

2126 <1> ; edx = audio buffer address (virtual)

2127 <1> ; 28/05/2017

2128 <1> ; 01/05/2017, 15/05/2017

2129 <1> ; 21/04/2017, 24/04/2017

2130 000101EA 803D[BC6B0100]00 <1> cmp byte [audio\_pci], 0

2131 000101F1 7708 <1> ja short snd\_alloc\_0

2132 <1> ; Max. 64KB DMA buffer !!!

2133 000101F3 81F900800000 <1> cmp ecx, 32768

2134 000101F9 77D4 <1> ja short sound\_buff\_error

2135 <1> snd\_alloc\_0:

2136 <1> ; 15/05/2017

2137 000101FB 81F900100000 <1> cmp ecx, 4096 ; PAGE\_SIZE

2138 00010201 72CC <1> jb short sound\_buff\_error

2139 <1> ;

2140 00010203 A1[D06B0100] <1> mov eax, [audio\_buffer] ; audio buffer address (current)

2141 00010208 09C0 <1> or eax, eax

2142 0001020A 7445 <1> jz short snd\_alloc\_2

2143 <1> ; audio buffer exists !

2144 0001020C 8A1D[B3030300] <1> mov bl, [u.uno]

2145 00010212 3A1D[E56B0100] <1> cmp bl, [audio\_user]

2146 00010218 0F85F5000000 <1> jne sndc\_owner\_error ; not owner !

2147 0001021E 39D0 <1> cmp eax, edx ; same virtual buffer address ?

2148 00010220 7508 <1> jne short snd\_alloc\_1

2149 00010222 3B0D[D86B0100] <1> cmp ecx, [audio\_buff\_size]

2150 00010228 746C <1> je short snd\_alloc\_3 ; Nothing to do !

2151 <1> ; Buffer has been set already!

2152 <1> snd\_alloc\_1:

2153 0001022A 51 <1> push ecx

2154 0001022B 52 <1> push edx

2155 0001022C 89C3 <1> mov ebx, eax ; audio buffer address (current)

2156 0001022E 8B0D[D86B0100] <1> mov ecx, [audio\_buff\_size]

2157 00010234 E84655FFFF <1> call deallocate\_user\_pages

2158 00010239 5A <1> pop edx

2159 0001023A 59 <1> pop ecx

2160 0001023B 31C0 <1> xor eax, eax ; 0

2161 0001023D A3[D06B0100] <1> mov [audio\_buffer], eax ; 0

2162 00010242 A3[D46B0100] <1> mov [audio\_p\_buffer], eax ; 0

2163 00010247 A3[D86B0100] <1> mov [audio\_buff\_size], eax

2164 0001024C A2[E56B0100] <1> mov [audio\_user], al ; 0

2165 <1> snd\_alloc\_2:

2166 00010251 89D3 <1> mov ebx, edx

2167 <1> ; 01/05/2017

2168 00010253 BA00F0FFFF <1> mov edx, ~PAGE\_OFF ; truncating page offsets

2169 <1> ; for aligning to page borders

2170 <1> ;and eax, edx

2171 00010258 21D3 <1> and ebx, edx

2172 0001025A 21D1 <1> and ecx, edx

2173 <1> ; 15/05/2017

2174 <1> ; EAX = Beginning address (physical)

2175 <1> ; EAX = 0 -> Allocate mem block from the 1st proper aperture

2176 <1> ; ECX = Number of bytes to be allocated

2177 0001025C E8C351FFFF <1> call allocate\_memory\_block

2178 00010261 0F8268FFFFFF <1> jc sound\_buff\_error

2179 <1> ; EAX = Physical address of the allocated memory block

2180 <1> ; ECX = Allocated bytes (as truncated to page border)

2181 <1> ; EBX = Virtual address (as truncated to page border)

2182 00010267 50 <1> push eax

2183 00010268 53 <1> push ebx

2184 00010269 51 <1> push ecx

2185 0001026A E80556FFFF <1> call allocate\_user\_pages

2186 0001026F 59 <1> pop ecx

2187 00010270 5B <1> pop ebx

2188 00010271 58 <1> pop eax

2189 00010272 7223 <1> jc short snd\_alloc\_4 ; insufficient memory, buff error

2190 <1> ; eax = physical address of the user's audio buffer

2191 <1> ; ebx = virtual address of the user's audio buffer

2192 <1> ; ecx = buffer size (in bytes)

2193 00010274 A3[D46B0100] <1> mov [audio\_p\_buffer], eax

2194 00010279 891D[D06B0100] <1> mov [audio\_buffer], ebx

2195 0001027F 890D[D86B0100] <1> mov [audio\_buff\_size], ecx

2196 00010285 8A15[B3030300] <1> mov dl, [u.uno]

2197 0001028B 8815[E56B0100] <1> mov [audio\_user], dl

2198 00010291 A3[64030300] <1> mov [u.r0], eax

2199 <1> snd\_alloc\_3:

2200 00010296 C3 <1> retn

2201 <1> snd\_alloc\_4:

2202 <1> ; 15/05/2017

2203 <1> ; EAX = Beginning address (physical)

2204 <1> ; ECX = Number of bytes to be deallocated

2205 00010297 E89553FFFF <1> call deallocate\_memory\_block

2206 0001029C E92EFFFFFF <1> jmp sound\_buff\_error ; insufficient memory, buff error

2207 <1>

2208 <1> soundc\_init:

2209 <1> ; FUNCTION = 3

2210 <1> ; bl = method (0= s.r.b., 1= callback, 2= auto incr s.r.b.)

2211 <1> ; cl = signal response byte (initial or fixed) value

2212 <1> ; edx = signal response byte or callback address

2213 <1> ; 28/05/2017

2214 <1> ; 12/05/2017, 20/05/2017

2215 <1> ; 22/04/2017, 23/04/2017, 24/04/2017

2216 <1> ; 13/04/2017, 14/04/2017, 16/04/2017, 21/04/2017

2217 <1> ; 03/04/2017, 10/04/2017

2218 <1>

2219 000102A1 A0[BD6B0100] <1> mov al, [audio\_device]

2220 000102A6 20C0 <1> and al, al

2221 000102A8 7549 <1> jnz short sndc\_init\_6

2222 <1> ;

2223 000102AA C605[BC6B0100]00 <1> mov byte [audio\_pci], 0

2224 000102B1 52 <1> push edx

2225 000102B2 53 <1> push ebx

2226 000102B3 51 <1> push ecx

2227 000102B4 E83D1C0000 <1> call DetectSB

2228 000102B9 7213 <1> jc short sndc\_init\_8

2229 000102BB 66B80103 <1> mov ax, 0301h ; Sound Blaster 16

2230 000102BF EB1E <1> jmp short sndc\_init\_7

2231 <1>

2232 <1> sndc\_init\_11:

2233 <1> ; 28/05/2017

2234 000102C1 E8F5160000 <1> call DetectICH ; Detect AC'97 (ICH) Audio Controller

2235 000102C6 7217 <1> jc short sndc\_init\_7

2236 000102C8 66B80203 <1> mov ax, 0302h ; Intel AC'97 Audio Device

2237 000102CC EB0B <1> jmp short sndc\_init\_12 ; (PCI device)

2238 <1>

2239 <1> sndc\_init\_8:

2240 000102CE E8F5160000 <1> call DetectVT8233

2241 <1> ;jc short sndc\_init\_7

2242 000102D3 72EC <1> jc sndc\_init\_11 ; 28/05/2017

2243 <1> ; eax = 0

2244 000102D5 B003 <1> mov al, 3 ; VIA VT8237R (VT3233) Audio Controller

2245 000102D7 88C4 <1> mov ah, al

2246 <1>

2247 <1> sndc\_init\_12:

2248 000102D9 FE05[BC6B0100] <1> inc byte [audio\_pci] ; = 1

2249 <1> sndc\_init\_7:

2250 000102DF 59 <1> pop ecx

2251 000102E0 5B <1> pop ebx

2252 000102E1 5A <1> pop edx

2253 000102E2 0F82E0FEFFFF <1> jc soundc\_dev\_err

2254 <1> ;

2255 000102E8 A2[BD6B0100] <1> mov [audio\_device], al

2256 000102ED 8825[BE6B0100] <1> mov [audio\_mode], ah ; stereo (bit0), 16 bit (bit1) capability

2257 <1>

2258 <1> sndc\_init\_6:

2259 000102F3 833D[D06B0100]00 <1> cmp dword [audio\_buffer], 0

2260 000102FA 0F86CFFEFFFF <1> jna sound\_buff\_error

2261 <1>

2262 00010300 A0[B3030300] <1> mov al, [u.uno]

2263 00010305 8A25[E56B0100] <1> mov ah, [audio\_user]

2264 0001030B 08E4 <1> or ah, ah

2265 0001030D 7418 <1> jz short sndc\_init0

2266 0001030F 38E0 <1> cmp al, ah

2267 00010311 7419 <1> je short sndc\_init1

2268 <1>

2269 <1> sndc\_owner\_error:

2270 00010313 B80B000000 <1> mov eax, ERR\_NOT\_OWNER ; 'permission denied !' error

2271 <1> sndc\_perm\_error:

2272 00010318 A3[64030300] <1> mov [u.r0], eax

2273 0001031D A3[C8030300] <1> mov [u.error], eax

2274 00010322 E997C3FFFF <1> jmp error

2275 <1> sndc\_init0:

2276 00010327 A2[E56B0100] <1> mov [audio\_user], al

2277 <1> sndc\_init1:

2278 0001032C 8915[E86B0100] <1> mov [audio\_cb\_addr], edx

2279 00010332 881D[E66B0100] <1> mov [audio\_cb\_mode], bl

2280 00010338 880D[E76B0100] <1> mov [audio\_srb], cl

2281 <1>

2282 <1> ; 24/04/2017

2283 0001033E 803D[BD6B0100]03 <1> cmp byte [audio\_device], 3 ; VT8233 (VT8237R)

2284 00010345 7438 <1> je short sndc\_init\_9

2285 <1> ;ja short soundc\_respond\_err ; temporary (28/05/2017)

2286 00010347 803D[BD6B0100]01 <1> cmp byte [audio\_device], 1 ; SB 16

2287 0001034E 7510 <1> jne short sndc\_init\_13

2288 00010350 BB[1B210100] <1> mov ebx, sb16\_int\_handler

2289 <1> ; Note: 'SbInit' is at 'Start to Play' stage

2290 <1> ; 20/05/2017

2291 00010355 66C705[F26B0100]08- <1> mov word [audio\_master\_volume], 0808h ; 2/8

2291 0001035D 08 <1>

2292 0001035E EB3F <1> jmp short sndc\_init\_10

2293 <1> sndc\_init\_13:

2294 <1> ; 28/05/2017

2295 00010360 803D[BD6B0100]02 <1> cmp byte [audio\_device], 2 ; AC 97 (ICH)

2296 00010367 0F8569FEFFFF <1> jne soundc\_respond\_err ; temporary (28/05/2017)

2297 <1>

2298 0001036D E8FE1E0000 <1> call ac97\_codec\_config

2299 00010372 0F825EFEFFFF <1> jc soundc\_respond\_err ; codec error !

2300 <1>

2301 00010378 BB[57240100] <1> mov ebx, ac97\_int\_handler

2302 0001037D EB20 <1> jmp short sndc\_init\_10

2303 <1>

2304 <1> sndc\_init\_9:

2305 <1> ;call reset\_codec

2306 <1> ;; eax = 1

2307 <1> ;call codec\_io\_w16 ; w32

2308 0001037F E8BB170000 <1> call init\_codec ; 28/05/2017

2309 00010384 0F824CFEFFFF <1> jc soundc\_respond\_err ; codec error !

2310 <1>

2311 0001038A E8EC190000 <1> call channel\_reset

2312 <1>

2313 <1> ; setup the Codec (actually mixer registers)

2314 0001038F E8F6180000 <1> call codec\_config ; unmute codec, set rates.

2315 00010394 0F823CFEFFFF <1> jc soundc\_respond\_err ; codec error !

2316 <1>

2317 0001039A BB[F71C0100] <1> mov ebx, vt8233\_int\_handler

2318 <1> sndc\_init\_10:

2319 <1> ; 13/04/2017

2320 0001039F A0[BF6B0100] <1> mov al, [audio\_intr] ; IRQ number

2321 000103A4 E832FDFFFF <1> call set\_dev\_IRQ\_service

2322 <1>

2323 <1> ; SETUP (audio) INTERRUPT CALLBACK SERVICE

2324 000103A9 8A1D[BF6B0100] <1> mov bl, [audio\_intr] ; IRQ number

2325 000103AF 8A3D[E66B0100] <1> mov bh, [audio\_cb\_mode]

2326 000103B5 FEC7 <1> inc bh ; 1 = Signal Response Byte method (fixed value)

2327 <1> ; 2 = Callback service method

2328 <1> ; 3 = Auto Increment S.R.B. method

2329 000103B7 8A0D[E76B0100] <1> mov cl, [audio\_srb]

2330 000103BD 8B15[E86B0100] <1> mov edx, [audio\_cb\_addr]

2331 000103C3 A0[E56B0100] <1> mov al, [audio\_user]

2332 <1> ; 14/04/2017

2333 000103C8 E8DB040000 <1> call set\_irq\_callback\_service

2334 <1> ; 16/04/2017

2335 000103CD A3[64030300] <1> mov [u.r0], eax

2336 <1> ;jnc sysret

2337 000103D2 7316 <1> jnc short sndc\_init2 ; 21/04/2017

2338 <1> ;

2339 000103D4 A3[C8030300] <1> mov dword [u.error], eax

2340 <1>

2341 000103D9 A0[BF6B0100] <1> mov al, [audio\_intr] ; IRQ number

2342 000103DE 31DB <1> xor ebx, ebx ; reset IRQ handler address

2343 000103E0 E8F6FCFFFF <1> call set\_dev\_IRQ\_service

2344 <1>

2345 000103E5 E9D4C2FFFF <1> jmp error

2346 <1>

2347 <1> sndc\_init2:

2348 <1> ; 21/04/2017

2349 000103EA 8B0D[D86B0100] <1> mov ecx, [audio\_buff\_size] ; audio buffer size

2350 000103F0 D1E1 <1> shl ecx, 1 ; \*2

2351 000103F2 A1[DC6B0100] <1> mov eax, [audio\_dma\_buff]

2352 000103F7 21C0 <1> and eax, eax

2353 000103F9 7415 <1> jz short sndc\_init3

2354 <1>

2355 000103FB 8B15[E06B0100] <1> mov edx, [audio\_dmabuff\_size] ; dma buffer size

2356 00010401 39D1 <1> cmp ecx, edx

2357 00010403 744D <1> je short sndc\_init5

2358 <1>

2359 00010405 87CA <1> xchg ecx, edx

2360 00010407 E82552FFFF <1> call deallocate\_memory\_block

2361 0001040C 87D1 <1> xchg edx, ecx

2362 0001040E 31C0 <1> xor eax, eax

2363 <1> sndc\_init3:

2364 <1> ; 12/05/2017

2365 00010410 803D[BD6B0100]01 <1> cmp byte [audio\_device], 1 ; SB 16

2366 00010417 7515 <1> jne short sndc\_init4

2367 00010419 C705[DC6B0100]- <1> mov dword [audio\_dma\_buff], sb16\_dma\_buffer

2367 0001041F [00000200] <1>

2368 00010423 C705[E06B0100]0000- <1> mov dword [audio\_dmabuff\_size], 65536

2368 0001042B 0100 <1>

2369 <1> ;xor eax, eax

2370 <1> ;mov [u.r0], eax ; 0 = no error, successful

2371 0001042D C3 <1> retn

2372 <1>

2373 <1> sndc\_init4:

2374 <1> ; EAX = Beginning address (physical)

2375 <1> ; EAX = 0 -> Allocate mem block from the 1st proper aperture

2376 <1> ; ECX = Number of bytes to be allocated (>0)

2377 0001042E E8F14FFFFF <1> call allocate\_memory\_block

2378 00010433 0F8296FDFFFF <1> jc sound\_buff\_error

2379 <1>

2380 <1> ; set dma buffer address and size parameters

2381 00010439 A3[DC6B0100] <1> mov [audio\_dma\_buff], eax ; dma buffer address

2382 0001043E 890D[E06B0100] <1> mov [audio\_dmabuff\_size], ecx ; dma buffer size

2383 <1> ; ; EAX = Beginning (physical) addr of the allocated mem block

2384 <1> ; ; ECX = Num of allocated bytes (rounded up to page borders)

2385 <1> ; cmp byte [audio\_pci], 0 ; AC97 audio controller ?

2386 <1> ; ja short sndc\_init4

2387 <1> ;

2388 <1> ; ; Sound Blaster 16 uses classic DMA

2389 <1> ; mov edx, eax

2390 <1> ; add edx, ecx

2391 <1> ; cmp edx, 1000000h ; 1st 16 MB

2392 <1> ; jna short sndc\_init4

2393 <1> ;

2394 <1> ; ; error !

2395 <1> ; ; restore Memory Allocation Table Content

2396 <1> ; ; EAX = Beginning address (physical)

2397 <1> ; ; ECX = Number of bytes to be deallocated

2398 <1> ; call deallocate\_memory\_block

2399 <1> ; ; reset dma buffer address and size parameters

2400 <1> ; xor eax, eax ; 0

2401 <1> ; mov [audio\_dma\_buff], eax ; 0

2402 <1> ; mov [audio\_dmabuff\_size], ecx ; 0

2403 <1> ; jmp sound\_buff\_error

2404 <1> ;

2405 <1> ;sndc\_init4:

2406 00010444 803D[BD6B0100]03 <1> cmp byte [audio\_device], 3

2407 <1> ;jne short sndc\_init5

2408 0001044B 7506 <1> jne short sndc\_init14 ; 28/05/2017

2409 0001044D E86A190000 <1> call set\_vt8233\_bdl

2410 <1> sndc\_init5:

2411 <1> ;sub eax, eax ; 0

2412 <1> ;mov [u.r0], eax ; 0 = no error, successful

2413 00010452 C3 <1> retn

2414 <1> sndc\_init14:

2415 00010453 E8311F0000 <1> call set\_ac97\_bdl

2416 <1> ;jmp short sndc\_init5

2417 00010458 C3 <1> retn

2418 <1>

2419 <1> sound\_play:

2420 <1> ; FUNCTION = 4

2421 <1> ; bl = Mode

2422 <1> ; bit 0 = mono/stereo (1 = stereo)

2423 <1> ; bit 1 = 8 bit / 16 bit (1 = 16 bit)

2424 <1> ; cx = Sampling Rate (Hz)

2425 <1>

2426 <1> ; 13/06/2017

2427 <1> ; Note: Even if Mode bits are not 11b,

2428 <1> ; AC'97 Audio Controller (&Codec)

2429 <1> ; will play audio samples as 16 bit, stereo

2430 <1> ; samples.

2431 <1> ; (Program must fill the audio buffer

2432 <1> ; as required; 8 bit samples must be converted

2433 <1> ; to 16 bit samples and mono samples must be

2434 <1> ; converted to stereo samples...)

2435 <1> ;

2436 <1> ; 28/05/2017

2437 <1> ; 15/05/2017, 20/05/2017

2438 <1> ; 21/04/2017, 24/04/2017

2439 <1> ; ... device check at first

2440 00010459 A0[BD6B0100] <1> mov al, [audio\_device]

2441 0001045E 08C0 <1> or al, al ; 0 ; pc speaker or invalid

2442 00010460 0F843519FFFF <1> jz beeper\_gfx ; 'video.s' ; temporary !

2443 <1> ; cmp al, 3 ; VIA VT 8237R (vt8233)

2444 <1> ; je short snd\_play\_1

2445 <1> ; cmp al, 1 ; SB 16

2446 <1> ; jne soundc\_dev\_err ; temporary !

2447 <1> ;snd\_play\_0:

2448 <1> ; ... buffer & (buffer) owner check at second

2449 00010466 833D[D06B0100]00 <1> cmp dword [audio\_buffer], 0

2450 0001046D 0F865CFDFFFF <1> jna sound\_buff\_error

2451 00010473 A0[B3030300] <1> mov al, [u.uno]

2452 00010478 3A05[E56B0100] <1> cmp al, [audio\_user]

2453 0001047E 0F858FFEFFFF <1> jne sndc\_owner\_error

2454 <1>

2455 00010484 66890D[EE6B0100] <1> mov [audio\_freq], cx ; sample frequency (Hertz)

2456 0001048B 88D8 <1> mov al, bl

2457 0001048D 2401 <1> and al, 1 ; mono/stereo (1= stereo)

2458 0001048F FEC0 <1> inc al ; channels

2459 00010491 A2[ED6B0100] <1> mov [audio\_stmo], al ; sound channels (1 or 2)

2460 00010496 B008 <1> mov al, 8

2461 00010498 F6C302 <1> test bl, 2 ; bits per sample (1= 16 bit)

2462 0001049B 7402 <1> jz short snd\_play\_bps

2463 0001049D D0E0 <1> shl al, 1

2464 <1> snd\_play\_bps:

2465 0001049F A2[EC6B0100] <1> mov [audio\_bps], al

2466 <1> ; Transfer ring 3 (user's) audio buffer content to dma buffer

2467 000104A4 8B3D[DC6B0100] <1> mov edi, [audio\_dma\_buff] ; dma buffer (ring 0)

2468 000104AA 09FF <1> or edi, edi

2469 000104AC 0F841DFDFFFF <1> jz sound\_buff\_error

2470 000104B2 8B35[D46B0100] <1> mov esi, [audio\_p\_buffer] ; physical address (ring 3)

2471 000104B8 8B0D[D86B0100] <1> mov ecx, [audio\_buff\_size] ; 15/05/2017

2472 <1> ;rep movsb

2473 000104BE C1E902 <1> shr ecx, 2

2474 000104C1 F3A5 <1> rep movsd

2475 <1> ; 20/05/2017

2476 000104C3 C605[E46B0100]01 <1> mov byte [audio\_flag], 1 ; next half (on next time)

2477 <1>

2478 <1> ; 24/04/2017

2479 000104CA A0[BD6B0100] <1> mov al, [audio\_device]

2480 000104CF 3C03 <1> cmp al, 3 ; VT8233 (VT8237R)

2481 000104D1 7410 <1> je short snd\_play\_1

2482 000104D3 3C01 <1> cmp al, 1 ; Sound Blaster 16

2483 000104D5 7512 <1> jne short snd\_play\_2 ; 28/05/2017

2484 000104D7 E8E81A0000 <1> call SbInit\_play

2485 000104DC 0F82F4FCFFFF <1> jc soundc\_respond\_err

2486 000104E2 C3 <1> retn

2487 <1>

2488 <1> snd\_play\_1:

2489 000104E3 E804190000 <1> call vt8233\_start\_play

2490 000104E8 C3 <1> retn

2491 <1>

2492 <1> snd\_play\_2:

2493 <1> ; 28/05/2017

2494 <1> ;cmp al, 2 ; AC'97

2495 <1> ;jne short snd\_play\_3

2496 <1>

2497 000104E9 E8CF1E0000 <1> call ac97\_start\_play

2498 000104EE C3 <1> retn

2499 <1>

2500 <1> ;snd\_play\_3:

2501 <1> ; ;call hda\_start\_play

2502 <1> ; retn

2503 <1>

2504 <1> sound\_pause:

2505 <1> ; FUNCTION = 5

2506 <1> ; Pause

2507 <1> ; 28/05/2017

2508 <1> ; 24/04/2017

2509 <1> ; 22/04/2017

2510 000104EF E814030000 <1> call snd\_dev\_check

2511 000104F4 7275 <1> jc short snd\_nothing ; temporary.

2512 000104F6 E81A030000 <1> call snd\_buf\_check

2513 000104FB 726E <1> jc short snd\_nothing ; temporary.

2514 000104FD A0[BD6B0100] <1> mov al, [audio\_device]

2515 00010502 3C03 <1> cmp al, 3 ; VIA VT 8237R (vt8233)

2516 00010504 7409 <1> je short snd\_pause\_1

2517 00010506 3C01 <1> cmp al, 1 ; Sound Blaster 16

2518 00010508 750A <1> jne short snd\_pause\_2 ; 28/05/2017

2519 0001050A E9931C0000 <1> jmp sb16\_pause

2520 <1> snd\_pause\_1:

2521 0001050F E996190000 <1> jmp vt8233\_pause

2522 <1> snd\_pause\_2:

2523 <1> ; 28/05/2017

2524 <1> ;cmp al, 2 ; AC'97

2525 <1> ;jne short snd\_nothing ; temporary.

2526 00010514 E932200000 <1> jmp ac97\_pause

2527 <1>

2528 <1> sound\_continue:

2529 <1> ; FUNCTION = 6

2530 <1> ; Continue to play

2531 <1> ; 28/05/2017

2532 <1> ; 22/04/2017

2533 00010519 E8EA020000 <1> call snd\_dev\_check

2534 0001051E 724B <1> jc short snd\_nothing ; temporary.

2535 00010520 E8F0020000 <1> call snd\_buf\_check

2536 00010525 7244 <1> jc short snd\_nothing ; temporary.

2537 00010527 A0[BD6B0100] <1> mov al, [audio\_device]

2538 0001052C 3C03 <1> cmp al, 3 ; VIA VT 8237R (vt8233)

2539 0001052E 7409 <1> je short snd\_cont\_1

2540 00010530 3C01 <1> cmp al, 1 ; Sound Blaster 16

2541 00010532 750A <1> jne short snd\_cont\_2 ; 28/05/2017

2542 00010534 E98C1C0000 <1> jmp sb16\_continue

2543 <1> snd\_cont\_1:

2544 00010539 E919190000 <1> jmp vt8233\_play

2545 <1> snd\_cont\_2:

2546 <1> ; 28/05/2017

2547 <1> ;cmp al, 2 ; AC'97

2548 <1> ;jne short snd\_nothing ; temporary.

2549 0001053E E9D01E0000 <1> jmp ac97\_play

2550 <1>

2551 <1> sound\_stop:

2552 <1> ; FUNCTION = 7

2553 <1> ; Stop playing

2554 <1> ; 28/05/2017

2555 <1> ; 24/05/2017

2556 <1> ; 21/04/2017, 22/04/2017, 24/04/2017

2557 00010543 E8C0020000 <1> call snd\_dev\_check

2558 00010548 7221 <1> jc short snd\_nothing ; temporary.

2559 <1> ;call snd\_buf\_check

2560 0001054A E8CF020000 <1> call snd\_user\_check ; 24/05/2017

2561 0001054F 721A <1> jc short snd\_nothing ; temporary.

2562 <1>

2563 00010551 A0[BD6B0100] <1> mov al, [audio\_device]

2564 00010556 3C03 <1> cmp al, 3 ; VIA VT 8237R (vt8233)

2565 00010558 0F8455180000 <1> je vt8233\_stop

2566 <1> ; 28/05/2017

2567 <1> ;ja short snd\_nothing

2568 0001055E 3C01 <1> cmp al, 1 ; Sound Blaster 16

2569 00010560 0F84821C0000 <1> je sb16\_stop

2570 <1> ;cmp al, 2

2571 <1> ;je short ac97\_stop

2572 00010566 E9B21F0000 <1> jmp ac97\_stop ; temporary.

2573 <1> ;jmp hda\_stop

2574 <1>

2575 <1> snd\_nothing:

2576 <1> ; 21/04/2017

2577 0001056B C3 <1> retn

2578 <1>

2579 <1> soundc\_reset:

2580 <1> ; FUNCTION = 8

2581 <1> ; Reset Audio Controller

2582 <1> ; 28/05/2017

2583 <1> ; 22/04/2017

2584 0001056C E897020000 <1> call snd\_dev\_check

2585 00010571 72F8 <1> jc snd\_nothing ; temporary.

2586 00010573 E89D020000 <1> call snd\_buf\_check

2587 00010578 72F1 <1> jc snd\_nothing ; temporary.

2588 <1>

2589 0001057A A0[BD6B0100] <1> mov al, [audio\_device]

2590 0001057F 3C03 <1> cmp al, 3 ; VIA VT 8237R (vt8233)

2591 00010581 0F8431190000 <1> je vt8233\_reset

2592 00010587 77E2 <1> ja short snd\_nothing ; temporary.

2593 <1> ;ja hda\_reset

2594 00010589 3C01 <1> cmp al, 1 ; Sound Blaster 16

2595 0001058B 0F850B200000 <1> jne ac97\_reset

2596 00010591 E8A41C0000 <1> call sb16\_reset

2597 00010596 0F823AFCFFFF <1> jc soundc\_respond\_err

2598 0001059C C3 <1> retn

2599 <1>

2600 <1> soundc\_cancel:

2601 <1> ; FUNCTION = 9

2602 <1> ; Cancel audio callback service

2603 <1> ; 22/04/2017

2604 0001059D A0[E56B0100] <1> mov al, [audio\_user]

2605 000105A2 3A05[B3030300] <1> cmp al, [u.uno]

2606 000105A8 75C1 <1> jne short snd\_nothing

2607 <1> ; RESET (audio) INTERRUPT CALLBACK SERVICE

2608 000105AA 8A1D[BF6B0100] <1> mov bl, [audio\_intr] ; IRQ number

2609 000105B0 A0[B3030300] <1> mov al, [u.uno]

2610 000105B5 28FF <1> sub bh, bh ; 0 ; unlink IRQ from user service

2611 000105B7 E8EC020000 <1> call set\_irq\_callback\_service

2612 000105BC 0F8256FDFFFF <1> jc sndc\_perm\_error ; 'permission denied' error

2613 000105C2 C3 <1> retn

2614 <1>

2615 <1> sound\_dalloc:

2616 <1> ; FUNCTION = 10

2617 <1> ; Deallocate (ring 3) audio buffer

2618 <1> ; 22/04/2017

2619 000105C3 A0[E56B0100] <1> mov al, [audio\_user]

2620 000105C8 3A05[B3030300] <1> cmp al, [u.uno]

2621 000105CE 759B <1> jne short snd\_nothing

2622 000105D0 8B1D[D06B0100] <1> mov ebx, [audio\_buffer]

2623 <1> ;or ebx, ebx

2624 <1> ;jz short snd\_nothing

2625 000105D6 8B0D[D86B0100] <1> mov ecx, [audio\_buff\_size]

2626 000105DC E89E51FFFF <1> call deallocate\_user\_pages

2627 000105E1 31C0 <1> xor eax, eax

2628 000105E3 A3[D06B0100] <1> mov [audio\_buffer], eax ; 0

2629 000105E8 A2[E56B0100] <1> mov [audio\_user], al ; 0

2630 000105ED C3 <1> retn

2631 <1>

2632 <1> sound\_volume:

2633 <1> ; FUNCTION = 11

2634 <1> ; Set sound volume level

2635 <1> ; 28/05/2017

2636 <1> ; 20/05/2017

2637 <1> ; 22/04/2017, 24/04/2017

2638 <1> ; bl = component (0 = master/playback/lineout volume)

2639 <1> ; cl = left channel volume level (0 to 31)

2640 <1> ; ch = right channel volume level (0 to 31)

2641 <1>

2642 000105EE 80FB80 <1> cmp bl, 80h

2643 000105F1 720E <1> jb short snd\_vol\_1

2644 000105F3 0F8772FFFFFF <1> ja snd\_nothing ; temporary.

2645 <1> ; Set volume level for next play (BL>= 80h)

2646 000105F9 66890D[F26B0100] <1> mov [audio\_master\_volume], cx

2647 00010600 C3 <1> retn

2648 <1> snd\_vol\_1:

2649 <1> ; set volume level immediate (BL< 80h)

2650 00010601 80FB00 <1> cmp bl, 0

2651 00010604 0F8761FFFFFF <1> ja snd\_nothing ; temporary.

2652 <1>

2653 0001060A E8F9010000 <1> call snd\_dev\_check

2654 0001060F 0F8256FFFFFF <1> jc snd\_nothing ; temporary.

2655 00010615 E8FB010000 <1> call snd\_buf\_check

2656 0001061A 0F824BFFFFFF <1> jc snd\_nothing ; temporary.

2657 <1>

2658 00010620 A0[BD6B0100] <1> mov al, [audio\_device]

2659 00010625 3C03 <1> cmp al, 3 ; VIA VT 8237R (vt8233)

2660 00010627 0F84A4180000 <1> je vt8233\_volume

2661 <1> ; 28/05/2017

2662 0001062D 0F8738FFFFFF <1> ja snd\_nothing ; temporary.

2663 <1> ;ja hda\_volume

2664 <1> ; Sound Blaster 16

2665 00010633 3C01 <1> cmp al, 1 ; SB 16

2666 00010635 0F84321B0000 <1> je sb16\_volume

2667 0001063B E9EF1D0000 <1> jmp ac97\_volume

2668 <1>

2669 <1> soundc\_disable:

2670 <1> ; FUNCTION = 12

2671 <1> ; Disable audio device (and unlink DMA memory)

2672 <1> ; 28/05/2017

2673 <1> ; 24/05/2017

2674 <1> ; 22/04/2017

2675 00010640 E8C3010000 <1> call snd\_dev\_check

2676 00010645 0F827DFBFFFF <1> jc soundc\_dev\_err ; temporary.

2677 <1> ;call snd\_buf\_check

2678 <1> ;jc sndc\_owner\_error ; temporary.

2679 <1>

2680 0001064B A0[BD6B0100] <1> mov al, [audio\_device]

2681 00010650 3C03 <1> cmp al, 3 ; VIA VT 8237R (vt8233)

2682 00010652 7418 <1> je short snd\_disable\_1

2683 00010654 0F8711FFFFFF <1> ja snd\_nothing ; temporary.

2684 0001065A 3C01 <1> cmp al, 1 ; Sound Blaster 16

2685 0001065C 7507 <1> jne short snd\_disable\_0

2686 0001065E E8851B0000 <1> call sb16\_stop

2687 00010663 EB0C <1> jmp short snd\_disable\_2

2688 <1> snd\_disable\_0:

2689 00010665 E8B31E0000 <1> call ac97\_stop

2690 0001066A EB05 <1> jmp short snd\_disable\_2

2691 <1> snd\_disable\_1:

2692 0001066C E842170000 <1> call vt8233\_stop

2693 <1> snd\_disable\_2:

2694 00010671 A0[BF6B0100] <1> mov al, [audio\_intr]

2695 00010676 29DB <1> sub ebx, ebx ; 0 = reset

2696 00010678 E85EFAFFFF <1> call set\_dev\_IRQ\_service

2697 <1>

2698 <1> ;mov al, [audio\_intr]

2699 0001067D 28E4 <1> sub ah, ah ; 0 = reset

2700 0001067F E8C0F6FFFF <1> call set\_hardware\_int\_vector

2701 <1>

2702 00010684 31C0 <1> xor eax, eax

2703 00010686 A2[BD6B0100] <1> mov byte [audio\_device], al

2704 0001068B A2[BF6B0100] <1> mov byte [audio\_intr], al

2705 00010690 8705[DC6B0100] <1> xchg eax, [audio\_dma\_buff]

2706 <1> ; 24/05/2017

2707 <1> ;or eax, eax

2708 <1> ;jz short snd\_disable\_3

2709 <1> ;cmp eax, sb16\_dma\_buffer ; default DMA buffer

2710 <1> ;je short snd\_disable\_3

2711 00010696 803D[BC6B0100]00 <1> cmp byte [audio\_pci], 0 ; AC97 audio controller ?

2712 0001069D 7612 <1> jna short snd\_disable\_3

2713 0001069F C605[BC6B0100]00 <1> mov byte [audio\_pci], 0

2714 <1> ;sub ecx, ecx

2715 <1> ;xchg ecx, [audio\_dmabuff\_size]

2716 000106A6 8B0D[E06B0100] <1> mov ecx, [audio\_dmabuff\_size]

2717 000106AC E8804FFFFF <1> call deallocate\_memory\_block

2718 <1> snd\_disable\_3:

2719 000106B1 C3 <1> retn

2720 <1>

2721 <1> sound\_dma\_map:

2722 <1> ; FUNCTION = 13

2723 <1> ; Map audio dma buff addr to user's buffer addr

2724 <1> ; 12/05/2017

2725 000106B2 21C9 <1> and ecx, ecx

2726 000106B4 0F8415FBFFFF <1> jz sound\_buff\_error

2727 000106BA 803D[BD6B0100]01 <1> cmp byte [audio\_device], 1

2728 000106C1 7229 <1> jb short snd\_dma\_map\_1

2729 <1> snd\_dma\_map\_0:

2730 000106C3 A1[DC6B0100] <1> mov eax, [audio\_dma\_buff]

2731 000106C8 21C0 <1> and eax, eax

2732 000106CA 7420 <1> jz short snd\_dma\_map\_1

2733 <1> ;

2734 000106CC 8A1D[E56B0100] <1> mov bl, [audio\_user]

2735 000106D2 08DB <1> or bl, bl

2736 000106D4 7416 <1> jz short snd\_dma\_map\_1

2737 000106D6 3A1D[B3030300] <1> cmp bl, [u.uno]

2738 000106DC 0F8531FCFFFF <1> jne sndc\_owner\_error

2739 <1> ;

2740 000106E2 8B1D[E06B0100] <1> mov ebx, [audio\_dmabuff\_size]

2741 000106E8 21DB <1> and ebx, ebx

2742 000106EA 750A <1> jnz short snd\_dma\_map\_2

2743 <1> snd\_dma\_map\_1:

2744 000106EC B8[00000200] <1> mov eax, sb16\_dma\_buffer

2745 000106F1 BB00000100 <1> mov ebx, 65536

2746 <1> snd\_dma\_map\_2:

2747 000106F6 81C1FF0F0000 <1> add ecx, PAGE\_SIZE-1 ; 4095

2748 000106FC 6681E100F0 <1> and cx, ~PAGE\_OFF ; not 4095

2749 00010701 39D9 <1> cmp ecx, ebx

2750 00010703 0F87C6FAFFFF <1> ja sound\_buff\_error

2751 00010709 50 <1> push eax

2752 0001070A 89D3 <1> mov ebx, edx

2753 0001070C C1E90C <1> shr ecx, 12 ; byte count to page count

2754 <1> ; eax = physical address of (audio) dma buffer

2755 <1> ; ebx = virtual address of (audio) dma buffer (user's pgdir)

2756 <1> ; ecx = page count (>0)

2757 0001070F E88D4FFFFF <1> call direct\_memory\_access

2758 00010714 58 <1> pop eax

2759 00010715 0F82B4FAFFFF <1> jc sound\_buff\_error

2760 0001071B A3[64030300] <1> mov [u.r0], eax

2761 00010720 C3 <1> retn

2762 <1>

2763 <1> soundc\_info:

2764 <1> ; FUNCTION = 14

2765 <1> ; Get Audio Controller Info

2766 <1> ; 10/06/2017

2767 <1> ; 05/06/2017

2768 00010721 20DB <1> and bl, bl ; 0

2769 00010723 740A <1> jz short sndc\_info\_0

2770 <1> ; invalid parameter !

2771 00010725 B817000000 <1> mov eax, ERR\_INV\_PARAMETER ; 23

2772 <1> ;sndc\_inf\_error:

2773 <1> ; mov [u.r0], eax

2774 <1> ; mov [u.error], eax

2775 <1> ; jmp error

2776 0001072A E9ACFAFFFF <1> jmp sysaudio\_err

2777 <1>

2778 <1> sndc\_info\_0:

2779 0001072F E8D4000000 <1> call snd\_dev\_check

2780 00010734 0F828EFAFFFF <1> jc soundc\_dev\_err

2781 <1>

2782 0001073A 8B1D[C86B0100] <1> mov ebx, [audio\_vendor]

2783 00010740 8B0D[C46B0100] <1> mov ecx, [audio\_dev\_id]

2784 <1> ;mov al, [audio\_device]

2785 00010746 3C02 <1> cmp al, 2 ; AC'97 (ICH)

2786 00010748 7513 <1> jne short sndc\_info\_1

2787 <1> ; Intel AC97 (ICH) Audio Controller (=2)

2788 0001074A 668B15[F66B0100] <1> mov dx, [NABMBAR]

2789 00010751 C1E210 <1> shl edx, 16

2790 00010754 668B15[F46B0100] <1> mov dx, [NAMBAR]

2791 0001075B EB07 <1> jmp short sndc\_info\_2

2792 <1> sndc\_info\_1:

2793 <1> ; 05/06/2017

2794 <1> ; Note: Intel HDA code (here) is not ready yet!

2795 <1> ; !!! SB16 or VT8233 (VT8237R) !!!

2796 0001075D 0FB715[C26B0100] <1> movzx edx, word [audio\_io\_base]

2797 <1> sndc\_info\_2:

2798 00010764 88C4 <1> mov ah, al ; [audio\_device]

2799 00010766 A0[BF6B0100] <1> mov al, [audio\_intr]

2800 <1>

2801 <1> ; EAX = IRQ Number in AL

2802 <1> ; Audio Device Number in AH

2803 <1> ; EBX = DEV/VENDOR ID

2804 <1> ; (DDDDDDDDDDDDDDDDVVVVVVVVVVVVVVVV)

2805 <1> ; ECX = BUS/DEV/FN

2806 <1> ; (00000000BBBBBBBBDDDDDFFF00000000)

2807 <1> ; EDX = NABMBAR/NAMBAR (for AC97)

2808 <1> ; (Low word, DX = NAMBAR address)

2809 <1> ; EDX = Base IO Addr (DX) for SB16 & VT8233

2810 <1>

2811 <1> ; 10/06/2017

2812 0001076B A3[64030300] <1> mov [u.r0], eax

2813 00010770 8B2D[60030300] <1> mov ebp, [u.usp]

2814 00010776 895D10 <1> mov [ebp+16], ebx ; ebx

2815 00010779 895514 <1> mov [ebp+20], edx ; edx

2816 0001077C 894D18 <1> mov [ebp+24], ecx ; ecx

2817 <1>

2818 0001077F C3 <1> retn

2819 <1>

2820 <1> sound\_data:

2821 <1> ; FUNCTION = 15

2822 <1> ; Get Current Sound data for graphics

2823 <1> ; 22/06/2017

2824 <1> ;

2825 00010780 E883000000 <1> call snd\_dev\_check

2826 00010785 0F823DFAFFFF <1> jc soundc\_dev\_err ; Device not ready !

2827 <1>

2828 0001078B 80FB00 <1> cmp bl, 0

2829 0001078E 760A <1> jna short sound\_data\_0

2830 <1>

2831 <1> ; Only PCM OUT buffer data is valid for now!

2832 00010790 B817000000 <1> mov eax, ERR\_INV\_PARAMETER ; 23

2833 00010795 E941FAFFFF <1> jmp sysaudio\_err

2834 <1>

2835 <1> sound\_data\_0:

2836 0001079A A1[DC6B0100] <1> mov eax, [audio\_dma\_buff]

2837 0001079F 09C0 <1> or eax, eax

2838 000107A1 0F8428FAFFFF <1> jz sound\_buff\_error

2839 <1>

2840 000107A7 803D[BD6B0100]04 <1> cmp byte [audio\_device], 4 ; Intel HDA

2841 000107AE 744F <1> je short sound\_data\_4 ; temporary ! (22/06/2017)

2842 <1>

2843 000107B0 21C9 <1> and ecx, ecx

2844 <1> ;jnz short sound\_data\_1 ; sample tranfer

2845 <1>

2846 <1> ; Return only DMA Buffer pointer/offset...

2847 <1> ; (If DMA Buffer has been mapped to user's

2848 <1> ; memory space; program can get graphics

2849 <1> ; data by using only this pointer value.)

2850 <1>

2851 <1> ;call get\_dma\_buffer\_offset

2852 <1> ;; eax = DMA buffer offset

2853 <1> ;; (!not half buffer offset!)

2854 <1> ;mov [u.r0], eax

2855 <1> ;retn

2856 <1>

2857 000107B2 0F845C1F0000 <1> jz get\_dma\_buffer\_offset

2858 <1>

2859 <1> sound\_data\_1:

2860 <1> ;mov eax, [audio\_dmabuff\_size]

2861 <1> ;shr eax, 1 ; half buffer size

2862 <1> ;cmp ecx, eax

2863 <1> ;ja short sound\_buff\_error

2864 <1>

2865 000107B8 3B0D[E06B0100] <1> cmp ecx, [audio\_dmabuff\_size]

2866 000107BE 0F870BFAFFFF <1> ja sound\_buff\_error

2867 <1>

2868 000107C4 89D0 <1> mov eax, edx

2869 000107C6 25FF0F0000 <1> and eax, PAGE\_OFF ; 4095 (0FFFh)

2870 000107CB 81F900100000 <1> cmp ecx, 4096

2871 000107D1 7605 <1> jna short sound\_data\_2

2872 000107D3 B900100000 <1> mov ecx, 4096 ; max. 1 page

2873 <1> sound\_data\_2:

2874 000107D8 01C8 <1> add eax, ecx

2875 000107DA 3D00100000 <1> cmp eax, 4096

2876 000107DF 7606 <1> jna short sound\_data\_3

2877 000107E1 6625FF0F <1> and ax, PAGE\_OFF ; 4095 (0FFFh)

2878 000107E5 29C1 <1> sub ecx, eax

2879 <1> ; here, ECX has been adjusted to fit

2880 <1> ; in page border.. (<= 4096, >0)

2881 <1> sound\_data\_3:

2882 000107E7 51 <1> push ecx

2883 000107E8 52 <1> push edx

2884 000107E9 89D3 <1> mov ebx, edx

2885 000107EB E89F4AFFFF <1> call get\_physical\_addr

2886 000107F0 5A <1> pop edx

2887 000107F1 59 <1> pop ecx

2888 000107F2 0F82D7F9FFFF <1> jc sound\_buff\_error

2889 <1>

2890 <1> ; eax = physical address of user's buffer

2891 000107F8 89C3 <1> mov ebx, eax

2892 <1> ; ecx = byte (transfer) count

2893 <1> ;call get\_current\_sound\_data

2894 <1> ;retn

2895 000107FA E9721E0000 <1> jmp get\_current\_sound\_data

2896 <1>

2897 <1> sound\_data\_4:

2898 <1> ; Intel HDA code is not ready yet !

2899 <1> ; 22/06/2017

2900 000107FF 31C0 <1> xor eax, eax

2901 00010801 48 <1> dec eax

2902 00010802 A3[64030300] <1> mov [u.r0], eax ; 0FFFFFFFFh

2903 00010807 C3 <1> retn

2904 <1>

2905 <1> snd\_dev\_check:

2906 <1> ; 10/06/2017

2907 <1> ; 05/06/2017

2908 <1> ; 24/05/2017

2909 <1> ; 22/04/2017

2910 <1> ; 21/04/2017

2911 <1> ; ... device check at first

2912 00010808 A0[BD6B0100] <1> mov al, [audio\_device]

2913 0001080D 3C01 <1> cmp al, 1 ; SB 16

2914 0001080F 7203 <1> jb short snd\_dev\_chk\_retn ; error !

2915 <1> ;cmp al, 4 ; Intel HDA

2916 <1> ;ja short snd\_dbchk\_stc ; invalid !

2917 <1> ; 10/06/2017

2918 00010811 3C05 <1> cmp al, 5

2919 00010813 F5 <1> cmc

2920 <1> snd\_dev\_chk\_retn:

2921 00010814 C3 <1> retn

2922 <1>

2923 <1> snd\_buf\_check:

2924 <1> ; 10/06/2017

2925 <1> ; 22/04/2017

2926 <1> ; 21/04/2017

2927 <1> ; ... buffer & (buffer) owner check at second

2928 00010815 833D[D06B0100]00 <1> cmp dword [audio\_buffer], 0

2929 0001081C 760D <1> jna short snd\_dbchk\_stc

2930 <1> snd\_user\_check:

2931 0001081E A0[B3030300] <1> mov al, [u.uno]

2932 00010823 3A05[E56B0100] <1> cmp al, [audio\_user]

2933 <1> ;jne short snd\_dbchk\_stc

2934 <1> ;retn

2935 00010829 74E9 <1> je short snd\_dev\_chk\_retn

2936 <1>

2937 <1> snd\_dbchk\_stc:

2938 0001082B F9 <1> stc

2939 0001082C C3 <1> retn

2940 <1>

2941 <1> sound\_update:

2942 <1> ; FUNCTION = 16

2943 <1> ; bl =

2944 <1> ; 0 = automatic (sequental) update (with flag switch!)

2945 <1> ; 1 = update dma half buffer 1 (without flag switch!)

2946 <1> ; 2 = update dma half buffer 2 (without flag switch!)

2947 <1> ; FFh = get current flag value

2948 <1> ; 0 = dma half buffer 1 (will be played next)

2949 <1> ; 1 = dma half buffer 2 (will be played next)

2950 <1>

2951 <1> ; 10/10/2017

2952 <1>

2953 <1> ; ... device check at first

2954 0001082D A0[BD6B0100] <1> mov al, [audio\_device]

2955 00010832 08C0 <1> or al, al ; 0 ; pc speaker or invalid

2956 00010834 0F848EF9FFFF <1> jz soundc\_dev\_err

2957 <1>

2958 <1> ; ... buffer & (buffer) owner check at second

2959 0001083A 833D[D06B0100]00 <1> cmp dword [audio\_buffer], 0

2960 00010841 0F8688F9FFFF <1> jna sound\_buff\_error

2961 00010847 A0[B3030300] <1> mov al, [u.uno]

2962 0001084C 3A05[E56B0100] <1> cmp al, [audio\_user]

2963 00010852 0F85BBFAFFFF <1> jne sndc\_owner\_error

2964 <1>

2965 <1> ; Transfer ring 3 (user's) audio buffer content to dma buffer

2966 00010858 8B3D[DC6B0100] <1> mov edi, [audio\_dma\_buff] ; dma buffer (ring 0)

2967 0001085E 09FF <1> or edi, edi

2968 00010860 0F8469F9FFFF <1> jz sound\_buff\_error

2969 00010866 8B35[D46B0100] <1> mov esi, [audio\_p\_buffer] ; physical address (ring 3)

2970 0001086C 8B0D[D86B0100] <1> mov ecx, [audio\_buff\_size]

2971 <1>

2972 <1> ;movzx eax, byte [audio\_flag]

2973 00010872 A0[E46B0100] <1> mov al, [audio\_flag]

2974 <1>

2975 00010877 FEC3 <1> inc bl

2976 00010879 7427 <1> jz short snd\_update\_3 ; bl = 0FFh

2977 0001087B FECB <1> dec bl

2978 0001087D 7411 <1> jz short snd\_update\_0 ; bl = 0

2979 <1>

2980 0001087F 80FB02 <1> cmp bl, 2

2981 00010882 7417 <1> je short snd\_update\_1 ; dma half buffer 2

2982 00010884 7217 <1> jb short snd\_update\_2 ; dma half buffer 1

2983 <1>

2984 <1> ; invalid parameter !

2985 00010886 B817000000 <1> mov eax, ERR\_INV\_PARAMETER ; 23

2986 <1> ; mov [u.r0], eax

2987 <1> ; mov [u.error], eax

2988 <1> ; jmp error

2989 0001088B E94BF9FFFF <1> jmp sysaudio\_err

2990 <1>

2991 <1> snd\_update\_0:

2992 00010890 8035[E46B0100]01 <1> xor byte [audio\_flag], 1 ; update flag !!!

2993 00010897 3C01 <1> cmp al, 1

2994 00010899 7202 <1> jb short snd\_update\_2 ; dma half buffer 1

2995 <1> snd\_update\_1:

2996 <1> ; dma half buffer 2

2997 0001089B 01CF <1> add edi, ecx

2998 <1> snd\_update\_2:

2999 <1> ;rep movsb

3000 0001089D C1E902 <1> shr ecx, 2

3001 000108A0 F3A5 <1> rep movsd

3002 <1> snd\_update\_3:

3003 000108A2 A3[64030300] <1> mov [u.r0], eax

3004 <1>

3005 000108A7 C3 <1> retn

3006 <1>

3007 <1>

3008 <1> set\_irq\_callback\_service:

3009 <1> ; 10/06/2017

3010 <1> ; 12/05/2017

3011 <1> ; 24/04/2017

3012 <1> ; 22/04/2017

3013 <1> ; caller: 'syscalbac' or 'sysaudio' or ...

3014 <1> ; 13/04/2017, 14/04/2017, 17/04/2017

3015 <1> ; 24/02/2017, 26/02/2017, 28/02/2017

3016 <1> ; 21/02/2017 - TRDOS 386 (TRDOS v2.0)

3017 <1> ;

3018 <1> ; Link or unlink IRQ callback service to/from user (ring 3)

3019 <1> ;

3020 <1> ; INPUT ->

3021 <1> ; If AL = 0, the caller is 'syscalbac';

3022 <1> ; otherwise, the caller is 'sysaudio' or ...

3023 <1> ; (AL = user number)

3024 <1> ;

3025 <1> ; BL = IRQ number (Hardware interrupt request number)

3026 <1> ; (0 t0 15 but IRQ 0,1,2,6,8,14,15 are prohibited)

3027 <1> ; IRQ numbers 3,4,5,7,9,10,11,12,13 are valid

3028 <1> ; (numbers >15 are invalid)

3029 <1> ;

3030 <1> ; BH = 0 = Unlink IRQ (in BL) from user (ring 3) service

3031 <1> ; 1 = Link IRQ by using Signal Response Byte method

3032 <1> ; 2 = Link IRQ by using Callback service method

3033 <1> ; 3 = Link IRQ by using Auto Increment S.R.B. method

3034 <1> ; >3 = invalid

3035 <1> ; (syscallback version will return to user)

3036 <1> ;

3037 <1> ; CL = Signal Return/Response Byte value

3038 <1> ;

3039 <1> ; If BH = 2, kernel will put a counter value

3040 <1> ; (into the S.R.B. addr)

3041 <1> ; between 0 to 255. (start value = CL+1)

3042 <1> ;

3043 <1> ; NOTE: counter value, for example: even and odd numbers

3044 <1> ; may be used for -audio- DMA buffer switch

3045 <1> ; within double buffer method, etc.

3046 <1> ;

3047 <1> ; EDX = Signal return (Response) byte address

3048 <1> ; - or -

3049 <1> ; Interrupt/Callback service/routine address

3050 <1> ;

3051 <1> ; (virtual address in user's memory space)

3052 <1> ;

3053 <1> ; OUTPUT ->

3054 <1> ; CF = 0 & EAX = 0 -> Successful setting

3055 <1> ; CF = 1 & EAX > 0 -> IRQ is prohibited or locked

3056 <1> ; by another process

3057 <1> ; eax = ERR\_PERM\_DENIED -> prohibited or locked

3058 <1> ; eax = ERR\_INV\_PARAMETER ->

3059 <1> ; invalid parameter/option or bad address

3060 <1> ;

3061 <1> ; TRDOS 386 - IRQ CALLBACK structures (parameters):

3062 <1> ;

3063 <1> ; [u.irqlock] = 1 word, IRQ flags (0-15) that indicates

3064 <1> ; which IRQs are locked by (that) user.

3065 <1> ; Lock and unlock (by user) will change

3066 <1> ; these flags or 'terminate process' (sysexit)

3067 <1> ; will clear these flags and unlock those IRQs.

3068 <1> ;

3069 <1> ; Bit 0 is for IRQ 0 and Bit 15 is for IRQ 15

3070 <1> ;

3071 <1> ; IRQ(x).owner : 1 byte, user, [u.uno], 0 = free (unlocked)

3072 <1> ;

3073 <1> ; IRQ(x).method : 1 byte for callback method & status

3074 <1> ; 0 = Signal Response Byte method

3075 <1> ; 1 = Callback service method

3076 <1> ; >1 = invalid for current 'syscalback'.

3077 <1> ; or(+) 80h = IRQ is in use by system (ring 0)

3078 <1> ; function (audio etc.) or

3079 <1> ; a device driver.

3080 <1> ; (system function will ignore the lock/owner)

3081 <1> ;

3082 <1> ; IRQ(x).srb : 1 byte, Signal Return/Response byte value

3083 <1> ; (a fixed value by user or a counter value

3084 <1> ; from 0 to 255, which is increased by every

3085 <1> ; interrupt just before putting it into

3086 <1> ; the Signal Response byte address

3087 <1> ; (This is not used in callback serv method)

3088 <1> ;

3089 <1> ; IRQ(x).addr : 1 dword

3090 <1> ; Signal Response Byte address (physical)

3091 <1> ; -or-

3092 <1> ; Callback service address (virtual)

3093 <1> ;

3094 <1> ; IRQ(x).dev : 1 byte

3095 <1> ; 0 = Default device or kernel function

3096 <1> ; -or-

3097 <1> ; 1-255 = Assigned device driver number

3098 <1> ;

3099 <1> ; (x) = 3,4,5,7,9,10,11,12,13

3100 <1> ;

3101 <1>

3102 000108A8 80FB0F <1> cmp bl, 15

3103 000108AB 7729 <1> ja short scbs\_2

3104 <1>

3105 000108AD 80FF03 <1> cmp bh, 3

3106 000108B0 7724 <1> ja short scbs\_2 ; invalid parameter

3107 <1>

3108 000108B2 0FB6FB <1> movzx edi, bl ; save IRQ number

3109 <1>

3110 <1> ; IRQ 0,1,2,6,8,14,15 are prohibited

3111 <1> ;IRQenum: ; 'trdosk9.s'

3112 <1> ; db 0,0,0,1,2,3,0,4,0,5,6,7,8,9,0,0

3113 <1>

3114 000108B5 0FB6B7[08160100] <1> movzx esi, byte [edi+IRQenum] ; IRQ availability

3115 <1> ; enumeration/index

3116 <1> ;dec esi

3117 000108BC 664E <1> dec si

3118 000108BE 780F <1> js short scbs\_1 ; 0 -> 0FFFFh

3119 <1>

3120 <1> ; ESI = IRQ callback parameters index number (0 to 8)

3121 <1>

3122 000108C0 08FF <1> or bh, bh

3123 000108C2 7419 <1> jz short scbs\_4 ; unlink the IRQ (in BL)

3124 <1>

3125 000108C4 FECF <1> dec bh

3126 <1> ; bh = method (0 = signal response byte, 1 = callback)

3127 <1> ; (2 = auto increment of signal response byte)

3128 <1>

3129 000108C6 80BE[6E6B0100]00 <1> cmp byte [esi+IRQ.owner], 0 ; locked ?

3130 000108CD 7637 <1> jna short scbs\_6 ; no... OK...

3131 <1>

3132 <1> scbs\_1:

3133 <1> ; permission denied (prohibited IRQ)

3134 000108CF B80B000000 <1> mov eax, ERR\_PERM\_DENIED

3135 000108D4 F9 <1> stc

3136 000108D5 C3 <1> retn

3137 <1> scbs\_2:

3138 000108D6 F9 <1> stc

3139 <1> scbs\_3:

3140 000108D7 B817000000 <1> mov eax, ERR\_INV\_PARAMETER

3141 000108DC C3 <1> retn

3142 <1>

3143 <1> scbs\_4: ; unlink the requested IRQ (if it belongs to current user)

3144 <1> ; 10/06/2017

3145 <1> ; 22/04/2017

3146 <1> ; 14/04/2017

3147 <1> ; If AL = 0 -> The caller is 'syscalbac'

3148 000108DD 8AA6[6E6B0100] <1> mov ah, [esi+IRQ.owner]

3149 000108E3 3A25[B3030300] <1> cmp ah, [u.uno]

3150 000108E9 75E4 <1> jne short scbs\_1

3151 <1>

3152 000108EB FE0D[D6030300] <1> dec byte [u.irqc] ; decrease IRQ count (in use)

3153 <1>

3154 <1> ;sub ah, ah

3155 <1> ;mov [esi+IRQ.owner], ah ; 0 ; free !!!

3156 <1> ;and byte [esi+IRQ.method], 80h

3157 <1> ;mov [esi+IRQ.srb], ah ; 0

3158 <1> ;mov [esi+IRQ.dev], ah ; 0

3159 <1> ;mov dword [esi+IRQ.addr], 0

3160 <1> ;mov dword [u.r0], 0

3161 <1>

3162 <1> ;mov byte [esi+IRQ.owner], 0

3163 <1>

3164 <1> ; 22/04/2017

3165 000108F1 29C0 <1> sub eax, eax

3166 000108F3 8886[6E6B0100] <1> mov [esi+IRQ.owner], al ; 0

3167 <1> ; 10/06/2017

3168 000108F9 8686[806B0100] <1> xchg al, [esi+IRQ.method]

3169 000108FF 2480 <1> and al, 80h

3170 00010901 745E <1> jz short scbs\_12

3171 <1> ; Audio device must be disabled -later- ! ([IRQ.medhod] = 80h)

3172 <1>

3173 <1> ; cmp byte [esi+IRQ.method], 80h ; device drv or kernel extension ?

3174 <1> ; jb short scbs\_12 ; bh = 0 reset to default IRQ handler

3175 <1> ;

3176 <1> ; and al, al

3177 <1> ; jz short scbs\_5 ; the caller is 'syscalbac'

3178 <1> ; ; The caller is 'sysaudio' or ...

3179 00010903 30C0 <1> xor al, al

3180 <1> ; mov [esi+IRQ.method], al ; 0 ; reset kernel extension flag

3181 <1> ;scbs\_5:

3182 <1> ; sub ah, ah

3183 <1> ;mov [u.r0], eax ; 0

3184 00010905 C3 <1> retn

3185 <1>

3186 <1> scbs\_6:

3187 <1> ; 14/04/2017

3188 00010906 20C0 <1> and al, al

3189 00010908 7405 <1> jz short scbs\_7 ; the caller is 'syscalbac'

3190 <1> ; AL = user number ([u.uno] or [audio.user] or ...)

3191 <1> ; The caller is 'sysaudio' or ...

3192 <1> ;

3193 <1> ; bh = method (0 = signal response byte, 1 = callback)

3194 <1> ; (2 = auto increment of signal response byte)

3195 <1>

3196 0001090A 80CF80 <1> or bh, 80h ; Kernel extension flag !

3197 0001090D EB0A <1> jmp short scbs\_8

3198 <1> scbs\_7:

3199 0001090F 8A86[806B0100] <1> mov al, [esi+IRQ.method] ; >= 80h = kernel is using this IRQ

3200 00010915 2480 <1> and al, 80h ; use only bit 7 (kernel function flag)

3201 00010917 08C7 <1> or bh, al ; method

3202 <1> ; 0 = signal response byte, 1 = callback

3203 <1> ; 2 = auto increment of s.r.b.

3204 <1> scbs\_8:

3205 00010919 A0[B3030300] <1> mov al, [u.uno] ; user (process) number (1 to 16)

3206 0001091E 8886[6E6B0100] <1> mov [esi+IRQ.owner], al ; lock the IRQ for user

3207 00010924 88BE[806B0100] <1> mov [esi+IRQ.method], bh

3208 <1>

3209 <1> ; test bh, 1

3210 <1> ; jnz short scbs\_9 ; Callback method, CX will not be used

3211 <1> ;

3212 <1> ; test bh, 2 ; use auto increment (counter) method

3213 <1> ; jz short scbs\_10 ; (count can be used for buffer switch)

3214 <1> ;scbs\_9:

3215 <1> ; xor ecx, ecx ; 0

3216 <1> scbs\_10:

3217 <1> ;mov [esi+IRQ.method], bh

3218 0001092A 888E[896B0100] <1> mov [esi+IRQ.srb], cl

3219 00010930 C686[776B0100]00 <1> mov byte [esi+IRQ.dev], 0 ; device number is always 0

3220 <1> ; for this system call

3221 <1> ;test bh, 1

3222 00010937 80E701 <1> and bh, 1 ; 17/04/2017

3223 0001093A 7513 <1> jnz short scbs\_11 ; callback method, use virtual address

3224 <1>

3225 0001093C 53 <1> push ebx ; IRQ number (in BL)

3226 0001093D 89D3 <1> mov ebx, edx

3227 <1> ; ebx = virtual address

3228 <1> ; [u.pgdir] = page directory's physical address

3229 0001093F FE05[0E6B0100] <1> inc byte [no\_page\_swap] ; 1

3230 <1> ; Do not add this page to swap queue

3231 <1> ; and remove it from swap queue if it is

3232 <1> ; on the queue.

3233 00010945 E84549FFFF <1> call get\_physical\_addr

3234 0001094A 5B <1> pop ebx

3235 0001094B 728A <1> jc scbs\_3 ; invalid address !

3236 <1> ; eax = physical address of the virtual address in user's space

3237 0001094D 89C2 <1> mov edx, eax

3238 <1> scbs\_11:

3239 0001094F 66C1E602 <1> shl si, 2 ; byte (index) to dword (offset)

3240 00010953 8996[926B0100] <1> mov [esi+IRQ.addr], edx

3241 <1>

3242 00010959 FE05[D6030300] <1> inc byte [u.irqc] ; increase IRQ (in use) count

3243 <1>

3244 0001095F FEC7 <1> inc bh ; 17/04/2017

3245 <1> ; bh > 0 -> set to requested IRQ handler (IRQ\_u\_list)

3246 <1> scbs\_12:

3247 00010961 88D8 <1> mov al, bl ; IRQ number

3248 00010963 88FC <1> mov ah, bh ; 0 = reset, >0 = set

3249 00010965 E8DAF3FFFF <1> call set\_hardware\_int\_vector

3250 <1>

3251 0001096A 31C0 <1> xor eax, eax

3252 <1> ;mov [u.r0], eax ; 0

3253 <1>

3254 0001096C C3 <1> retn ; return with success (cf=0, eax=0)

3255 <1>

3256 <1>

3257 <1> sysdma: ; DMA FUNCTIONS

3258 <1> ; 02/09/2017

3259 <1> ; 28/08/2017

3260 <1> ; 20/08/2017 - TRDOS 386 (TRDOS v2.0)

3261 <1> ;

3262 <1> ; Inputs:

3263 <1> ; BH = 0 -> Allocate DMA buffer

3264 <1> ; BL = 0 -> Use the system's default DMA

3265 <1> ; (SB16) Buffer

3266 <1> ; Buffer Size (max.) = 65536 bytes

3267 <1> ; BL > 0 -> Allocate (a new) DMA buffer

3268 <1> ; ECX = DMA Buffer Size in bytes (<=128KB)

3269 <1> ; EDX = Virtual Address of DMA buffer

3270 <1> ;

3271 <1> ; BH = 1 -> Initialize (Start) DMA service

3272 <1> ; BL, bit 0 to 3 = Channel Number (0 to 7)

3273 <1> ; BL, bit 7 = Auto Initialized Mode

3274 <1> ; (If bit 7 is set)

3275 <1> ; bit 6 = Record (read) mode (0= playback)

3276 <1> ; ECX = byte count (0 = use dma buffer size)

3277 <1> ; EDX = physical buffer address

3278 <1> ; (0 = use dma buffer -start- address)

3279 <1> ;

3280 <1> ; BH = 2 -> Get Current DMA Buffer Offset

3281 <1> ; BL = DMA channel number

3282 <1> ;

3283 <1> ; BH = 3 -> Get Current DMA count down value

3284 <1> ; BL = DMA channel number (0 tO 7)

3285 <1> ;

3286 <1> ; BH = 4 -> Get Current DMA channel (in progress)

3287 <1> ;

3288 <1> ; BH = 5 -> Get System's Default DMA Buffer Address

3289 <1> ;

3290 <1> ; BH = 6 -> Get Current DMA Buffer Address

3291 <1> ;

3292 <1> ; BH = 7 -> Stop DMA service

3293 <1> ;

3294 <1> ;

3295 <1> ; Outputs:

3296 <1> ;

3297 <1> ; For BH = 0 ; Allocate DMA buffer

3298 <1> ; EAX = Physical address of DMA buffer

3299 <1> ; ECX = Allocated buffer size in bytes

3300 <1> ; - page count \* 4096 -

3301 <1> ; (may be bigger than requested)

3302 <1> ; If BL input > 0,

3303 <1> ; 'sysalloc:' system call will be used with

3304 <1> ; EBX (for 'sysalloc') = EDX (for 'sysdma')

3305 <1> ; ECX is same, byte count (buffer size)

3306 <1> ; EDX = 1024\*1024\*16 ; 16 MB upper limit

3307 <1> ; If BL input = 0,

3308 <1> ; Default DMA buffer (SB16 buffer) will be

3309 <1> ; checked and if it is free, it's address

3310 <1> ; will be returned in EAX and it's size

3311 <1> ; will be returned in ECX (as 65536)

3312 <1> ;

3313 <1> ; If CF = 1, error code is in EAX

3314 <1> ; EAX = -1 ; DMA buffer allocation error!

3315 <1> ; EAX = 11 ; 'Permission Denied' error !

3316 <1> ;

3317 <1> ; Note: 'sysalloc' error return method

3318 <1> ; will be applied if BL input > 0 !

3319 <1> ;

3320 <1> ; For BH = 1 ; Initialize (Start) DMA

3321 <1> ; EAX = 0 (Successful)

3322 <1> ; If CF = 1, error code is in EAX

3323 <1> ;

3324 <1> ; For BH = 2 ; Get Current DMA Buffer Offset

3325 <1> ; EAX = DMA Buffer Offset (in bytes)

3326 <1> ; ;

3327 <1> ; AX = DMA buffer offset

3328 <1> ; EAX bits 16 to 23 = Page register value

3329 <1> ;

3330 <1> ; For BH = 3 ; Get Current DMA count down value

3331 <1> ; EAX = Count down value (remain bytes)

3332 <1> ;

3333 <1> ; For BH = 4 ; Get Current DMA channel (in progress)

3334 <1> ; EAX = DMA channel number (0 to 7)

3335 <1> ; AH = 0 if the owner is the caller process

3336 <1> ; AH > 0 if the dma channel is in use by

3337 <1> ; another user/process

3338 <1> ; EAX = -1 (0FFFFFFFFh)

3339 <1> ; if DMA service is not in use

3340 <1> ; (stopped or not initialized/started)

3341 <1> ;

3342 <1> ; For BH = 5 ; Get System's Default DMA Buff Addr

3343 <1> ; EAX = Default DMA Buffer Address (Physical)

3344 <1> ; = offset 'sb16\_dma\_buffer:'

3345 <1> ; ECX = Buffer size

3346 <1> ; = 65536

3347 <1> ;

3348 <1> ; For BH = 6 ; Get Current DMA Buffer Address

3349 <1> ; EAX = Current DMA buffer address (Physical)

3350 <1> ; ECX = Current DMA buffer size (setting value)

3351 <1> ; Note: These values are for current dma channel

3352 <1> ; settings for the user/process

3353 <1> ; \*\* For now (for current TRDOS 386 version)

3354 <1> ; only one user/process can use only one

3355 <1> ; dma channel & one dma buffer at same time

3356 <1> ; (no multi tasking on DMA service) !!! \*\*

3357 <1> ; (Once, current DMA user must stop it's own DMA

3358 <1> ; DMA service, than another user/program

3359 <1> ; can use DMA service with same dma channel

3360 <1> ; or with another DMA channel.)

3361 <1> ;

3362 <1> ; For BH = 7 ; Stop DMA service (for current user

3363 <1> ; and current DMA channel)

3364 <1> ; EAX = 0 ; successful

3365 <1> ; CF = 1 & EAX > 0 (= -1) -> Error

3366 <1>

3367 0001096D 80FF07 <1> cmp bh, 7

3368 00010970 7612 <1> jna short sysdma\_0

3369 <1>

3370 <1> sysdma\_err:

3371 00010972 31C0 <1> xor eax, eax

3372 00010974 48 <1> dec eax ; -1

3373 <1> sysdma\_perm\_err:

3374 00010975 A3[64030300] <1> mov [u.r0], eax

3375 0001097A A3[C8030300] <1> mov [u.error], eax ; DMA service error !

3376 0001097F E93ABDFFFF <1> jmp error

3377 <1>

3378 <1> sysdma\_0:

3379 00010984 08FF <1> or bh, bh

3380 00010986 0F85BA000000 <1> jnz sysdma\_1

3381 <1>

3382 0001098C 20DB <1> and bl, bl

3383 0001098E 7416 <1> jz short sysdma\_01

3384 <1>

3385 <1> ; redirect system call to 'sysalloc'

3386 00010990 89D3 <1> mov ebx, edx ; virtual address of DMA buffer

3387 <1> ;ecx = Buffer size in bytes

3388 <1> ; DMA buffer address <= 16MB upper limit

3389 00010992 BA00000001 <1> mov edx, 1024\*1024\*16 ; 16MB limit for DMA buff

3390 <1>

3391 00010997 C705[00700100]FFFF- <1> mov dword [dma\_addr], 0FFFFFFFFh ; -1

3391 0001099F FFFF <1>

3392 <1>

3393 000109A1 E9A8E5FFFF <1> jmp sysalloc

3394 <1>

3395 <1> sysdma\_01:

3396 000109A6 B8[00000200] <1> mov eax, sb16\_dma\_buffer

3397 <1>

3398 000109AB 803D[BD6B0100]01 <1> cmp byte [audio\_device], 1

3399 000109B2 722A <1> jb short sysdma\_03

3400 <1>

3401 000109B4 3B05[DC6B0100] <1> cmp eax, [audio\_dma\_buff]

3402 000109BA 7507 <1> jne short sysdma\_02

3403 <1>

3404 <1> sysdma\_0\_err:

3405 000109BC B80B000000 <1> mov eax, ERR\_PERM\_DENIED

3406 000109C1 EBB2 <1> jmp short sysdma\_perm\_err

3407 <1>

3408 <1> sysdma\_02:

3409 <1> ; Only one user is permitted for audio/dma functions

3410 <1>

3411 000109C3 833D[DC6B0100]00 <1> cmp dword [audio\_dma\_buff], 0

3412 000109CA 7612 <1> jna short sysdma\_03

3413 <1>

3414 000109CC 8A1D[E56B0100] <1> mov bl, [audio\_user]

3415 000109D2 08DB <1> or bl, bl

3416 000109D4 7408 <1> jz short sysdma\_03

3417 <1>

3418 000109D6 3A1D[B3030300] <1> cmp bl, [u.uno]

3419 000109DC 75DE <1> jne short sysdma\_0\_err

3420 <1>

3421 <1> sysdma\_03:

3422 000109DE 8A1D[FD6F0100] <1> mov bl, [dma\_user]

3423 000109E4 20DB <1> and bl, bl

3424 000109E6 750E <1> jnz short sysdma\_04

3425 <1>

3426 000109E8 8A1D[B3030300] <1> mov bl, [u.uno]

3427 000109EE 881D[FD6F0100] <1> mov [dma\_user], bl

3428 <1>

3429 000109F4 EB15 <1> jmp short sysdma\_05

3430 <1>

3431 <1> sysdma\_04:

3432 000109F6 8B35[00700100] <1> mov esi, [dma\_addr]

3433 000109FC 21F6 <1> and esi, esi

3434 000109FE 740B <1> jz short sysdma\_05

3435 <1>

3436 00010A00 46 <1> inc esi ; -1 -> 0

3437 00010A01 7408 <1> jz short sysdma\_05

3438 <1>

3439 00010A03 3A1D[B3030300] <1> cmp bl, [u.uno]

3440 00010A09 75B1 <1> jne short sysdma\_0\_err

3441 <1>

3442 <1> sysdma\_05:

3443 <1> ; edx = virtual address (user's buffer address)

3444 <1> ;

3445 00010A0B 81F900000100 <1> cmp ecx, 65536 ; byte count (buffer size)

3446 00010A11 0F875BFFFFFF <1> ja sysdma\_err

3447 <1> ;

3448 00010A17 81C1FF0F0000 <1> add ecx, PAGE\_SIZE-1 ; 4095

3449 00010A1D 6681E100F0 <1> and cx, ~PAGE\_OFF ; not 4095

3450 <1> ;cmp ecx, 65536

3451 <1> ;ja sysdma\_err ;

3452 00010A22 51 <1> push ecx ; buffer size (allocated pages \* 4096)

3453 00010A23 50 <1> push eax ; offset sb16\_dma\_buffer

3454 00010A24 89D3 <1> mov ebx, edx

3455 00010A26 C1E90C <1> shr ecx, 12 ; byte count to page count

3456 <1> ; eax = physical address of (audio) dma buffer

3457 <1> ; ebx = virtual address of (audio) dma buffer (user's pgdir)

3458 <1> ; ecx = page count (>0)

3459 00010A29 E8734CFFFF <1> call direct\_memory\_access

3460 00010A2E 58 <1> pop eax

3461 00010A2F 59 <1> pop ecx

3462 00010A30 0F823CFFFFFF <1> jc sysdma\_err

3463 <1>

3464 00010A36 A3[00700100] <1> mov [dma\_addr], eax

3465 00010A3B 890D[04700100] <1> mov [dma\_size], ecx ; dma buffer size (in bytes)

3466 <1>

3467 <1> ;mov [u.r0], eax ; DMA Buffer Address (Physical)

3468 <1>

3469 <1> ;mov ebp, [u.usp] ; ebp points to user's registers

3470 <1> ;mov [ebp+24], ecx ; return to user with ecx value

3471 <1>

3472 <1> ;jmp sysret

3473 <1>

3474 <1> ; 28/08/2017

3475 00010A41 E9C4000000 <1> jmp sysdma\_51

3476 <1>

3477 <1> sysdma\_1:

3478 00010A46 80FF01 <1> cmp bh, 1

3479 00010A49 0F87A6000000 <1> ja sysdma\_5

3480 <1>

3481 00010A4F F6C340 <1> test bl, 40h ; record (read) mode -BL, bit 6-

3482 00010A52 0F851AFFFFFF <1> jnz sysdma\_err ; not ready yet!

3483 <1>

3484 00010A58 A1[00700100] <1> mov eax, [dma\_addr] ; physical address of dma buffer

3485 00010A5D 21C0 <1> and eax, eax

3486 00010A5F 0F840DFFFFFF <1> jz sysdma\_err

3487 <1>

3488 00010A65 09D2 <1> or edx, edx

3489 00010A67 7504 <1> jnz short sysdma\_11

3490 <1>

3491 00010A69 89C2 <1> mov edx, eax

3492 00010A6B EB08 <1> jmp short sysdma\_12

3493 <1> sysdma\_11:

3494 00010A6D 39C2 <1> cmp edx, eax

3495 00010A6F 0F82FDFEFFFF <1> jb sysdma\_err

3496 <1> sysdma\_12:

3497 00010A75 21C9 <1> and ecx, ecx

3498 00010A77 7508 <1> jnz short sysdma\_13

3499 <1>

3500 00010A79 8B0D[04700100] <1> mov ecx, [dma\_size]

3501 00010A7F EB0C <1> jmp short sysdma\_14

3502 <1> sysdma\_13:

3503 00010A81 3B0D[04700100] <1> cmp ecx, [dma\_size]

3504 00010A87 0F87E5FEFFFF <1> ja sysdma\_err

3505 <1> sysdma\_14:

3506 00010A8D 89C6 <1> mov esi, eax

3507 00010A8F 0335[04700100] <1> add esi, [dma\_size]

3508 <1>

3509 00010A95 89D0 <1> mov eax, edx

3510 00010A97 01C8 <1> add eax, ecx

3511 00010A99 0F82D3FEFFFF <1> jc sysdma\_err ; 02/09/2017

3512 <1>

3513 00010A9F 39F0 <1> cmp eax, esi

3514 00010AA1 0F87CBFEFFFF <1> ja sysdma\_err

3515 <1>

3516 00010AA7 8B3D[DC6B0100] <1> mov edi, [audio\_dma\_buff]

3517 00010AAD 8B35[00700100] <1> mov esi, [dma\_addr]

3518 <1>

3519 00010AB3 09FF <1> or edi, edi

3520 00010AB5 7424 <1> jz short sysdma\_16

3521 <1>

3522 00010AB7 803D[BD6B0100]01 <1> cmp byte [audio\_device], 1

3523 00010ABE 7208 <1> jb short sysdma\_15

3524 <1>

3525 <1> ; Sound Blaster 16

3526 00010AC0 39FE <1> cmp esi, edi

3527 00010AC2 0F84F4FEFFFF <1> je sysdma\_0\_err ; permmission denied !

3528 <1>

3529 <1> sysdma\_15:

3530 00010AC8 C605[FF6F0100]48 <1> mov byte [dma\_mode], 48h ; single mode playback

3531 <1>

3532 00010ACF F6C380 <1> test bl, 80h ; DMA mode - BL, bit 7, auto init -

3533 00010AD2 7407 <1> jz short sysdma\_16

3534 <1> ; Auto initialized playback (write) mode

3535 00010AD4 8005[FF6F0100]10 <1> add byte [dma\_mode], 10h ; = 58h

3536 <1> sysdma\_16:

3537 00010ADB 80E307 <1> and bl, 07h

3538 00010ADE 881D[FE6F0100] <1> mov [dma\_channel], bl

3539 00010AE4 8915[08700100] <1> mov [dma\_start], edx

3540 00010AEA 890D[0C700100] <1> mov [dma\_count], ecx

3541 <1>

3542 <1> ; 28/08/2017

3543 <1> ;call dma\_init

3544 <1> ;jmp sysret

3545 00010AF0 E94B010000 <1> jmp dma\_init

3546 <1>

3547 <1> sysdma\_5:

3548 00010AF5 80FF05 <1> cmp bh, 5

3549 00010AF8 7223 <1> jb short sysdma\_3

3550 00010AFA 0F87CE000000 <1> ja sysdma\_6

3551 <1>

3552 <1> ; Get the system's default dma buffer addr and size

3553 00010B00 B8[00000200] <1> mov eax, sb16\_dma\_buffer

3554 00010B05 B900000100 <1> mov ecx, 65536 ; Buffer size in bytes

3555 <1>

3556 <1> sysdma\_51:

3557 <1> ; 0 = there is not a dma buffer (in use or available)

3558 00010B0A A3[64030300] <1> mov [u.r0], eax

3559 <1>

3560 00010B0F 8B2D[60030300] <1> mov ebp, [u.usp] ; ebp points to user's registers

3561 00010B15 894D18 <1> mov [ebp+24], ecx ; return to user with ecx value

3562 <1>

3563 00010B18 E9C1BBFFFF <1> jmp sysret

3564 <1>

3565 <1> sysdma\_3:

3566 00010B1D 80FF03 <1> cmp bh, 3

3567 00010B20 7231 <1> jb short sysdma\_2

3568 00010B22 776B <1> ja short sysdma\_4

3569 <1>

3570 <1> ; Get current dma count down value (remain bytes)

3571 <1> ; 28/08/2017

3572 00010B24 0FB635[FE6F0100] <1> movzx esi, byte [dma\_channel]

3573 00010B2B 0FB696[40160100] <1> movzx edx, byte [dma\_flip+esi]

3574 00010B32 EE <1> out dx, al ; flip-flop clear

3575 00010B33 8A96[20160100] <1> mov dl, [dma\_cnt+esi] ; dma count register addr

3576 00010B39 EC <1> in al, dx

3577 00010B3A 0FB6D8 <1> movzx ebx, al

3578 00010B3D EC <1> in al, dx

3579 00010B3E 88C7 <1> mov bh, al

3580 <1>

3581 00010B40 6683FE04 <1> cmp si, 4 ; channel number ?

3582 00010B44 7202 <1> jb short sysdma\_31 ; 8 bit dma channel

3583 <1>

3584 00010B46 D1E3 <1> shl ebx, 1 ; word count to byte count

3585 <1>

3586 <1> sysdma\_31:

3587 00010B48 891D[64030300] <1> mov [u.r0], ebx

3588 <1>

3589 00010B4E E98BBBFFFF <1> jmp sysret

3590 <1>

3591 <1> sysdma\_2:

3592 <1> ; Get current dma buffer offset (& page)

3593 <1> ; 28/08/2017

3594 00010B53 0FB635[FE6F0100] <1> movzx esi, byte [dma\_channel]

3595 00010B5A 0FB696[40160100] <1> movzx edx, byte [dma\_flip+esi]

3596 00010B61 EE <1> out dx, al ; flip-flop clear

3597 00010B62 8A96[18160100] <1> mov dl, [dma\_adr+esi]

3598 00010B68 EC <1> in al, dx ; get dma position

3599 00010B69 0FB6D8 <1> movzx ebx, al

3600 00010B6C EC <1> in al, dx

3601 00010B6D 88C7 <1> mov bh, al

3602 <1>

3603 00010B6F 6683FE04 <1> cmp si, 4 ; channel number ?

3604 00010B73 7202 <1> jb short sysdma\_21 ; 8 bit dma channel

3605 <1>

3606 00010B75 D1E3 <1> shl ebx, 1 ; word offset to byte offset

3607 <1>

3608 <1> sysdma\_21:

3609 00010B77 891D[64030300] <1> mov [u.r0], ebx

3610 <1>

3611 00010B7D 8A96[28160100] <1> mov dl, [dma\_page+esi]

3612 00010B83 EC <1> in al, dx ; get dma page

3613 <1>

3614 <1> ;add [u.ro+2], al

3615 00010B84 0805[66030300] <1> or [u.r0+2], al

3616 <1>

3617 00010B8A E94FBBFFFF <1> jmp sysret

3618 <1>

3619 <1> sysdma\_4:

3620 <1> ; Get current DMA channel number

3621 <1> ; 28/08/2017

3622 00010B8F 8A25[FD6F0100] <1> mov ah, [dma\_user]

3623 00010B95 20E4 <1> and ah, ah

3624 00010B97 750F <1> jnz short sysdma\_42

3625 <1>

3626 <1> sysdma\_41:

3627 <1> ; Not a valid dma channel (in use)

3628 00010B99 C705[64030300]FFFF- <1> mov dword [u.r0], -1 ; 0FFFFFFFFh

3628 00010BA1 FFFF <1>

3629 00010BA3 E936BBFFFF <1> jmp sysret

3630 <1>

3631 <1> sysdma\_42:

3632 00010BA8 8B35[00700100] <1> mov esi, [dma\_addr]

3633 00010BAE 21F6 <1> and esi, esi

3634 00010BB0 74E7 <1> jz short sysdma\_41

3635 <1>

3636 00010BB2 46 <1> inc esi ; -1 -> 0

3637 00010BB3 74E4 <1> jz short sysdma\_41

3638 <1>

3639 00010BB5 A0[FE6F0100] <1> mov al, [dma\_channel]

3640 <1>

3641 00010BBA 3A25[B3030300] <1> cmp ah, [u.uno]

3642 00010BC0 7502 <1> jne short sysdma\_43

3643 <1>

3644 00010BC2 30E4 <1> xor ah, ah ; DMA channel in use by current user

3645 <1>

3646 <1> sysdma\_43:

3647 00010BC4 A3[64030300] <1> mov [u.r0], eax ; AL = dma channel number

3648 <1> ; AH > 0 if the the channel

3649 <1> ; in use by another user/process

3650 00010BC9 E910BBFFFF <1> jmp sysret

3651 <1>

3652 <1> sysdma\_6:

3653 00010BCE 80FF06 <1> cmp bh, 6

3654 00010BD1 7710 <1> ja short sysdma\_7

3655 <1>

3656 <1> ; 28/08/2017

3657 <1> ; Get current DMA buffer addr and size

3658 00010BD3 A1[00700100] <1> mov eax, [dma\_addr] ; dma buffer address

3659 00010BD8 8B0D[04700100] <1> mov ecx, [dma\_size] ; dma buffer size (in bytes)

3660 <1>

3661 00010BDE E927FFFFFF <1> jmp sysdma\_51

3662 <1>

3663 <1> sysdma\_7:

3664 <1> ; DMA service STOP

3665 00010BE3 A0[B3030300] <1> mov al, [u.uno]

3666 00010BE8 3A05[FD6F0100] <1> cmp al, [dma\_user]

3667 00010BEE 751D <1> jne short sysdma\_72

3668 <1>

3669 00010BF0 28C0 <1> sub al, al ; 0

3670 <1>

3671 00010BF2 A2[FD6F0100] <1> mov [dma\_user], al ; clear user

3672 <1>

3673 00010BF7 8605[FF6F0100] <1> xchg al, [dma\_mode]

3674 00010BFD 20C0 <1> and al, al

3675 <1> ;jz short sysdma\_err

3676 00010BFF 7527 <1> jnz short sysdma\_73

3677 <1>

3678 <1> sysdma\_71:

3679 00010C01 31C0 <1> xor eax, eax

3680 00010C03 A3[64030300] <1> mov [u.r0], eax; 0

3681 00010C08 E9D1BAFFFF <1> jmp sysret

3682 <1>

3683 <1> sysdma\_72:

3684 <1> ; 28/08/2017

3685 00010C0D 803D[FD6F0100]00 <1> cmp byte [dma\_user], 0

3686 00010C14 76EB <1> jna short sysdma\_71 ; Nothing to do !

3687 <1>

3688 00010C16 833D[00700100]00 <1> cmp dword [dma\_addr], 0

3689 00010C1D 0F8799FDFFFF <1> ja sysdma\_0\_err

3690 <1>

3691 00010C23 A2[FD6F0100] <1> mov [dma\_user], al ; reset to current user

3692 <1>

3693 <1> sysdma\_73:

3694 <1> ; 28/08/2017

3695 00010C28 0FB635[FE6F0100] <1> movzx esi, byte [dma\_channel]

3696 00010C2F 0FB696[30160100] <1> movzx edx, byte [dma\_mask+esi]

3697 00010C36 A0[FE6F0100] <1> mov al, [dma\_channel]

3698 00010C3B 0C04 <1> or al, 4

3699 00010C3D EE <1> out dx, al

3700 <1>

3701 00010C3E EBC1 <1> jmp short sysdma\_71

3702 <1>

3703 <1> dma\_init:

3704 <1> ; 28/08/2017

3705 <1> ; 20/08/2017

3706 <1> ; DMA initialization

3707 <1> ; 14/08/2017

3708 <1> ; 03/08/2017, 06/08/2017, 08/08/2017

3709 <1> ; 02/07/2017, 13/07/2017, 16/07/2017, 30/07/2017

3710 <1> ; (Derived from 'DMA\_INIT' procedure in SB16MOD.ASM)

3711 <1> ; Modified for TRDOS 386 DMA buffer allocation & initialization !

3712 <1>

3713 00010C40 8B1D[08700100] <1> mov ebx, [dma\_start]

3714 00010C46 8B0D[0C700100] <1> mov ecx, [dma\_count]

3715 <1>

3716 00010C4C 0FB635[FE6F0100] <1> movzx esi, byte [dma\_channel]

3717 <1>

3718 00010C53 6683FE04 <1> cmp si, 4

3719 00010C57 7205 <1> jb short gdmi1

3720 <1> ; 08/08/2017

3721 00010C59 66D1E9 <1> shr cx, 1 ; word count

3722 00010C5C D1EB <1> shr ebx, 1 ; convert byte offset to word offset

3723 <1> gdmi1:

3724 <1> ;mov [dma\_poff], bx ; 08/08/2017

3725 00010C5E 6649 <1> dec cx ; dma size = block size - 1

3726 <1>

3727 00010C60 0FB696[30160100] <1> movzx edx, byte [dma\_mask+esi] ; 30/07/2017

3728 00010C67 A0[FE6F0100] <1> mov al, [dma\_channel]

3729 00010C6C 0C04 <1> or al, 4

3730 00010C6E EE <1> out dx, al ; dma channel mask

3731 <1>

3732 00010C6F 30C0 <1> xor al, al ; 0 ; any value ! 08/08/2017

3733 00010C71 8A96[40160100] <1> mov dl, [dma\_flip+esi]

3734 00010C77 EE <1> out dx, al ; flip-flop clear

3735 <1>

3736 00010C78 8A96[38160100] <1> mov dl, [dma\_mod+esi]

3737 00010C7E A0[FE6F0100] <1> mov al, [dma\_channel] ; 13/07/2017

3738 00010C83 2403 <1> and al, 3

3739 <1> ; 08/08/2017

3740 00010C85 0A05[FF6F0100] <1> or al, [dma\_mode] ; 58h ; dma mode for SB16

3741 00010C8B EE <1> out dx, al

3742 <1>

3743 00010C8C 8A96[18160100] <1> mov dl, [dma\_adr+esi]

3744 00010C92 88D8 <1> mov al, bl

3745 00010C94 EE <1> out dx, al ; offset low

3746 <1>

3747 00010C95 88F8 <1> mov al, bh

3748 00010C97 EE <1> out dx, al ; offset high

3749 <1>

3750 00010C98 8A96[20160100] <1> mov dl, [dma\_cnt+esi]

3751 00010C9E 88C8 <1> mov al, cl

3752 00010CA0 EE <1> out dx, al ; size low

3753 <1>

3754 00010CA1 88E8 <1> mov al, ch

3755 00010CA3 EE <1> out dx, al ; size high

3756 <1>

3757 00010CA4 8A96[28160100] <1> mov dl, [dma\_page+esi]

3758 <1> ; 14/08/2017

3759 00010CAA 6683FE04 <1> cmp si, 4

3760 00010CAE 7305 <1> jnb short gdmi2

3761 00010CB0 C1EB10 <1> shr ebx, 16

3762 00010CB3 EB06 <1> jmp short gdmi3

3763 <1> gdmi2:

3764 <1> ; 09/08/2017

3765 00010CB5 C1EB0F <1> shr ebx, 15 ; complete 16 bit shift

3766 00010CB8 80E3FE <1> and bl, 0FEh ; clear bit 0 (not necessary)

3767 <1> gdmi3:

3768 00010CBB 88D8 <1> mov al, bl

3769 00010CBD EE <1> out dx, al ; page

3770 <1>

3771 00010CBE 8A96[30160100] <1> mov dl, [dma\_mask+esi]

3772 00010CC4 A0[FE6F0100] <1> mov al, [dma\_channel] ; 13/07/2017

3773 00010CC9 2403 <1> and al, 3

3774 00010CCB EE <1> out dx, al ; dma channel unmask

3775 <1>

3776 <1> ;retn

3777 <1> ; 28/08/2017

3778 00010CCC E90DBAFFFF <1> jmp sysret

3779 <1>

3780 <1> otty:

3781 <1> sret:

3782 <1> ocvt:

3783 <1> ctty:

3784 <1> cret:

3785 <1> ccvt:

3786 <1> rtty:

3787 <1> wtty:

3788 <1> rmem:

3789 <1> wmem:

3790 <1> rfd:

3791 <1> rhd:

3792 <1> wfd:

3793 <1> whd:

3794 <1> rlpt:

3795 <1> wlpt:

3796 <1> rcvt:

3797 <1> xmtt:

3798 00010CD1 C3 <1> retn

2313 %include 'trdosk9.s' ; 04/01/2016

1 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2 <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.0) - INITIALIZED DATA : trdosk9.s

3 <1> ; ----------------------------------------------------------------------------

4 <1> ; Last Update: 31/12/2017

5 <1> ; ----------------------------------------------------------------------------

6 <1> ; Beginning: 04/01/2016

7 <1> ; ----------------------------------------------------------------------------

8 <1> ; ----------------------------------------------------------------------------

9 <1> ; Assembler: NASM version 2.11 (trdos386.s)

10 <1> ; ----------------------------------------------------------------------------

11 <1> ; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan

12 <1> ; TRDOS2.ASM (09/11/2011)

13 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

14 <1> ; DRV\_INIT.ASM [26/09/2009] Last Update: 07/08/2011

15 <1> ; MAINPROG.ASM [17/01/2004] Last Update: 09/11/2011

16 <1> ; CMD\_INTR.ASM [29/01/2005] Last Update: 09/11/2011

17 <1> ; FILE.ASM [29/10/2009] Last Update: 09/10/2011

18 <1>

19 <1> ; 12/02/2016

20 <1> Last\_DOS\_DiskNo:

21 00010CD2 01 <1> db 1 ; A: = 0 & B: = 1

22 <1>

23 <1> Restore\_CDIR:

24 00010CD3 FF <1> db 0FFh ; Initial value -> any number except 0

25 <1>

26 <1> msg\_CRLF\_temp:

27 00010CD4 070D0A00 <1> db 07h, 0Dh, 0Ah, 0

28 <1>

29 <1> Magic\_Bytes:

30 00010CD8 04 <1> db 4

31 00010CD9 01 <1> db 1

32 <1> mainprog\_Version:

33 00010CDA 07 <1> db 7

34 00010CDB 5B5452444F535D204D- <1> db "[TRDOS] Main Program v2.0.311217"

34 00010CE4 61696E2050726F6772- <1>

34 00010CED 616D2076322E302E33- <1>

34 00010CF6 3131323137 <1>

35 00010CFB 0D0A <1> db 0Dh, 0Ah

36 00010CFD 286329204572646F67- <1> db "(c) Erdogan Tan 2005-2017"

36 00010D06 616E2054616E203230- <1>

36 00010D0F 30352D32303137 <1>

37 00010D16 0D0A00 <1> db 0Dh, 0Ah, 0

38 <1>

39 <1> MainProgCfgFile: ; 14/04/2016

40 00010D19 4D41494E50524F472E- <1> db "MAINPROG.CFG", 0

40 00010D22 43464700 <1>

41 <1>

42 <1> TRDOSPromptLabel:

43 00010D26 5452444F53 <1> db "TRDOS"

44 00010D2B 00 <1> db 0

45 00010D2C 00<rept> <1> times 5 db 0

46 00010D31 00 <1> db 0

47 <1>

48 <1> ; INTERNAL COMMANDS

49 <1> Command\_List:

50 00010D32 44495200 <1> Cmd\_Dir: db "DIR", 0

51 00010D36 434400 <1> Cmd\_Cd: db "CD", 0

52 00010D39 433A00 <1> Cmd\_Drive: db "C:", 0

53 00010D3C 56455200 <1> Cmd\_Ver: db "VER", 0

54 00010D40 4558495400 <1> Cmd\_Exit: db "EXIT", 0

55 00010D45 50524F4D505400 <1> Cmd\_Prompt: db "PROMPT", 0

56 00010D4C 564F4C554D4500 <1> Cmd\_Volume: db "VOLUME", 0

57 00010D53 4C4F4E474E414D4500 <1> Cmd\_LongName: db "LONGNAME", 0

58 00010D5C 4441544500 <1> Cmd\_Date: db "DATE", 0

59 00010D61 54494D4500 <1> Cmd\_Time: db "TIME", 0

60 00010D66 52554E00 <1> Cmd\_Run: db "RUN", 0

61 00010D6A 53455400 <1> Cmd\_Set: db "SET", 0

62 00010D6E 434C5300 <1> Cmd\_Cls: db "CLS", 0

63 00010D72 53484F5700 <1> Cmd\_Show: db "SHOW", 0

64 00010D77 44454C00 <1> Cmd\_Del: db "DEL", 0

65 00010D7B 41545452494200 <1> Cmd\_Attrib: db "ATTRIB", 0

66 00010D82 52454E414D4500 <1> Cmd\_Rename: db "RENAME", 0

67 00010D89 524D44495200 <1> Cmd\_Rmdir: db "RMDIR", 0

68 00010D8F 4D4B44495200 <1> Cmd\_Mkdir: db "MKDIR", 0

69 00010D95 434F505900 <1> Cmd\_Copy: db "COPY", 0

70 00010D9A 4D4F564500 <1> Cmd\_Move: db "MOVE", 0

71 00010D9F 5041544800 <1> Cmd\_Path: db "PATH", 0

72 00010DA4 4D454D00 <1> Cmd\_Mem: db "MEM", 0

73 00010DA8 00 <1> db 0

74 00010DA9 46494E4400 <1> Cmd\_Find: db "FIND", 0

75 00010DAE 4543484F00 <1> Cmd\_Echo: db "ECHO", 0

76 00010DB3 2A00 <1> Cmd\_Remark: db "\*", 0

77 00010DB5 3F00 <1> Cmd\_Help: db "?", 0

78 00010DB7 44455649434500 <1> Cmd\_Device: db "DEVICE", 0

79 00010DBE 4445564C49535400 <1> Cmd\_DevList: db "DEVLIST", 0

80 00010DC6 434844495200 <1> Cmd\_Chdir: db "CHDIR", 0

81 00010DCC 4245455000 <1> Cmd\_Beep: db "BEEP", 0

82 <1>

83 00010DD1 00 <1> db 0

84 <1>

85 <1> ; 15/02/2016 (FILE.ASM, 09/10/2011)

86 <1> invalid\_fname\_chars:

87 00010DD2 222728292A2B2C2F <1> db 22h, 27h, 28h, 29h, 2Ah, 2Bh, 2Ch, 2Fh

88 00010DDA 3A3B3C3D3E3F40 <1> db 3Ah, 3Bh, 3Ch, 3Dh, 3Eh, 3Fh, 40h

89 00010DE1 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 00010DE6 456E746572206E6577- <1> db 'Enter new date (dd-mm-yy): '

94 00010DEF 206461746520286464- <1>

94 00010DF8 2D6D6D2D7979293A20 <1>

95 00010E01 00 <1> db 0

96 <1> Msg\_Show\_Date:

97 00010E02 43757272656E742064- <1> db 'Current date is '

97 00010E0B 61746520697320 <1>

98 00010E12 30 <1> Day: db '0'

99 00010E13 30 <1> db '0'

100 00010E14 2F <1> db '/'

101 00010E15 30 <1> Month: db '0'

102 00010E16 30 <1> db '0'

103 00010E17 2F <1> db '/'

104 00010E18 30 <1> Century: db '0'

105 00010E19 30 <1> db '0'

106 00010E1A 30 <1> Year: db '0'

107 00010E1B 30 <1> db '0'

108 00010E1C 0D0A00 <1> db 0Dh, 0Ah, 0

109 <1>

110 <1> Msg\_Enter\_Time:

111 00010E1F 456E746572206E6577- <1> db 'Enter new time: '

111 00010E28 2074696D653A20 <1>

112 00010E2F 00 <1> db 0

113 <1> Msg\_Show\_Time:

114 00010E30 43757272656E742074- <1> db 'Current time is '

114 00010E39 696D6520697320 <1>

115 00010E40 30 <1> Hour: db '0'

116 00010E41 30 <1> db '0'

117 00010E42 3A <1> db ':'

118 00010E43 30 <1> Minute: db '0'

119 00010E44 30 <1> db '0'

120 00010E45 3A <1> db ':'

121 00010E46 30 <1> Second: db '0'

122 00010E47 30 <1> db '0'

123 00010E48 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 00010E4B 206B696C6F62797465- <1> db " kilobytes", 0Dh, 0Ah, 0

129 00010E54 730D0A00 <1>

130 <1> VolSize\_Bytes:

131 00010E58 2062797465730D0A00 <1> db " bytes", 0Dh, 0Ah, 0

132 <1> Volume\_in\_drive:

133 00010E61 0D0A <1> db 0Dh, 0Ah

134 <1> Vol\_FS\_Name:

135 00010E63 54522046533120 <1> db "TR FS1 "

136 00010E6A 566F6C756D6520696E- <1> db "Volume in drive "

136 00010E73 20647269766520 <1>

137 00010E7A 30 <1> Vol\_Drv\_Name: db 30h

138 00010E7B 3A <1> db ":"

139 00010E7C 20697320 <1> db " is "

140 00010E80 0D0A00 <1> db 0Dh, 0Ah, 0

141 <1> Dir\_Drive\_Str:

142 00010E83 54522D444F53204472- <1> db "TR-DOS Drive "

142 00010E8C 69766520 <1>

143 <1> Dir\_Drive\_Name:

144 00010E90 303A <1> db "0:"

145 00010E92 0D0A <1> db 0Dh, 0Ah

146 <1> Vol\_Str\_Header:

147 00010E94 566F6C756D65204E61- <1> db "Volume Name: "

147 00010E9D 6D653A20 <1>

148 <1> Vol\_Name:

149 00010EA1 00<rept> <1> times 64 db 0

150 00010EE1 00 <1> db 0

151 <1> Vol\_Serial\_Header:

152 00010EE2 0D0A <1> db 0Dh, 0Ah

153 00010EE4 566F6C756D65205365- <1> db "Volume Serial No: "

153 00010EED 7269616C204E6F3A20 <1>

154 <1> Vol\_Serial1:

155 00010EF6 30303030 <1> db "0000"

156 00010EFA 2D <1> db "-"

157 <1> Vol\_Serial2:

158 00010EFB 30303030 <1> db "0000"

159 00010EFF 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 00010F02 0D0A <1> db 0Dh, 0Ah

165 00010F04 566F6C756D65205369- <1> db "Volume Size : ", 0

165 00010F0D 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 00010F13 467265652053706163- <1> db "Free Space : ", 0

173 00010F1C 6520203A2000 <1>

174 <1> ;Vol\_Free\_Sectors\_Str:

175 <1> ; db "0000000000"

176 <1> ;Vol\_Free\_Sectors\_Str\_End:

177 <1> ; db 0

178 <1>

179 <1> Dir\_Str\_Header:

180 00010F22 4469726563746F7279- <1> db "Directory: "

180 00010F2B 3A20 <1>

181 00010F2D 2F <1> Dir\_Str\_Root: db "/"

182 00010F2E 00<rept> <1> Dir\_Str: times 64 db 0

183 00010F6E 00000000 <1> dd 0

184 00010F72 00 <1> db 0

185 <1>

186 <1> Msg\_Bad\_Command:

187 00010F73 42616420636F6D6D61- <1> db "Bad command or file name!"

187 00010F7C 6E64206F722066696C- <1>

187 00010F85 65206E616D6521 <1>

188 00010F8C 0D0A00 <1> db 0Dh, 0Ah, 0

189 <1>

190 <1> msgl\_drv\_not\_ready:

191 00010F8F 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 00010F92 4472697665206E6F74- <1> db "Drive not ready or read error!"

196 00010F9B 207265616479206F72- <1>

196 00010FA4 207265616420657272- <1>

196 00010FAD 6F7221 <1>

197 00010FB0 0D0A00 <1> db 0Dh, 0Ah, 0

198 <1>

199 <1> Msg\_Not\_Ready\_Write\_Err:

200 00010FB3 4472697665206E6F74- <1> db "Drive not ready or write error!"

200 00010FBC 207265616479206F72- <1>

200 00010FC5 207772697465206572- <1>

200 00010FCE 726F7221 <1>

201 00010FD2 0D0A00 <1> db 0Dh, 0Ah, 0

202 <1>

203 <1> Msg\_Dir\_Not\_Found:

204 00010FD5 4469726563746F7279- <1> db "Directory not found!"

204 00010FDE 206E6F7420666F756E- <1>

204 00010FE7 6421 <1>

205 00010FE9 0D0A00 <1> db 0Dh, 0Ah, 0

206 <1>

207 <1> Msg\_File\_Not\_Found:

208 00010FEC 46696C65206E6F7420- <1> db "File not found!"

208 00010FF5 666F756E6421 <1>

209 00010FFB 0D0A00 <1> db 0Dh, 0Ah, 0

210 <1>

211 <1> Msg\_File\_Directory\_Not\_Found:

212 00010FFE 46696C65206F722064- <1> db "File or directory not found!"

212 00011007 69726563746F727920- <1>

212 00011010 6E6F7420666F756E64- <1>

212 00011019 21 <1>

213 0001101A 0D0A00 <1> db 0Dh, 0Ah, 0

214 <1>

215 <1> Msg\_LongName\_Not\_Found:

216 0001101D 4C6F6E67206E616D65- <1> db "Long name not found!"

216 00011026 206E6F7420666F756E- <1>

216 0001102F 6421 <1>

217 00011031 0D0A00 <1> db 0Dh, 0Ah, 0

218 <1>

219 <1> beep\_Insufficient\_Memory: ; 20/02/2017

220 00011034 0D0A <1> db 0Dh, 0Ah

221 00011036 07 <1> db 07h

222 <1> Msg\_Insufficient\_Memory:

223 00011037 496E73756666696369- <1> db "Insufficient memory!"

223 00011040 656E74206D656D6F72- <1>

223 00011049 7921 <1>

224 0001104B 0D0A00 <1> db 0Dh, 0Ah, 0

225 <1>

226 <1> Msg\_Error\_Code:

227 0001104E 436F6D6D616E642066- <1> db 'Command failed! Error code : '

227 00011057 61696C656421204572- <1>

227 00011060 726F7220636F646520- <1>

227 00011069 3A20 <1>

228 0001106B 303068 <1> error\_code\_hex: db '00h'

229 0001106E 0A0A00 <1> db 0Ah, 0Ah, 0

230 <1>

231 00011071 90 <1> align 2

232 <1>

233 <1> ; 10/02/2016

234 <1> ; DIR.ASM - 09/10/2011

235 <1>

236 00011072 3C4449523E20202020- <1> Type\_Dir: db '<DIR> ' ; 10 bytes

236 0001107B 20 <1>

237 <1>

238 <1> File\_Name:

239 0001107C 20<rept> <1> times 12 db 20h

240 00011088 20 <1> db 20h

241 <1> Dir\_Or\_FileSize:

242 00011089 20<rept> <1> times 10 db 20h

243 00011093 20 <1> db 20h

244 <1> File\_Attribute:

245 00011094 20202020 <1> dd 20202020h

246 00011098 20 <1> db 20h

247 <1> File\_Day:

248 00011099 3030 <1> db '0','0'

249 0001109B 2F <1> db '/'

250 <1> File\_Month:

251 0001109C 3030 <1> db '0','0'

252 0001109E 2F <1> db '/'

253 <1> File\_Year:

254 0001109F 30303030 <1> db '0','0','0','0'

255 000110A3 20 <1> db 20h

256 <1> File\_Hour:

257 000110A4 3030 <1> db '0','0'

258 000110A6 3A <1> db ':'

259 <1> File\_Minute:

260 000110A7 3030 <1> db '0','0'

261 000110A9 00 <1> db 0

262 <1>

263 <1> Decimal\_File\_Count\_Header:

264 000110AA 0D0A <1> db 0Dh, 0Ah

265 <1> Decimal\_File\_Count:

266 000110AC 00<rept> <1> times 6 db 0

267 <1>

268 000110B2 2066696C6528732920- <1> str\_files: db " file(s) & "

268 000110BB 2620 <1>

269 <1> Decimal\_Dir\_Count:

270 000110BD 00<rept> <1> times 6 db 0

271 <1> str\_dirs:

272 000110C3 206469726563746F72- <1> db " directory(s) "

272 000110CC 7928732920 <1>

273 000110D1 0D0A00 <1> db 0Dh, 0Ah, 0

274 <1>

275 000110D4 206279746528732920- <1> str\_bytes: db " byte(s) in file(s)"

275 000110DD 696E2066696C652873- <1>

275 000110E6 29 <1>

276 000110E7 0D0A00 <1> db 0Dh, 0Ah, 0

277 <1>

278 <1> ; CMD\_INTR.ASM - 09/11/2011

279 <1> ; 07/10/2010

280 <1> Msg\_invalid\_name\_chars:

281 000110EA 496E76616C69642066- <1> db "Invalid file or directory name characters!"

281 000110F3 696C65206F72206469- <1>

281 000110FC 726563746F7279206E- <1>

281 00011105 616D65206368617261- <1>

281 0001110E 637465727321 <1>

282 00011114 0D0A00 <1> db 0Dh, 0Ah, 0

283 <1> ; 21/02/2016

284 00011117 46696C65206F722064- <1> Msg\_Name\_Exists: db "File or directory name exists!"

284 00011120 69726563746F727920- <1>

284 00011129 6E616D652065786973- <1>

284 00011132 747321 <1>

285 00011135 0D0A00 <1> db 0Dh, 0Ah, 0

286 <1> Msg\_DoYouWantMkdir:

287 00011138 446F20796F75207761- <1> db "Do you want to make directory ", 0

287 00011141 6E7420746F206D616B- <1>

287 0001114A 65206469726563746F- <1>

287 00011153 72792000 <1>

288 00011157 2028592F4E29203F20- <1> Msg\_YesNo: db " (Y/N) ? ", 0

288 00011160 00 <1>

289 00011161 000D0A00 <1> Y\_N\_nextline: db 0, 0Dh, 0Ah, 0

290 00011165 4F4B2E0D0A00 <1> Msg\_OK: db "OK.", 0Dh, 0Ah, 0

291 <1>

292 <1> ; 27/02/2016

293 <1> Msg\_DoYouWantRmDir:

294 0001116B 446F20796F75207761- <1> db "Do you want to delete directory ", 0

294 00011174 6E7420746F2064656C- <1>

294 0001117D 657465206469726563- <1>

294 00011186 746F72792000 <1>

295 <1> Msg\_Dir\_Not\_Empty:

296 0001118C 4469726563746F7279- <1> db "Directory not empty!"

296 00011195 206E6F7420656D7074- <1>

296 0001119E 7921 <1>

297 000111A0 0D0A00 <1> db 0Dh, 0Ah, 0

298 <1>

299 <1> Msg\_DoYouWantDelete:

300 000111A3 446F20796F75207761- <1> db "Do you want to delete file ",0

300 000111AC 6E7420746F2064656C- <1>

300 000111B5 6574652066696C6520- <1>

300 000111BE 00 <1>

301 <1>

302 000111BF 44656C657465642E2E- <1> Msg\_Deleted: db "Deleted...", 0Dh, 0Ah, 0

302 000111C8 2E0D0A00 <1>

303 <1>

304 <1> Msg\_Permission\_Denied:

305 000111CC 07 <1> db 7

306 000111CD 5065726D697373696F- <1> db "Permission denied!", 0Dh, 0Ah, 0

306 000111D6 6E2064656E69656421- <1>

306 000111DF 0D0A00 <1>

307 <1>

308 <1> ; 04/03/2016

309 000111E2 4E657720 <1> Msg\_New: db "New "

310 000111E6 00 <1> db 0

311 <1> Str\_Attributes:

312 000111E7 417474726962757465- <1> db "Attributes : "

312 000111F0 73203A20 <1>

313 000111F4 4E4F524D414C <1> Attr\_Chars: db "NORMAL"

314 000111FA 00 <1> db 0

315 <1>

316 <1> ; 06/03/2016

317 <1> ; CMD\_INTR.ASM - 16/11/2010

318 <1> Msg\_DoYouWantRename:

319 000111FB 446F20796F75207761- <1> db "Do you want to rename ", 0

319 00011204 6E7420746F2072656E- <1>

319 0001120D 616D652000 <1>

320 00011212 66696C652000 <1> Rename\_File: db "file ", 0

321 00011218 6469726563746F7279- <1> Rename\_Directory: db "directory ", 0

321 00011221 2000 <1>

322 00011223 00<rept> <1> Rename\_OldName: times 13 db 0

323 00011230 20617320 <1> Msg\_File\_rename\_as: db " as "

324 00011234 00<rept> <1> Rename\_NewName: times 13 db 0

325 <1>

326 <1> ; 08/03/2016

327 <1> ; CMD\_INTR.ASM - 01/08/2010 - 23/04/2011

328 <1> msg\_not\_same\_drv:

329 00011241 4E6F742073616D6520- <1> db "Not same drive!"

329 0001124A 647269766521 <1>

330 00011250 0D0A00 <1> db 0Dh, 0Ah, 0

331 <1>

332 <1> Msg\_DoYouWantMoveFile:

333 00011253 446F20796F75207761- <1> db "Do you want to move file", 0

333 0001125C 6E7420746F206D6F76- <1>

333 00011265 652066696C6500 <1>

334 <1>

335 <1> msg\_insufficient\_disk\_space:

336 0001126C 496E73756666696369- <1> db "Insufficient disk space!"

336 00011275 656E74206469736B20- <1>

336 0001127E 737061636521 <1>

337 00011284 0D0A00 <1> db 0Dh, 0Ah, 0

338 <1>

339 <1> ; 01/08/2010

340 <1> msg\_source\_file:

341 00011287 0D0A536F7572636520- <1> db 0Dh, 0Ah, "Source file name : "

341 00011290 66696C65206E616D65- <1>

341 00011299 2020202020203A2020- <1>

341 000112A2 20 <1>

342 <1> msg\_source\_file\_drv:

343 000112A3 203A00 <1> db " :", 0

344 <1> msg\_destination\_file:

345 000112A6 0D0A44657374696E61- <1> db 0Dh, 0Ah, "Destination file name : "

345 000112AF 74696F6E2066696C65- <1>

345 000112B8 206E616D65203A2020- <1>

345 000112C1 20 <1>

346 <1> msg\_destination\_file\_drv:

347 000112C2 203A00 <1> db " :", 0

348 <1> msg\_copy\_nextline:

349 000112C5 0D0A00 <1> db 0Dh, 0Ah, 0

350 <1>

351 <1> ; 15/03/2016

352 <1> ; CMD\_INTR.ASM

353 <1>

354 <1> Msg\_DoYouWantOverWriteFile:

355 000112C8 446F20796F75207761- <1> db "Do you want to overwrite file ",0

355 000112D1 6E7420746F206F7665- <1>

355 000112DA 727772697465206669- <1>

355 000112E3 6C652000 <1>

356 <1>

357 <1> Msg\_DoYouWantCopyFile:

358 000112E7 446F20796F75207761- <1> db "Do you want to copy file",0

358 000112F0 6E7420746F20636F70- <1>

358 000112F9 792066696C6500 <1>

359 <1>

360 <1> Msg\_read\_file\_error\_before\_EOF:

361 00011300 46696C652072656164- <1> db "File reading error! (before EOF)"

361 00011309 696E67206572726F72- <1>

361 00011312 2120286265666F7265- <1>

361 0001131B 20454F4629 <1>

362 00011320 0A0A00 <1> db 0Ah, 0Ah, 0

363 <1>

364 <1> ; 18/03/2016

365 <1> ; TRDOS 386 (v2.0) mainprog copy procedure

366 <1> msg\_reading:

367 00011323 52656164696E672E2E- <1> db "Reading... ", 0

367 0001132C 2E2000 <1>

368 <1> msg\_writing:

369 0001132F 57726974696E672E2E- <1> db "Writing... ", 0

369 00011338 2E2000 <1>

370 <1> percentagestr:

371 0001133B 2020202500 <1> db " %", 0 ; " 0%" .. "100%"

372 <1> ; 11/04/2016

373 <1> Msg\_No\_Set\_Space:

374 00011340 496E73756666696369- <1> db "Insufficient environment space!"

374 00011349 656E7420656E766972- <1>

374 00011352 6F6E6D656E74207370- <1>

374 0001135B 61636521 <1>

375 0001135F 0D0A00 <1> db 0Dh, 0Ah, 0

376 <1> ; 18/04/2016

377 <1> isc\_msg:

378 00011362 0D0A <1> db 0Dh, 0Ah

379 00011364 494E56414C49442053- <1> db "INVALID SYSTEM CALL", 0

379 0001136D 595354454D2043414C- <1>

379 00011376 4C00 <1>

380 <1> usi\_msg:

381 00011378 0D0A <1> db 0Dh, 0Ah

382 0001137A 554E444546494E4544- <1> db "UNDEFINED SOFTWARE INTERRUPT", 0

382 00011383 20534F465457415245- <1>

382 0001138C 20494E544552525550- <1>

382 00011395 5400 <1>

383 <1> ifc\_msg:

384 00011397 0D0A <1> db 0Dh, 0Ah

385 00011399 494E56414C49442046- <1> db "INVALID FUNCTION CALL"

385 000113A2 554E4354494F4E2043- <1>

385 000113AB 414C4C <1>

386 <1> inv\_msg\_for\_trdos\_v2:

387 000113AE 20 <1> db 20h

388 000113AF 666F72205452444F53- <1> db "for TRDOS v2!"

388 000113B8 20763221 <1>

389 000113BC 07 <1> db 07h

390 000113BD 0D0A <1> db 0Dh, 0Ah

391 000113BF 0D0A <1> db 0Dh, 0Ah

392 000113C1 494E5420 <1> db "INT "

393 000113C5 303068 <1> int\_num\_str: db "00h"

394 000113C8 0D0A <1> db 0Dh, 0Ah

395 000113CA 454158203A20 <1> db "EAX : "

396 000113D0 303030303030303068- <1> eax\_str: db "00000000h", 0Dh, 0Ah

396 000113D9 0D0A <1>

397 000113DB 454950203A20 <1> db "EIP : "

398 000113E1 303030303030303068- <1> eip\_str: db "00000000h", 0Dh, 0Ah, 0

398 000113EA 0D0A00 <1>

399 <1>

400 <1> ; 07/10/2016

401 <1> ; Device names & parameters (for kernel devices)

402 <1>

403 000113ED 90 <1> align 2

404 <1> KDEV\_NAME:

405 000113EE 5454590000000000 <1> db 'TTY',0,0,0,0,0 ; 1

406 000113F6 4D454D0000000000 <1> db 'MEM',0,0,0,0,0 ; 2

407 000113FE 4644300000000000 <1> db 'FD0',0,0,0,0,0 ; 3

408 00011406 4644310000000000 <1> db 'FD1',0,0,0,0,0 ; 4

409 0001140E 4844300000000000 <1> db 'HD0',0,0,0,0,0 ; 5

410 00011416 4844310000000000 <1> db 'HD1',0,0,0,0,0 ; 6

411 0001141E 4844320000000000 <1> db 'HD2',0,0,0,0,0 ; 7

412 00011426 4844330000000000 <1> db 'HD3',0,0,0,0,0 ; 8

413 0001142E 4C50540000000000 <1> db 'LPT',0,0,0,0,0 ; 9

414 00011436 5454593000000000 <1> db 'TTY0',0,0,0,0 ; 10

415 0001143E 5454593100000000 <1> db 'TTY1',0,0,0,0 ; 11

416 00011446 5454593200000000 <1> db 'TTY2',0,0,0,0 ; 12

417 0001144E 5454593300000000 <1> db 'TTY3',0,0,0,0 ; 13

418 00011456 5454593400000000 <1> db 'TTY4',0,0,0,0 ; 14

419 0001145E 5454593500000000 <1> db 'TTY5',0,0,0,0 ; 15

420 00011466 5454593600000000 <1> db 'TTY6',0,0,0,0 ; 16

421 0001146E 5454593700000000 <1> db 'TTY7',0,0,0,0 ; 17

422 00011476 5454593800000000 <1> db 'TTY8',0,0,0,0 ; 18

423 0001147E 5454593900000000 <1> db 'TTY9',0,0,0,0 ; 19

424 00011486 434F4D3100000000 <1> db 'COM1',0,0,0,0 ; 18

425 0001148E 434F4D3200000000 <1> db 'COM2',0,0,0,0 ; 19

426 <1> ;db 'CONSOLE',0 ; 1

427 <1> ;db 'PRINTER',0 ; 9

428 <1> ;db 'CDROM' ; 20

429 <1> ;db 'CDROM0' ; 20

430 <1> ;db 'CDROM1' ; 21

431 <1> ;db 'DVD' ; 22

432 <1> ;db 'DVD0' ; 22

433 <1> ;db 'DVD1' ; 23

434 <1> ;db 'USB' ; 24

435 <1> ;db 'USB0' ; 24

436 <1> ;db 'USB1' ; 25

437 <1> ;db 'USB2' ; 26

438 <1> ;db 'USB3' ; 27

439 <1> ;db 'KEYBOARD' ; 1

440 <1> ;db 'MOUSE' ; 28

441 <1> ;db 'SOUND' ; 29

442 <1> ;db 'VGA',0,0,0,0 ; 30

443 <1> ;db 'CGA',0,0,0,0 ; 31

444 <1> ;db 'AUDIO',0,0,0 ; 29

445 <1> ;db 'VIDEO',0,0,0 ; 32

446 <1> ;db 'MUSIC',0,0,0 ; 33

447 <1> ;db 'ETHERNET' ; 34

448 <1> ;db 'SD0',0,0,0,0,0 ; 35

449 <1> ;db 'SD1',0,0,0,0,0 ; 36

450 <1> ;db 'SD2',0,0,0,0,0 ; 37

451 <1> ;db 'SD3',0,0,0,0,0 ; 38

452 <1> ;db 'SATA0' ; 35

453 <1> ;db 'SATA1' ; 36

454 <1> ;db 'SATA2' ; 37

455 <1> ;db 'SATA3' ; 38

456 <1> ;db 'PATA0',0,0,0 ; 5

457 <1> ;db 'PATA1',0,0,0 ; 6

458 <1> ;db 'PATA2',0,0,0 ; 7

459 <1> ;db 'PATA3',0,0,0 ; 8

460 <1> ;db 'WIRELESS' ; 39

461 <1> ;db 'HDMI',0,0,0,0 ; 40

462 00011496 4E554C4C00000000 <1> db 'NULL',0,0,0,0 ; 0

463 <1>

464 <1> NumOfKernelDevNames equ ($-KDEV\_NAME) / 8 ; 20 (07/10/2016)

465 <1>

466 <1> KDEV\_NUMBER:

467 0001149E 010203040506070809 <1> db 1,2,3,4,5,6,7,8,9

468 000114A7 0A0B0C0D0E0F101112- <1> db 10,11,12,13,14,15,16,17,18,19

468 000114B0 13 <1>

469 000114B1 121300 <1> db 18,19,0

470 <1>

471 <1> NumOfKernelDevices equ $ - KDEV\_NUMBER

472 <1>

473 <1> KDEV\_OADDR:

474 000114B4 [D10C0100] <1> dd otty ;tty ; 1

475 000114B8 [D10C0100] <1> dd sret ;mem ; 2

476 000114BC [D10C0100] <1> dd sret ;fd0 ; 3

477 000114C0 [D10C0100] <1> dd sret ;fd1 ; 4

478 000114C4 [D10C0100] <1> dd sret ;hd0 ; 5

479 000114C8 [D10C0100] <1> dd sret ;hd1 ; 6

480 000114CC [D10C0100] <1> dd sret ;hd2 ; 7

481 000114D0 [D10C0100] <1> dd sret ;hd3 ; 8

482 000114D4 [D10C0100] <1> dd sret ;lpt ; 9

483 000114D8 [D10C0100] <1> dd ocvt ;tty0 ; 10

484 000114DC [D10C0100] <1> dd ocvt ;tty1 ; 11

485 000114E0 [D10C0100] <1> dd ocvt ;tty2 ; 12

486 000114E4 [D10C0100] <1> dd ocvt ;tty3 ; 13

487 000114E8 [D10C0100] <1> dd ocvt ;tty4 ; 14

488 000114EC [D10C0100] <1> dd ocvt ;tty5 ; 15

489 000114F0 [D10C0100] <1> dd ocvt ;tty6 ; 16

490 000114F4 [D10C0100] <1> dd ocvt ;tty7 ; 17

491 000114F8 [D10C0100] <1> dd ocvt ;tty8 ; 18

492 000114FC [D10C0100] <1> dd ocvt ;tty9 ; 19

493 <1> ;dd ocvt ;com1 ; 18

494 <1> ;dd ocvt ;com2 ; 19

495 00011500 [D10C0100] <1> dd sret ;null ; 20

496 <1> KDEV\_CADDR:

497 00011504 [D10C0100] <1> dd ctty ;tty ; 1

498 00011508 [D10C0100] <1> dd cret ;mem ; 2

499 0001150C [D10C0100] <1> dd cret ;fd0 ; 3

500 00011510 [D10C0100] <1> dd cret ;fd1 ; 4

501 00011514 [D10C0100] <1> dd cret ;hd0 ; 5

502 00011518 [D10C0100] <1> dd cret ;hd1 ; 6

503 0001151C [D10C0100] <1> dd cret ;hd2 ; 7

504 00011520 [D10C0100] <1> dd cret ;hd3 ; 8

505 00011524 [D10C0100] <1> dd cret ;lpt ; 9

506 00011528 [D10C0100] <1> dd ocvt ;tty0 ; 10

507 0001152C [D10C0100] <1> dd ccvt ;tty1 ; 11

508 00011530 [D10C0100] <1> dd ccvt ;tty2 ; 12

509 00011534 [D10C0100] <1> dd ccvt ;tty3 ; 13

510 00011538 [D10C0100] <1> dd ccvt ;tty4 ; 14

511 0001153C [D10C0100] <1> dd ccvt ;tty5 ; 15

512 00011540 [D10C0100] <1> dd ccvt ;tty6 ; 16

513 00011544 [D10C0100] <1> dd ccvt ;tty7 ; 17

514 00011548 [D10C0100] <1> dd ccvt ;tty8 ; 18

515 0001154C [D10C0100] <1> dd ccvt ;tty9 ; 19

516 <1> ;dd ccvt ;com1 ; 18

517 <1> ;dd ccvt ;com2 ; 19

518 00011550 [D10C0100] <1> dd cret ;null ; 20

519 <1>

520 <1> KDEV\_RADDR:

521 00011554 [D10C0100] <1> dd rtty ;tty ; 1

522 00011558 [D10C0100] <1> dd rmem ;mem ; 2

523 0001155C [D10C0100] <1> dd rfd ;fd0 ; 3

524 00011560 [D10C0100] <1> dd rfd ;fd1 ; 4

525 00011564 [D10C0100] <1> dd rhd ;hd0 ; 5

526 00011568 [D10C0100] <1> dd rhd ;hd1 ; 6

527 0001156C [D10C0100] <1> dd rhd ;hd2 ; 7

528 00011570 [D10C0100] <1> dd rhd ;hd3 ; 8

529 00011574 [D10C0100] <1> dd rlpt ;lpt ; 9

530 00011578 [D10C0100] <1> dd rcvt ;tty0 ; 10

531 0001157C [D10C0100] <1> dd rcvt ;tty1 ; 11

532 00011580 [D10C0100] <1> dd rcvt ;tty2 ; 12

533 00011584 [D10C0100] <1> dd rcvt ;tty3 ; 13

534 00011588 [D10C0100] <1> dd rcvt ;tty4 ; 14

535 0001158C [D10C0100] <1> dd rcvt ;tty5 ; 15

536 00011590 [D10C0100] <1> dd rcvt ;tty6 ; 16

537 00011594 [D10C0100] <1> dd rcvt ;tty7 ; 17

538 00011598 [D10C0100] <1> dd rcvt ;tty8 ; 18

539 0001159C [D10C0100] <1> dd rcvt ;tty9 ; 19

540 <1> ;dd rcvt ;com1 ; 18

541 <1> ;dd rcvt ;com2 ; 19

542 000115A0 [C2000100] <1> dd rnull ;null ; 20

543 <1> KDEV\_WADDR:

544 000115A4 [D10C0100] <1> dd wtty ;tty ; 1

545 000115A8 [D10C0100] <1> dd wmem ;mem ; 2

546 000115AC [D10C0100] <1> dd wfd ;fd0 ; 3

547 000115B0 [D10C0100] <1> dd wfd ;fd1 ; 4

548 000115B4 [D10C0100] <1> dd whd ;hd0 ; 5

549 000115B8 [D10C0100] <1> dd whd ;hd1 ; 6

550 000115BC [D10C0100] <1> dd whd ;hd2 ; 7

551 000115C0 [D10C0100] <1> dd whd ;hd3 ; 8

552 000115C4 [D10C0100] <1> dd wlpt ;lpt ; 9

553 000115C8 [D10C0100] <1> dd xmtt ;tty0 ; 10

554 000115CC [D10C0100] <1> dd xmtt ;tty1 ; 11

555 000115D0 [D10C0100] <1> dd xmtt ;tty2 ; 12

556 000115D4 [D10C0100] <1> dd xmtt ;tty3 ; 13

557 000115D8 [D10C0100] <1> dd xmtt ;tty4 ; 14

558 000115DC [D10C0100] <1> dd xmtt ;tty5 ; 15

559 000115E0 [D10C0100] <1> dd xmtt ;tty6 ; 16

560 000115E4 [D10C0100] <1> dd xmtt ;tty7 ; 17

561 000115E8 [D10C0100] <1> dd xmtt ;tty8 ; 18

562 000115EC [D10C0100] <1> dd xmtt ;tty9 ; 19

563 <1> ;dd xmtt ;com1 ; 18

564 <1> ;dd xmtt ;com2 ; 19

565 000115F0 [C3000100] <1> dd wnull ;null ; 20

566 <1>

567 <1> ; DEV\_ACCESS bits:

568 <1> ; bit 0 = accessable by normal users

569 <1> ; bit 1 = read access permission

570 <1> ; bit 2 = write access permission

571 <1> ; bit 3 = IOCTL permission to users

572 <1> ; bit 4 = block device if it is set

573 <1> ; bit 5 = 16 bit or 1024 byte data

574 <1> ; bit 6 = 32 bit or 2048 byte data

575 <1> ; bit 7 = installable device driver

576 <1>

577 <1> KDEV\_ACCESS: ; 08/10/2016

578 000115F4 07 <1> db 00000111b ; tty, 1

579 000115F5 07 <1> db 00000111b ; mem, 2

580 000115F6 8F <1> db 10001111b ; fd0, 3

581 000115F7 8F <1> db 10001111b ; fd1, 4

582 000115F8 8F <1> db 10001111b ; hd0, 5

583 000115F9 8F <1> db 10001111b ; hd1, 6

584 000115FA 8F <1> db 10001111b ; hd2, 7

585 000115FB 8F <1> db 10001111b ; hd3, 8

586 000115FC 07 <1> db 00000111b ; lpt, 9

587 000115FD 07 <1> db 00000111b ; tty0, 10

588 000115FE 07 <1> db 00000111b ; tty1, 11

589 000115FF 07 <1> db 00000111b ; tty2, 12

590 00011600 07 <1> db 00000111b ; tty3, 13

591 00011601 07 <1> db 00000111b ; tty4, 14

592 00011602 07 <1> db 00000111b ; tty5, 15

593 00011603 07 <1> db 00000111b ; tty6, 16

594 00011604 07 <1> db 00000111b ; tty7, 17

595 00011605 07 <1> db 00000111b ; tty8, 18

596 00011606 07 <1> db 00000111b ; tty9, 19

597 <1> ;db 00000111b ; com1, 18

598 <1> ;db 00000111b ; com2, 19

599 00011607 00 <1> db 00000000b ; null, 0

600 <1>

601 <1> ; 07/10/2016

602 <1> NumOfInstallableDevices equ 8

603 <1> NUMIDEV equ NumOfInstallableDevices ; 8

604 <1> NUMOFDEVICES equ NumOfKernelDevices + NumOfInstallableDevices

605 <1>

606 <1> ; 26/02/2017

607 <1> ; IRQ Callback (& Signal Response Byte) service availibity

608 <1> ; 'syscalbac'

609 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

610 <1> ; IRQ 00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15

611 <1> ; --- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

612 <1> ; --- 00 00 00 01 02 03 00 04 00 05 06 07 08 09 00 00

613 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

614 <1> IRQenum:

615 00011608 000000010203000400- <1> db 0,0,0,1,2,3,0,4,0,5,6,7,8,9,0,0

615 00011611 05060708090000 <1>

616 <1>

617 <1> ; 28/08/2017

618 <1> ; 20/08/2017

619 <1> ; DMA Registers (for 'sysdma')

620 <1> ; 02/07/2017 (sb16mod.s)

621 00011618 00020406C0C4C8CC <1> dma\_adr: db 0,2,4,6,0C0h,0C4h,0C8h,0CCh

622 00011620 01030507C2C6CACE <1> dma\_cnt: db 1,3,5,7,0C2h,0C6h,0CAh,0CEh

623 00011628 878381828F8B898A <1> dma\_page: db 87h,83h,81h,82h,8Fh,8Bh,89h,8Ah ; 03/08/2017

624 00011630 0A0A0A0AD4D4D4D4 <1> dma\_mask: db 0Ah,0Ah,0Ah,0Ah,0D4h,0D4h,0D4h,0D4h

625 00011638 0B0B0B0BD6D6D6D6 <1> dma\_mod: db 0Bh,0Bh,0Bh,0Bh,0D6h,0D6h,0D6h,0D6h

626 00011640 0C0C0C0CD8D8D8D8 <1> dma\_flip: db 0Ch,0Ch,0Ch,0Ch,0D8h,0D8h,0D8h,0D8h

2314

2315 ; 27/08/2014

2316 scr\_row:

2317 00011648 E0810B00 dd 0B8000h + 0A0h + 0A0h + 0A0h ; Row 3

2318 scr\_col:

2319 0001164C 00000000 dd 0

2320

2321 Align 4

2322 ; 15/04/2016

2323 ; TRDOS 386 (TRDOS v2.0)

2324

2325 ; 21/08/2014

2326 ilist:

2327 ;times 32 dd cpu\_except ; INT 0 to INT 1Fh

2328 ;

2329 ; Exception list

2330 ; 25/08/2014

2331 00011650 [17090000] dd exc0 ; 0h, Divide-by-zero Error

2332 00011654 [1E090000] dd exc1

2333 00011658 [25090000] dd exc2

2334 0001165C [2C090000] dd exc3

2335 00011660 [30090000] dd exc4

2336 00011664 [34090000] dd exc5

2337 00011668 [38090000] dd exc6 ; 06h, Invalid Opcode

2338 0001166C [3C090000] dd exc7

2339 00011670 [40090000] dd exc8

2340 00011674 [44090000] dd exc9

2341 00011678 [48090000] dd exc10

2342 0001167C [4C090000] dd exc11

2343 00011680 [50090000] dd exc12

2344 00011684 [54090000] dd exc13 ; 0Dh, General Protection Fault

2345 00011688 [58090000] dd exc14 ; 0Eh, Page Fault

2346 0001168C [5C090000] dd exc15

2347 00011690 [60090000] dd exc16

2348 00011694 [64090000] dd exc17

2349 00011698 [68090000] dd exc18

2350 0001169C [6C090000] dd exc19

2351 000116A0 [70090000] dd exc20

2352 000116A4 [74090000] dd exc21

2353 000116A8 [78090000] dd exc22

2354 000116AC [7C090000] dd exc23

2355 000116B0 [80090000] dd exc24

2356 000116B4 [84090000] dd exc25

2357 000116B8 [88090000] dd exc26

2358 000116BC [8C090000] dd exc27

2359 000116C0 [90090000] dd exc28

2360 000116C4 [94090000] dd exc29

2361 000116C8 [98090000] dd exc30

2362 000116CC [9C090000] dd exc31

2363 IRQ\_list: ; 28/02/2017 ('syscalbac')

2364 ; Interrupt list

2365 000116D0 [8B060000] dd timer\_int ; INT 20h

2366 ;dd irq0

2367 000116D4 [FF0D0000] dd kb\_int ; 24/01/2016

2368 ;dd irq1

2369 000116D8 [6D080000] dd irq2

2370 ; COM2 int

2371 000116DC [71080000] dd irq3

2372 ; COM1 int

2373 000116E0 [7C080000] dd irq4

2374 000116E4 [87080000] dd irq5

2375 ;DISKETTE\_INT: ;06/02/2015

2376 000116E8 [B0410000] dd fdc\_int ; 16/02/2015, IRQ 6 handler

2377 ;dd irq6

2378 ; Default IRQ 7 handler against spurious IRQs (from master PIC)

2379 ; 25/02/2015 (source: http://wiki.osdev.org/8259\_PIC)

2380 000116EC [F60B0000] dd default\_irq7 ; 25/02/2015

2381 ;dd irq7

2382 ; Real Time Clock Interrupt

2383 000116F0 [F6070000] dd rtc\_int ; 23/02/2015, IRQ 8 handler

2384 ;dd irq8 ; INT 28h

2385 000116F4 [97080000] dd irq9

2386 000116F8 [9B080000] dd irq10

2387 000116FC [9F080000] dd irq11

2388 00011700 [A3080000] dd irq12

2389 00011704 [A7080000] dd irq13

2390 ;HDISK\_INT1: ;06/02/2015

2391 00011708 [2D4B0000] dd hdc1\_int ; 21/02/2015, IRQ 14 handler

2392 ;dd irq14

2393 ;HDISK\_INT2: ;06/02/2015

2394 0001170C [544B0000] dd hdc2\_int ; 21/02/2015, IRQ 15 handler

2395 ;dd irq15 ; INT 2Fh

2396 ; 14/08/2015

2397 ;dd sysent ; INT 30h (system calls)

2398

2399 ; 15/04/2016

2400 ; TRDOS 386(TRDOS v2.0) Software Interrupts

2401

2402 00011710 [6D170100] dd int30h ; Reserved for

2403 ; !!! Retro UNIX (RUNIX) !!!

2404 ; !!! SINGLIX !!! System Calls

2405 00011714 [F2140000] dd int31h ; Video BIOS (IBM PC/AT, Int 10h)

2406 00011718 [1E0C0000] dd int32h ; Keyboard Functions (IBM PC/AT, Int 16h)

2407 0001171C [66420000] dd int33h ; DISK I/O (IBM PC/AT, Int 13h)

2408 00011720 [73F90000] dd int34h ; #IOCTL# (I/O port access support for ring 3)

2409 00011724 [82590000] dd int35h ; Time/Date Functions (IBM PC/AT, Int 1Ah)

2410 00011728 [AA0A0000] dd ignore\_int ; INT 36h : Timer Functions

2411 0001172C [AA0A0000] dd ignore\_int ; INT 37h

2412 00011730 [AA0A0000] dd ignore\_int ; INT 38h

2413 00011734 [AA0A0000] dd ignore\_int ; INT 39h

2414 00011738 [AA0A0000] dd ignore\_int ; INT 3Ah

2415 0001173C [AA0A0000] dd ignore\_int ; INT 3Bh

2416 00011740 [AA0A0000] dd ignore\_int ; INT 3Ch

2417 00011744 [AA0A0000] dd ignore\_int ; INT 3Dh

2418 00011748 [AA0A0000] dd ignore\_int ; INT 3Eh

2419 0001174C [AA0A0000] dd ignore\_int ; INT 3Fh

2420 00011750 [8CC50000] dd sysent ; INT 40h : !!! TRDOS 386 System Calls !!!

2421 ;dd ignore\_int

2422 00011754 00000000 dd 0

2423

2424 ; 20/08/2014

2425 ; /\* This is the default interrupt "handler" :-) \*/

2426 ; Linux v0.12 (head.s)

2427 int\_msg:

2428 00011758 556E6B6E6F776E2069- db "Unknown interrupt ! ", 0

2428 00011761 6E7465727275707420-

2428 0001176A 212000

2429

2430 ; 15/04/2016

2431 ; TRDOS 386 (TRDOS v2.0)

2432

2433 ; 29/04/2016

2434 int30h:

2435 trdos\_isc\_routine:

2436 ; 02/05/2016

2437 ; 01/05/2016

2438 ; 29/04/2016

2439 ; 18/04/2016

2440 ; 15/04/2016 (TRDOS 386 = TRDOS v2.0)

2441 ; 17/04/2011 (TRDOS v1.0, 'IFC.ASM')

2442 ; 03/02/2011 ('trdos\_ifc\_routine')

2443 ;

2444 0001176D 8B1C24 mov ebx, [esp] ; EIP (next)

2445 00011770 83EB02 sub ebx, 2 ; EIP (CD ##h)

2446

2447 00011773 89C1 mov ecx, eax

2448 00011775 8A4301 mov al, [ebx+1] ; CDh ##h

2449

2450 00011778 66BA1000 mov dx, KDATA

2451 0001177C 8EDA mov ds, dx

2452 0001177E 8EC2 mov es, dx

2453

2454 00011780 FC cld

2455 00011781 8B15[38580100] mov edx, [k\_page\_dir]

2456 00011787 0F22DA mov cr3, edx

2457

2458 0001178A E83A1BFFFF call bytetohex

2459 0001178F 66A3[C5130100] mov [int\_num\_str], ax

2460

2461 00011795 89D8 mov eax, ebx ; EIP

2462 00011797 E86D1BFFFF call dwordtohex

2463 0001179C 8915[E1130100] mov [eip\_str], edx

2464 000117A2 A3[E5130100] mov [eip\_str+4], eax

2465

2466 000117A7 89C8 mov eax, ecx

2467 000117A9 E85B1BFFFF call dwordtohex

2468 000117AE 8915[D0130100] mov [eax\_str], edx

2469 000117B4 A3[D4130100] mov [eax\_str+4], eax

2470

2471 000117B9 43 inc ebx

2472 000117BA 8A03 mov al, [ebx] ; Interrupt number

2473

2474 trdos\_isc\_handler:

2475 000117BC 80FE30 cmp dh, 30h ; Retro UNIX, SINGLIX System calls

2476 000117BF 7507 jne short trdos\_usi\_handler

2477 000117C1 BE[62130100] mov esi, isc\_msg

2478 000117C6 EB05 jmp short loc\_write\_inv\_system\_call\_msg

2479

2480 trdos\_usi\_handler:

2481 000117C8 BE[78130100] mov esi, usi\_msg

2482

2483 loc\_write\_inv\_system\_call\_msg:

2484 000117CD E88B4BFFFF call print\_msg

2485 ; 29/04/2016

2486 000117D2 BE[AE130100] mov esi, inv\_msg\_for\_trdos\_v2

2487 000117D7 E8814BFFFF call print\_msg

2488

2489 loc\_ifc\_terminate\_process:

2490 ; u.uno = process number

2491 ; 29/04/2016

2492

2493 ; 02/05/2016

2494 000117DC FE05[5B030300] inc byte [sysflg] ; 0FFh -> 0

2495

2496 000117E2 B801000000 mov eax, 1

2497 000117E7 E979B0FFFF jmp sysexit

2498

2499 ; 07/03/2015

2500 ; Temporary Code

2501 display\_disks:

2502 000117EC 803D[F65C0000]00 cmp byte [fd0\_type], 0

2503 000117F3 7605 jna short ddsks1

2504 000117F5 E87D000000 call pdskm

2505 ddsks1:

2506 000117FA 803D[F75C0000]00 cmp byte [fd1\_type], 0

2507 00011801 760C jna short ddsks2

2508 00011803 C605[47190100]31 mov byte [dskx], '1'

2509 0001180A E868000000 call pdskm

2510 ddsks2:

2511 0001180F 803D[F85C0000]00 cmp byte [hd0\_type], 0

2512 00011816 7654 jna short ddsk6

2513 00011818 66C705[45190100]68- mov word [dsktype], 'hd'

2513 00011820 64

2514 00011821 C605[47190100]30 mov byte [dskx], '0'

2515 00011828 E84A000000 call pdskm

2516 ddsks3:

2517 0001182D 803D[F95C0000]00 cmp byte [hd1\_type], 0

2518 00011834 7636 jna short ddsk6

2519 00011836 C605[47190100]31 mov byte [dskx], '1'

2520 0001183D E835000000 call pdskm

2521 ddsks4:

2522 00011842 803D[FA5C0000]00 cmp byte [hd2\_type], 0

2523 00011849 7621 jna short ddsk6

2524 0001184B C605[47190100]32 mov byte [dskx], '2'

2525 00011852 E820000000 call pdskm

2526 ddsks5:

2527 00011857 803D[FB5C0000]00 cmp byte [hd3\_type], 0

2528 0001185E 760C jna short ddsk6

2529 00011860 C605[47190100]33 mov byte [dskx], '3'

2530 00011867 E80B000000 call pdskm

2531 ddsk6:

2532 0001186C BE[6F190100] mov esi, nextline

2533 00011871 E806000000 call pdskml

2534 pdskm\_ok:

2535 00011876 C3 retn

2536 pdskm:

2537 00011877 BE[43190100] mov esi, dsk\_ready\_msg

2538 pdskml:

2539 0001187C AC lodsb

2540 0001187D 08C0 or al, al

2541 0001187F 74F5 jz short pdskm\_ok

2542 00011881 56 push esi

2543 ; 13/05/2016

2544 00011882 BB07000000 mov ebx, 7 ; Black background,

2545 ; light gray forecolor

2546 ; Video page 0 (bh=0)

2547 00011887 E82604FFFF call \_write\_tty

2548 0001188C 5E pop esi

2549 0001188D EBED jmp short pdskml

2550

2551 0001188F 90 Align 2

2552 ; 21/08/2014

2553 exc\_msg:

2554 00011890 435055206578636570- db "CPU exception ! "

2554 00011899 74696F6E202120

2555 excnstr: ; 25/08/2014

2556 000118A0 3F3F68202045495020- db "??h", " EIP : "

2556 000118A9 3A20

2557 EIPstr: ; 29/08/2014

2558 000118AB 00<rept> times 12 db 0

2559

2560 ; 23/02/2015

2561 ; 25/08/2014

2562 ;scounter:

2563 ; db 5

2564 ; db 19

2565

2566 ; 06/11/2014

2567 ; Memory Information message

2568 ; 14/08/2015

2569 msg\_memory\_info:

2570 000118B7 07 db 07h

2571 000118B8 0D0A db 0Dh, 0Ah

2572 ;db "MEMORY ALLOCATION INFO", 0Dh, 0Ah, 0Dh, 0Ah

2573 000118BA 546F74616C206D656D- db "Total memory : "

2573 000118C3 6F7279203A20

2574 mem\_total\_b\_str: ; 10 digits

2575 000118C9 303030303030303030- db "0000000000 bytes", 0Dh, 0Ah

2575 000118D2 302062797465730D0A

2576 000118DB 202020202020202020- db " ", 20h, 20h, 20h

2576 000118E4 202020202020202020

2577 mem\_total\_p\_str: ; 7 digits

2578 000118ED 303030303030302070- db "0000000 pages", 0Dh, 0Ah

2578 000118F6 616765730D0A

2579 000118FC 0D0A db 0Dh, 0Ah

2580 000118FE 46726565206D656D6F- db "Free memory : "

2580 00011907 727920203A20

2581 free\_mem\_b\_str: ; 10 digits

2582 0001190D 3F3F3F3F3F3F3F3F3F- db "?????????? bytes", 0Dh, 0Ah

2582 00011916 3F2062797465730D0A

2583 0001191F 202020202020202020- db " ", 20h, 20h, 20h

2583 00011928 202020202020202020

2584 free\_mem\_p\_str: ; 7 digits

2585 00011931 3F3F3F3F3F3F3F2070- db "??????? pages", 0Dh, 0Ah

2585 0001193A 616765730D0A

2586 00011940 0D0A00 db 0Dh, 0Ah, 0

2587

2588 dsk\_ready\_msg:

2589 00011943 0D0A db 0Dh, 0Ah

2590 dsktype:

2591 00011945 6664 db 'fd'

2592 dskx:

2593 00011947 30 db '0'

2594 00011948 20 db 20h

2595 00011949 697320524541445920- db 'is READY ...'

2595 00011952 2E2E2E

2596 00011955 00 db 0

2597

2598 setup\_error\_msg:

2599 00011956 0D0A db 0Dh, 0Ah

2600 00011958 4469736B2053657475- db 'Disk Setup Error !'

2600 00011961 70204572726F722021

2601 0001196A 0D0A00 db 0Dh, 0Ah,0

2602

2603 next2line: ; 08/02/2016

2604 0001196D 0D0A db 0Dh, 0Ah

2605 nextline:

2606 0001196F 0D0A00 db 0Dh, 0Ah, 0

2607

2608 ; KERNEL - SYSINIT Messages

2609 ; 24/08/2015

2610 ; 13/04/2015 - (Retro UNIX 386 v1 Beginning)

2611 ; 14/07/2013

2612 ;kernel\_init\_err\_msg:

2613 ; db 0Dh, 0Ah

2614 ; db 07h

2615 ; db 'Kernel initialization ERROR !'

2616 ; db 0Dh, 0Ah, 0

2617

2618 ;welcome\_msg:

2619 ; db 0Dh, 0Ah

2620 ; db 07h

2621 ; db 'Welcome to TRDOS 386 Operating System !'

2622 ; db 0Dh, 0Ah

2623 ; db 'by Erdogan Tan - 31/12/2017 (v2.0.0)'

2624 ; db 0Dh, 0Ah, 0

2625

2626 panic\_msg:

2627 00011972 0D0A07 db 0Dh, 0Ah, 07h

2628 00011975 4552524F523A204B65- db 'ERROR: Kernel Panic !'

2628 0001197E 726E656C2050616E69-

2628 00011987 632021

2629 0001198A 0D0A00 db 0Dh, 0Ah, 0

2630

2631 ;msgl\_drv\_not\_ready:

2632 ; db 07h, 0Dh, 0Ah

2633 ; db 'Drive not ready or read error !'

2634 ; db 0Dh, 0Ah, 0

2635

2636 starting\_msg:

2637 0001198D 5475726B6973682052- db "Turkish Rational DOS v2.0 [31/12/2017] ...", 0

2637 00011996 6174696F6E616C2044-

2637 0001199F 4F532076322E30205B-

2637 000119A8 33312F31322F323031-

2637 000119B1 375D202E2E2E00

2638 NextLine:

2639 000119B8 0D0A00 db 0Dh, 0Ah, 0

2640

2641 %include 'audio.s' ; 03/04/2017

1 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2 <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0 - audio.s

3 <1> ; ----------------------------------------------------------------------------

4 <1> ; Last Update: 28/10/2017

5 <1> ; ----------------------------------------------------------------------------

6 <1> ; Beginning: 03/04/2017

7 <1> ; ----------------------------------------------------------------------------

8 <1> ; Assembler: NASM version 2.11 (trdos386.s)

9 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

10 <1>

11 <1> ; AUDIO CONTROLLER & CODEC DEFINITIONS & CODE FOR TRDOS 386

12 <1>

13 <1> ;=============================================================================

14 <1> ; EQUATES

15 <1> ;=============================================================================

16 <1>

17 <1> ; PCI EQUATES

18 <1>

19 <1> BIT0 EQU 1

20 <1> BIT1 EQU 2

21 <1> BIT2 EQU 4

22 <1> BIT3 EQU 8

23 <1> BIT4 EQU 10h

24 <1> BIT5 EQU 20h

25 <1> BIT6 EQU 40h

26 <1> BIT7 EQU 80h

27 <1> BIT8 EQU 100h

28 <1> BIT9 EQU 200h

29 <1> BIT10 EQU 400h

30 <1> BIT11 EQU 800h

31 <1> BIT12 EQU 1000h

32 <1> BIT13 EQU 2000h

33 <1> BIT14 EQU 4000h

34 <1> BIT15 EQU 8000h

35 <1> BIT16 EQU 10000h

36 <1> BIT17 EQU 20000h

37 <1> BIT18 EQU 40000h

38 <1> BIT19 EQU 80000h

39 <1> BIT20 EQU 100000h

40 <1> BIT21 EQU 200000h

41 <1> BIT22 EQU 400000h

42 <1> BIT23 EQU 800000h

43 <1> BIT24 EQU 1000000h

44 <1> BIT25 EQU 2000000h

45 <1> BIT26 EQU 4000000h

46 <1> BIT27 EQU 8000000h

47 <1> BIT28 EQU 10000000h

48 <1> BIT29 EQU 20000000h

49 <1> BIT30 EQU 40000000h

50 <1> BIT31 EQU 80000000h

51 <1> NOT\_BIT31 EQU 7FFFFFFFh

52 <1>

53 <1> ; PCI equates

54 <1> ; PCI function address (PFA)

55 <1> ; bit 31 = 1

56 <1> ; bit 23:16 = bus number (0-255)

57 <1> ; bit 15:11 = device number (0-31)

58 <1> ; bit 10:8 = function number (0-7)

59 <1> ; bit 7:0 = register number (0-255)

60 <1>

61 <1> IO\_ADDR\_MASK EQU 0FFFEh ; mask off bit 0 for reading BARs

62 <1> PCI\_INDEX\_PORT EQU 0CF8h

63 <1> PCI\_DATA\_PORT EQU 0CFCh

64 <1> PCI32 EQU BIT31 ; bitflag to signal 32bit access

65 <1> PCI16 EQU BIT30 ; bitflag for 16bit access

66 <1> NOT\_PCI32\_PCI16 EQU 03FFFFFFFh ; 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

103 <1> VIA\_ACLINK\_CTRL\_INIT equ (VIA\_ACLINK\_CTRL\_ENABLE + VIA\_ACLINK\_CTRL\_RESET + VIA\_ACLINK\_CTRL\_PCM + VIA\_ACLINK\_CTRL\_VRA)

104 <1>

105 <1> CODEC\_AUX\_VOL equ 04h

106 <1> VIA\_REG\_AC97\_BUSY equ 01000000h ;(1<<24)

107 <1> VIA\_REG\_AC97\_CMD\_SHIFT equ 10h ; 16

108 <1> VIA\_REG\_AC97\_PRIMARY\_VALID equ 02000000h ;(1<<25)

109 <1> VIA\_REG\_AC97\_READ equ 00800000h ;(1<<23)

110 <1> VIA\_REG\_AC97\_CODEC\_ID\_SHIFT equ 1Eh ; 30

111 <1> VIA\_REG\_AC97\_CODEC\_ID\_PRIMARY equ 0

112 <1> VIA\_REG\_AC97\_DATA\_SHIFT equ 0

113 <1> VIADEV\_PLAYBACK equ 0

114 <1> VIA\_REG\_OFFSET\_STATUS equ 0 ;; byte - channel status

115 <1> VIA\_REG\_OFFSET\_CONTROL equ 01h ;; byte - channel control

116 <1> VIA\_REG\_CTRL\_START equ 80h ;; WO

117 <1> VIA\_REG\_CTRL\_TERMINATE equ 40h ;; WO

118 <1> VIA\_REG\_CTRL\_PAUSE equ 08h ;; RW

119 <1> VIA\_REG\_CTRL\_RESET equ 01h ;; RW - probably reset? undocumented

120 <1> VIA\_REG\_OFFSET\_STOP\_IDX equ 08h ;; dword - stop index, channel type, sample rate

121 <1> VIA8233\_REG\_TYPE\_16BIT equ 200000h ;; RW

122 <1> VIA8233\_REG\_TYPE\_STEREO equ 100000h ;; RW

123 <1> VIA\_REG\_OFFSET\_CURR\_INDEX equ 0Fh ;; byte - channel current index (for via8233 only)

124 <1> VIA\_REG\_OFFSET\_TABLE\_PTR equ 04h ;; dword - channel table pointer

125 <1> VIA\_REG\_OFFSET\_CURR\_PTR equ 04h ;; dword - channel current pointer

126 <1> VIA\_REG\_OFS\_PLAYBACK\_VOLUME\_L equ 02h ;; byte

127 <1> VIA\_REG\_OFS\_PLAYBACK\_VOLUME\_R equ 03h ;; byte

128 <1> VIA\_REG\_CTRL\_AUTOSTART equ 20h

129 <1> VIA\_REG\_CTRL\_INT\_EOL equ 02h

130 <1> VIA\_REG\_CTRL\_INT\_FLAG equ 01h

133 <1> VIA\_REG\_CTRL\_INT equ (VIA\_REG\_CTRL\_INT\_FLAG + VIA\_REG\_CTRL\_INT\_EOL + VIA\_REG\_CTRL\_AUTOSTART)

134 <1>

135 <1> VIA\_REG\_STAT\_STOPPED equ 04h ;; RWC

136 <1> VIA\_REG\_STAT\_EOL equ 02h ;; RWC

137 <1> VIA\_REG\_STAT\_FLAG equ 01h ;; RWC

138 <1> VIA\_REG\_STAT\_ACTIVE equ 80h ;; RO

139 <1> ; 28/11/2016

140 <1> VIA\_REG\_STAT\_LAST equ 40h ;; RO

141 <1> VIA\_REG\_STAT\_TRIGGER\_QUEUED equ 08h ;; RO

142 <1> VIA\_REF\_CTRL\_INT\_STOP equ 04h ; Interrupt on Current Index = Stop Index

143 <1> ; and End of Block

144 <1>

145 <1> VIA\_REG\_OFFSET\_CURR\_COUNT equ 0Ch ;; dword - channel current count, index

146 <1>

147 <1> PORTB EQU 061h

148 <1> REFRESH\_STATUS EQU 010h ; Refresh signal status

149 <1>

150 <1> ; AC97 Codec registers.

151 <1>

152 <1> ; each codec/mixer register is 16bits

153 <1>

154 <1> CODEC\_RESET\_REG equ 00h ; reset codec

155 <1> CODEC\_MASTER\_VOL\_REG equ 02h ; master volume

156 <1> CODEC\_HP\_VOL\_REG equ 04h ; headphone volume

157 <1> CODEC\_MASTER\_MONO\_VOL\_REG equ 06h ; master mono volume

158 <1> CODEC\_MASTER\_TONE\_REG equ 08h ; master tone (R+L)

159 <1> CODEC\_PCBEEP\_VOL\_REG equ 0Ah ; PC beep volume

160 <1> CODEC\_PHONE\_VOL\_REG equ 0Bh ; phone volume

161 <1> CODEC\_MIC\_VOL\_REG equ 0Eh ; MIC volume

162 <1> CODEC\_LINE\_IN\_VOL\_REG equ 10h ; line input volume

163 <1> CODEC\_CD\_VOL\_REG equ 12h ; CD volume

164 <1> CODEC\_VID\_VOL\_REG equ 14h ; video volume

165 <1> CODEC\_AUX\_VOL\_REG equ 16h ; aux volume

166 <1> CODEC\_PCM\_OUT\_REG equ 18h ; PCM output volume

167 <1> CODEC\_RECORD\_SELECT\_REG equ 1Ah ; record select input

168 <1> CODEC\_RECORD\_VOL\_REG equ 1Ch ; record volume

169 <1> CODEC\_RECORD\_MIC\_VOL\_REG equ 1Eh ; record mic volume

170 <1> CODEC\_GP\_REG equ 20h ; general purpose

171 <1> CODEC\_3D\_CONTROL\_REG equ 22h ; 3D control

172 <1> ; 24h is reserved

173 <1> CODEC\_POWER\_CTRL\_REG equ 26h ; powerdown control

174 <1> CODEC\_EXT\_AUDIO\_REG equ 28h ; extended audio

175 <1> CODEC\_EXT\_AUDIO\_CTRL\_REG equ 2Ah ; extended audio control

176 <1> CODEC\_PCM\_FRONT\_DACRATE\_REG equ 2Ch ; PCM out sample rate

177 <1> CODEC\_PCM\_SURND\_DACRATE\_REG equ 2Eh ; surround sound sample rate

178 <1> CODEC\_PCM\_LFE\_DACRATE\_REG equ 30h ; LFE sample rate

179 <1> CODEC\_LR\_ADCRATE\_REG equ 32h ; PCM in sample rate

180 <1> CODEC\_MIC\_ADCRATE\_REG equ 34h ; mic in sample rate

181 <1>

182 <1> ; VT8233 SGD bits (21/04/2017)

183 <1> FLAG EQU BIT30

184 <1> EOL EQU BIT31

185 <1>

186 <1> ; INTEL ICH EQUATES

187 <1> ; 28/05/2017

188 <1> INTEL\_VID equ 8086h ; Intel's PCI vendor ID

189 <1> ICH\_DID equ 2415h ; ICH (82801AA) device ID

190 <1> NAMBAR\_REG equ 10h ; native audio mixer Base Address Register

191 <1> NABMBAR\_REG equ 14h ; native audio bus mastering Base Addr Reg

192 <1>

193 <1> PI\_CR\_REG equ 0Bh ; PCM in Control Register

194 <1> PO\_CR\_REG equ 1Bh ; PCM out Control Register

195 <1> MC\_CR\_REG equ 2Bh ; MIC in Control Register

196 <1>

197 <1> PI\_SR\_REG equ 6 ; PCM in Status register

198 <1> PO\_SR\_REG equ 16h ; PCM out Status register

199 <1> MC\_SR\_REG equ 26h ; MIC in Status register

200 <1>

201 <1> IOCE equ BIT4 ; interrupt on complete enable.

202 <1> FEIFE equ BIT3 ; set if you want an interrupt to fire

203 <1> LVBIE equ BIT2 ; last valid buffer interrupt enable.

204 <1> RR equ BIT1 ; reset registers. Nukes all regs

205 <1> ; except bits 4:2 of this register.

206 <1> ; Only set this bit if BIT 0 is 0

207 <1> RPBM equ BIT0 ; Run/Pause

208 <1> ; set this bit to start the codec!

209 <1>

210 <1> PI\_BDBAR\_REG equ 0 ; PCM in buffer descriptor BAR

211 <1> PO\_BDBAR\_REG equ 10h ; PCM out buffer descriptor BAR

212 <1> MC\_BDBAR\_REG equ 20h ; MIC in buffer descriptor BAR

213 <1>

214 <1> PI\_CIV\_REG equ 4 ; PCM in current Index value (RO)

215 <1> PO\_CIV\_REG equ 14h ; PCM out current Index value (RO)

216 <1> MC\_CIV\_REG equ 24h ; MIC in current Index value (RO)

217 <1>

218 <1> PI\_LVI\_REG equ 5 ; PCM in Last Valid Index

219 <1> PO\_LVI\_REG equ 15h ; PCM out Last Valid Index

220 <1> MC\_LVI\_REG equ 25h ; MIC in Last Valid Index

221 <1>

222 <1> IOC equ BIT31 ; Fire an interrupt whenever this

223 <1> ; buffer is complete.

224 <1> BUP equ BIT30 ; Buffer Underrun Policy.

225 <1>

226 <1> GLOB\_CNT\_REG equ 2Ch ; Global Control Register

227 <1> GLOB\_STS\_REG equ 30h ; Global Status register (RO)

228 <1>

229 <1> CTRL\_ST\_CREADY equ BIT8+BIT9+BIT28 ; Primary Codec Ready

230 <1>

231 <1> CODEC\_REG\_POWERDOWN equ 26h

232 <1> CODEC\_REG\_ST equ 26h

233 <1>

234 <1> ; 22/06/2017

235 <1> PO\_PICB\_REG equ 18h ; PCM Out Position In Current Buffer Register

236 <1>

237 <1> ;=============================================================================

238 <1> ; CODE

239 <1> ;=============================================================================

240 <1>

241 <1> ; CODE for INTEL ICH AC'97 AUDIO CONTROLLER

242 <1>

243 <1> DetectICH:

244 <1> ; 10/06/2017

245 <1> ; 05/06/2017

246 <1> ; 29/05/2017

247 <1> ; 28/05/2017

248 000119BB B886801524 <1> mov eax, (ICH\_DID << 16) + INTEL\_VID

249 000119C0 E876000000 <1> call pciFindDevice

250 000119C5 730D <1> jnc short d\_ac97\_1

251 <1> d\_ac97\_0:

252 <1> ; couldn't find the audio device!

253 000119C7 C3 <1> retn

254 <1>

255 <1> ; CODE for VIA VT8233 AUDIO CONTROLLER

256 <1>

257 <1> DetectVT8233:

258 <1> ; 10/06/2017

259 <1> ; 05/06/2017

260 <1> ; 29/05/2017

261 <1> ; 03/04/2017

262 000119C8 B806115930 <1> mov eax, (VT8233\_DID << 16) + VIA\_VID

263 000119CD E869000000 <1> call pciFindDevice

264 <1> ; jnc short d\_vt8233\_0

265 <1> ; couldn't find the audio device!

266 <1> ; retn

267 000119D2 72F3 <1> jc short d\_ac97\_0 ; 28/05/2017

268 <1> d\_vt8233\_0:

269 <1> ; 24/03/2017 ('player.asm')

270 <1> ; 12/11/2016

271 <1> ; Erdogan Tan - 8/11/2016

272 <1> ; References: Kolibrios - vt823x.asm (2016)

273 <1> ; VIA VT8235 V-Link South Bridge (VT8235-VIA.PDF)(2002)

274 <1> ; lowlevel.eu - AC97 (2016)

275 <1> ; .wav player for DOS by Jeff Leyda (2002) -this file-

276 <1> ; Linux kernel - via82xx.c (2016)

277 <1> d\_ac97\_1:

278 <1> ; eax = BUS/DEV/FN

279 <1> ; 00000000BBBBBBBBDDDDDFFF00000000

280 <1> ; edx = DEV/VENDOR

281 <1> ; DDDDDDDDDDDDDDDDVVVVVVVVVVVVVVVV

282 <1>

283 000119D4 A3[C46B0100] <1> mov [audio\_dev\_id], eax

284 000119D9 8915[C86B0100] <1> mov [audio\_vendor], edx

285 <1>

286 <1> ; init controller

287 000119DF B004 <1> mov al, PCI\_CMD\_REG ; command register (04h)

288 000119E1 E8E2000000 <1> call pciRegRead32

289 <1>

290 <1> ; eax = BUS/DEV/FN/REG

291 <1> ; edx = STATUS/COMMAND

292 <1> ; SSSSSSSSSSSSSSSSCCCCCCCCCCCCCCCC

293 000119E6 8915[CC6B0100] <1> mov [audio\_stats\_cmd], edx

294 <1>

295 000119EC B010 <1> mov al, PCI\_IO\_BASE ; IO base address register (10h)

296 <1> ;mov al, NAMBAR\_REG ; Native Audio Mixer BAR (10h)

297 000119EE E8D5000000 <1> call pciRegRead32

298 <1>

299 000119F3 66813D[C86B0100]86- <1> cmp word [audio\_vendor], 8086h ; AC'97 ?

299 000119FB 80 <1>

300 000119FC 751F <1> jne short d\_vt8233\_1

301 <1>

302 000119FE 6683E2FE <1> and dx, 0FFFEh ; Audio Codec IO\_ADDR\_MASK

303 00011A02 668915[F46B0100] <1> mov [NAMBAR], dx

304 <1>

305 00011A09 B014 <1> mov al, NABMBAR\_REG ; Native Audio Bus Mastering BAR (14h)

306 00011A0B E8B8000000 <1> call pciRegRead32

307 <1>

308 00011A10 6683E2C0 <1> and dx, 0FFC0h ; Audio Controller IO\_ADDR\_MASK

309 00011A14 668915[F66B0100] <1> mov [NABMBAR], dx

310 <1> ;mov [audio\_io\_base], dx

311 <1>

312 00011A1B EB0B <1> jmp short d\_ac97\_2

313 <1>

314 <1> d\_vt8233\_1:

315 00011A1D 6683E2C0 <1> and dx, 0FFC0h ; Audio Controller IO\_ADDR\_MASK

316 00011A21 668915[C26B0100] <1> mov [audio\_io\_base], dx

317 <1>

318 <1> d\_ac97\_2:

319 <1> ; 10/06/2017

320 00011A28 B03C <1> mov al, AC97\_INT\_LINE ; Interrupt Line Register (3Ch)

321 <1> ;call pciRegRead32

322 00011A2A E886000000 <1> call pciRegRead8

323 <1>

324 <1> ;and edx, 0FFh

325 00011A2F 6681E2FF00 <1> and dx, 0FFh

326 <1>

327 00011A34 8815[BF6B0100] <1> mov [audio\_intr], dl

328 <1>

329 00011A3A C3 <1> retn

330 <1>

331 <1> ;; (Note: Interrupts are already enabled by TRDOS 386 kernel!)

332 <1> ;mov cx, dx

333 <1>

334 <1> ;in al, 0A1h ; irq 8-15

335 <1> ;mov ah, al

336 <1> ;in al, 21h ; irq 0-7

337 <1> ;btr ax, dx ; unmask ; 17/03/2017

338 <1> ;;bts ax, dx ; MASK interrupt ; 10/06/2017

339 <1> ;out 21h, al ; irq <= 7

340 <1> ;mov al, ah

341 <1> ;out 0A1h, al ; irq > 7

342 <1> ;

343 <1>

344 <1> ; 10/06/2017

345 <1> ; === Intel ICH I/O Controller Hub Datasheet, Section 8.1.16 ===

346 <1> ; PRQ[n]\_ROUT Register (61h, PRQB) Bit 7:

347 <1> ; Interrupt Routing Enable (IRQEN).

348 <1> ; 0 = The corresponding PIRQ is routed to one of the ISA-compatible

349 <1> ; interrupts specified in bits[3:0].

350 <1> ; 1 = The PIRQ is not routed to the 8259.

351 <1> ; Note: If the PIRQ is intended to cause an interrupt to the ICH’s

352 <1> ; integrated I/O APIC, then this bit should be set to 0 and

353 <1> ; the APIC\_EN bit should be set to 1.

354 <1> ; The IRQEN must be set to 0 and the PIRQ routed to

355 <1> ; an 8259 interrupt via the IRQ Routing filed (bits[3:0).

356 <1> ; The corresponding 8259 interrupt must be masked via the

357 <1> ; appropriated bit in the 8259’s OCW1 (Interrupt Mask)

358 <1> ; register. The IOAPIC must then be enabled by setting

359 <1> ; the APIC\_EN bit in the GEN\_CNTL register.

360 <1>

361 <1> ;mov eax, 0F861h ; D31:F0

362 <1> ;AL=61h : PIRQ[B] Routing Control Reg, LPC interface

363 <1> ;;mov dl, [audio\_intr]

364 <1> ;call pciRegWrite8

365 <1> ;;mov al, 0D0h ; General Control Register (GEN\_CTL)

366 <1> ;;call pciRegRead32

367 <1> ;;or edx, 100h ; Bit 8, APIC\_EN (Enable I/O APIC)

368 <1> ;;;call pciRegWrite32

369 <1> ;;and edx, ~100h

370 <1> ;;call pciRegWrite32 ; ; Bit 8, APIC\_EN (Disable I/O APIC)

371 <1> ;

372 <1>

373 <1> ;mov dx, 4D1h ; 8259 ELCR2

374 <1> ;in al, dx

375 <1> ;mov ah, al

376 <1> ;;mov dx, 4D0h ; 8259 ELCR1

377 <1> ;dec dl

378 <1> ;in al, dx

379 <1> ;bts ax, cx

380 <1> ;;mov dx, 4D0h

381 <1> ;out dx, al ; set level-triggered mode

382 <1> ;mov al, ah ; 29/05/2017

383 <1> ;;mov dx, 4D1h

384 <1> ;inc dl

385 <1> ;out dx, al ; set level-triggered mode

386 <1>

387 <1> ;xor eax, eax ; 0

388 <1>

389 <1> ;retn

390 <1>

391 <1> ; CODE for PCI

392 <1>

393 <1> pciFindDevice:

394 <1> ; 03/04/2017 ('pci.asm', 20/03/2017)

395 <1> ;

396 <1> ; scan through PCI space looking for a device+vendor ID

397 <1> ;

398 <1> ; Entry: EAX=Device+Vendor ID

399 <1> ;

400 <1> ; Exit: EAX=PCI address if device found

401 <1> ; EDX=Device+Vendor ID

402 <1> ; CY clear if found, set if not found. EAX invalid if CY set.

403 <1> ;

404 <1> ; Destroys: ebx, esi, edi, cl

405 <1> ;

406 <1>

407 <1> ;push ecx

408 00011A3B 50 <1> push eax

409 <1> ;push esi

410 <1> ;push edi

411 <1>

412 00011A3C 89C6 <1> mov esi, eax ; save off vend+device ID

413 00011A3E BF00FFFF7F <1> mov edi, (80000000h - 100h) ; start with bus 0, dev 0 func 0

414 <1>

415 <1> nextPCIdevice:

416 00011A43 81C700010000 <1> add edi, 100h

417 00011A49 81FF00F8FF80 <1> cmp edi, 80FFF800h ; scanned all devices?

418 00011A4F F9 <1> stc

419 00011A50 740C <1> je short PCIScanExit ; not found

420 <1>

421 00011A52 89F8 <1> mov eax, edi ; read PCI registers

422 00011A54 E86F000000 <1> call pciRegRead32

423 00011A59 39F2 <1> cmp edx, esi ; found device?

424 00011A5B 75E6 <1> jne short nextPCIdevice

425 00011A5D F8 <1> clc

426 <1>

427 <1> PCIScanExit:

428 00011A5E 9C <1> pushf

429 00011A5F B8FFFFFF7F <1> mov eax, NOT\_BIT31 ; 19/03/2017

430 00011A64 21F8 <1> and eax, edi ; return only bus/dev/fn #

431 00011A66 9D <1> popf

432 <1>

433 <1> ;pop edi

434 <1> ;pop esi

435 00011A67 5A <1> pop edx

436 <1> ;pop ecx

437 00011A68 C3 <1> retn

438 <1>

439 <1> pciRegRead:

440 <1> ; 03/04/2017 ('pci.asm', 20/03/2017)

441 <1> ;

442 <1> ; 8/16/32bit PCI reader

443 <1> ;

444 <1> ; Entry: EAX=PCI Bus/Device/fn/register number

445 <1> ; BIT30 set if 32 bit access requested

446 <1> ; BIT29 set if 16 bit access requested

447 <1> ; otherwise defaults to 8 bit read

448 <1> ;

449 <1> ; Exit: DL,DX,EDX register data depending on requested read size

450 <1> ;

451 <1> ; Note1: this routine is meant to be called via pciRegRead8,

452 <1> ; pciRegread16 or pciRegRead32, listed below.

453 <1> ;

454 <1> ; Note2: don't attempt to read 32 bits of data from a non dword

455 <1> ; aligned reg number. Likewise, don't do 16 bit reads from

456 <1> ; non word aligned reg #

457 <1>

458 00011A69 53 <1> push ebx

459 00011A6A 51 <1> push ecx

460 00011A6B 89C3 <1> mov ebx, eax ; save eax, dh

461 00011A6D 88F1 <1> mov cl, dh

462 <1>

463 00011A6F 25FFFFFF3F <1> and eax, NOT\_PCI32\_PCI16 ; clear out data size request

464 00011A74 0D00000080 <1> or eax, BIT31 ; make a PCI access request

465 00011A79 24FC <1> and al, ~3 ; NOT 3 ; force index to be dword

466 <1>

467 00011A7B 66BAF80C <1> mov dx, PCI\_INDEX\_PORT

468 00011A7F EF <1> out dx, eax ; write PCI selector

469 <1>

470 00011A80 66BAFC0C <1> mov dx, PCI\_DATA\_PORT

471 00011A84 88D8 <1> mov al, bl

472 00011A86 2403 <1> and al, 3 ; figure out which port to

473 00011A88 00C2 <1> add dl, al ; read to

474 <1>

475 00011A8A F7C3000000C0 <1> test ebx, PCI32+PCI16

476 00011A90 7507 <1> jnz short \_pregr0

477 00011A92 EC <1> in al, dx ; return 8 bits of data

478 00011A93 88C2 <1> mov dl, al

479 00011A95 88CE <1> mov dh, cl ; restore dh for 8 bit read

480 00011A97 EB12 <1> jmp short \_pregr2

481 <1> \_pregr0:

482 00011A99 F7C300000080 <1> test ebx, PCI32

483 00011A9F 7507 <1> jnz short \_pregr1

484 00011AA1 66ED <1> in ax, dx

485 00011AA3 6689C2 <1> mov dx, ax ; return 16 bits of data

486 00011AA6 EB03 <1> jmp short \_pregr2

487 <1> \_pregr1:

488 00011AA8 ED <1> in eax, dx ; return 32 bits of data

489 00011AA9 89C2 <1> mov edx, eax

490 <1> \_pregr2:

491 00011AAB 89D8 <1> mov eax, ebx ; restore eax

492 00011AAD 25FFFFFF3F <1> and eax, NOT\_PCI32\_PCI16 ; clear out data size request

493 00011AB2 59 <1> pop ecx

494 00011AB3 5B <1> pop ebx

495 00011AB4 C3 <1> retn

496 <1>

497 <1> pciRegRead8:

498 00011AB5 25FFFFFF3F <1> and eax, NOT\_PCI32\_PCI16 ; set up 8 bit read size

499 00011ABA EBAD <1> jmp short pciRegRead ; call generic PCI access

500 <1>

501 <1> pciRegRead16:

502 00011ABC 25FFFFFF3F <1> and eax, NOT\_PCI32\_PCI16 ; set up 16 bit read size

503 00011AC1 0D00000040 <1> or eax, PCI16 ; call generic PCI access

504 00011AC6 EBA1 <1> jmp short pciRegRead

505 <1>

506 <1> pciRegRead32:

507 00011AC8 25FFFFFF3F <1> and eax, NOT\_PCI32\_PCI16 ; set up 32 bit read size

508 00011ACD 0D00000080 <1> or eax, PCI32 ; call generic PCI access

509 00011AD2 EB95 <1> jmp pciRegRead

510 <1>

511 <1> pciRegWrite:

512 <1> ; 03/04/2017 ('pci.asm', 29/11/2016)

513 <1> ;

514 <1> ; 8/16/32bit PCI writer

515 <1> ;

516 <1> ; Entry: EAX=PCI Bus/Device/fn/register number

517 <1> ; BIT31 set if 32 bit access requested

518 <1> ; BIT30 set if 16 bit access requested

519 <1> ; otherwise defaults to 8bit read

520 <1> ; DL/DX/EDX data to write depending on size

521 <1> ;

522 <1> ; Note1: this routine is meant to be called via pciRegWrite8,

523 <1> ; pciRegWrite16 or pciRegWrite32 as detailed below.

524 <1> ;

525 <1> ; Note2: don't attempt to write 32bits of data from a non dword

526 <1> ; aligned reg number. Likewise, don't do 16 bit writes from

527 <1> ; non word aligned reg #

528 <1>

529 00011AD4 53 <1> push ebx

530 00011AD5 51 <1> push ecx

531 00011AD6 89C3 <1> mov ebx, eax ; save eax, edx

532 00011AD8 89D1 <1> mov ecx, edx

533 00011ADA 25FFFFFF3F <1> and eax, NOT\_PCI32\_PCI16 ; clear out data size request

534 00011ADF 0D00000080 <1> or eax, BIT31 ; make a PCI access request

535 00011AE4 24FC <1> and al, ~3 ; NOT 3 ; force index to be dword

536 <1>

537 00011AE6 66BAF80C <1> mov dx, PCI\_INDEX\_PORT

538 00011AEA EF <1> out dx, eax ; write PCI selector

539 <1>

540 00011AEB 66BAFC0C <1> mov dx, PCI\_DATA\_PORT

541 00011AEF 88D8 <1> mov al, bl

542 00011AF1 2403 <1> and al, 3 ; figure out which port to

543 00011AF3 00C2 <1> add dl, al ; write to

544 <1>

545 00011AF5 F7C3000000C0 <1> test ebx, PCI32+PCI16

546 00011AFB 7505 <1> jnz short \_pregw0

547 00011AFD 88C8 <1> mov al, cl ; put data into al

548 00011AFF EE <1> out dx, al

549 00011B00 EB12 <1> jmp short \_pregw2

550 <1> \_pregw0:

551 00011B02 F7C300000080 <1> test ebx, PCI32

552 00011B08 7507 <1> jnz short \_pregw1

553 00011B0A 6689C8 <1> mov ax, cx ; put data into ax

554 00011B0D 66EF <1> out dx, ax

555 00011B0F EB03 <1> jmp short \_pregw2

556 <1> \_pregw1:

557 00011B11 89C8 <1> mov eax, ecx ; put data into eax

558 00011B13 EF <1> out dx, eax

559 <1> \_pregw2:

560 00011B14 89D8 <1> mov eax, ebx ; restore eax

561 00011B16 25FFFFFF3F <1> and eax, NOT\_PCI32\_PCI16 ; clear out data size request

562 00011B1B 89CA <1> mov edx, ecx ; restore dx

563 00011B1D 59 <1> pop ecx

564 00011B1E 5B <1> pop ebx

565 00011B1F C3 <1> retn

566 <1>

567 <1> pciRegWrite8:

568 00011B20 25FFFFFF3F <1> and eax, NOT\_PCI32\_PCI16 ; set up 8 bit write size

569 00011B25 EBAD <1> jmp short pciRegWrite ; call generic PCI access

570 <1>

571 <1> pciRegWrite16:

572 00011B27 25FFFFFF3F <1> and eax, NOT\_PCI32\_PCI16 ; set up 16 bit write size

573 00011B2C 0D00000040 <1> or eax, PCI16 ; call generic PCI access

574 00011B31 EBA1 <1> jmp short pciRegWrite

575 <1>

576 <1> pciRegWrite32:

577 00011B33 25FFFFFF3F <1> and eax, NOT\_PCI32\_PCI16 ; set up 32 bit write size

578 00011B38 0D00000080 <1> or eax, PCI32 ; call generic PCI access

579 00011B3D EB95 <1> jmp pciRegWrite

580 <1>

581 <1> init\_codec:

582 <1> ; 05/06/2017

583 <1> ; 28/05/2017 - Erdogan Tan (Ref: KolibriOS, vt823x.asm)

584 <1> ;

585 00011B3F A1[C46B0100] <1> mov eax, [audio\_dev\_id]

586 00011B44 B041 <1> mov al, VIA\_ACLINK\_CTRL

587 00011B46 E86AFFFFFF <1> call pciRegRead8

588 <1> ; ?

589 00011B4B B040 <1> mov al, VIA\_ACLINK\_STAT

590 00011B4D E863FFFFFF <1> call pciRegRead8

591 00011B52 F6C201 <1> test dl, VIA\_ACLINK\_C00\_READY

592 00011B55 7508 <1> jnz short \_codec\_ready\_1

593 00011B57 E80E000000 <1> call reset\_codec

594 00011B5C 7306 <1> jnc short \_codec\_ready\_2 ; eax = 1

595 00011B5E C3 <1> retn

596 <1> \_codec\_ready\_1:

597 00011B5F B801000000 <1> mov eax, 1

598 <1> \_codec\_ready\_2:

599 00011B64 E87A000000 <1> call codec\_io\_w16

600 <1> detect\_codec:

601 00011B69 C3 <1> retn

602 <1>

603 <1> reset\_codec:

604 <1> ; 16/04/2017

605 <1> ; 23/03/2017

606 <1> ; ('codec.asm')

607 <1> ; 12/11/2016 - Erdogan Tan (Ref: KolibriOS, vt823x.asm)

608 00011B6A A1[C46B0100] <1> mov eax, [audio\_dev\_id]

609 00011B6F B041 <1> mov al, VIA\_ACLINK\_CTRL

610 00011B71 B2E0 <1> mov dl, VIA\_ACLINK\_CTRL\_ENABLE + VIA\_ACLINK\_CTRL\_RESET + VIA\_ACLINK\_CTRL\_SYNC

611 00011B73 E8A8FFFFFF <1> call pciRegWrite8

612 <1>

613 00011B78 E83D000000 <1> call delay\_100ms ; wait 100 ms

614 <1> \_rc\_cold:

615 00011B7D E808000000 <1> call cold\_reset

616 00011B82 7301 <1> jnc short \_reset\_codec\_ok

617 <1>

618 <1> ; 16/04/2017

619 <1> ;xor eax, eax ; timeout error

620 <1> ;stc

621 00011B84 C3 <1> retn

622 <1>

623 <1> \_reset\_codec\_ok:

624 00011B85 31C0 <1> xor eax, eax

625 <1> ;mov al, VIA\_ACLINK\_C00\_READY ; 1

626 00011B87 FEC0 <1> inc al

627 00011B89 C3 <1> retn

628 <1>

629 <1> cold\_reset:

630 <1> ; 16/04/2017

631 <1> ; 23/03/2017

632 <1> ; ('codec.asm')

633 <1> ; 12/11/2016 - Erdogan Tan (Ref: KolibriOS, vt823x.asm)

634 <1> ;mov eax, [audio\_dev\_id]

635 <1> ;mov al, VIA\_ACLINK\_CTRL

636 00011B8A 30D2 <1> xor dl, dl ; 0

637 00011B8C E88FFFFFFF <1> call pciRegWrite8

638 <1>

639 00011B91 E824000000 <1> call delay\_100ms ; wait 100 ms

640 <1>

641 <1> ;; ACLink on, deassert ACLink reset, VSR, SGD data out

642 <1> ;; note - FM data out has trouble with non VRA codecs !!

643 <1>

644 <1> ;mov eax, [audio\_dev\_id]

645 <1> ;mov al, VIA\_ACLINK\_CTRL

646 00011B96 B2CC <1> mov dl, VIA\_ACLINK\_CTRL\_INIT

647 00011B98 E883FFFFFF <1> call pciRegWrite8

648 <1>

649 00011B9D B910000000 <1> mov ecx, 16 ; total 2s

650 <1>

651 <1> \_crst\_wait:

652 <1> ;mov eax, [audio\_dev\_id]

653 00011BA2 B040 <1> mov al, VIA\_ACLINK\_STAT

654 00011BA4 E80CFFFFFF <1> call pciRegRead8

655 <1>

656 00011BA9 F6C201 <1> test dl, VIA\_ACLINK\_C00\_READY

657 00011BAC 750B <1> jnz short \_crst\_ok

658 <1>

659 00011BAE 51 <1> push ecx

660 00011BAF E806000000 <1> call delay\_100ms

661 00011BB4 59 <1> pop ecx

662 <1>

663 00011BB5 49 <1> dec ecx

664 00011BB6 75EA <1> jnz short \_crst\_wait

665 <1>

666 <1> \_crst\_fail:

667 00011BB8 F9 <1> stc

668 <1> \_crst\_ok:

669 00011BB9 C3 <1> retn

670 <1>

671 <1> delay\_100ms:

672 <1> ; 29/05/2017

673 <1> ; 24/03/2017 ('codec.asm')

674 <1> ; wait 100 ms

675 00011BBA B990010000 <1> mov ecx, 400 ; 400\*0.25ms

676 <1> \_delay\_x\_ms:

677 00011BBF E803000000 <1> call delay1\_4ms

678 00011BC4 E2F9 <1> loop \_delay\_x\_ms

679 00011BC6 C3 <1> retn

680 <1>

681 <1> ; delay1\_4ms - Delay for 1/4 millisecond.

682 <1> ; 1mS = 1000us

683 <1> ; Entry:

684 <1> ; None

685 <1> ; Exit:

686 <1> ; None

687 <1> ;

688 <1> ; Modified:

689 <1> ; None

690 <1> ;

691 <1>

692 <1> ; 29/05/2017

693 <1> ; 23/04/2017

694 <1> ; 05/03/2017 (TRDOS 386)

695 <1> ; ('UTILS.ASM')

696 <1> delay1\_4ms:

697 00011BC7 50 <1> push eax

698 00011BC8 51 <1> push ecx

699 00011BC9 B110 <1> mov cl, 16 ; close enough.

700 <1>

701 00011BCB E461 <1> in al, PORTB ; 61h

702 <1>

703 00011BCD 2410 <1> and al, REFRESH\_STATUS ; 10h

704 00011BCF 88C5 <1> mov ch, al ; Start toggle state

705 <1> \_d4ms1:

706 00011BD1 E461 <1> in al, PORTB ; Read system control port

707 <1>

708 00011BD3 2410 <1> and al, REFRESH\_STATUS ; Refresh toggles 15.085 microseconds

709 00011BD5 38C5 <1> cmp ch, al

710 00011BD7 74F8 <1> je short \_d4ms1 ; Wait for state change

711 <1>

712 00011BD9 88C5 <1> mov ch, al ; Update with new state

713 00011BDB FEC9 <1> dec cl

714 00011BDD 75F2 <1> jnz short \_d4ms1

715 <1>

716 00011BDF F8 <1> clc ; 29/05/2017

717 <1>

718 00011BE0 59 <1> pop ecx

719 00011BE1 58 <1> pop eax

720 00011BE2 C3 <1> retn

721 <1>

722 <1> ; 10/04/2017 (TRDOS 386)

723 <1> ; 12/11/2016

724 <1>

725 <1> codec\_io\_w16: ;w32

726 <1> ; ('codec.asm')

727 00011BE3 668B15[C26B0100] <1> mov dx, [audio\_io\_base]

728 00011BEA 6681C28000 <1> add dx, VIA\_REG\_AC97

729 00011BEF EF <1> out dx, eax

730 00011BF0 C3 <1> retn

731 <1>

732 <1> codec\_io\_r16: ;r32

733 <1> ; ('codec.asm')

734 00011BF1 668B15[C26B0100] <1> mov dx, [audio\_io\_base]

735 00011BF8 6681C28000 <1> add dx, VIA\_REG\_AC97

736 00011BFD ED <1> in eax, dx

737 00011BFE C3 <1> retn

738 <1>

739 <1> ctrl\_io\_w8:

740 <1> ; ('codec.asm')

741 00011BFF 660315[C26B0100] <1> add dx, [audio\_io\_base]

742 00011C06 EE <1> out dx, al

743 00011C07 C3 <1> retn

744 <1>

745 <1> ctrl\_io\_r8:

746 <1> ; ('codec.asm')

747 00011C08 660315[C26B0100] <1> add dx, [audio\_io\_base]

748 00011C0F EC <1> in al, dx

749 00011C10 C3 <1> retn

750 <1>

751 <1> ctrl\_io\_w32:

752 <1> ; ('codec.asm')

753 00011C11 660315[C26B0100] <1> add dx, [audio\_io\_base]

754 00011C18 EF <1> out dx, eax

755 00011C19 C3 <1> retn

756 <1>

757 <1> ctrl\_io\_r32:

758 <1> ; ('codec.asm')

759 00011C1A 660315[C26B0100] <1> add dx, [audio\_io\_base]

760 00011C21 ED <1> in eax, dx

761 00011C22 C3 <1> retn

762 <1>

763 <1> codec\_read:

764 <1> ; 12/11/2016 - Erdogan Tan (Ref: KolibriOS, vt823x.asm)

765 <1> ; Use only primary codec.

766 <1> ; eax = register

767 00011C23 C1E010 <1> shl eax, VIA\_REG\_AC97\_CMD\_SHIFT

768 00011C26 0D00008002 <1> or eax, VIA\_REG\_AC97\_PRIMARY\_VALID + VIA\_REG\_AC97\_READ

769 <1>

770 00011C2B E8B3FFFFFF <1> call codec\_io\_w16

771 <1>

772 <1> ; codec\_valid

773 00011C30 E831000000 <1> call codec\_check\_ready

774 00011C35 7301 <1> jnc short \_cr\_ok

775 <1>

776 00011C37 C3 <1> retn

777 <1>

778 <1> \_cr\_ok:

779 <1> ; wait 25 ms

780 00011C38 B950000000 <1> mov ecx, 80 ; (100\*0.25 ms)

781 <1> \_cr\_wloop:

782 00011C3D E885FFFFFF <1> call delay1\_4ms

783 00011C42 E2F9 <1> loop \_cr\_wloop

784 <1>

785 00011C44 E8A8FFFFFF <1> call codec\_io\_r16

786 00011C49 25FFFF0000 <1> and eax, 0FFFFh

787 00011C4E C3 <1> retn

788 <1>

789 <1> codec\_write:

790 <1> ; 12/11/2016 - Erdogan Tan (Ref: KolibriOS, vt823x.asm)

791 <1> ; Use only primary codec.

792 <1>

793 <1> ; eax = data (volume)

794 <1> ; edx = register (mixer register)

795 <1>

796 00011C4F C1E210 <1> shl edx, VIA\_REG\_AC97\_CMD\_SHIFT

797 <1>

798 00011C52 C1E000 <1> shl eax, VIA\_REG\_AC97\_DATA\_SHIFT ; shl eax, 0

799 00011C55 09C2 <1> or edx, eax

800 <1>

801 00011C57 B800000000 <1> mov eax, VIA\_REG\_AC97\_CODEC\_ID\_PRIMARY

802 00011C5C C1E01E <1> shl eax, VIA\_REG\_AC97\_CODEC\_ID\_SHIFT

803 00011C5F 09D0 <1> or eax, edx

804 <1>

805 00011C61 E87DFFFFFF <1> call codec\_io\_w16

806 <1> ;mov [codec.regs+esi], ax

807 <1>

808 <1> ;call codec\_check\_ready

809 <1> ;retn

810 <1> ;jmp short \_codec\_check\_ready

811 <1>

812 <1> codec\_check\_ready:

813 <1> ; 12/11/2016 - Erdogan Tan (Ref: KolibriOS, vt823x.asm)

814 <1>

815 <1> \_codec\_check\_ready:

816 00011C66 B914000000 <1> mov ecx, 20 ; total 2s

817 <1> \_ccr\_wait:

818 00011C6B 51 <1> push ecx

819 <1>

820 00011C6C E880FFFFFF <1> call codec\_io\_r16

821 00011C71 A900000001 <1> test eax, VIA\_REG\_AC97\_BUSY

822 00011C76 740B <1> jz short \_ccr\_ok

823 <1>

824 00011C78 E83DFFFFFF <1> call delay\_100ms

825 <1>

826 00011C7D 59 <1> pop ecx

827 <1>

828 00011C7E 49 <1> dec ecx

829 00011C7F 75EA <1> jnz short \_ccr\_wait

830 <1>

831 00011C81 F9 <1> stc

832 00011C82 C3 <1> retn

833 <1>

834 <1> \_ccr\_ok:

835 00011C83 59 <1> pop ecx

836 00011C84 25FFFF0000 <1> and eax, 0FFFFh

837 00011C89 C3 <1> retn

838 <1>

839 <1> codec\_config:

840 <1> ; 10/06/2017

841 <1> ; 29/05/2017

842 <1> ; 24/04/2017

843 <1> ; 21/04/2017

844 <1> ; 16/04/2017 (TRDOS 386 Kernel)

845 <1> ; 15/11/2016 ('codec.asm', 'player.com')

846 <1> ; 14/11/2016

847 <1> ; 12/11/2016 - Erdogan Tan

848 <1> ; (Ref: KolibriOS, 'setup\_codec', codec.inc)

849 <1>

850 00011C8A B802020000 <1> mov eax, 0202h

851 00011C8F 66A3[F26B0100] <1> mov [audio\_master\_volume], ax

852 00011C95 66B81F1F <1> mov ax, 1F1Fh ; 31,31

853 00011C99 BA02000000 <1> mov edx, CODEC\_MASTER\_VOL\_REG ; 02h ; Line Out

854 00011C9E E8ACFFFFFF <1> call codec\_write

855 <1> ;jc short cconfig\_error

856 <1>

857 <1> ;mov eax, 0202h

858 00011CA3 66B80202 <1> mov ax, 0202h

859 00011CA7 BA18000000 <1> mov edx, CODEC\_PCM\_OUT\_REG ; 18h ; Wave Output (Stereo)

860 00011CAC E89EFFFFFF <1> call codec\_write

861 <1> ;jc short cconfig\_error

862 <1>

863 <1> ;mov eax, 0202h

864 00011CB1 66B80202 <1> mov ax, 0202h

865 00011CB5 BA04000000 <1> mov edx, CODEC\_AUX\_VOL ; 04h ; CODEC\_HP\_VOL\_REG ; HeadPhone

866 00011CBA E890FFFFFF <1> call codec\_write

867 <1> ;jc short cconfig\_error

868 <1>

869 <1> ;mov eax, 08h

870 <1> ;mov ax, 08h

871 00011CBF 66B80880 <1> mov ax, 8008h ; Mute

872 00011CC3 BA0C000000 <1> mov edx, 0Ch ; AC97\_PHONE\_VOL ; TAD Input (Mono)

873 00011CC8 E882FFFFFF <1> call codec\_write

874 <1> ;jc short cconfig\_error

875 <1>

876 <1> ;mov eax, 0808h

877 00011CCD 66B80808 <1> mov ax, 0808h

878 00011CD1 BA10000000 <1> mov edx, CODEC\_LINE\_IN\_VOL\_REG ; 10h ; Line Input (Stereo)

879 00011CD6 E874FFFFFF <1> call codec\_write

880 <1> ;jc short cconfig\_error

881 <1>

882 <1> ;mov eax, 0808h

883 00011CDB 66B80808 <1> mov ax, 0808h

884 00011CDF BA12000000 <1> mov edx, CODEC\_CD\_VOL\_REG ; 12h ; CR Input (Stereo)

885 00011CE4 E866FFFFFF <1> call codec\_write

886 <1> ;jc short cconfig\_error

887 <1>

888 <1> ;mov eax, 0808h

889 00011CE9 66B80808 <1> mov ax, 0808h

890 00011CED BA16000000 <1> mov edx, CODEC\_AUX\_VOL\_REG ; 16h ; Aux Input (Stereo)

891 <1> ;call codec\_write

892 <1> ;;jc short cconfig\_error

893 00011CF2 E958FFFFFF <1> jmp codec\_write ; 10/06/2017

894 <1>

895 <1> ; ; Extended Audio Status (2Ah)

896 <1> ; mov eax, CODEC\_EXT\_AUDIO\_CTRL\_REG ; 2Ah

897 <1> ; call codec\_read

898 <1> ; and eax, 0FFFFh - 2 ; clear DRA (BIT1)

899 <1> ; ;or eax, 1 ; set VRA (BIT0)

900 <1> ; or eax, 5 ; VRA (BIT0) & S/PDIF (BIT2) ; 14/11/2016

901 <1> ; mov edx, CODEC\_EXT\_AUDIO\_CTRL\_REG

902 <1> ; call codec\_write

903 <1> ; ;jc short cconfig\_error

904 <1> ;

905 <1> ;set\_sample\_rate:

906 <1> ; ;movzx eax, word [audio\_freq]

907 <1> ; mov ax, [audio\_freq]

908 <1> ; mov edx, CODEC\_PCM\_FRONT\_DACRATE\_REG ; 2Ch ; PCM Front DAC Rate

909 <1> ; ;call codec\_write

910 <1> ; ;retn

911 <1> ; jmp codec\_write

912 <1>

913 <1> ;cconfig\_error:

914 <1> ; retn

915 <1>

916 <1> vt8233\_int\_handler:

917 <1> ; Interrupt Handler for VIA VT8237R Audio Controller

918 <1> ; Note: called by 'dev\_IRQ\_service'

919 <1> ; 14/10/2017

920 <1> ; 09/10/2017, 10/10/2017, 12/10/2017

921 <1> ; 13/06/2017

922 <1> ; 21/04/2017 (TRDOS 386 kernel, 'audio.s')

923 <1> ; 24/03/2017 - 'PLAYER.COM' ('player.asm')

924 <1>

925 <1> ;push eax ; \* must be saved !

926 <1> ;push edx

927 <1> ;push ecx

928 <1> ;push ebx ; \* must be saved !

929 <1> ;push esi

930 <1> ;push edi

931 <1>

932 <1> ;cmp byte [audio\_busy], 1

933 <1> ;jnb short \_ih0 ; 09/10/2017

934 <1>

935 <1> ;mov byte [audio\_flag\_eol], 0

936 <1>

937 00011CF7 66BA0000 <1> mov dx, VIADEV\_PLAYBACK + VIA\_REG\_OFFSET\_STATUS

938 00011CFB E808FFFFFF <1> call ctrl\_io\_r8

939 <1>

940 00011D00 A880 <1> test al, VIA\_REG\_STAT\_ACTIVE

941 00011D02 7417 <1> jz short \_ih0 ; 09/10/2017

942 <1>

943 00011D04 2407 <1> and al, VIA\_REG\_STAT\_EOL + VIA\_REG\_STAT\_FLAG + VIA\_REG\_STAT\_STOPPED

944 00011D06 A2[F16B0100] <1> mov [audio\_flag\_eol], al

945 00011D0B 740E <1> jz short \_ih0 ; 09/10/2017

946 <1>

947 <1> ; 09/10/2017

948 <1> ;mov byte [audio\_busy], 1

949 <1>

950 00011D0D 803D[F06B0100]01 <1> cmp byte [audio\_play\_cmd], 1

951 00011D14 7315 <1> jnb short \_ih1 ; 10/10/2017

952 <1>

953 00011D16 E860000000 <1> call channel\_reset

954 <1> \_ih0:

955 <1> ; 09/10/2017

956 00011D1B A0[F16B0100] <1> mov al, [audio\_flag\_eol] ;; ack ;;

957 00011D20 66BA0000 <1> mov dx, VIADEV\_PLAYBACK + VIA\_REG\_OFFSET\_STATUS

958 00011D24 E8D6FEFFFF <1> call ctrl\_io\_w8

959 00011D29 EB4F <1> jmp short \_ih4

960 <1> \_ih1:

961 <1> vt8233\_tuneLoop:

962 00011D2B A0[F16B0100] <1> mov al, [audio\_flag\_eol] ;; ack ;;

963 00011D30 66BA0000 <1> mov dx, VIADEV\_PLAYBACK + VIA\_REG\_OFFSET\_STATUS

964 00011D34 E8C6FEFFFF <1> call ctrl\_io\_w8

965 <1>

966 <1> ; 12/10/2017

967 00011D39 C605[E46B0100]00 <1> mov byte [audio\_flag], 0 ; Reset

968 <1>

969 <1> ; 10/10/2017

970 <1> ; 09/10/2017

971 <1> ;test byte [audio\_flag\_eol], VIA\_REG\_STAT\_FLAG

972 <1> ;jz short \_ih2 ; EOL

973 <1>

974 <1> ; 14/10/2017

975 00011D40 F605[F16B0100]02 <1> test byte [audio\_flag\_eol], VIA\_REG\_STAT\_EOL

976 00011D47 7506 <1> jnz short \_ih2 ; EOL

977 <1> ; (Half Buffer 2 has been completed

978 <1> ; and Half Buffer 1 will be played.)

979 <1> ; FLAG

980 <1> ; (Half Buffer 1 has been completed

981 <1> ; and Half Buffer 2 will be played.)

982 <1>

983 <1> ; 14/10/2017

984 <1> ;; (Continue to play.)

985 <1> ;mov al, VIA\_REG\_CTRL\_INT

986 <1> ;or al, VIA\_REG\_CTRL\_START

987 <1> ;mov dx, VIADEV\_PLAYBACK + VIA\_REG\_OFFSET\_CONTROL

988 <1> ;call ctrl\_io\_w8

989 <1> ; 12/10/2017

990 <1> ;mov byte [audio\_flag], 1

991 00011D49 FE05[E46B0100] <1> inc byte [audio\_flag] ; = 1

992 <1> \_ih2:

993 <1> ; 10/10/2017

994 00011D4F 8B3D[DC6B0100] <1> mov edi, [audio\_dma\_buff]

995 00011D55 8B0D[E06B0100] <1> mov ecx, [audio\_dmabuff\_size]

996 00011D5B D1E9 <1> shr ecx, 1 ; dma buff size / 2 = half buffer size

997 <1>

998 <1> ; 12/10/2017

999 00011D5D 803D[E46B0100]00 <1> cmp byte [audio\_flag], 0

1000 00011D64 7702 <1> ja short \_ih3 ; Playing Half Buffer 2 (Current: FLAG)

1001 <1> ; Playing Half Buffer 1 (Current: EOL)

1002 00011D66 01CF <1> add edi, ecx

1003 <1> \_ih3:

1004 <1> ; Update half buffer 2 while playing half buffer 1 (FLAG)

1005 <1> ; Update half buffer 1 while playing half buffer 2 (EOL)

1006 <1>

1007 00011D68 8B35[D46B0100] <1> mov esi, [audio\_p\_buffer] ; phy addr of audio buff

1008 00011D6E C1E902 <1> shr ecx, 2 ; half buff size / 4

1009 00011D71 F3A5 <1> rep movsd

1010 <1> ; switch flag value ;

1011 00011D73 8035[E46B0100]01 <1> xor byte [audio\_flag], 1 ; 10/10/2017

1012 <1> ; 12/10/2017

1013 <1> ; [audio\_flag] = 0 : Playing dma half buffer 2 (just after FLAG)

1014 <1> ; Next buffer (to update) is dma half buff 1

1015 <1> ; = 1 : Playing dma half buffer 1 (just after EOL)

1016 <1> ; Next buffer (to update) is dma half buff 2

1017 <1> \_ih4:

1018 <1> ; 28/05/2017

1019 <1> ;mov byte [audio\_busy], 0 ; 09/10/2017

1020 <1> ;

1021 <1> ;pop edi

1022 <1> ;pop esi

1023 <1> ;pop ebx ; \* must be restored !

1024 <1> ;pop ecx

1025 <1> ;pop edx

1026 <1> ;pop eax ; \* must be restored !

1027 <1>

1028 00011D7A C3 <1> retn

1029 <1>

1030 <1> channel\_reset:

1031 <1> ; 24/06/2017

1032 <1> ; 29/05/2017

1033 <1> ; 23/03/2017

1034 <1> ; 14/11/2016 - Erdogan Tan

1035 <1> ; 12/11/2016 - Erdogan Tan (Ref: KolibriOS, vt823x.asm)

1036 00011D7B BA01000000 <1> mov edx, VIA\_REG\_OFFSET\_CONTROL

1037 <1> ;mov eax, VIA\_REG\_CTRL\_PAUSE + VIA\_REG\_CTRL\_TERMINATE + VIA\_REG\_CTRL\_RESET

1038 00011D80 B848000000 <1> mov eax, VIA\_REG\_CTRL\_PAUSE + VIA\_REG\_CTRL\_TERMINATE ; 24/06/2017

1039 00011D85 E875FEFFFF <1> call ctrl\_io\_w8

1040 <1>

1041 <1> ;mov edx, VIA\_REG\_OFFSET\_CONTROL

1042 <1> ;call ctrl\_io\_r8

1043 <1>

1044 <1> ; wait for 50 ms

1045 00011D8A B9A0000000 <1> mov ecx, 160 ; (200\*0.25 ms) ; 29/05/2017

1046 <1> \_ch\_rst\_wait:

1047 00011D8F E833FEFFFF <1> call delay1\_4ms

1048 00011D94 49 <1> dec ecx

1049 00011D95 75F8 <1> jnz short \_ch\_rst\_wait

1050 <1>

1051 <1> ; disable interrupts

1052 00011D97 BA01000000 <1> mov edx, VIA\_REG\_OFFSET\_CONTROL

1053 00011D9C 31C0 <1> xor eax, eax

1054 00011D9E E85CFEFFFF <1> call ctrl\_io\_w8

1055 <1>

1056 <1> ; clear interrupts

1057 00011DA3 BA00000000 <1> mov edx, VIA\_REG\_OFFSET\_STATUS

1058 00011DA8 B803000000 <1> mov eax, 3

1059 00011DAD E84DFEFFFF <1> call ctrl\_io\_w8

1060 <1>

1061 <1> ;mov edx, VIA\_REG\_OFFSET\_CURR\_PTR

1062 <1> ;xor eax, eax

1063 <1> ;call ctrl\_io\_w32

1064 <1>

1065 00011DB2 C3 <1> retn

1066 <1>

1067 <1> vt8233\_stop: ; 22/04/2017

1068 00011DB3 C605[F06B0100]00 <1> mov byte [audio\_play\_cmd], 0 ; stop !

1069 <1> \_tlp2:

1070 <1> ; 24/06/2017

1071 <1> ; finished with song, stop everything

1072 <1> ;mov al, VIA\_REG\_CTRL\_INT

1073 <1> ;or al, VIA\_REG\_CTRL\_TERMINATE

1074 <1> ;mov dx, VIADEV\_PLAYBACK + VIA\_REG\_OFFSET\_CONTROL

1075 <1> ;call ctrl\_io\_w8

1076 <1>

1077 <1> ;call channel\_reset

1078 <1> ;retn

1079 00011DBA EBBF <1> jmp short channel\_reset

1080 <1>

1081 <1> set\_vt8233\_bdl: ; Set VT8237R Buffer Descriptor List

1082 <1> ; 28/05/2017

1083 <1> ; 21/04/2017 (TRDOS 386 kernel, 'audio.s')

1084 <1> ; 24/03/2017 - 'PLAYER.COM' ('via\_wav.asm' - 29/11/2016)

1085 <1>

1086 <1> ; eax = dma buffer address = [audio\_DMA\_buff]

1087 <1> ; ecx = dma buffer buffer size = [audio\_dmabuff\_size]

1088 <1>

1089 00011DBC D1E9 <1> shr ecx, 1 ; dma half buffer size

1090 00011DBE 89CE <1> mov esi, ecx

1091 <1>

1092 00011DC0 BF[F86B0100] <1> mov edi, audio\_bdl\_buff ; get BDL address

1093 00011DC5 B910000000 <1> mov ecx, 32 / 2 ; make 32 entries in BDL

1094 <1>

1095 00011DCA EB05 <1> jmp short s\_vt8233\_bdl1

1096 <1>

1097 <1> s\_vt8233\_bdl0:

1098 <1> ; set buffer descriptor 0 to start of data file in memory

1099 <1>

1100 00011DCC A1[DC6B0100] <1> mov eax, [audio\_dma\_buff] ; Physical address of DMA buffer

1101 <1>

1102 <1> s\_vt8233\_bdl1:

1103 00011DD1 AB <1> stosd ; store dmabuffer1 address

1104 <1>

1105 00011DD2 89C2 <1> mov edx, eax

1106 <1>

1107 <1> ; VIA VT8235.PDF: (Page 110) (Erdogan Tan, 29/11/2016)

1108 <1> ;

1109 <1> ; Audio SGD Table Format

1110 <1> ; -------------------------------

1111 <1> ; 63 62 61-56 55-32 31-0

1112 <1> ; -- -- -------- ----- ----

1113 <1> ; EOL FLAG -reserved- Base Base

1114 <1> ; Count Address

1115 <1> ; [23:0] [31:0]

1116 <1> ; EOL: End Of Link.

1117 <1> ; 1 indicates this block is the last of the link.

1118 <1> ; If the channel “Interrupt on EOL” bit is set, then

1119 <1> ; an interrupt is generated at the end of the transfer.

1120 <1> ;

1121 <1> ; FLAG: Block Flag. If set, transfer pauses at the end of this

1122 <1> ; block. If the channel “Interrupt on FLAG” bit is set,

1123 <1> ; then an interrupt is generated at the end of this block.

1124 <1>

1125 00011DD4 89F0 <1> mov eax, esi ; DMA half buffer size

1126 00011DD6 01C2 <1> add edx, eax

1127 00011DD8 0D00000040 <1> or eax, FLAG

1128 <1> ;or eax, EOL

1129 00011DDD AB <1> stosd

1130 <1>

1131 <1> ; 2nd buffer:

1132 <1>

1133 00011DDE 89D0 <1> mov eax, edx ; Physical address of the 2nd half of DMA buffer

1134 00011DE0 AB <1> stosd ; store dmabuffer2 address

1135 <1>

1136 <1> ; set length to [audio\_dmabuff\_size]/2

1137 <1> ; Set control (bits 31:16) to BUP, bits 15:0=number of samples

1138 <1> ;

1139 00011DE1 89F0 <1> mov eax, esi ; DMA half buffer size

1140 00011DE3 0D00000080 <1> or eax, EOL

1141 <1> ;or eax, FLAG

1142 00011DE8 AB <1> stosd

1143 <1>

1144 00011DE9 E2E1 <1> loop s\_vt8233\_bdl0

1145 <1>

1146 00011DEB C3 <1> retn

1147 <1>

1148 <1> vt8233\_start\_play:

1149 <1> ; start to play audio data via VT8233 audio controller

1150 <1> ; 13/06/2017

1151 <1> ; 10/06/2017

1152 <1> ; 24/04/2017

1153 <1> ; 21/04/2017 (TRDOS 386 kernel, 'audio.s')

1154 <1> ; 24/03/2017 - 'PLAYER.COM' ('via\_wav.asm' - 29/11/2016)

1155 <1> ; write buffer descriptor list address

1156 <1> ;

1157 <1>

1158 <1> ; Extended Audio Status (2Ah)

1159 00011DEC B82A000000 <1> mov eax, CODEC\_EXT\_AUDIO\_CTRL\_REG ; 2Ah

1160 00011DF1 E82DFEFFFF <1> call codec\_read

1161 00011DF6 25FDFF0000 <1> and eax, 0FFFFh - 2 ; clear DRA (BIT1)

1162 <1> ;or eax, 1 ; set VRA (BIT0)

1163 00011DFB 83C805 <1> or eax, 5 ; VRA (BIT0) & S/PDIF (BIT2) ; 14/11/2016

1164 00011DFE BA2A000000 <1> mov edx, CODEC\_EXT\_AUDIO\_CTRL\_REG

1165 00011E03 E847FEFFFF <1> call codec\_write

1166 <1> ;jc short cconfig\_error

1167 <1>

1168 <1> set\_sample\_rate:

1169 <1> ;movzx eax, word [audio\_freq]

1170 00011E08 66A1[EE6B0100] <1> mov ax, [audio\_freq]

1171 00011E0E BA2C000000 <1> mov edx, CODEC\_PCM\_FRONT\_DACRATE\_REG ; 2Ch ; PCM Front DAC Rate

1172 00011E13 E837FEFFFF <1> call codec\_write

1173 <1>

1174 00011E18 B8[F86B0100] <1> mov eax, audio\_bdl\_buff

1175 <1>

1176 <1> ; 12/11/2016 - Erdogan Tan

1177 <1> ; (Ref: KolibriOS, vt823x.asm, 'create\_primary\_buff')

1178 00011E1D BA04000000 <1> mov edx, VIADEV\_PLAYBACK + VIA\_REG\_OFFSET\_TABLE\_PTR

1179 00011E22 E8EAFDFFFF <1> call ctrl\_io\_w32

1180 <1>

1181 <1> ;call codec\_check\_ready

1182 <1>

1183 00011E27 66BA0200 <1> mov dx, VIADEV\_PLAYBACK + VIA\_REG\_OFS\_PLAYBACK\_VOLUME\_L

1184 <1> ;mov eax, 2 ; 31

1185 00011E2B B01F <1> mov al, 31

1186 00011E2D 2A05[F26B0100] <1> sub al, [audio\_master\_volume\_l]

1187 00011E33 E8C7FDFFFF <1> call ctrl\_io\_w8

1188 <1>

1189 <1> ;call codec\_check\_ready

1190 <1>

1191 00011E38 66BA0300 <1> mov dx, VIADEV\_PLAYBACK + VIA\_REG\_OFS\_PLAYBACK\_VOLUME\_R

1192 <1> ;mov ax, 2 ; 31

1193 00011E3C B01F <1> mov al, 31

1194 00011E3E 2A05[F36B0100] <1> sub al, [audio\_master\_volume\_r]

1195 00011E44 E8B6FDFFFF <1> call ctrl\_io\_w8

1196 <1>

1197 <1> ;call codec\_check\_ready

1198 <1> ;

1199 <1> ;

1200 <1> ; All set. Let's play some music.

1201 <1> ;

1202 <1> ;

1203 <1> ;mov dx, VIADEV\_PLAYBACK + VIA\_REG\_OFFSET\_STOP\_IDX

1204 <1> ;mov ax, VIA8233\_REG\_TYPE\_16BIT or VIA8233\_REG\_TYPE\_STEREO or 0xfffff or 0xff000000

1205 <1> ;call ctrl\_io\_w32

1206 <1>

1207 <1> ;call codec\_check\_ready

1208 <1>

1209 <1> ; 08/12/2016

1210 <1> ; 07/10/2016

1211 <1> ;mov al, 1

1212 00011E49 B01F <1> mov al, 31

1213 00011E4B E815000000 <1> call set\_VT8233\_LastValidIndex

1214 <1>

1215 00011E50 C605[F06B0100]01 <1> mov byte [audio\_play\_cmd], 1 ; play command (do not stop) !

1216 <1>

1217 <1> vt8233\_play: ; continue to play

1218 <1> ; 22/04/2017

1219 00011E57 B023 <1> mov al, VIA\_REG\_CTRL\_INT

1220 00011E59 0C80 <1> or al, VIA\_REG\_CTRL\_START

1221 <1> ;mov al, VIA\_REG\_CTRL\_AUTOSTART + VIA\_REG\_CTRL\_START

1222 <1> ;mov al, VIA\_REG\_CTRL\_AUTOSTART + VIA\_REG\_CTRL\_START + VIA\_REG\_CTRL\_INT\_FLAG

1223 00011E5B 66BA0100 <1> mov dx, VIADEV\_PLAYBACK + VIA\_REG\_OFFSET\_CONTROL

1224 00011E5F E89BFDFFFF <1> call ctrl\_io\_w8

1225 <1> ;call codec\_check\_ready

1226 <1> ;retn

1227 <1> ;jmp codec\_check\_ready

1228 00011E64 C3 <1> retn

1229 <1>

1230 <1> ;input AL = index # to stop on

1231 <1> set\_VT8233\_LastValidIndex:

1232 <1> ; 10/06/2017

1233 <1> ; 21/04/2017 (TRDOS 386 kernel, 'audio.s')

1234 <1> ; 24/03/2017 - 'PLAYER.COM' ('via\_wav.asm' - 29/11/2016)

1235 <1> ; 19/11/2016

1236 <1> ; 14/11/2016 - Erdogan Tan (Ref: VIA VT8235.PDF, Page 110)

1237 <1> ; 12/11/2016 - Erdogan Tan

1238 <1> ; (Ref: KolibriOS, vt823x.asm, 'create\_primary\_buff')

1239 <1> ;push edx

1240 00011E65 6650 <1> push ax

1241 <1> ;push ecx

1242 00011E67 0FB705[EE6B0100] <1> movzx eax, word [audio\_freq] ; Hertz

1243 00011E6E BA00001000 <1> mov edx, 100000h ; 2^20 = 1048576

1244 00011E73 F7E2 <1> mul edx

1245 00011E75 B980BB0000 <1> mov ecx, 48000

1246 00011E7A F7F1 <1> div ecx

1247 <1> ;and eax, 0FFFFFh

1248 <1> ;pop ecx

1249 00011E7C 665A <1> pop dx

1250 00011E7E C1E218 <1> shl edx, 24 ; STOP Index Setting: Bit 24 to 31

1251 00011E81 09D0 <1> or eax, edx

1252 <1> ; 19/11/2016

1253 00011E83 803D[EC6B0100]10 <1> cmp byte [audio\_bps], 16

1254 00011E8A 7505 <1> jne short sLVI\_1

1255 00011E8C 0D00002000 <1> or eax, VIA8233\_REG\_TYPE\_16BIT

1256 <1> sLVI\_1:

1257 00011E91 803D[ED6B0100]02 <1> cmp byte [audio\_stmo], 2

1258 00011E98 7505 <1> jne short sLVI\_2

1259 00011E9A 0D00001000 <1> or eax, VIA8233\_REG\_TYPE\_STEREO

1260 <1> sLVI\_2:

1261 00011E9F BA08000000 <1> mov edx, VIADEV\_PLAYBACK + VIA\_REG\_OFFSET\_STOP\_IDX

1262 00011EA4 E868FDFFFF <1> call ctrl\_io\_w32

1263 <1> ;call codec\_check\_ready

1264 <1> ;pop edx

1265 00011EA9 C3 <1> retn

1266 <1>

1267 <1> vt8233\_pause: ; pause

1268 <1> ; 10/06/2017

1269 <1> ; 22/04/2017

1270 00011EAA B023 <1> mov al, VIA\_REG\_CTRL\_INT

1271 00011EAC 0C08 <1> or al, VIA\_REG\_CTRL\_PAUSE

1272 00011EAE 66BA0100 <1> mov dx, VIADEV\_PLAYBACK + VIA\_REG\_OFFSET\_CONTROL

1273 00011EB2 E848FDFFFF <1> call ctrl\_io\_w8

1274 <1> ;call codec\_check\_ready

1275 <1> ;retn

1276 <1> ;jmp codec\_check\_ready

1277 00011EB7 C3 <1> retn

1278 <1>

1279 <1> vt8233\_reset:

1280 <1> ; 22/04/2017

1281 <1> ; reset VT8237R (vt8233) Audio Controller

1282 <1> ;cmp byte [audio\_play\_cmd], 1

1283 <1> ;jna short vt8233\_rst\_0

1284 00011EB8 C605[F06B0100]00 <1> mov byte [audio\_play\_cmd], 0 ; stop !

1285 <1> vt8233\_rst\_0:

1286 00011EBF E8A6FCFFFF <1> call reset\_codec

1287 00011EC4 720A <1> jc short vt8233\_rst\_1 ; codec error !

1288 <1> ; eax = 1

1289 00011EC6 E818FDFFFF <1> call codec\_io\_w16 ; w32

1290 00011ECB E8ABFEFFFF <1> call channel\_reset

1291 <1> vt8233\_rst\_1:

1292 00011ED0 C3 <1> retn

1293 <1>

1294 <1> vt8233\_volume:

1295 <1> ; set VT8237R (vt8233) sound volume level

1296 <1> ; 24/04/2017

1297 <1> ; 22/04/2017

1298 <1> ; bl = component (0 = master/playback/lineout volume)

1299 <1> ; cl = left channel volume level (0 to 31)

1300 <1> ; ch = right channel volume level (0 to 31)

1301 <1>

1302 00011ED1 08DB <1> or bl, bl

1303 00011ED3 7520 <1> jnz short vt8233\_vol\_1 ; temporary !

1304 00011ED5 66B81F1F <1> mov ax, 1F1Fh ; 31,31

1305 00011ED9 38C1 <1> cmp cl, al

1306 00011EDB 7718 <1> ja short vt8233\_vol\_1 ; temporary !

1307 00011EDD 38E5 <1> cmp ch, ah

1308 00011EDF 7714 <1> ja short vt8233\_vol\_1 ; temporary !

1309 00011EE1 66890D[F26B0100] <1> mov [audio\_master\_volume], cx

1310 00011EE8 6629C8 <1> sub ax, cx

1311 00011EEB BA02000000 <1> mov edx, CODEC\_MASTER\_VOL\_REG ; 02h ; Line Out

1312 00011EF0 E85AFDFFFF <1> call codec\_write

1313 <1> vt8233\_vol\_1:

1314 00011EF5 C3 <1> retn

1315 <1>

1316 <1> ; CODE for SOUND BLASTER 16

1317 <1>

1318 <1> DetectSB:

1319 <1> ; 24/04/2017

1320 <1> ;pushad

1321 <1> ScanPort:

1322 00011EF6 66BB1002 <1> mov bx, 210h ; start scanning ports

1323 <1> ; 210h, 220h, .. 260h

1324 <1> ResetDSP:

1325 00011EFA 6689DA <1> mov dx, bx ; try to reset the DSP.

1326 00011EFD 6683C206 <1> add dx, 06h

1327 00011F01 B001 <1> mov al, 1

1328 00011F03 EE <1> out dx, al

1329 <1>

1330 00011F04 EC <1> in al, dx

1331 00011F05 EC <1> in al, dx

1332 00011F06 EC <1> in al, dx

1333 00011F07 EC <1> in al, dx

1334 <1>

1335 00011F08 30C0 <1> xor al, al

1336 00011F0A EE <1> out dx, al

1337 <1>

1338 00011F0B 6683C208 <1> add dx, 08h

1339 00011F0F 66B96400 <1> mov cx, 100

1340 <1> WaitID:

1341 00011F13 EC <1> in al, dx

1342 00011F14 08C0 <1> or al, al

1343 00011F16 7804 <1> js short GetID

1344 00011F18 E2F9 <1> loop WaitID

1345 00011F1A EB0F <1> jmp short NextPort

1346 <1> GetID:

1347 00011F1C 6683EA04 <1> sub dx, 04h

1348 00011F20 EC <1> in al, dx

1349 00011F21 3CAA <1> cmp al, 0AAh

1350 00011F23 7413 <1> je short Found

1351 00011F25 6683C204 <1> add dx, 04h

1352 00011F29 E2E8 <1> loop WaitID

1353 <1> NextPort:

1354 00011F2B 6683C310 <1> add bx, 10h ; if not response,

1355 00011F2F 6681FB6002 <1> cmp bx, 260h ; try the next port.

1356 00011F34 76C4 <1> jbe short ResetDSP

1357 00011F36 F9 <1> stc

1358 00011F37 C3 <1> retn

1359 <1> Found:

1360 00011F38 66891D[C26B0100] <1> mov [audio\_io\_base], bx ; SB Port Address Found!

1361 <1> ScanIRQ:

1362 <1> SetIrqs:

1363 00011F3F 28C0 <1> sub al, al ; 0

1364 00011F41 A2[BA6B0100] <1> mov [IRQnum], al ; reset

1365 00011F46 A2[BF6B0100] <1> mov [audio\_intr], al ; reset

1366 <1>

1367 <1> ; ah > 0 -> set IRQ vector

1368 <1> ; al = IRQ number

1369 <1> ;mov ax, 103h ; IRQ 3

1370 <1> ;call set\_hardware\_int\_vector

1371 <1> ;mov ax, 104h ; IRQ 4

1372 <1> ;call set\_hardware\_int\_vector

1373 00011F4B 66B80501 <1> mov ax, 105h ; IRQ 5

1374 00011F4F E8F0DDFFFF <1> call set\_hardware\_int\_vector

1375 00011F54 66B80701 <1> mov ax, 107h ; IRQ 7

1376 00011F58 E8E7DDFFFF <1> call set\_hardware\_int\_vector

1377 <1>

1378 00011F5D 668B15[C26B0100] <1> mov dx, [audio\_io\_base] ; tells to the SB to

1379 00011F64 6683C20C <1> add dx, 0Ch ; generate a IRQ!

1380 <1> WaitSb:

1381 00011F68 EC <1> in al, dx

1382 00011F69 08C0 <1> or al, al

1383 00011F6B 78FB <1> js short WaitSb

1384 00011F6D B0F2 <1> mov al, 0F2h

1385 00011F6F EE <1> out dx, al

1386 <1>

1387 00011F70 31C9 <1> xor ecx, ecx ; wait until IRQ level

1388 <1> WaitIRQ:

1389 00011F72 A0[BA6B0100] <1> mov al, [IRQnum]

1390 00011F77 3C00 <1> cmp al, 0 ; is changed or timeout.

1391 00011F79 7706 <1> ja short IrqOk

1392 00011F7B 6649 <1> dec cx

1393 00011F7D 75F3 <1> jnz short WaitIRQ

1394 00011F7F EB15 <1> jmp short RestoreIrqs

1395 <1> IrqOk:

1396 00011F81 A2[BF6B0100] <1> mov [audio\_intr], al ; set

1397 00011F86 668B15[C26B0100] <1> mov dx, [audio\_io\_base]

1398 00011F8D 6683C20E <1> add dx, 0Eh

1399 00011F91 EC <1> in al, dx ; SB acknowledge.

1400 00011F92 B020 <1> mov al, 20h

1401 00011F94 E620 <1> out 20h, al ; Hardware acknowledge.

1402 <1>

1403 <1> RestoreIrqs:

1404 <1> ; ah = 0 -> reset IRQ vector

1405 <1> ; al = IRQ number

1406 <1> ;mov ax, 3 ; IRQ 3

1407 <1> ;call set\_hardware\_int\_vector

1408 <1> ;mov ax, 4 ; IRQ 4

1409 <1> ;call set\_hardware\_int\_vector

1410 00011F96 66B80500 <1> mov ax, 5 ; IRQ 5

1411 00011F9A E8A5DDFFFF <1> call set\_hardware\_int\_vector

1412 00011F9F 66B80700 <1> mov ax, 7 ; IRQ 7

1413 00011FA3 E89CDDFFFF <1> call set\_hardware\_int\_vector

1414 <1>

1415 00011FA8 31D2 <1> xor edx, edx

1416 00011FAA 8915[C46B0100] <1> mov [audio\_dev\_id], edx ; 0

1417 00011FB0 8915[C86B0100] <1> mov [audio\_vendor], edx ; 0

1418 00011FB6 8915[CC6B0100] <1> mov [audio\_stats\_cmd], edx ; 0

1419 <1>

1420 <1> ;popad

1421 <1>

1422 00011FBC 803D[BF6B0100]01 <1> cmp byte [audio\_intr], 1 ; IRQ level was changed?

1423 <1>

1424 00011FC3 C3 <1> retn

1425 <1>

1426 <1> %macro SbOut 1

1427 <1> %%Wait:

1428 <1> in al, dx

1429 <1> or al, al

1430 <1> js short %%Wait

1431 <1> mov al, %1

1432 <1> out dx, al

1433 <1> %endmacro

1434 <1>

1435 <1> SbInit\_play:

1436 <1> ; 22/10/2017

1437 <1> ; 20/10/2017

1438 <1> ; 06/10/2017

1439 <1> ; 13/07/2017, 09/08/2017

1440 <1> ; 24/04/2017, 15/05/2017, 24/06/2017

1441 <1> ;pushad

1442 <1> SetBuffer:

1443 <1> ;mov byte [DmaFlag], 0

1444 <1>

1445 00011FC4 8B1D[DC6B0100] <1> mov ebx, [audio\_dma\_buff] ; physical addr of DMA buff

1446 00011FCA 89DF <1> mov edi, ebx

1447 00011FCC 8B0D[E06B0100] <1> mov ecx, [audio\_dmabuff\_size]

1448 <1>

1449 00011FD2 803D[EC6B0100]10 <1> cmp byte [audio\_bps], 16

1450 00011FD9 7531 <1> jne short sbInit\_0 ; set 8 bit DMA buffer

1451 <1>

1452 <1> ; 09/08/2017

1453 <1> ; convert byte count to word count

1454 00011FDB D1E9 <1> shr ecx, 1

1455 00011FDD 49 <1> dec ecx ; word count - 1

1456 <1> ; convert byte offset to word offset

1457 00011FDE D1EB <1> shr ebx, 1

1458 <1>

1459 <1> ; 16 bit DMA buffer setting (DMA channel 5)

1460 00011FE0 B005 <1> mov al, 05h ; set mask bit for channel 5 (4+1)

1461 00011FE2 E6D4 <1> out 0D4h, al

1462 <1>

1463 00011FE4 30C0 <1> xor al, al ; stops all DMA processes on selected channel

1464 00011FE6 E6D8 <1> out 0D8h, al ; clear selected channel register

1465 <1>

1466 00011FE8 88D8 <1> mov al, bl ; byte 0 of DMA buffer offset in words (physical)

1467 00011FEA E6C4 <1> out 0C4h, al ; DMA channel 5 port number

1468 <1>

1469 00011FEC 88F8 <1> mov al, bh ; byte 1 of DMA buffer offset in words (physical)

1470 00011FEE E6C4 <1> out 0C4h, al

1471 <1>

1472 <1> ; 09/08/2017

1473 00011FF0 C1EB0F <1> shr ebx, 15 ; complete 16 bit shift

1474 00011FF3 80E3FE <1> and bl, 0FEh ; clear bit 0 (not necessary, it will be ignored)

1475 <1>

1476 00011FF6 88D8 <1> mov al, bl ; byte 2 of DMA buffer address (physical)

1477 00011FF8 E68B <1> out 8Bh, al ; page register port addr for channel 5 ; 13/07/2017

1478 <1>

1479 00011FFA 88C8 <1> mov al, cl ; low byte of DMA count - 1

1480 00011FFC E6C6 <1> out 0C6h, al ; count register port addr for channel 1

1481 <1>

1482 00011FFE 88E8 <1> mov al, ch ; high byte of DMA count - 1

1483 00012000 E6C6 <1> out 0C6h, al

1484 <1>

1485 <1> ; channel 5, read, autoinitialized, single mode

1486 <1> ;mov al, 49h

1487 00012002 B059 <1> mov al, 59h ; 06/10/2017

1488 00012004 E6D6 <1> out 0D6h, al ; DMA mode register port address

1489 <1>

1490 00012006 B001 <1> mov al, 01h ; clear mask bit for channel 1

1491 00012008 E6D4 <1> out 0D4h, al ; DMA mask register port address

1492 <1>

1493 0001200A EB28 <1> jmp short ClearBuffer

1494 <1>

1495 <1> sbInit\_0:

1496 0001200C 49 <1> dec ecx ; 09/08/2017

1497 <1>

1498 <1> ; 8 bit DMA buffer setting (DMA channel 1)

1499 0001200D B005 <1> mov al, 05h ; set mask bit for channel 1 (4+1)

1500 0001200F E60A <1> out 0Ah, al ; DMA mask register

1501 <1>

1502 00012011 30C0 <1> xor al, al ; stops all DMA processes on selected channel

1503 00012013 E60C <1> out 0Ch, al ; clear selected channel register

1504 <1>

1505 00012015 88D8 <1> mov al, bl ; byte 0 of DMA buffer address (physical)

1506 00012017 E602 <1> out 02h, al ; DMA channel 1 port number

1507 <1>

1508 00012019 88F8 <1> mov al, bh ; byte 1 of DMA buffer address (physical)

1509 0001201B E602 <1> out 02h, al

1510 <1>

1511 0001201D C1EB10 <1> shr ebx, 16

1512 <1>

1513 00012020 88D8 <1> mov al, bl ; byte 2 of DMA buffer address (physical)

1514 00012022 E683 <1> out 83h, al ; page register port addr for channel 1

1515 <1>

1516 00012024 88C8 <1> mov al, cl ; low byte of DMA count - 1

1517 00012026 E603 <1> out 03h, al ; count register port addr for channel 1

1518 <1>

1519 00012028 88E8 <1> mov al, ch ; high byte of DMA count - 1

1520 0001202A E603 <1> out 03h, al

1521 <1>

1522 <1> ; channel 1, read, autoinitialized, single mode

1523 <1> ;mov al, 49h

1524 0001202C B059 <1> mov al, 59h ; 06/10/2017

1525 0001202E E60B <1> out 0Bh, al ; DMA mode register port address

1526 <1>

1527 00012030 B001 <1> mov al, 01h ; clear mask bit for channel 1

1528 00012032 E60A <1> out 0Ah, al ; DMA mask register port address

1529 <1>

1530 <1> ClearBuffer:

1531 <1> ;;mov edi, [audio\_dma\_buff]

1532 <1> ;;mov ecx, [audio\_dmabuff\_size]

1533 <1> ;inc ecx

1534 <1> ;mov al, 80h

1535 <1> ;;cld

1536 <1> ;rep stosb

1537 <1> SetIrq:

1538 <1> ;mov ebx, SbIrqhandler

1539 <1> ;mov al, [audio\_intr] ; IRQ number

1540 <1> ;call set\_dev\_IRQ\_service

1541 <1> ;; SETUP (audio) INTERRUPT CALLBACK SERVICE

1542 <1> ;mov bl, [audio\_intr] ; IRQ number

1543 <1> ;mov bh, [audio\_cb\_mode]

1544 <1> ;inc bh ; 1 = Signal Response Byte method (fixed value)

1545 <1> ; ; 2 = Callback service method

1546 <1> ; ; 3 = Auto Increment S.R.B. method

1547 <1> ;mov cl, [audio\_srb]

1548 <1> ;mov edx, [audio\_cb\_addr]

1549 <1> ;mov al, [audio\_user]

1550 <1> ;call set\_irq\_callback\_service

1551 <1> ResetDsp:

1552 00012034 668B15[C26B0100] <1> mov dx, [audio\_io\_base]

1553 0001203B 6683C206 <1> add dx, 06h

1554 0001203F B001 <1> mov al, 1

1555 00012041 EE <1> out dx, al

1556 <1>

1557 00012042 EC <1> in al, dx

1558 00012043 EC <1> in al, dx

1559 00012044 EC <1> in al, dx

1560 00012045 EC <1> in al, dx

1561 <1>

1562 00012046 30C0 <1> xor al, al

1563 00012048 EE <1> out dx, al

1564 <1>

1565 00012049 66B96400 <1> mov cx, 100

1566 0001204D 28E4 <1> sub ah, ah ; 0

1567 <1> WaitId:

1568 0001204F 668B15[C26B0100] <1> mov dx, [audio\_io\_base]

1569 00012056 6683C20E <1> add dx, 0Eh

1570 0001205A EC <1> in al, dx

1571 0001205B 08C0 <1> or al, al

1572 0001205D 7807 <1> js short sb\_GetId

1573 0001205F E2EE <1> loop WaitId

1574 00012061 E9B4000000 <1> jmp sb\_Exit

1575 <1> sb\_GetId:

1576 00012066 668B15[C26B0100] <1> mov dx, [audio\_io\_base]

1577 0001206D 6683C20A <1> add dx, 0Ah

1578 00012071 EC <1> in al, dx

1579 00012072 3CAA <1> cmp al, 0AAh

1580 00012074 7407 <1> je short SbOk

1581 00012076 E2D7 <1> loop WaitId

1582 00012078 E99D000000 <1> jmp sb\_Exit

1583 <1> SbOk:

1584 0001207D 668B15[C26B0100] <1> mov dx, [audio\_io\_base]

1585 00012084 6683C20C <1> add dx, 0Ch

1586 <1> SbOut 0D1h ; Turn on speaker

1586 <2> %%Wait:

1586 00012088 EC <2> in al, dx

1586 00012089 08C0 <2> or al, al

1586 0001208B 78FB <2> js short %%Wait

1586 0001208D B0D1 <2> mov al, %1

1586 0001208F EE <2> out dx, al

1587 <1> SbOut 41h ; 8 bit or 16 bit transfer

1587 <2> %%Wait:

1587 00012090 EC <2> in al, dx

1587 00012091 08C0 <2> or al, al

1587 00012093 78FB <2> js short %%Wait

1587 00012095 B041 <2> mov al, %1

1587 00012097 EE <2> out dx, al

1588 00012098 668B1D[EE6B0100] <1> mov bx, [audio\_freq] ; sampling rate (Hz)

1589 <1> SbOut bh ; sampling rate high byte

1589 <2> %%Wait:

1589 0001209F EC <2> in al, dx

1589 000120A0 08C0 <2> or al, al

1589 000120A2 78FB <2> js short %%Wait

1589 000120A4 88F8 <2> mov al, %1

1589 000120A6 EE <2> out dx, al

1590 <1> SbOut bl ; sampling rate low byte

1590 <2> %%Wait:

1590 000120A7 EC <2> in al, dx

1590 000120A8 08C0 <2> or al, al

1590 000120AA 78FB <2> js short %%Wait

1590 000120AC 88D8 <2> mov al, %1

1590 000120AE EE <2> out dx, al

1591 <1>

1592 <1> ; 22/05/2017

1593 000120AF E8C0000000 <1> call sb16\_volume\_initial ; 15/05/2017

1594 <1> ; 20/05/2017

1595 <1> ;call sb16\_volume

1596 <1>

1597 <1> StartDma:

1598 <1> ; autoinitialized mode

1599 000120B4 803D[EC6B0100]10 <1> cmp byte [audio\_bps], 16 ; 16 bit samples

1600 000120BB 7411 <1> je short sb\_play\_1

1601 <1> ; 8 bit samples

1602 000120BD 66BBC600 <1> mov bx, 0C6h ; 8 bit output (0C6h)

1603 000120C1 803D[ED6B0100]02 <1> cmp byte [audio\_stmo], 2 ; 1 = mono, 2 = stereo

1604 000120C8 7214 <1> jb short sb\_play\_2

1605 000120CA B720 <1> mov bh, 20h ; 8 bit stereo (20h)

1606 000120CC EB10 <1> jmp short sb\_play\_2

1607 <1> sb\_play\_1:

1608 <1> ; 16 bit samples

1609 000120CE 66BBB610 <1> mov bx, 10B6h ; 16 bit output (0B6h)

1610 000120D2 803D[ED6B0100]02 <1> cmp byte [audio\_stmo], 2 ; 1 = mono, 2 = stereo

1611 000120D9 7203 <1> jb short sb\_play\_2

1612 000120DB 80C720 <1> add bh, 20h ; 16 bit stereo (30h)

1613 <1> sb\_play\_2:

1614 <1> ; PCM output (8/16 bit mono autoinitialized transfer)

1615 <1> SbOut bl ; bCommand

1615 <2> %%Wait:

1615 000120DE EC <2> in al, dx

1615 000120DF 08C0 <2> or al, al

1615 000120E1 78FB <2> js short %%Wait

1615 000120E3 88D8 <2> mov al, %1

1615 000120E5 EE <2> out dx, al

1616 <1> SbOut bh ; bMode

1616 <2> %%Wait:

1616 000120E6 EC <2> in al, dx

1616 000120E7 08C0 <2> or al, al

1616 000120E9 78FB <2> js short %%Wait

1616 000120EB 88F8 <2> mov al, %1

1616 000120ED EE <2> out dx, al

1617 000120EE 8B1D[E06B0100] <1> mov ebx, [audio\_dmabuff\_size] ; 15/05/2017

1618 000120F4 D1EB <1> shr ebx, 1 ; half buffer size

1619 <1> ; 20/10/2017

1620 000120F6 803D[EC6B0100]10 <1> cmp byte [audio\_bps], 16 ; 16 bit DMA

1621 000120FD 7502 <1> jne short sb\_play\_3

1622 000120FF D1EB <1> shr ebx, 1 ; byte count to word count

1623 <1> sb\_play\_3:

1624 00012101 664B <1> dec bx ; wBlkSize is one less than the actual size

1625 <1> SbOut bl

1625 <2> %%Wait:

1625 00012103 EC <2> in al, dx

1625 00012104 08C0 <2> or al, al

1625 00012106 78FB <2> js short %%Wait

1625 00012108 88D8 <2> mov al, %1

1625 0001210A EE <2> out dx, al

1626 <1> SbOut bh

1626 <2> %%Wait:

1626 0001210B EC <2> in al, dx

1626 0001210C 08C0 <2> or al, al

1626 0001210E 78FB <2> js short %%Wait

1626 00012110 88F8 <2> mov al, %1

1626 00012112 EE <2> out dx, al

1627 <1>

1628 00012113 C605[F06B0100]01 <1> mov byte [audio\_play\_cmd], 1 ; playing !

1629 <1>

1630 <1> ;; Set Voice and master volumes

1631 <1> ;mov dx, [audio\_io\_base]

1632 <1> ;add dl, 4 ; Mixer chip Register Address Port

1633 <1> ;SbOut 30h ; select Master Volume Register (L)

1634 <1> ;inc dl ; Mixer chip Register Data Port

1635 <1> ;SbOut 0F8h ; Max. volume value is 31 (31\*8)

1636 <1> ;dec dl

1637 <1> ;SbOut 31h ; select Master Volume Register (R)

1638 <1> ;inc dl

1639 <1> ;SbOut 0F8h ; Max. volume value is 31 (31\*8)

1640 <1> ;dec dl

1641 <1> ;SbOut 32h ; select Voice Volume Register (L)

1642 <1> ;inc dl

1643 <1> ;SbOut 0F8h ; Max. volume value is 31 (31\*8)

1644 <1> ;dec dl

1645 <1> ;SbOut 33h ; select Voice Volume Register (R)

1646 <1> ;inc dl

1647 <1> ;SbOut 0F8h ; Max. volume value is 31 (31\*8)

1648 <1> ;;

1649 <1> ;dec dl

1650 <1> ;SbOut 44h ; select Treble Register (L)

1651 <1> ;inc dl

1652 <1> ;SbOut 0F0h ; Max. Treble value is 15 (15\*16)

1653 <1> ;dec dl

1654 <1> ;SbOut 45h ; select Treble Register (R)

1655 <1> ;inc dl

1656 <1> ;SbOut 0F0h ; Max. Treble value is 15 (15\*16)

1657 <1> ;dec dl

1658 <1> ;SbOut 46h ; select Bass Register (L)

1659 <1> ;inc dl

1660 <1> ;SbOut 0F0h ; Max. Bass value is 15 (15\*16)

1661 <1> ;dec dl

1662 <1> ;SbOut 47h ; select Bass Register (R)

1663 <1> ;inc dl

1664 <1> ;SbOut 0F0h ; Max. Bass value is 15 (15\*16)

1665 <1>

1666 <1> sb\_Exit:

1667 <1> ;popad

1668 0001211A C3 <1> retn

1669 <1>

1670 <1> sb16\_int\_handler:

1671 <1> ; Interrupt Handler for Sound Blaster 16 Audio Card

1672 <1> ; Note: called by 'dev\_IRQ\_service'

1673 <1> ; 20/10/2017

1674 <1> ; 12/10/2017

1675 <1> ; 10/10/2017

1676 <1> ; 12/05/2017, 09/10/2017

1677 <1> ; 24/04/2017 (TRDOS 386 kernel, 'audio.s')

1678 <1> ; 10/03/2017 - 'PLAYWAV.PRG' ('playwav.s')

1679 <1>

1680 <1> ;push eax ; \* must be saved !

1681 <1> ;push ebx ; \* must be saved !

1682 <1> ;push ecx

1683 <1> ;push edx

1684 <1> ;push esi

1685 <1> ;push edi

1686 <1>

1687 0001211B 668B15[C26B0100] <1> mov dx, [audio\_io\_base]

1688 <1> ; 20/10/2017

1689 00012122 80C20F <1> add dl, 0Fh ; 2xFh (DSP 16 bit intr ack)

1690 00012125 803D[EC6B0100]10 <1> cmp byte [audio\_bps], 16

1691 0001212C 7402 <1> je short sb\_irq\_16bit\_ack

1692 <1> sb\_irq\_8bit\_ack:

1693 0001212E FECA <1> dec dl ; 2xEh (DSP 8 bit intr ack)

1694 <1> sb\_irq\_16bit\_ack:

1695 00012130 EC <1> in al, dx

1696 <1>

1697 <1> ;cmp byte [audio\_busy], 0

1698 <1> ;ja short sb\_irq\_h3

1699 <1>

1700 <1> ;mov byte [audio\_busy], 1

1701 <1>

1702 00012131 803D[F06B0100]01 <1> cmp byte [audio\_play\_cmd], 1

1703 00012138 7307 <1> jnb short sb\_irq\_h1

1704 <1> sb\_irq\_h0:

1705 0001213A E8A9000000 <1> call sb16\_stop

1706 0001213F EB2B <1> jmp short sb\_irq\_h3

1707 <1> sb\_irq\_h1:

1708 <1> ;call sb16\_tuneloop

1709 <1> ; 09/10/2017

1710 <1> sb16\_tuneloop:

1711 00012141 8B3D[DC6B0100] <1> mov edi, [audio\_dma\_buff]

1712 00012147 8B0D[E06B0100] <1> mov ecx, [audio\_dmabuff\_size]

1713 0001214D D1E9 <1> shr ecx, 1 ; dma buff size / 2 = half buffer size

1714 <1>

1715 <1> ; 22/05/2017

1716 0001214F F605[E46B0100]01 <1> test byte [audio\_flag], 1 ; Current flag value

1717 00012156 7402 <1> jz short sb\_tlp1 ; EOL (Half Buffer 1 must be filled)

1718 <1> ; FLAG (Half Buffer 2 must be filled)

1719 00012158 01CF <1> add edi, ecx

1720 <1> ; 15/05/2017

1721 <1> sb\_tlp1:

1722 0001215A 8B35[D46B0100] <1> mov esi, [audio\_p\_buffer] ; phy addr of audio buff

1723 <1> ;rep movsb

1724 00012160 C1E902 <1> shr ecx, 2 ; half buff size / 4

1725 00012163 F3A5 <1> rep movsd

1726 <1> ;retn

1727 <1>

1728 <1> ; 10/10/2017

1729 <1> ; switch flag value

1730 00012165 8035[E46B0100]01 <1> xor byte [audio\_flag], 1

1731 <1>

1732 <1> ; 12/10/2017

1733 <1> ; [audio\_flag] = 0 : Playing dma half buffer 2 (odd intr count)

1734 <1> ; Next buffer (to update) is dma half buff 1

1735 <1> ; = 1 : Playing dma half buffer 1 (even intr count)

1736 <1> ; Next buffer (to update) is dma half buff 2

1737 <1>

1738 <1> sb\_irq\_h3:

1739 <1> ;mov byte [audio\_busy], 0

1740 <1>

1741 <1> ;pop edi

1742 <1> ;pop esi

1743 <1> ;pop edx

1744 <1> ;pop ecx

1745 <1> ;pop ebx ; \* must be restored !

1746 <1> ;pop eax ; \* must be restored !

1747 <1>

1748 0001216C C3 <1> retn

1749 <1>

1750 <1> sb16\_volume:

1751 <1> ; 22/10/2017

1752 <1> ; mov [audio\_master\_volume\_l], cl

1753 <1> ; mov [audio\_master\_volume\_h], ch

1754 0001216D 66890D[F26B0100] <1> mov [audio\_master\_volume], cx

1755 <1> sb16\_volume\_initial:

1756 00012174 6652 <1> push dx ; DX (port address) must be saved

1757 00012176 668B15[C26B0100] <1> mov dx, [audio\_io\_base]

1758 0001217D 6683C204 <1> add dx, 4 ; Mixer chip address port

1759 00012181 B022 <1> mov al, 22h ; master volume

1760 00012183 EE <1> out dx, al

1761 00012184 6642 <1> inc dx

1762 00012186 8A25[F26B0100] <1> mov ah, [audio\_master\_volume\_l]

1763 0001218C C0EC02 <1> shr ah, 2 ; 32 -> 8 level

1764 0001218F C0E405 <1> shl ah, 5 ; bit 5 to 7

1765 00012192 A0[F36B0100] <1> mov al, [audio\_master\_volume\_r]

1766 00012197 C0E802 <1> shr al, 2 ; 32 -> 8 level

1767 <1> ;and al, 0Fh

1768 0001219A D0E0 <1> shl al, 1 ; bit 1 to 3

1769 0001219C 08E0 <1> or al, ah

1770 0001219E EE <1> out dx, al

1771 0001219F 665A <1> pop dx ; DX (port address) must be restored

1772 000121A1 C3 <1> retn

1773 <1>

1774 <1> sb16\_pause:

1775 000121A2 668B15[C26B0100] <1> mov dx, [audio\_io\_base]

1776 000121A9 6683C20C <1> add dx, 0Ch ; Command & Data Port

1777 000121AD 803D[EC6B0100]10 <1> cmp byte [audio\_bps], 16 ; 16 bit samples

1778 000121B4 7404 <1> je short sb\_pause\_1

1779 <1> ; 8 bit samples

1780 000121B6 B3D0 <1> mov bl, 0D0h ; 8 bit DMA mode

1781 000121B8 EB02 <1> jmp short sb\_pause\_2

1782 <1> sb\_pause\_1:

1783 <1> ; 16 bit samples

1784 000121BA B3D5 <1> mov bl, 0D5h ; 16 bit DMA mode

1785 <1> sb\_pause\_2:

1786 <1> SbOut bl ; bCommand

1786 <2> %%Wait:

1786 000121BC EC <2> in al, dx

1786 000121BD 08C0 <2> or al, al

1786 000121BF 78FB <2> js short %%Wait

1786 000121C1 88D8 <2> mov al, %1

1786 000121C3 EE <2> out dx, al

1787 <1> sb\_pause\_3:

1788 000121C4 C3 <1> retn

1789 <1>

1790 <1> sb16\_continue:

1791 000121C5 668B15[C26B0100] <1> mov dx, [audio\_io\_base]

1792 000121CC 6683C20C <1> add dx, 0Ch ; Command & Data Port

1793 000121D0 803D[EC6B0100]10 <1> cmp byte [audio\_bps], 16 ; 16 bit samples

1794 000121D7 7404 <1> je short sb\_cont\_1

1795 <1> ; 8 bit samples

1796 000121D9 B3D4 <1> mov bl, 0D4h ; 8 bit DMA mode

1797 000121DB EB02 <1> jmp short sb\_cont\_2

1798 <1> sb\_cont\_1:

1799 <1> ; 16 bit samples

1800 000121DD B3D6 <1> mov bl, 0D6h ; 16 bit DMA mode

1801 <1> sb\_cont\_2:

1802 <1> SbOut bl ; bCommand

1802 <2> %%Wait:

1802 000121DF EC <2> in al, dx

1802 000121E0 08C0 <2> or al, al

1802 000121E2 78FB <2> js short %%Wait

1802 000121E4 88D8 <2> mov al, %1

1802 000121E6 EE <2> out dx, al

1803 <1> sb\_cont\_3:

1804 000121E7 C3 <1> retn

1805 <1>

1806 <1> sb16\_stop:

1807 <1> ; 24/04/2017

1808 000121E8 803D[F06B0100]00 <1> cmp byte [audio\_play\_cmd], 0

1809 000121EF 7648 <1> jna short sb16\_stop\_4

1810 <1>

1811 <1> ; 22/05/2017

1812 000121F1 668B15[C26B0100] <1> mov dx, [audio\_io\_base]

1813 000121F8 6683C20C <1> add dx, 0Ch

1814 <1>

1815 000121FC B3D9 <1> mov bl, 0D9h ; exit auto-initialize 16 bit transfer

1816 <1> ; stop autoinitialized DMA transfer mode

1817 000121FE 803D[EC6B0100]10 <1> cmp byte [audio\_bps], 16 ; 16 bit samples

1818 00012205 7402 <1> je short sb16\_stop\_1

1819 <1> ;mov bl, 0DAh ; exit auto-initialize 8 bit transfer

1820 00012207 FEC3 <1> inc bl

1821 <1> sb16\_stop\_1:

1822 <1> SbOut bl ; exit auto-initialize transfer command

1822 <2> %%Wait:

1822 00012209 EC <2> in al, dx

1822 0001220A 08C0 <2> or al, al

1822 0001220C 78FB <2> js short %%Wait

1822 0001220E 88D8 <2> mov al, %1

1822 00012210 EE <2> out dx, al

1823 <1>

1824 00012211 30C0 <1> xor al, al ; stops all DMA processes on selected channel

1825 <1>

1826 00012213 803D[EC6B0100]10 <1> cmp byte [audio\_bps], 16 ; 16 bit samples

1827 0001221A 7404 <1> je short sb16\_stop\_2

1828 0001221C E60C <1> out 0Ch, al ; clear selected channel register

1829 0001221E EB02 <1> jmp short sb16\_stop\_3

1830 <1>

1831 <1> sb16\_stop\_2:

1832 00012220 E6D8 <1> out 0D8h, al ; clear selected channel register

1833 <1>

1834 <1> sb16\_stop\_3:

1835 00012222 C605[F06B0100]00 <1> mov byte [audio\_play\_cmd], 0 ; stop !

1836 <1> SbDone:

1837 <1> ;mov dx, [audio\_io\_base]

1838 <1> ;add dx, 0Ch

1839 <1> SbOut 0D0h

1839 <2> %%Wait:

1839 00012229 EC <2> in al, dx

1839 0001222A 08C0 <2> or al, al

1839 0001222C 78FB <2> js short %%Wait

1839 0001222E B0D0 <2> mov al, %1

1839 00012230 EE <2> out dx, al

1840 <1> SbOut 0D3h

1840 <2> %%Wait:

1840 00012231 EC <2> in al, dx

1840 00012232 08C0 <2> or al, al

1840 00012234 78FB <2> js short %%Wait

1840 00012236 B0D3 <2> mov al, %1

1840 00012238 EE <2> out dx, al

1841 <1> sb16\_stop\_4:

1842 00012239 C3 <1> retn

1843 <1>

1844 <1> sb16\_reset:

1845 <1> ; 24/04/2017

1846 0001223A 668B15[C26B0100] <1> mov dx, [audio\_io\_base] ; try to reset the DSP.

1847 00012241 6683C206 <1> add dx, 06h

1848 00012245 B001 <1> mov al, 1

1849 00012247 EE <1> out dx, al

1850 <1>

1851 00012248 EC <1> in al, dx

1852 00012249 EC <1> in al, dx

1853 0001224A EC <1> in al, dx

1854 0001224B EC <1> in al, dx

1855 <1>

1856 0001224C 30C0 <1> xor al, al

1857 0001224E EE <1> out dx, al

1858 <1>

1859 0001224F 6683C208 <1> add dx, 08h

1860 00012253 66B96400 <1> mov cx, 100

1861 <1> sbrstWaitID:

1862 00012257 EC <1> in al, dx

1863 00012258 08C0 <1> or al, al

1864 0001225A 7804 <1> js short sbrstGetID

1865 0001225C E2F9 <1> loop sbrstWaitID

1866 0001225E F9 <1> stc

1867 0001225F C3 <1> retn

1868 <1> sbrstGetID:

1869 00012260 6683EA04 <1> sub dx, 04h

1870 00012264 EC <1> in al, dx

1871 00012265 3CAA <1> cmp al, 0AAh

1872 00012267 7406 <1> je short sb\_rst\_retn

1873 00012269 6683C204 <1> add dx, 04h

1874 0001226D E2E8 <1> loop sbrstWaitID

1875 <1> sb\_rst\_retn:

1876 0001226F C3 <1> retn

1877 <1>

1878 <1> ac97\_codec\_config:

1879 <1> ; 10/06/2017

1880 <1> ; 05/06/2017

1881 <1> ; 29/05/2017

1882 <1> ; 28/05/2017 (TRDOS 386, 'audio.s')

1883 <1> ; 07/11/2016 (Erdogan Tan)

1884 <1> ; Derived from 'codecConfig' procedure in 'CODEC.ASM'

1885 <1> ; .wav player for DOS by Jeff Leyda (02/09/2002)

1886 <1>

1887 <1> ;; 'PLAYER.ASM'

1888 <1> ;; get ICH base address regs for mixer and bus master

1889 <1>

1890 <1> init\_ac97\_controller: ; 10/06/2017

1891 00012270 A1[C46B0100] <1> mov eax, [audio\_dev\_id]

1892 <1> ;mov al, NAMBAR\_REG

1893 <1> ;;call pciRegRead16 ; read PCI registers 10-11

1894 <1> ;call pciRegRead32

1895 <1> ;and dx, IO\_ADDR\_MASK ; mask off BIT0

1896 <1> ;;and edx, IO\_ADDR\_MASK

1897 <1>

1898 <1> ;mov [NAMBAR], dx ; save audio mixer base addr

1899 <1>

1900 <1> ;mov al, NABMBAR\_REG

1901 <1> ;;call pciRegRead16

1902 <1> ;call pciRegRead32

1903 <1> ;and dx, 0FFC0h ; IO\_ADDR\_MASK

1904 <1> ;;and edx, 0FFC0h

1905 <1>

1906 <1> ;mov [NABMBAR], dx ; save bus master base addr

1907 <1>

1908 <1> ;mov eax, [audio\_dev\_id]

1909 00012275 B004 <1> mov al, PCI\_CMD\_REG

1910 <1> ;call pciRegRead8 ; read PCI command register

1911 00012277 E840F8FFFF <1> call pciRegRead16

1912 0001227C 80CA05 <1> or dl, IO\_ENA+BM\_ENA ; enable IO and bus master

1913 <1> ;call pciRegWrite8

1914 0001227F E8A3F8FFFF <1> call pciRegWrite16

1915 <1>

1916 <1> ; 'CODEC.ASM'

1917 <1>

1918 <1> ; enable codec, unmute stuff, set output rate

1919 <1> ; ; entry: [audio\_freq] = desired sample rate

1920 <1>

1921 <1> ; mov dx, [NAMBAR]

1922 <1> ; add dx, CODEC\_EXT\_AUDIO\_CTRL\_REG ; 2Ah

1923 <1> ; in ax, dx

1924 <1> ; or ax, 1

1925 <1> ; out dx, ax ; Enable variable rate audio

1926 <1>

1927 <1> ; ;call delay1\_4ms

1928 <1> ; ;call delay1\_4ms

1929 <1> ; ;call delay1\_4ms

1930 <1> ; ;call delay1\_4ms

1931 <1>

1932 <1> ; mov ax, [audio\_freq] ; sample rate

1933 <1>

1934 <1> ; mov dx, [NAMBAR]

1935 <1> ; add dx, CODEC\_PCM\_FRONT\_DACRATE\_REG ; 2Ch

1936 <1> ; out dx, ax ; out sample rate

1937 <1>

1938 <1> ; ;call delay1\_4ms

1939 <1> ; ;call delay1\_4ms

1940 <1> ; ;call delay1\_4ms

1941 <1> ; ;call delay1\_4ms

1942 <1>

1943 <1> ;mov dx, [NAMBAR] ; mixer base address

1944 <1> ;add dx, CODEC\_RESET\_REG ; reset register

1945 <1> ;mov ax, 42

1946 <1> ;out dx, ax ; reset

1947 <1>

1948 <1> ;mov dx, [NABMBAR] ; bus master base address

1949 <1> ;add dx, GLOB\_STS\_REG

1950 <1> ;mov ax, 2

1951 <1> ;out dx, ax

1952 <1>

1953 00012284 E831F9FFFF <1> call delay\_100ms ; 29/05/2017

1954 <1>

1955 <1> init\_ac97\_codec:

1956 <1> ; 10/06/2017

1957 <1> ; 29/05/2017

1958 <1> ; 28/05/2017 - Erdogan Tan (Ref: KolibriOS, intelac97.asm)

1959 <1> ;

1960 00012289 66BA2C00 <1> mov dx, GLOB\_CNT\_REG ; 2Ch

1961 0001228D 660315[F66B0100] <1> add dx, [NABMBAR]

1962 00012294 ED <1> in eax, dx

1963 <1> ; ?

1964 00012295 66BA3000 <1> mov dx, GLOB\_STS\_REG ; 30h

1965 00012299 660315[F66B0100] <1> add dx, [NABMBAR]

1966 000122A0 ED <1> in eax, dx

1967 <1>

1968 000122A1 83F8FF <1> cmp eax, 0FFFFFFFFh ; -1

1969 000122A4 744B <1> je short init\_ac97\_codec\_err1

1970 <1>

1971 000122A6 A900030010 <1> test eax, CTRL\_ST\_CREADY

1972 000122AB 7507 <1> jnz short \_ac97\_codec\_ready

1973 <1>

1974 000122AD E8EF020000 <1> call reset\_ac97\_codec

1975 000122B2 723E <1> jc short init\_ac97\_codec\_err2

1976 <1>

1977 <1> \_ac97\_codec\_ready:

1978 000122B4 668B15[F46B0100] <1> mov dx, [NAMBAR]

1979 <1> ;add dx, 0 ; ac\_reg\_0 ; reset register

1980 000122BB 66EF <1> out dx, ax

1981 <1>

1982 000122BD 31C0 <1> xor eax, eax ; 0

1983 000122BF 668B15[F46B0100] <1> mov dx, [NAMBAR]

1984 000122C6 6683C226 <1> add dx, CODEC\_REG\_POWERDOWN

1985 000122CA 66EF <1> out dx, ax

1986 <1>

1987 <1> ; 10/06/2017

1988 <1> ; 29/05/2017

1989 <1> ; wait for 1 second

1990 000122CC B9E8030000 <1> mov ecx, 1000 ; 1000\*0.25ms = 1s

1991 <1> \_ac97\_codec\_rloop:

1992 000122D1 E8F1F8FFFF <1> call delay1\_4ms

1993 000122D6 E8ECF8FFFF <1> call delay1\_4ms

1994 000122DB E8E7F8FFFF <1> call delay1\_4ms

1995 000122E0 E8E2F8FFFF <1> call delay1\_4ms

1996 <1> ;mov dx, [NAMBAR]

1997 <1> ;add dx, CODEC\_REG\_POWERDOWN

1998 000122E5 66ED <1> in ax, dx

1999 000122E7 6683E00F <1> and ax, 0Fh

2000 000122EB 3C0F <1> cmp al, 0Fh

2001 000122ED 7404 <1> je short \_ac97\_codec\_init\_ok

2002 000122EF E2E0 <1> loop \_ac97\_codec\_rloop

2003 <1>

2004 <1> init\_ac97\_codec\_err1:

2005 000122F1 F9 <1> stc

2006 <1> init\_ac97\_codec\_err2:

2007 000122F2 C3 <1> retn

2008 <1>

2009 <1> \_ac97\_codec\_init\_ok:

2010 000122F3 B002 <1> mov al, 2 ; force set 16-bit 2-channel PCM

2011 000122F5 66BA2C00 <1> mov dx, GLOB\_CNT\_REG ; 2Ch

2012 000122F9 660315[F66B0100] <1> add dx, [NABMBAR]

2013 00012300 EF <1> out dx, eax

2014 <1>

2015 <1> ;call delay1\_4ms

2016 <1>

2017 <1> ; 10/06/2017

2018 00012301 E849020000 <1> call reset\_ac97\_controller

2019 <1>

2020 <1> ; call setup\_ac97\_codec

2021 <1> ;

2022 <1> ;detect\_ac97\_codec:

2023 <1> ; retn

2024 <1>

2025 <1> setup\_ac97\_codec:

2026 <1> ; 10/06/2017

2027 <1> ; 29/05/2017

2028 00012306 B802020000 <1> mov eax, 0202h

2029 0001230B 66A3[F26B0100] <1> mov [audio\_master\_volume], ax

2030 00012311 66B81F1F <1> mov ax, 1F1Fh ; 31, 31

2031 <1>

2032 00012315 668B15[F46B0100] <1> mov dx, [NAMBAR]

2033 0001231C 6683C202 <1> add dx, CODEC\_MASTER\_VOL\_REG ;02h

2034 00012320 6631C0 <1> xor ax, ax ; volume attenuation = 0 (max. volume)

2035 00012323 66EF <1> out dx, ax

2036 <1>

2037 00012325 668B15[F46B0100] <1> mov dx, [NAMBAR]

2038 0001232C 6683C206 <1> add dx, CODEC\_MASTER\_MONO\_VOL\_REG ;06h

2039 <1> ;xor ax, ax

2040 00012330 66EF <1> out dx, ax

2041 <1>

2042 00012332 668B15[F46B0100] <1> mov dx, [NAMBAR]

2043 00012339 6683C20A <1> add dx, CODEC\_PCBEEP\_VOL\_REG ;0Ah

2044 <1> ;xor ax, ax

2045 0001233D 66EF <1> out dx, ax

2046 <1>

2047 0001233F 668B15[F46B0100] <1> mov dx, [NAMBAR]

2048 00012346 6683C218 <1> add dx, CODEC\_PCM\_OUT\_REG ;18h

2049 <1> ;xor ax, ax

2050 0001234A 66EF <1> out dx, ax

2051 <1>

2052 0001234C 66B80880 <1> mov ax, 8008h ; Mute

2053 00012350 668B15[F46B0100] <1> mov dx, [NAMBAR]

2054 00012357 6683C20C <1> add dx, 0Ch ; AC97\_PHONE\_VOL ; TAD Input (Mono)

2055 0001235B 66EF <1> out dx, ax

2056 <1>

2057 0001235D 66B80808 <1> mov ax, 0808h

2058 00012361 668B15[F46B0100] <1> mov dx, [NAMBAR]

2059 00012368 6683C210 <1> add dx, CODEC\_LINE\_IN\_VOL\_REG ; 10h ; Line Input (Stereo)

2060 0001236C 66EF <1> out dx, ax

2061 <1>

2062 <1> ;mov ax, 0808h

2063 0001236E 668B15[F46B0100] <1> mov dx, [NAMBAR]

2064 00012375 6683C212 <1> add dx, CODEC\_CD\_VOL\_REG ; 12h ; CR Input (Stereo)

2065 00012379 66EF <1> out dx, ax

2066 <1>

2067 <1> ;mov ax, 0808h

2068 0001237B 668B15[F46B0100] <1> mov dx, [NAMBAR]

2069 00012382 6683C216 <1> add dx, CODEC\_AUX\_VOL\_REG ; 16h ; Aux Input (Stereo)

2070 00012386 66EF <1> out dx, ax

2071 <1>

2072 <1> ;call delay1\_4ms

2073 <1> ;call delay1\_4ms

2074 <1> ;call delay1\_4ms

2075 <1> ;call delay1\_4ms

2076 <1>

2077 <1> detect\_ac97\_codec:

2078 00012388 C3 <1> retn

2079 <1>

2080 <1> set\_ac97\_bdl: ; Set AC97 (ICH) Buffer Descriptor List

2081 <1> ; 17/06/2017

2082 <1> ; 11/06/2017

2083 <1> ; 28/05/2017

2084 <1> ; eax = dma buffer address = [audio\_DMA\_buff]

2085 <1> ; ecx = dma buffer buffer size = [audio\_dmabuff\_size]

2086 <1>

2087 00012389 D1E9 <1> shr ecx, 1 ; dma half buffer size

2088 0001238B 89CE <1> mov esi, ecx

2089 <1>

2090 0001238D BF[F86B0100] <1> mov edi, audio\_bdl\_buff ; get BDL address

2091 00012392 B910000000 <1> mov ecx, 32 / 2 ; make 32 entries in BDL

2092 <1>

2093 00012397 EB05 <1> jmp short s\_ac97\_bdl1

2094 <1>

2095 <1> s\_ac97\_bdl0:

2096 <1> ; set buffer descriptor 0 to start of data file in memory

2097 <1>

2098 00012399 A1[DC6B0100] <1> mov eax, [audio\_dma\_buff] ; Physical address of DMA buffer

2099 <1>

2100 <1> s\_ac97\_bdl1:

2101 0001239E AB <1> stosd ; store dmabuffer1 address

2102 <1>

2103 0001239F 89C2 <1> mov edx, eax

2104 <1>

2105 <1> ;

2106 <1> ; Buffer Descriptors List

2107 <1> ; As stated earlier, each buffer descriptor list is a set of (up to) 32

2108 <1> ; descriptors, each 8 bytes in length. Bytes 0-3 of a descriptor entry point

2109 <1> ; to a chunk of memory to either play from or record to. Bytes 4-7 of an

2110 <1> ; entry describe various control things detailed below.

2111 <1> ;

2112 <1> ; Buffer pointers must always be aligned on a Dword boundry.

2113 <1> ;

2114 <1> ;

2115 <1>

2116 <1> ;IOC equ BIT31 ; Fire an interrupt whenever this

2117 <1> ; buffer is complete.

2118 <1>

2119 <1> ;BUP equ BIT30 ; Buffer Underrun Policy.

2120 <1> ; if this buffer is the last buffer

2121 <1> ; in a playback, fill the remaining

2122 <1> ; samples with 0 (silence) or not.

2123 <1> ; It's a good idea to set this to 1

2124 <1> ; for the last buffer in playback,

2125 <1> ; otherwise you're likely to get a lot

2126 <1> ; of noise at the end of the sound.

2127 <1>

2128 <1> ;

2129 <1> ; Bits 15:0 contain the length of the buffer, in number of samples, which

2130 <1> ; are 16 bits each, coupled in left and right pairs, or 32bits each.

2131 <1> ; Luckily for us, that's the same format as .wav files.

2132 <1> ;

2133 <1> ; A value of FFFF is 65536 samples. Running at 44.1Khz, that's just about

2134 <1> ; 1.5 seconds of sample time. FFFF \* 32bits is 1FFFFh bytes or 128k of data.

2135 <1> ;

2136 <1> ; A value of 0 in these bits means play no samples.

2137 <1> ;

2138 <1>

2139 000123A1 89F0 <1> mov eax, esi ; DMA half buffer size

2140 000123A3 01C2 <1> add edx, eax

2141 000123A5 D1E8 <1> shr eax, 1 ; count of 16 bit samples

2142 <1> ;or eax, IOC+BUP

2143 000123A7 0D00000080 <1> or eax, IOC ; 11/06/2017

2144 000123AC AB <1> stosd

2145 <1>

2146 <1> ; 2nd buffer:

2147 <1>

2148 000123AD 89D0 <1> mov eax, edx ; Physical address of the 2nd half of DMA buffer

2149 000123AF AB <1> stosd ; store dmabuffer2 address

2150 <1>

2151 <1> ; set length to [audio\_dmabuff\_size]/2

2152 <1> ; Set control (bits 31:16) to BUP, bits 15:0=number of samples

2153 <1> ;

2154 000123B0 89F0 <1> mov eax, esi ; DMA half buffer size

2155 000123B2 D1E8 <1> shr eax, 1 ; count of 16 bit samples

2156 <1> ;or eax, IOC+BUP

2157 000123B4 0D00000080 <1> or eax, IOC ; 11/06/2017

2158 000123B9 AB <1> stosd

2159 <1>

2160 000123BA E2DD <1> loop s\_ac97\_bdl0

2161 <1>

2162 000123BC C3 <1> retn

2163 <1>

2164 <1> ac97\_start\_play:

2165 <1> ; 28/05/2017

2166 <1> ; Derived from 'playWav' procedure in 'ICHWAV.ASM'

2167 <1> ; .wav player for DOS by Jeff Leyda (02/09/2002)

2168 <1>

2169 <1> ; set output rate

2170 <1> ; entry: [audio\_freq] = desired sample rate

2171 <1>

2172 000123BD 668B15[F46B0100] <1> mov dx, [NAMBAR]

2173 000123C4 6683C22A <1> add dx, CODEC\_EXT\_AUDIO\_CTRL\_REG ; 2Ah

2174 000123C8 66ED <1> in ax, dx

2175 000123CA 6683C801 <1> or ax, 1

2176 000123CE 66EF <1> out dx, ax ; Enable variable rate audio

2177 <1>

2178 <1> ;call delay1\_4ms

2179 <1> ;call delay1\_4ms

2180 <1> ;call delay1\_4ms

2181 <1> ;call delay1\_4ms

2182 <1>

2183 000123D0 66A1[EE6B0100] <1> mov ax, [audio\_freq] ; sample rate

2184 <1>

2185 000123D6 668B15[F46B0100] <1> mov dx, [NAMBAR]

2186 000123DD 6683C22C <1> add dx, CODEC\_PCM\_FRONT\_DACRATE\_REG ; 2Ch

2187 000123E1 66EF <1> out dx, ax ; out sample rate

2188 <1>

2189 <1> ;call delay1\_4ms

2190 <1> ;call delay1\_4ms

2191 <1> ;call delay1\_4ms

2192 <1> ;call delay1\_4ms

2193 <1>

2194 <1> ;

2195 <1> ; register reset the DMA engine. This may cause a pop noise on the output

2196 <1> ; lines when the device is reset. Prolly a better idea to mute output, then

2197 <1> ; reset.

2198 <1> ;

2199 000123E3 668B15[F66B0100] <1> mov dx, [NABMBAR]

2200 000123EA 6683C21B <1> add dx, PO\_CR\_REG ; set pointer to Cntl reg

2201 000123EE B002 <1> mov al, RR ; set reset

2202 000123F0 EE <1> out dx, al ; self clearing bit

2203 <1> ;

2204 <1> ; mov edi, audio\_bdl\_buff

2205 <1> ; mov edx, [audio\_dmabuff\_size]

2206 <1> ; shr edx, 1

2207 <1> ; mov ecx, 32/2

2208 <1> ;ac97\_set\_bdl\_buffer:

2209 <1> ; ; 1st half of DMA buffer

2210 <1> ; mov eax, [audio\_dma\_buff]

2211 <1> ; push eax

2212 <1> ; stosd

2213 <1> ; mov eax, edx ; dma buffer size / 2

2214 <1> ; or eax, IOC+BUP

2215 <1> ; stosd

2216 <1> ; pop eax

2217 <1> ; ; 2nd half of DMA buffer

2218 <1> ; add eax, edx

2219 <1> ; stosd

2220 <1> ; mov eax, edx ; dma buffer size / 2

2221 <1> ; or eax, IOC+BUP

2222 <1> ; stosd

2223 <1> ; loop ac97\_set\_bdl\_buffer

2224 <1>

2225 <1> ; tell the DMA engine where to find our list of Buffer Descriptors.

2226 <1> ; this 32bit value is a flat mode memory offset (ie no segment:offset)

2227 <1> ;

2228 <1> ; write NABMBAR+10h with offset of buffer descriptor list

2229 <1> ;

2230 000123F1 B8[F86B0100] <1> mov eax, audio\_bdl\_buff

2231 000123F6 668B15[F66B0100] <1> mov dx, [NABMBAR]

2232 000123FD 6683C210 <1> add dx, PO\_BDBAR\_REG

2233 00012401 EF <1> out dx, eax

2234 <1> ;

2235 <1> ; All set. Let's play some music.

2236 <1> ;

2237 <1> ;

2238 00012402 B81F000000 <1> mov eax, 31

2239 00012407 E816000000 <1> call set\_ac97\_LastValidIndex

2240 <1>

2241 0001240C C605[F06B0100]01 <1> mov byte [audio\_play\_cmd], 1 ; play command (do not stop) !

2242 <1>

2243 <1> ac97\_play: ; continue to play (after pause)

2244 <1> ; 11/06/2017

2245 <1> ; 29/05/2017

2246 <1> ; 28/05/2017

2247 00012413 668B15[F66B0100] <1> mov dx, [NABMBAR]

2248 0001241A 6683C21B <1> add dx, PO\_CR\_REG ; PCM out control register

2249 0001241E B011 <1> mov al, IOCE+RPBM ; 29/05/2017

2250 <1> ;mov al, 1Dh ; (Ref: KolibriOS, intelac97.asm, 'play:')

2251 00012420 EE <1> out dx, al ; set start!

2252 <1>

2253 <1> ;mov byte [audio\_play\_cmd], 1 ; play command (do not stop) !

2254 <1>

2255 00012421 C3 <1> retn

2256 <1>

2257 <1> ;input AL = index # to stop on

2258 <1> set\_ac97\_LastValidIndex:

2259 <1> ; 28/05/2017

2260 <1> ; Derived from 'setLastValidIndex' procedure in 'ICHWAV.ASM'

2261 <1> ; .wav player for DOS by Jeff Leyda (02/09/2002)

2262 00012422 668B15[F66B0100] <1> mov dx, [NABMBAR]

2263 00012429 6683C215 <1> add dx, PO\_LVI\_REG

2264 0001242D EE <1> out dx, al

2265 <1> ;mov [audio\_lvi], al ; for ac97\_int\_handler

2266 0001242E C3 <1> retn

2267 <1>

2268 <1> ac97\_volume:

2269 <1> ; 28/05/2017

2270 <1> ; bl = component (0 = master/playback/lineout volume)

2271 <1> ; cl = left channel volume level (0 to 31)

2272 <1> ; ch = right channel volume level (0 to 31)

2273 <1>

2274 0001242F 08DB <1> or bl, bl

2275 00012431 7523 <1> jnz short ac97\_vol\_1 ; temporary !

2276 00012433 66B81F1F <1> mov ax, 1F1Fh ; 31,31

2277 00012437 38C1 <1> cmp cl, al

2278 00012439 771B <1> ja short ac97\_vol\_1 ; temporary !

2279 0001243B 38E5 <1> cmp ch, ah

2280 0001243D 7717 <1> ja short ac97\_vol\_1 ; temporary !

2281 0001243F 66890D[F26B0100] <1> mov [audio\_master\_volume], cx

2282 00012446 6629C8 <1> sub ax, cx

2283 00012449 668B15[F46B0100] <1> mov dx, [NAMBAR]

2284 00012450 6683C202 <1> add dx, CODEC\_MASTER\_VOL\_REG ; 02h ; Line Out

2285 00012454 66EF <1> out dx, ax

2286 <1> ac97\_vol\_1:

2287 00012456 C3 <1> retn

2288 <1>

2289 <1> ac97\_int\_handler:

2290 <1> ; 12/10/2017

2291 <1> ; 10/10/2017

2292 <1> ; 09/10/2017

2293 <1> ; 13/06/2017, 13/06/2017

2294 <1> ; 10/06/2017, 11/06/2017

2295 <1> ; Interrupt Handler for AC97 (ICH) Audio Controller

2296 <1> ; Note: called by 'dev\_IRQ\_service'

2297 <1> ; 28/05/2017

2298 <1>

2299 <1> ;push eax ; \* must be saved !

2300 <1> ;push edx

2301 <1> ;push ecx

2302 <1> ;push ebx ; \* must be saved !

2303 <1> ;push esi

2304 <1> ;push edi

2305 <1>

2306 <1> ;cmp byte [audio\_busy], 1

2307 <1> ;jnb \_ac97\_ih2 ; busy !

2308 <1>

2309 00012457 66BA3000 <1> mov dx, GLOB\_STS\_REG

2310 0001245B 660315[F66B0100] <1> add dx, [NABMBAR]

2311 00012462 ED <1> in eax, dx

2312 <1>

2313 00012463 83F8FF <1> cmp eax, 0FFFFFFFFh ; -1

2314 00012466 0F849A000000 <1> je \_ac97\_ih3 ; exit

2315 <1>

2316 0001246C A940000000 <1> test eax, 40h ; PCM Out Interrupt

2317 00012471 750E <1> jnz short \_ac97\_ih0

2318 <1>

2319 00012473 85C0 <1> test eax, eax

2320 00012475 0F848B000000 <1> jz \_ac97\_ih3 ; exit

2321 <1>

2322 <1> ;mov dx, GLOB\_STS\_REG

2323 <1> ;add dx, [NABMBAR]

2324 0001247B EF <1> out dx, eax

2325 <1>

2326 0001247C E985000000 <1> jmp \_ac97\_ih3 ; exit

2327 <1>

2328 <1> \_ac97\_ih0:

2329 00012481 50 <1> push eax

2330 <1> ; 09/10/2017

2331 00012482 803D[F06B0100]01 <1> cmp byte [audio\_play\_cmd], 1

2332 00012489 727C <1> jb short \_ac97\_ih4 ; stop command !

2333 <1>

2334 <1> ;mov byte [audio\_busy], 1

2335 <1>

2336 <1> ;mov al, 10h

2337 <1> ;mov dx, PO\_CR\_REG

2338 <1> ;add dx, [NABMBAR]

2339 <1> ;out dx, al

2340 <1>

2341 0001248B 66B81C00 <1> mov ax, 1Ch ; FIFOE(=16)+BCIS(=8)+LVBCI(=4)

2342 0001248F 66BA1600 <1> mov dx, PO\_SR\_REG

2343 00012493 660315[F66B0100] <1> add dx, [NABMBAR]

2344 0001249A 66EF <1> out dx, ax

2345 <1>

2346 0001249C 66BA1400 <1> mov dx, PO\_CIV\_REG

2347 000124A0 660315[F66B0100] <1> add dx, [NABMBAR]

2348 000124A7 EC <1> in al, dx

2349 <1>

2350 <1> ;cmp al, [audio\_civ] ; [audio\_flag]

2351 <1> ;je short \_ac97\_ih2

2352 <1>

2353 000124A8 A2[F16B0100] <1> mov [audio\_civ], al

2354 000124AD FEC8 <1> dec al

2355 <1> ;inc al ; 11/06/2017

2356 000124AF 241F <1> and al, 1Fh

2357 <1>

2358 000124B1 66BA1500 <1> mov dx, PO\_LVI\_REG

2359 000124B5 660315[F66B0100] <1> add dx, [NABMBAR]

2360 000124BC EE <1> out dx, al

2361 <1>

2362 <1> ; 12/10/2017

2363 000124BD A0[F16B0100] <1> mov al, [audio\_civ]

2364 000124C2 FEC0 <1> inc al

2365 000124C4 2401 <1> and al, 1

2366 000124C6 A2[E46B0100] <1> mov [audio\_flag], al

2367 <1> ;; [audio\_flag] : 0 = Buffer 1, 1 = Buffer 2

2368 <1> ;

2369 000124CB 58 <1> pop eax

2370 <1> ;

2371 000124CC 83E040 <1> and eax, 40h

2372 000124CF 668B15[F66B0100] <1> mov dx, [NABMBAR]

2373 000124D6 6683C230 <1> add dx, GLOB\_STS\_REG

2374 000124DA EF <1> out dx, eax

2375 <1>

2376 <1> ;; 13/06/2017

2377 <1> ;mov al, 11h ; IOCE + RPBM

2378 <1> ;mov dx, PO\_CR\_REG

2379 <1> ;add dx, [NABMBAR]

2380 <1> ;out dx, al

2381 <1>

2382 <1> ac97\_tuneloop:

2383 <1> ; 09/10/2017

2384 000124DB 8B3D[DC6B0100] <1> mov edi, [audio\_dma\_buff]

2385 000124E1 8B0D[E06B0100] <1> mov ecx, [audio\_dmabuff\_size]

2386 000124E7 D1E9 <1> shr ecx, 1 ; dma buff size / 2 = half buffer size

2387 <1>

2388 <1> ; 12/10/2017

2389 000124E9 803D[E46B0100]00 <1> cmp byte [audio\_flag], 0

2390 000124F0 7702 <1> ja short \_ac97\_ih1 ; Playing Half Buffer 2 (Current: FLAG)

2391 <1> ; Playing Half Buffer 1 (Current: EOL)

2392 000124F2 01CF <1> add edi, ecx

2393 <1> \_ac97\_ih1:

2394 <1> ; Update half buffer 2 while playing half buffer 1 (next: FLAG)

2395 <1> ; Update half buffer 1 while playing half buffer 2 (next: EOL)

2396 <1>

2397 000124F4 8B35[D46B0100] <1> mov esi, [audio\_p\_buffer] ; phy addr of audio buff

2398 000124FA C1E902 <1> shr ecx, 2 ; half buff size / 4

2399 000124FD F3A5 <1> rep movsd

2400 <1>

2401 <1> ; 10/10/2017

2402 <1> ; switch flag value

2403 000124FF 8035[E46B0100]01 <1> xor byte [audio\_flag], 1

2404 <1> ; 12/10/2017

2405 <1> ; [audio\_flag] = 0 : Playing dma half buffer 2 (even index value)

2406 <1> ; Next buffer (to update) is dma half buff 1

2407 <1> ; = 1 : Playing dma half buffer 1 (odd index value)

2408 <1> ; Next buffer (to update) is dma half buff 2

2409 <1>

2410 <1> \_ac97\_ih2:

2411 <1> ;mov byte [audio\_busy], 0

2412 <1> \_ac97\_ih3:

2413 <1> ;pop edi

2414 <1> ;pop esi

2415 <1> ;pop ebx ; \* must be restored !

2416 <1> ;pop ecx

2417 <1> ;pop edx

2418 <1> ;pop eax ; \* must be restored !

2419 <1>

2420 00012506 C3 <1> retn

2421 <1>

2422 <1> \_ac97\_ih4:

2423 <1> ; 09/10/2017

2424 00012507 E818000000 <1> call \_ac97\_stop

2425 <1> ;

2426 0001250C 58 <1> pop eax

2427 <1> ;

2428 0001250D 83E040 <1> and eax, 40h

2429 00012510 668B15[F66B0100] <1> mov dx, [NABMBAR]

2430 00012517 6683C230 <1> add dx, GLOB\_STS\_REG

2431 0001251B EF <1> out dx, eax

2432 <1>

2433 <1> ;; 13/06/2017

2434 <1> ;mov al, 11h ; IOCE + RPBM

2435 <1> ;dx, PO\_CR\_REG

2436 <1> ;add dx, [NABMBAR]

2437 <1> ;out dx, al

2438 <1>

2439 <1> ; 10/10/2017

2440 <1> ;jmp short \_ac97\_ih3 ; exit

2441 0001251C C3 <1> retn

2442 <1>

2443 <1> ac97\_stop:

2444 <1> ; 28/05/2017

2445 0001251D C605[F06B0100]00 <1> mov byte [audio\_play\_cmd], 0 ; stop !

2446 <1> \_ac97\_stop: ; 09/10/2017

2447 <1> ; 29/05/2017

2448 <1> ;mov dx, [NABMBAR]

2449 <1> ;add dx, PO\_CR\_REG

2450 <1> ;mov al, 0

2451 <1> ;out dx, al

2452 <1>

2453 <1> ; 11/06/2017

2454 00012524 30C0 <1> xor al, al ; 0

2455 00012526 E813000000 <1> call ac97\_po\_cmd

2456 <1>

2457 <1> ; (Ref: KolibriOS, intelac97.asm, 'stop:')

2458 <1> ; Clear FIFOE, BCIS, LVBCI (Ref: Intel ICH hub manual)

2459 0001252B 66B81C00 <1> mov ax, 1Ch

2460 0001252F 668B15[F66B0100] <1> mov dx, [NABMBAR]

2461 00012536 6683C216 <1> add dx, PO\_SR\_REG

2462 0001253A 66EF <1> out dx, ax

2463 <1>

2464 <1> ;retn

2465 <1>

2466 <1> ; 11/06/2017

2467 0001253C B002 <1> mov al, RR

2468 <1> ac97\_po\_cmd:

2469 <1> ;11/06/2017

2470 <1> ; 29/05/2017

2471 0001253E 668B15[F66B0100] <1> mov dx, [NABMBAR]

2472 00012545 6683C21B <1> add dx, PO\_CR\_REG ; PCM out control register

2473 00012549 EE <1> out dx, al

2474 0001254A C3 <1> retn

2475 <1>

2476 <1> ac97\_pause:

2477 <1> ; 11/06/2017

2478 <1> ; 29/05/2017

2479 0001254B B010 <1> mov al, IOCE

2480 0001254D EBEF <1> jmp short ac97\_po\_cmd

2481 <1>

2482 <1> reset\_ac97\_controller:

2483 <1> ; 10/06/2017

2484 <1> ; 29/05/2017

2485 <1> ; 28/05/2017

2486 <1> ; reset AC97 audio controller registers

2487 0001254F 31C0 <1> xor eax, eax

2488 00012551 66BA0B00 <1> mov dx, PI\_CR\_REG

2489 00012555 660315[F66B0100] <1> add dx, [NABMBAR]

2490 0001255C EE <1> out dx, al

2491 <1>

2492 0001255D 66BA1B00 <1> mov dx, PO\_CR\_REG

2493 00012561 660315[F66B0100] <1> add dx, [NABMBAR]

2494 00012568 EE <1> out dx, al

2495 <1>

2496 00012569 66BA2B00 <1> mov dx, MC\_CR\_REG

2497 0001256D 660315[F66B0100] <1> add dx, [NABMBAR]

2498 00012574 EE <1> out dx, al

2499 <1>

2500 00012575 B002 <1> mov al, RR

2501 00012577 66BA0B00 <1> mov dx, PI\_CR\_REG

2502 0001257B 660315[F66B0100] <1> add dx, [NABMBAR]

2503 00012582 EE <1> out dx, al

2504 <1>

2505 00012583 66BA1B00 <1> mov dx, PO\_CR\_REG

2506 00012587 660315[F66B0100] <1> add dx, [NABMBAR]

2507 0001258E EE <1> out dx, al

2508 <1>

2509 0001258F 66BA2B00 <1> mov dx, MC\_CR\_REG

2510 00012593 660315[F66B0100] <1> add dx, [NABMBAR]

2511 0001259A EE <1> out dx, al

2512 <1>

2513 0001259B C3 <1> retn

2514 <1>

2515 <1> ac97\_reset:

2516 <1> ; 10/06/2017

2517 <1> ; 29/05/2017

2518 <1> ; 28/05/2017

2519 0001259C E8AEFFFFFF <1> call reset\_ac97\_controller

2520 <1> ; 29/05/2017

2521 <1> ;jmp reset\_ac97\_codec

2522 <1> reset\_ac97\_codec:

2523 <1> ; 28/05/2017 - Erdogan Tan (Ref: KolibriOS, intelac97.asm)

2524 000125A1 66BA2C00 <1> mov dx, GLOB\_CNT\_REG ; 2Ch

2525 000125A5 660315[F66B0100] <1> add dx, [NABMBAR]

2526 000125AC ED <1> in eax, dx

2527 <1>

2528 000125AD A902000000 <1> test eax, 2

2529 000125B2 7407 <1> jz short \_r\_ac97codec\_cold

2530 <1>

2531 000125B4 E80F000000 <1> call warm\_ac97codec\_reset

2532 000125B9 7308 <1> jnc short \_r\_ac97codec\_ok

2533 <1> \_r\_ac97codec\_cold:

2534 000125BB E83D000000 <1> call cold\_ac97codec\_reset

2535 000125C0 7301 <1> jnc short \_r\_ac97codec\_ok

2536 <1>

2537 <1> ; 16/04/2017

2538 <1> ;xor eax, eax ; timeout error

2539 <1> ;stc

2540 000125C2 C3 <1> retn

2541 <1>

2542 <1> \_r\_ac97codec\_ok:

2543 000125C3 31C0 <1> xor eax, eax

2544 <1> ;mov al, VIA\_ACLINK\_C00\_READY ; 1

2545 000125C5 FEC0 <1> inc al

2546 000125C7 C3 <1> retn

2547 <1>

2548 <1> warm\_ac97codec\_reset:

2549 <1> ; 28/05/2017 - Erdogan Tan (Ref: KolibriOS, intelac97.asm)

2550 000125C8 B806000000 <1> mov eax, 6

2551 000125CD 66BA2C00 <1> mov dx, GLOB\_CNT\_REG ; 2Ch

2552 000125D1 660315[F66B0100] <1> add dx, [NABMBAR]

2553 000125D8 EF <1> out dx, eax

2554 <1>

2555 000125D9 B90A000000 <1> mov ecx, 10 ; total 1s

2556 <1> \_warm\_ac97c\_rst\_wait:

2557 000125DE 51 <1> push ecx

2558 000125DF E8D6F5FFFF <1> call delay\_100ms

2559 000125E4 59 <1> pop ecx

2560 <1>

2561 000125E5 66BA3000 <1> mov dx, GLOB\_STS\_REG ; 30h

2562 000125E9 660315[F66B0100] <1> add dx, [NABMBAR]

2563 000125F0 ED <1> in eax, dx

2564 <1>

2565 000125F1 A900030010 <1> test eax, CTRL\_ST\_CREADY

2566 000125F6 7504 <1> jnz short \_warm\_ac97c\_rst\_ok

2567 <1>

2568 000125F8 49 <1> dec ecx

2569 000125F9 75E3 <1> jnz short \_warm\_ac97c\_rst\_wait

2570 <1>

2571 <1> \_warm\_ac97c\_rst\_fail:

2572 000125FB F9 <1> stc

2573 <1> \_warm\_ac97c\_rst\_ok:

2574 000125FC C3 <1> retn

2575 <1>

2576 <1> cold\_ac97codec\_reset:

2577 <1> ; 28/05/2017 - Erdogan Tan (Ref: KolibriOS, intelac97.asm)

2578 000125FD B802000000 <1> mov eax, 2

2579 00012602 66BA2C00 <1> mov dx, GLOB\_CNT\_REG ; 2Ch

2580 00012606 660315[F66B0100] <1> add dx, [NABMBAR]

2581 0001260D EF <1> out dx, eax

2582 <1>

2583 0001260E E8A7F5FFFF <1> call delay\_100ms ; wait 100 ms

2584 00012613 E8A2F5FFFF <1> call delay\_100ms ; wait 100 ms

2585 00012618 E89DF5FFFF <1> call delay\_100ms ; wait 100 ms

2586 0001261D E898F5FFFF <1> call delay\_100ms ; wait 100 ms

2587 <1>

2588 00012622 B910000000 <1> mov ecx, 16 ; total 20\*100 ms = 2s

2589 <1> \_cold\_ac97c\_rst\_wait:

2590 00012627 66BA3000 <1> mov dx, GLOB\_STS\_REG ; 30h

2591 0001262B 660315[F66B0100] <1> add dx, [NABMBAR]

2592 00012632 ED <1> in eax, dx

2593 <1>

2594 00012633 A900030010 <1> test eax, CTRL\_ST\_CREADY

2595 00012638 750B <1> jnz short \_cold\_ac97c\_rst\_ok

2596 <1>

2597 0001263A 51 <1> push ecx

2598 0001263B E87AF5FFFF <1> call delay\_100ms

2599 00012640 59 <1> pop ecx

2600 <1>

2601 00012641 49 <1> dec ecx

2602 00012642 75E3 <1> jnz short \_cold\_ac97c\_rst\_wait

2603 <1>

2604 <1> \_cold\_ac97c\_rst\_fail:

2605 00012644 F9 <1> stc

2606 <1> \_cold\_ac97c\_rst\_ok:

2607 00012645 C3 <1> retn

2608 <1>

2609 <1> sb16\_current\_sound\_data:

2610 <1> ; 20/08/2017

2611 <1> ; 24/06/2017

2612 <1> ; 22/06/2017

2613 <1> ; get current sound (PCM out) data for graphics

2614 <1> ; (for Sound Blaster 16)

2615 <1> ; ebx = Physical address (on page boundary)

2616 <1> ; ecx = Byte count

2617 <1> ; [audio\_buff\_size]

2618 <1>

2619 <1> ;;mov edi, [audio\_buff\_size]

2620 <1> ;mov edi, [audio\_dmabuff\_size]

2621 <1> ;mov esi, [audio\_dma\_buff]

2622 00012646 39CF <1> cmp edi, ecx

2623 00012648 7302 <1> jnb short sb16\_gcd\_0

2624 0001264A 89F9 <1> mov ecx, edi

2625 <1> sb16\_gcd\_0:

2626 <1> ; 20/08/2017

2627 0001264C 803D[EC6B0100]10 <1> cmp byte [audio\_bps], 16

2628 00012653 750F <1> jne short sb16\_gcd\_1 ; 8 bit DMA channel

2629 00012655 E4C6 <1> in al, 0C6h ; DMA channel 5 count register

2630 00012657 88C2 <1> mov dl, al

2631 00012659 E4C6 <1> in al, 0C6h

2632 0001265B 88C6 <1> mov dh, al

2633 0001265D 0FB7C2 <1> movzx eax, dx

2634 00012660 D1E0 <1> shl eax, 1 ; word count -> byte count

2635 00012662 EB4E <1> jmp short sb16\_gcd\_2

2636 <1> sb16\_gcd\_1:

2637 00012664 E403 <1> in al, 03h ; DMA channel 1 count register

2638 00012666 88C2 <1> mov dl, al

2639 00012668 E403 <1> in al, 03h

2640 0001266A 88C6 <1> mov dh, al

2641 0001266C 0FB7C2 <1> movzx eax, dx

2642 0001266F EB41 <1> jmp short sb16\_gcd\_2

2643 <1> ;sb16\_gcd\_2:

2644 <1> ; cmp eax, ecx

2645 <1> ; jnb short sb16\_gcd\_3

2646 <1> ; ; remain count < graphics bytes

2647 <1> ; mov eax, ecx ; fix remain count to data size

2648 <1> ;sb16\_gcd\_3:

2649 <1> ; sub edi, eax

2650 <1> ; jna short sb16\_gcd\_4

2651 <1> ; add esi, edi ; dma buffer offset

2652 <1> ;sb16\_gcd\_4:

2653 <1> ; mov edi, ebx ; buffer address (for graphics)

2654 <1> ; mov [u.r0], ecx

2655 <1> ; rep movsb

2656 <1> ; retn

2657 <1>

2658 <1> get\_current\_sound\_data:

2659 <1> ; 24/06/2017

2660 <1> ; 22/06/2017

2661 <1> ; get current sound (PCM out) data for graphics

2662 <1> ;

2663 <1> ; ebx = Physical address (on page boundary)

2664 <1> ; ecx = Byte count

2665 <1> ; [audio\_buff\_size]

2666 <1>

2667 <1> ;mov edi, [audio\_buff\_size]

2668 00012671 8B3D[E06B0100] <1> mov edi, [audio\_dmabuff\_size]

2669 00012677 8B35[DC6B0100] <1> mov esi, [audio\_dma\_buff]

2670 0001267D 803D[BD6B0100]02 <1> cmp byte [audio\_device], 2

2671 00012684 72C0 <1> jb short sb16\_current\_sound\_data ; = 1

2672 00012686 D1EF <1> shr edi, 1

2673 00012688 39CF <1> cmp edi, ecx

2674 0001268A 7302 <1> jnb short gcd\_0

2675 0001268C 89F9 <1> mov ecx, edi

2676 <1> gcd\_0:

2677 0001268E 803D[BD6B0100]03 <1> cmp byte [audio\_device], 3

2678 00012695 7232 <1> jb short ac97\_current\_sound\_data ; = 2

2679 <1> ; = 3

2680 <1> vt8233\_current\_sound\_data:

2681 <1> ; 22/06/2017

2682 <1> ; 21/06/2017

2683 <1> ; get current sound (PCM out) data for graphics

2684 <1> ; (for VT 8233, VT 8237R)

2685 <1> ; ebx = Physical address (on page boundary)

2686 <1> ; ecx = Byte count

2687 <1> ; [audio\_buff\_size]

2688 <1>

2689 <1> ;;mov edi, [audio\_buff\_size]

2690 <1> ;mov edi, [audio\_dmabuff\_size]

2691 <1> ;mov esi, [audio\_dma\_buff]

2692 <1> ;shr edi, 1

2693 <1> ;cmp edi, ecx

2694 <1> ;jnb short vt8233\_gcd\_1

2695 <1> ;mov ecx, edi

2696 <1> vt8233\_gcd\_1:

2697 00012697 BA0C000000 <1> mov edx, VIA\_REG\_OFFSET\_CURR\_COUNT

2698 0001269C E879F5FFFF <1> call ctrl\_io\_r32

2699 000126A1 89C2 <1> mov edx, eax ; remain count (bits 23-0),

2700 <1> ; SGD index (bits 31-24)

2701 000126A3 81E200000001 <1> and edx, 1000000h ; SGD index (0 = 1st half)

2702 000126A9 7402 <1> jz short vt8233\_gcd\_2

2703 <1> ; the second half of DMA buffer

2704 000126AB 01FE <1> add esi, edi

2705 <1> vt8233\_gcd\_2:

2706 000126AD 25FFFFFF00 <1> and eax, 0FFFFFFh ; bits 23-0

2707 <1> ac97\_gcd\_2:

2708 <1> sb16\_gcd\_2:

2709 000126B2 39C8 <1> cmp eax, ecx

2710 000126B4 7302 <1> jnb short vt8233\_gcd\_3

2711 <1> ; remain count < graphics bytes

2712 000126B6 89C8 <1> mov eax, ecx ; fix remain count to data size

2713 <1> vt8233\_gcd\_3:

2714 000126B8 29C7 <1> sub edi, eax

2715 000126BA 7602 <1> jna short vt8233\_gcd\_4

2716 000126BC 01FE <1> add esi, edi ; dma buffer offset

2717 <1> vt8233\_gcd\_4:

2718 000126BE 89DF <1> mov edi, ebx ; buffer address (for graphics)

2719 000126C0 890D[64030300] <1> mov [u.r0], ecx

2720 000126C6 F3A4 <1> rep movsb

2721 <1> vt8233\_gcd\_5:

2722 000126C8 C3 <1> retn

2723 <1>

2724 <1> ac97\_current\_sound\_data:

2725 <1> ; 23/06/2017

2726 <1> ; 22/06/2017

2727 <1> ; get current sound (PCM out) data for graphics

2728 <1> ; (for AC'97, ICH)

2729 <1> ; ebx = Physical address (on page boundary)

2730 <1> ; ecx = Byte count

2731 <1> ; [audio\_buff\_size]

2732 <1>

2733 <1> ;;mov edi, [audio\_buff\_size]

2734 <1> ;mov edi, [audio\_dmabuff\_size]

2735 <1> ;mov esi, [audio\_dma\_buff]

2736 <1> ;shr edi, 1

2737 <1> ;cmp edi, ecx

2738 <1> ;jnb short ac97\_gcd\_0

2739 <1> ;mov ecx, edi

2740 <1> ac97\_gcd\_0:

2741 000126C9 66BA1400 <1> mov dx, PO\_CIV\_REG ; Position In Current Buff Reg

2742 000126CD 660315[F66B0100] <1> add dx, [NABMBAR]

2743 000126D4 EC <1> in al, dx ; current index value

2744 000126D5 A801 <1> test al, 1

2745 000126D7 7402 <1> jz short ac97\_gcd\_1

2746 000126D9 01FE <1> add esi, edi

2747 <1> ac97\_gcd\_1:

2748 000126DB 31C0 <1> xor eax, eax

2749 000126DD 66BA1800 <1> mov dx, PO\_PICB\_REG ; Position In Current Buff Reg

2750 000126E1 660315[F66B0100] <1> add dx, [NABMBAR]

2751 000126E8 66ED <1> in ax, dx ; remain dwords

2752 000126EA C1E002 <1> shl eax, 2 ; remain bytes ; 23/06/2017

2753 000126ED EBC3 <1> jmp short ac97\_gcd\_2

2754 <1> ; cmp eax, ecx

2755 <1> ; jnb short ac97\_gcd\_2

2756 <1> ; ; remain count < graphics bytes

2757 <1> ; mov eax, ecx ; fix remain count to data size

2758 <1> ;ac97\_gcd\_2:

2759 <1> ; sub edi, eax

2760 <1> ; jna short ac97\_gcd\_3

2761 <1> ; add esi, edi ; dma buffer offset

2762 <1> ;ac97\_gcd\_3:

2763 <1> ; mov edi, ebx ; buffer address (for graphics)

2764 <1> ; mov [u.r0], ecx

2765 <1> ; rep movsb

2766 <1> ; retn

2767 <1>

2768 <1> sb16\_get\_dma\_buff\_off:

2769 <1> ; 28/10/2017

2770 <1> ; 24/06/2017

2771 <1> ; 22/06/2017

2772 <1> ; get current (PCM OUT DMA buffer) pointer

2773 <1> ; (for Sound Blaster 16)

2774 <1>

2775 <1> ;mov ecx, [audio\_dmabuff\_size]

2776 <1> ;xor ebx, ebx

2777 <1> ;shr ecx, 1

2778 <1> sb16\_gdmabo\_0:

2779 <1> ; 28/10/2017

2780 000126EF 803D[EC6B0100]10 <1> cmp byte [audio\_bps], 16

2781 000126F6 750F <1> jne short sb16\_gdmabo\_1 ; 8 bit DMA channel

2782 <1> ; 16 bit DMA channel

2783 000126F8 E4C6 <1> in al, 0C6h ; DMA channel 5 count register

2784 000126FA 88C2 <1> mov dl, al

2785 000126FC E4C6 <1> in al, 0C6h

2786 000126FE 88C6 <1> mov dh, al

2787 00012700 0FB7C2 <1> movzx eax, dx

2788 00012703 D1E0 <1> shl eax, 1 ; word count -> byte count

2789 00012705 EB3D <1> jmp short sb16\_gdmabo\_2

2790 <1> sb16\_gdmabo\_1:

2791 00012707 E403 <1> in al, 03h ; DMA channel 1 count register

2792 00012709 88C2 <1> mov dl, al

2793 0001270B E403 <1> in al, 03h

2794 0001270D 88C6 <1> mov dh, al

2795 0001270F 0FB7C2 <1> movzx eax, dx

2796 00012712 EB30 <1> jmp short sb16\_gdmabo\_2

2797 <1>

2798 <1> get\_dma\_buffer\_offset:

2799 <1> ; 24/06/2017

2800 <1> ; 22/06/2017

2801 <1> ; get current sound (PCM out) data for graphics

2802 <1> ;

2803 <1> ; ebx = Physical address (on page boundary)

2804 <1> ; ecx = Byte count

2805 <1> ; [audio\_buff\_size]

2806 <1>

2807 00012714 8B0D[E06B0100] <1> mov ecx, [audio\_dmabuff\_size]

2808 0001271A 31DB <1> xor ebx, ebx

2809 <1> gdmabo\_0:

2810 0001271C 803D[BD6B0100]02 <1> cmp byte [audio\_device], 2

2811 00012723 72CA <1> jb short sb16\_get\_dma\_buff\_off

2812 00012725 742A <1> je short ac97\_get\_dma\_buff\_off

2813 <1>

2814 <1> vt8233\_get\_dma\_buff\_off:

2815 <1> ; 24/06/2017

2816 <1> ; 22/06/2017

2817 <1> ; get current (PCM OUT DMA buffer) pointer

2818 <1> ; (for VT 8233, VT 8237R)

2819 <1>

2820 <1> ;mov ecx, [audio\_dmabuff\_size]

2821 <1> ;xor ebx, ebx

2822 00012727 D1E9 <1> shr ecx, 1

2823 <1> vt8233\_gdmabo\_0:

2824 00012729 BA0C000000 <1> mov edx, VIA\_REG\_OFFSET\_CURR\_COUNT

2825 0001272E E8E7F4FFFF <1> call ctrl\_io\_r32

2826 00012733 89C2 <1> mov edx, eax ; remain count (bits 23-0),

2827 <1> ; SGD index (bits 31-24)

2828 00012735 81E200000001 <1> and edx, 1000000h ; SGD index (0 = 1st half)

2829 0001273B 7402 <1> jz short vt8233\_gdmabo\_1

2830 <1> ; the second half of DMA buffer

2831 0001273D 89CB <1> mov ebx, ecx

2832 <1> vt8233\_gdmabo\_1:

2833 0001273F 25FFFFFF00 <1> and eax, 0FFFFFFh ; bits 23-0

2834 <1> sb16\_gdmabo\_2:

2835 <1> ac97\_gdmabo\_2:

2836 00012744 29C1 <1> sub ecx, eax

2837 00012746 7602 <1> jna short vt8233\_gdmabo\_2

2838 00012748 01CB <1> add ebx, ecx ; dma buffer offset

2839 <1> vt8233\_gdmabo\_2:

2840 0001274A 891D[64030300] <1> mov [u.r0], ebx

2841 00012750 C3 <1> retn

2842 <1>

2843 <1> ac97\_get\_dma\_buff\_off:

2844 <1> ; 24/06/2017

2845 <1> ; 22/06/2017

2846 <1> ; get current (PCM OUT DMA buffer) pointer

2847 <1> ; (for AC'97, ICH)

2848 <1> ; ebx = Physical address (on page boundary)

2849 <1> ; ecx = Byte count

2850 <1> ; [audio\_buff\_size]

2851 <1>

2852 <1> ;mov ecx, [audio\_dmabuff\_size]

2853 <1> ;xor ebx, ebx

2854 00012751 D1E9 <1> shr ecx, 1

2855 <1> ac97\_gdmabo\_0:

2856 00012753 66BA1400 <1> mov dx, PO\_CIV\_REG ; Position In Current Buff Reg

2857 00012757 660315[F66B0100] <1> add dx, [NABMBAR]

2858 0001275E EC <1> in al, dx ; current index value

2859 0001275F A801 <1> test al, 1

2860 00012761 7402 <1> jz short ac97\_gdmabo\_1

2861 00012763 89CB <1> mov ebx, ecx

2862 <1> ac97\_gdmabo\_1:

2863 00012765 31C0 <1> xor eax, eax

2864 00012767 66BA1800 <1> mov dx, PO\_PICB\_REG ; Position In Current Buff Reg

2865 0001276B 660315[F66B0100] <1> add dx, [NABMBAR]

2866 00012772 66ED <1> in ax, dx ; remain dwords

2867 00012774 EBCE <1> jmp short ac97\_gdmabo\_2

2642

2643 00012776 90<rept> align 4

2644

2645 %include 'vgadata.s' ; 04/07/2016

1 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2 <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0 - vgadata.s (palette and fond data)

3 <1> ; -----------------------------------------------------------------------------

4 <1> ; Last Update: 04/07/2016

5 <1> ; -----------------------------------------------------------------------------

6 <1> ; Beginning: 16/01/2016

7 <1> ; -----------------------------------------------------------------------------

8 <1> ; Assembler: NASM version 2.11 (trdos386.s)

9 <1> ; -----------------------------------------------------------------------------

10 <1> ; Turkish Rational DOS

11 <1> ; Operating System Project v2.0 by ERDOGAN TAN (Beginning: 04/01/2016)

12 <1> ;

13 <1> ; Derived from 'Plex86/Bochs VGABios' source code, vgabios-0.7a (2011)

14 <1> ; by the LGPL VGABios Developers Team (2001-2008), 'vgatables.h'

15 <1> ;

16 <1> ; Oracle VirtualBox 5.0.24 VGABios Source Code

17 <1> ; ('vgabios.c', 'vgatables.h', 'vgafonts.h', 'vgarom.asm')

18 <1> ;

19 <1> ; Palette and font data in assembly language format:

20 <1> ; 'VBoxVgaBiosAlternative.asm'

21 <1>

22 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

23 <1>

24 <1> ; 04/07/2016

25 <1> ; COLOR DATA

26 <1>

27 <1> palette0:

28 00012778 000000000000000000- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

28 00012781 00000000000000 <1>

29 00012788 00000000000000002A- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah

29 00012791 2A2A2A2A2A2A2A <1>

30 00012798 2A2A2A2A2A2A2A2A2A- <1> db 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah

30 000127A1 2A2A2A2A2A2A2A <1>

31 000127A8 2A2A2A2A2A2A2A2A2A- <1> db 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah

31 000127B1 2A2A2A2A2A2A2A <1>

32 000127B8 2A2A2A2A2A2A2A2A3F- <1> db 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh

32 000127C1 3F3F3F3F3F3F3F <1>

33 000127C8 3F3F3F3F3F3F3F3F3F- <1> db 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh

33 000127D1 3F3F3F3F3F3F3F <1>

34 000127D8 000000000000000000- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

34 000127E1 00000000000000 <1>

35 000127E8 00000000000000002A- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah

35 000127F1 2A2A2A2A2A2A2A <1>

36 000127F8 2A2A2A2A2A2A2A2A2A- <1> db 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah

36 00012801 2A2A2A2A2A2A2A <1>

37 00012808 2A2A2A2A2A2A2A2A2A- <1> db 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah

37 00012811 2A2A2A2A2A2A2A <1>

38 00012818 2A2A2A2A2A2A2A2A3F- <1> db 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh

38 00012821 3F3F3F3F3F3F3F <1>

39 00012828 3F3F3F3F3F3F3F3F3F- <1> db 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh

39 00012831 3F3F3F3F3F3F3F <1>

40 <1> palette1:

41 00012838 00000000002A002A00- <1> db 000h, 000h, 000h, 000h, 000h, 02ah, 000h, 02ah, 000h, 000h, 02ah, 02ah, 02ah, 000h, 000h, 02ah

41 00012841 002A2A2A00002A <1>

42 00012848 002A2A15002A2A2A00- <1> db 000h, 02ah, 02ah, 015h, 000h, 02ah, 02ah, 02ah, 000h, 000h, 000h, 000h, 000h, 02ah, 000h, 02ah

42 00012851 000000002A002A <1>

43 00012858 00002A2A2A00002A00- <1> db 000h, 000h, 02ah, 02ah, 02ah, 000h, 000h, 02ah, 000h, 02ah, 02ah, 015h, 000h, 02ah, 02ah, 02ah

43 00012861 2A2A15002A2A2A <1>

44 00012868 15151515153F153F15- <1> db 015h, 015h, 015h, 015h, 015h, 03fh, 015h, 03fh, 015h, 015h, 03fh, 03fh, 03fh, 015h, 015h, 03fh

44 00012871 153F3F3F15153F <1>

45 00012878 153F3F3F153F3F3F15- <1> db 015h, 03fh, 03fh, 03fh, 015h, 03fh, 03fh, 03fh, 015h, 015h, 015h, 015h, 015h, 03fh, 015h, 03fh

45 00012881 151515153F153F <1>

46 00012888 15153F3F3F15153F15- <1> db 015h, 015h, 03fh, 03fh, 03fh, 015h, 015h, 03fh, 015h, 03fh, 03fh, 03fh, 015h, 03fh, 03fh, 03fh

46 00012891 3F3F3F153F3F3F <1>

47 00012898 00000000002A002A00- <1> db 000h, 000h, 000h, 000h, 000h, 02ah, 000h, 02ah, 000h, 000h, 02ah, 02ah, 02ah, 000h, 000h, 02ah

47 000128A1 002A2A2A00002A <1>

48 000128A8 002A2A15002A2A2A00- <1> db 000h, 02ah, 02ah, 015h, 000h, 02ah, 02ah, 02ah, 000h, 000h, 000h, 000h, 000h, 02ah, 000h, 02ah

48 000128B1 000000002A002A <1>

49 000128B8 00002A2A2A00002A00- <1> db 000h, 000h, 02ah, 02ah, 02ah, 000h, 000h, 02ah, 000h, 02ah, 02ah, 015h, 000h, 02ah, 02ah, 02ah

49 000128C1 2A2A15002A2A2A <1>

50 000128C8 15151515153F153F15- <1> db 015h, 015h, 015h, 015h, 015h, 03fh, 015h, 03fh, 015h, 015h, 03fh, 03fh, 03fh, 015h, 015h, 03fh

50 000128D1 153F3F3F15153F <1>

51 000128D8 153F3F3F153F3F3F15- <1> db 015h, 03fh, 03fh, 03fh, 015h, 03fh, 03fh, 03fh, 015h, 015h, 015h, 015h, 015h, 03fh, 015h, 03fh

51 000128E1 151515153F153F <1>

52 000128E8 15153F3F3F15153F15- <1> db 015h, 015h, 03fh, 03fh, 03fh, 015h, 015h, 03fh, 015h, 03fh, 03fh, 03fh, 015h, 03fh, 03fh, 03fh

52 000128F1 3F3F3F153F3F3F <1>

53 <1> palette2:

54 000128F8 00000000002A002A00- <1> db 000h, 000h, 000h, 000h, 000h, 02ah, 000h, 02ah, 000h, 000h, 02ah, 02ah, 02ah, 000h, 000h, 02ah

54 00012901 002A2A2A00002A <1>

55 00012908 002A2A2A002A2A2A00- <1> db 000h, 02ah, 02ah, 02ah, 000h, 02ah, 02ah, 02ah, 000h, 000h, 015h, 000h, 000h, 03fh, 000h, 02ah

55 00012911 001500003F002A <1>

56 00012918 15002A3F2A00152A00- <1> db 015h, 000h, 02ah, 03fh, 02ah, 000h, 015h, 02ah, 000h, 03fh, 02ah, 02ah, 015h, 02ah, 02ah, 03fh

56 00012921 3F2A2A152A2A3F <1>

57 00012928 00150000152A003F00- <1> db 000h, 015h, 000h, 000h, 015h, 02ah, 000h, 03fh, 000h, 000h, 03fh, 02ah, 02ah, 015h, 000h, 02ah

57 00012931 003F2A2A15002A <1>

58 00012938 152A2A3F002A3F2A00- <1> db 015h, 02ah, 02ah, 03fh, 000h, 02ah, 03fh, 02ah, 000h, 015h, 015h, 000h, 015h, 03fh, 000h, 03fh

58 00012941 151500153F003F <1>

59 00012948 15003F3F2A15152A15- <1> db 015h, 000h, 03fh, 03fh, 02ah, 015h, 015h, 02ah, 015h, 03fh, 02ah, 03fh, 015h, 02ah, 03fh, 03fh

59 00012951 3F2A3F152A3F3F <1>

60 00012958 15000015002A152A00- <1> db 015h, 000h, 000h, 015h, 000h, 02ah, 015h, 02ah, 000h, 015h, 02ah, 02ah, 03fh, 000h, 000h, 03fh

60 00012961 152A2A3F00003F <1>

61 00012968 002A3F2A003F2A2A15- <1> db 000h, 02ah, 03fh, 02ah, 000h, 03fh, 02ah, 02ah, 015h, 000h, 015h, 015h, 000h, 03fh, 015h, 02ah

61 00012971 001515003F152A <1>

62 00012978 15152A3F3F00153F00- <1> db 015h, 015h, 02ah, 03fh, 03fh, 000h, 015h, 03fh, 000h, 03fh, 03fh, 02ah, 015h, 03fh, 02ah, 03fh

62 00012981 3F3F2A153F2A3F <1>

63 00012988 15150015152A153F00- <1> db 015h, 015h, 000h, 015h, 015h, 02ah, 015h, 03fh, 000h, 015h, 03fh, 02ah, 03fh, 015h, 000h, 03fh

63 00012991 153F2A3F15003F <1>

64 00012998 152A3F3F003F3F2A15- <1> db 015h, 02ah, 03fh, 03fh, 000h, 03fh, 03fh, 02ah, 015h, 015h, 015h, 015h, 015h, 03fh, 015h, 03fh

64 000129A1 151515153F153F <1>

65 000129A8 15153F3F3F15153F15- <1> db 015h, 015h, 03fh, 03fh, 03fh, 015h, 015h, 03fh, 015h, 03fh, 03fh, 03fh, 015h, 03fh, 03fh, 03fh

65 000129B1 3F3F3F153F3F3F <1>

66 <1> palette3:

67 000129B8 00000000002A002A00- <1> db 000h, 000h, 000h, 000h, 000h, 02ah, 000h, 02ah, 000h, 000h, 02ah, 02ah, 02ah, 000h, 000h, 02ah

67 000129C1 002A2A2A00002A <1>

68 000129C8 002A2A15002A2A2A15- <1> db 000h, 02ah, 02ah, 015h, 000h, 02ah, 02ah, 02ah, 015h, 015h, 015h, 015h, 015h, 03fh, 015h, 03fh

68 000129D1 151515153F153F <1>

69 000129D8 15153F3F3F15153F15- <1> db 015h, 015h, 03fh, 03fh, 03fh, 015h, 015h, 03fh, 015h, 03fh, 03fh, 03fh, 015h, 03fh, 03fh, 03fh

69 000129E1 3F3F3F153F3F3F <1>

70 000129E8 000000050505080808- <1> db 000h, 000h, 000h, 005h, 005h, 005h, 008h, 008h, 008h, 00bh, 00bh, 00bh, 00eh, 00eh, 00eh, 011h

70 000129F1 0B0B0B0E0E0E11 <1>

71 000129F8 11111414141818181C- <1> db 011h, 011h, 014h, 014h, 014h, 018h, 018h, 018h, 01ch, 01ch, 01ch, 020h, 020h, 020h, 024h, 024h

71 00012A01 1C1C2020202424 <1>

72 00012A08 242828282D2D2D3232- <1> db 024h, 028h, 028h, 028h, 02dh, 02dh, 02dh, 032h, 032h, 032h, 038h, 038h, 038h, 03fh, 03fh, 03fh

72 00012A11 323838383F3F3F <1>

73 00012A18 00003F10003F1F003F- <1> db 000h, 000h, 03fh, 010h, 000h, 03fh, 01fh, 000h, 03fh, 02fh, 000h, 03fh, 03fh, 000h, 03fh, 03fh

73 00012A21 2F003F3F003F3F <1>

74 00012A28 002F3F001F3F00103F- <1> db 000h, 02fh, 03fh, 000h, 01fh, 03fh, 000h, 010h, 03fh, 000h, 000h, 03fh, 010h, 000h, 03fh, 01fh

74 00012A31 00003F10003F1F <1>

75 00012A38 003F2F003F3F002F3F- <1> db 000h, 03fh, 02fh, 000h, 03fh, 03fh, 000h, 02fh, 03fh, 000h, 01fh, 03fh, 000h, 010h, 03fh, 000h

75 00012A41 001F3F00103F00 <1>

76 00012A48 003F00003F10003F1F- <1> db 000h, 03fh, 000h, 000h, 03fh, 010h, 000h, 03fh, 01fh, 000h, 03fh, 02fh, 000h, 03fh, 03fh, 000h

76 00012A51 003F2F003F3F00 <1>

77 00012A58 2F3F001F3F00103F1F- <1> db 02fh, 03fh, 000h, 01fh, 03fh, 000h, 010h, 03fh, 01fh, 01fh, 03fh, 027h, 01fh, 03fh, 02fh, 01fh

77 00012A61 1F3F271F3F2F1F <1>

78 00012A68 3F371F3F3F1F3F3F1F- <1> db 03fh, 037h, 01fh, 03fh, 03fh, 01fh, 03fh, 03fh, 01fh, 037h, 03fh, 01fh, 02fh, 03fh, 01fh, 027h

78 00012A71 373F1F2F3F1F27 <1>

79 00012A78 3F1F1F3F271F3F2F1F- <1> db 03fh, 01fh, 01fh, 03fh, 027h, 01fh, 03fh, 02fh, 01fh, 03fh, 037h, 01fh, 03fh, 03fh, 01fh, 037h

79 00012A81 3F371F3F3F1F37 <1>

80 00012A88 3F1F2F3F1F273F1F1F- <1> db 03fh, 01fh, 02fh, 03fh, 01fh, 027h, 03fh, 01fh, 01fh, 03fh, 01fh, 01fh, 03fh, 027h, 01fh, 03fh

80 00012A91 3F1F1F3F271F3F <1>

81 00012A98 2F1F3F371F3F3F1F37- <1> db 02fh, 01fh, 03fh, 037h, 01fh, 03fh, 03fh, 01fh, 037h, 03fh, 01fh, 02fh, 03fh, 01fh, 027h, 03fh

81 00012AA1 3F1F2F3F1F273F <1>

82 00012AA8 2D2D3F312D3F362D3F- <1> db 02dh, 02dh, 03fh, 031h, 02dh, 03fh, 036h, 02dh, 03fh, 03ah, 02dh, 03fh, 03fh, 02dh, 03fh, 03fh

82 00012AB1 3A2D3F3F2D3F3F <1>

83 00012AB8 2D3A3F2D363F2D313F- <1> db 02dh, 03ah, 03fh, 02dh, 036h, 03fh, 02dh, 031h, 03fh, 02dh, 02dh, 03fh, 031h, 02dh, 03fh, 036h

83 00012AC1 2D2D3F312D3F36 <1>

84 00012AC8 2D3F3A2D3F3F2D3A3F- <1> db 02dh, 03fh, 03ah, 02dh, 03fh, 03fh, 02dh, 03ah, 03fh, 02dh, 036h, 03fh, 02dh, 031h, 03fh, 02dh

84 00012AD1 2D363F2D313F2D <1>

85 00012AD8 2D3F2D2D3F312D3F36- <1> db 02dh, 03fh, 02dh, 02dh, 03fh, 031h, 02dh, 03fh, 036h, 02dh, 03fh, 03ah, 02dh, 03fh, 03fh, 02dh

85 00012AE1 2D3F3A2D3F3F2D <1>

86 00012AE8 3A3F2D363F2D313F00- <1> db 03ah, 03fh, 02dh, 036h, 03fh, 02dh, 031h, 03fh, 000h, 000h, 01ch, 007h, 000h, 01ch, 00eh, 000h

86 00012AF1 001C07001C0E00 <1>

87 00012AF8 1C15001C1C001C1C00- <1> db 01ch, 015h, 000h, 01ch, 01ch, 000h, 01ch, 01ch, 000h, 015h, 01ch, 000h, 00eh, 01ch, 000h, 007h

87 00012B01 151C000E1C0007 <1>

88 00012B08 1C00001C07001C0E00- <1> db 01ch, 000h, 000h, 01ch, 007h, 000h, 01ch, 00eh, 000h, 01ch, 015h, 000h, 01ch, 01ch, 000h, 015h

88 00012B11 1C15001C1C0015 <1>

89 00012B18 1C000E1C00071C0000- <1> db 01ch, 000h, 00eh, 01ch, 000h, 007h, 01ch, 000h, 000h, 01ch, 000h, 000h, 01ch, 007h, 000h, 01ch

89 00012B21 1C00001C07001C <1>

90 00012B28 0E001C15001C1C0015- <1> db 00eh, 000h, 01ch, 015h, 000h, 01ch, 01ch, 000h, 015h, 01ch, 000h, 00eh, 01ch, 000h, 007h, 01ch

90 00012B31 1C000E1C00071C <1>

91 00012B38 0E0E1C110E1C150E1C- <1> db 00eh, 00eh, 01ch, 011h, 00eh, 01ch, 015h, 00eh, 01ch, 018h, 00eh, 01ch, 01ch, 00eh, 01ch, 01ch

91 00012B41 180E1C1C0E1C1C <1>

92 00012B48 0E181C0E151C0E111C- <1> db 00eh, 018h, 01ch, 00eh, 015h, 01ch, 00eh, 011h, 01ch, 00eh, 00eh, 01ch, 011h, 00eh, 01ch, 015h

92 00012B51 0E0E1C110E1C15 <1>

93 00012B58 0E1C180E1C1C0E181C- <1> db 00eh, 01ch, 018h, 00eh, 01ch, 01ch, 00eh, 018h, 01ch, 00eh, 015h, 01ch, 00eh, 011h, 01ch, 00eh

93 00012B61 0E151C0E111C0E <1>

94 00012B68 0E1C0E0E1C110E1C15- <1> db 00eh, 01ch, 00eh, 00eh, 01ch, 011h, 00eh, 01ch, 015h, 00eh, 01ch, 018h, 00eh, 01ch, 01ch, 00eh

94 00012B71 0E1C180E1C1C0E <1>

95 00012B78 181C0E151C0E111C14- <1> db 018h, 01ch, 00eh, 015h, 01ch, 00eh, 011h, 01ch, 014h, 014h, 01ch, 016h, 014h, 01ch, 018h, 014h

95 00012B81 141C16141C1814 <1>

96 00012B88 1C1A141C1C141C1C14- <1> db 01ch, 01ah, 014h, 01ch, 01ch, 014h, 01ch, 01ch, 014h, 01ah, 01ch, 014h, 018h, 01ch, 014h, 016h

96 00012B91 1A1C14181C1416 <1>

97 00012B98 1C14141C16141C1814- <1> db 01ch, 014h, 014h, 01ch, 016h, 014h, 01ch, 018h, 014h, 01ch, 01ah, 014h, 01ch, 01ch, 014h, 01ah

97 00012BA1 1C1A141C1C141A <1>

98 00012BA8 1C14181C14161C1414- <1> db 01ch, 014h, 018h, 01ch, 014h, 016h, 01ch, 014h, 014h, 01ch, 014h, 014h, 01ch, 016h, 014h, 01ch

98 00012BB1 1C14141C16141C <1>

99 00012BB8 18141C1A141C1C141A- <1> db 018h, 014h, 01ch, 01ah, 014h, 01ch, 01ch, 014h, 01ah, 01ch, 014h, 018h, 01ch, 014h, 016h, 01ch

99 00012BC1 1C14181C14161C <1>

100 00012BC8 000010040010080010- <1> db 000h, 000h, 010h, 004h, 000h, 010h, 008h, 000h, 010h, 00ch, 000h, 010h, 010h, 000h, 010h, 010h

100 00012BD1 0C001010001010 <1>

101 00012BD8 000C10000810000410- <1> db 000h, 00ch, 010h, 000h, 008h, 010h, 000h, 004h, 010h, 000h, 000h, 010h, 004h, 000h, 010h, 008h

101 00012BE1 00001004001008 <1>

102 00012BE8 00100C001010000C10- <1> db 000h, 010h, 00ch, 000h, 010h, 010h, 000h, 00ch, 010h, 000h, 008h, 010h, 000h, 004h, 010h, 000h

102 00012BF1 00081000041000 <1>

103 00012BF8 001000001004001008- <1> db 000h, 010h, 000h, 000h, 010h, 004h, 000h, 010h, 008h, 000h, 010h, 00ch, 000h, 010h, 010h, 000h

103 00012C01 00100C00101000 <1>

104 00012C08 0C1000081000041008- <1> db 00ch, 010h, 000h, 008h, 010h, 000h, 004h, 010h, 008h, 008h, 010h, 00ah, 008h, 010h, 00ch, 008h

104 00012C11 08100A08100C08 <1>

105 00012C18 100E08101008101008- <1> db 010h, 00eh, 008h, 010h, 010h, 008h, 010h, 010h, 008h, 00eh, 010h, 008h, 00ch, 010h, 008h, 00ah

105 00012C21 0E10080C10080A <1>

106 00012C28 100808100A08100C08- <1> db 010h, 008h, 008h, 010h, 00ah, 008h, 010h, 00ch, 008h, 010h, 00eh, 008h, 010h, 010h, 008h, 00eh

106 00012C31 100E081010080E <1>

107 00012C38 10080C10080A100808- <1> db 010h, 008h, 00ch, 010h, 008h, 00ah, 010h, 008h, 008h, 010h, 008h, 008h, 010h, 00ah, 008h, 010h

107 00012C41 100808100A0810 <1>

108 00012C48 0C08100E081010080E- <1> db 00ch, 008h, 010h, 00eh, 008h, 010h, 010h, 008h, 00eh, 010h, 008h, 00ch, 010h, 008h, 00ah, 010h

108 00012C51 10080C10080A10 <1>

109 00012C58 0B0B100C0B100D0B10- <1> db 00bh, 00bh, 010h, 00ch, 00bh, 010h, 00dh, 00bh, 010h, 00fh, 00bh, 010h, 010h, 00bh, 010h, 010h

109 00012C61 0F0B10100B1010 <1>

110 00012C68 0B0F100B0D100B0C10- <1> db 00bh, 00fh, 010h, 00bh, 00dh, 010h, 00bh, 00ch, 010h, 00bh, 00bh, 010h, 00ch, 00bh, 010h, 00dh

110 00012C71 0B0B100C0B100D <1>

111 00012C78 0B100F0B10100B0F10- <1> db 00bh, 010h, 00fh, 00bh, 010h, 010h, 00bh, 00fh, 010h, 00bh, 00dh, 010h, 00bh, 00ch, 010h, 00bh

111 00012C81 0B0D100B0C100B <1>

112 00012C88 0B100B0B100C0B100D- <1> db 00bh, 010h, 00bh, 00bh, 010h, 00ch, 00bh, 010h, 00dh, 00bh, 010h, 00fh, 00bh, 010h, 010h, 00bh

112 00012C91 0B100F0B10100B <1>

113 00012C98 0F100B0D100B0C1000- <1> db 00fh, 010h, 00bh, 00dh, 010h, 00bh, 00ch, 010h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

113 00012CA1 00000000000000 <1>

114 00012CA8 000000000000000000- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

114 00012CB1 00000000000000 <1>

115 <1>

116 <1>

117 <1> ; 04/07/2016

118 <1> ; FONT DATA

119 <1>

120 <1> CRT\_CHAR\_GEN:

121 <1> vgafont8:

122 00012CB8 00000000000000007E- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07eh, 081h, 0a5h, 081h, 0bdh, 099h, 081h, 07eh

122 00012CC1 81A581BD99817E <1>

123 00012CC8 7EFFDBFFC3E7FF7E6C- <1> db 07eh, 0ffh, 0dbh, 0ffh, 0c3h, 0e7h, 0ffh, 07eh, 06ch, 0feh, 0feh, 0feh, 07ch, 038h, 010h, 000h

123 00012CD1 FEFEFE7C381000 <1>

124 00012CD8 10387CFE7C38100038- <1> db 010h, 038h, 07ch, 0feh, 07ch, 038h, 010h, 000h, 038h, 07ch, 038h, 0feh, 0feh, 07ch, 038h, 07ch

124 00012CE1 7C38FEFE7C387C <1>

125 00012CE8 1010387CFE7C387C00- <1> db 010h, 010h, 038h, 07ch, 0feh, 07ch, 038h, 07ch, 000h, 000h, 018h, 03ch, 03ch, 018h, 000h, 000h

125 00012CF1 00183C3C180000 <1>

126 00012CF8 FFFFE7C3C3E7FFFF00- <1> db 0ffh, 0ffh, 0e7h, 0c3h, 0c3h, 0e7h, 0ffh, 0ffh, 000h, 03ch, 066h, 042h, 042h, 066h, 03ch, 000h

126 00012D01 3C664242663C00 <1>

127 00012D08 FFC399BDBD99C3FF0F- <1> db 0ffh, 0c3h, 099h, 0bdh, 0bdh, 099h, 0c3h, 0ffh, 00fh, 007h, 00fh, 07dh, 0cch, 0cch, 0cch, 078h

127 00012D11 070F7DCCCCCC78 <1>

128 00012D18 3C6666663C187E183F- <1> db 03ch, 066h, 066h, 066h, 03ch, 018h, 07eh, 018h, 03fh, 033h, 03fh, 030h, 030h, 070h, 0f0h, 0e0h

128 00012D21 333F303070F0E0 <1>

129 00012D28 7F637F636367E6C099- <1> db 07fh, 063h, 07fh, 063h, 063h, 067h, 0e6h, 0c0h, 099h, 05ah, 03ch, 0e7h, 0e7h, 03ch, 05ah, 099h

129 00012D31 5A3CE7E73C5A99 <1>

130 00012D38 80E0F8FEF8E0800002- <1> db 080h, 0e0h, 0f8h, 0feh, 0f8h, 0e0h, 080h, 000h, 002h, 00eh, 03eh, 0feh, 03eh, 00eh, 002h, 000h

130 00012D41 0E3EFE3E0E0200 <1>

131 00012D48 183C7E18187E3C1866- <1> db 018h, 03ch, 07eh, 018h, 018h, 07eh, 03ch, 018h, 066h, 066h, 066h, 066h, 066h, 000h, 066h, 000h

131 00012D51 66666666006600 <1>

132 00012D58 7FDBDB7B1B1B1B003E- <1> db 07fh, 0dbh, 0dbh, 07bh, 01bh, 01bh, 01bh, 000h, 03eh, 063h, 038h, 06ch, 06ch, 038h, 0cch, 078h

132 00012D61 63386C6C38CC78 <1>

133 00012D68 000000007E7E7E0018- <1> db 000h, 000h, 000h, 000h, 07eh, 07eh, 07eh, 000h, 018h, 03ch, 07eh, 018h, 07eh, 03ch, 018h, 0ffh

133 00012D71 3C7E187E3C18FF <1>

134 00012D78 183C7E181818180018- <1> db 018h, 03ch, 07eh, 018h, 018h, 018h, 018h, 000h, 018h, 018h, 018h, 018h, 07eh, 03ch, 018h, 000h

134 00012D81 1818187E3C1800 <1>

135 00012D88 00180CFE0C18000000- <1> db 000h, 018h, 00ch, 0feh, 00ch, 018h, 000h, 000h, 000h, 030h, 060h, 0feh, 060h, 030h, 000h, 000h

135 00012D91 3060FE60300000 <1>

136 00012D98 0000C0C0C0FE000000- <1> db 000h, 000h, 0c0h, 0c0h, 0c0h, 0feh, 000h, 000h, 000h, 024h, 066h, 0ffh, 066h, 024h, 000h, 000h

136 00012DA1 2466FF66240000 <1>

137 00012DA8 00183C7EFFFF000000- <1> db 000h, 018h, 03ch, 07eh, 0ffh, 0ffh, 000h, 000h, 000h, 0ffh, 0ffh, 07eh, 03ch, 018h, 000h, 000h

137 00012DB1 FFFF7E3C180000 <1>

138 00012DB8 000000000000000030- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 030h, 078h, 078h, 030h, 030h, 000h, 030h, 000h

138 00012DC1 78783030003000 <1>

139 00012DC8 6C6C6C00000000006C- <1> db 06ch, 06ch, 06ch, 000h, 000h, 000h, 000h, 000h, 06ch, 06ch, 0feh, 06ch, 0feh, 06ch, 06ch, 000h

139 00012DD1 6CFE6CFE6C6C00 <1>

140 00012DD8 307CC0780CF8300000- <1> db 030h, 07ch, 0c0h, 078h, 00ch, 0f8h, 030h, 000h, 000h, 0c6h, 0cch, 018h, 030h, 066h, 0c6h, 000h

140 00012DE1 C6CC183066C600 <1>

141 00012DE8 386C3876DCCC760060- <1> db 038h, 06ch, 038h, 076h, 0dch, 0cch, 076h, 000h, 060h, 060h, 0c0h, 000h, 000h, 000h, 000h, 000h

141 00012DF1 60C00000000000 <1>

142 00012DF8 183060606030180060- <1> db 018h, 030h, 060h, 060h, 060h, 030h, 018h, 000h, 060h, 030h, 018h, 018h, 018h, 030h, 060h, 000h

142 00012E01 30181818306000 <1>

143 00012E08 00663CFF3C66000000- <1> db 000h, 066h, 03ch, 0ffh, 03ch, 066h, 000h, 000h, 000h, 030h, 030h, 0fch, 030h, 030h, 000h, 000h

143 00012E11 3030FC30300000 <1>

144 00012E18 000000000030306000- <1> db 000h, 000h, 000h, 000h, 000h, 030h, 030h, 060h, 000h, 000h, 000h, 0fch, 000h, 000h, 000h, 000h

144 00012E21 0000FC00000000 <1>

145 00012E28 000000000030300006- <1> db 000h, 000h, 000h, 000h, 000h, 030h, 030h, 000h, 006h, 00ch, 018h, 030h, 060h, 0c0h, 080h, 000h

145 00012E31 0C183060C08000 <1>

146 00012E38 7CC6CEDEF6E67C0030- <1> db 07ch, 0c6h, 0ceh, 0deh, 0f6h, 0e6h, 07ch, 000h, 030h, 070h, 030h, 030h, 030h, 030h, 0fch, 000h

146 00012E41 7030303030FC00 <1>

147 00012E48 78CC0C3860CCFC0078- <1> db 078h, 0cch, 00ch, 038h, 060h, 0cch, 0fch, 000h, 078h, 0cch, 00ch, 038h, 00ch, 0cch, 078h, 000h

147 00012E51 CC0C380CCC7800 <1>

148 00012E58 1C3C6CCCFE0C1E00FC- <1> db 01ch, 03ch, 06ch, 0cch, 0feh, 00ch, 01eh, 000h, 0fch, 0c0h, 0f8h, 00ch, 00ch, 0cch, 078h, 000h

148 00012E61 C0F80C0CCC7800 <1>

149 00012E68 3860C0F8CCCC7800FC- <1> db 038h, 060h, 0c0h, 0f8h, 0cch, 0cch, 078h, 000h, 0fch, 0cch, 00ch, 018h, 030h, 030h, 030h, 000h

149 00012E71 CC0C1830303000 <1>

150 00012E78 78CCCC78CCCC780078- <1> db 078h, 0cch, 0cch, 078h, 0cch, 0cch, 078h, 000h, 078h, 0cch, 0cch, 07ch, 00ch, 018h, 070h, 000h

150 00012E81 CCCC7C0C187000 <1>

151 00012E88 003030000030300000- <1> db 000h, 030h, 030h, 000h, 000h, 030h, 030h, 000h, 000h, 030h, 030h, 000h, 000h, 030h, 030h, 060h

151 00012E91 30300000303060 <1>

152 00012E98 183060C06030180000- <1> db 018h, 030h, 060h, 0c0h, 060h, 030h, 018h, 000h, 000h, 000h, 0fch, 000h, 000h, 0fch, 000h, 000h

152 00012EA1 00FC0000FC0000 <1>

153 00012EA8 6030180C1830600078- <1> db 060h, 030h, 018h, 00ch, 018h, 030h, 060h, 000h, 078h, 0cch, 00ch, 018h, 030h, 000h, 030h, 000h

153 00012EB1 CC0C1830003000 <1>

154 00012EB8 7CC6DEDEDEC0780030- <1> db 07ch, 0c6h, 0deh, 0deh, 0deh, 0c0h, 078h, 000h, 030h, 078h, 0cch, 0cch, 0fch, 0cch, 0cch, 000h

154 00012EC1 78CCCCFCCCCC00 <1>

155 00012EC8 FC66667C6666FC003C- <1> db 0fch, 066h, 066h, 07ch, 066h, 066h, 0fch, 000h, 03ch, 066h, 0c0h, 0c0h, 0c0h, 066h, 03ch, 000h

155 00012ED1 66C0C0C0663C00 <1>

156 00012ED8 F86C6666666CF800FE- <1> db 0f8h, 06ch, 066h, 066h, 066h, 06ch, 0f8h, 000h, 0feh, 062h, 068h, 078h, 068h, 062h, 0feh, 000h

156 00012EE1 6268786862FE00 <1>

157 00012EE8 FE6268786860F0003C- <1> db 0feh, 062h, 068h, 078h, 068h, 060h, 0f0h, 000h, 03ch, 066h, 0c0h, 0c0h, 0ceh, 066h, 03eh, 000h

157 00012EF1 66C0C0CE663E00 <1>

158 00012EF8 CCCCCCFCCCCCCC0078- <1> db 0cch, 0cch, 0cch, 0fch, 0cch, 0cch, 0cch, 000h, 078h, 030h, 030h, 030h, 030h, 030h, 078h, 000h

158 00012F01 30303030307800 <1>

159 00012F08 1E0C0C0CCCCC7800E6- <1> db 01eh, 00ch, 00ch, 00ch, 0cch, 0cch, 078h, 000h, 0e6h, 066h, 06ch, 078h, 06ch, 066h, 0e6h, 000h

159 00012F11 666C786C66E600 <1>

160 00012F18 F06060606266FE00C6- <1> db 0f0h, 060h, 060h, 060h, 062h, 066h, 0feh, 000h, 0c6h, 0eeh, 0feh, 0feh, 0d6h, 0c6h, 0c6h, 000h

160 00012F21 EEFEFED6C6C600 <1>

161 00012F28 C6E6F6DECEC6C60038- <1> db 0c6h, 0e6h, 0f6h, 0deh, 0ceh, 0c6h, 0c6h, 000h, 038h, 06ch, 0c6h, 0c6h, 0c6h, 06ch, 038h, 000h

161 00012F31 6CC6C6C66C3800 <1>

162 00012F38 FC66667C6060F00078- <1> db 0fch, 066h, 066h, 07ch, 060h, 060h, 0f0h, 000h, 078h, 0cch, 0cch, 0cch, 0dch, 078h, 01ch, 000h

162 00012F41 CCCCCCDC781C00 <1>

163 00012F48 FC66667C6C66E60078- <1> db 0fch, 066h, 066h, 07ch, 06ch, 066h, 0e6h, 000h, 078h, 0cch, 0e0h, 070h, 01ch, 0cch, 078h, 000h

163 00012F51 CCE0701CCC7800 <1>

164 00012F58 FCB4303030307800CC- <1> db 0fch, 0b4h, 030h, 030h, 030h, 030h, 078h, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 0fch, 000h

164 00012F61 CCCCCCCCCCFC00 <1>

165 00012F68 CCCCCCCCCC783000C6- <1> db 0cch, 0cch, 0cch, 0cch, 0cch, 078h, 030h, 000h, 0c6h, 0c6h, 0c6h, 0d6h, 0feh, 0eeh, 0c6h, 000h

165 00012F71 C6C6D6FEEEC600 <1>

166 00012F78 C6C66C38386CC600CC- <1> db 0c6h, 0c6h, 06ch, 038h, 038h, 06ch, 0c6h, 000h, 0cch, 0cch, 0cch, 078h, 030h, 030h, 078h, 000h

166 00012F81 CCCC7830307800 <1>

167 00012F88 FEC68C183266FE0078- <1> db 0feh, 0c6h, 08ch, 018h, 032h, 066h, 0feh, 000h, 078h, 060h, 060h, 060h, 060h, 060h, 078h, 000h

167 00012F91 60606060607800 <1>

168 00012F98 C06030180C06020078- <1> db 0c0h, 060h, 030h, 018h, 00ch, 006h, 002h, 000h, 078h, 018h, 018h, 018h, 018h, 018h, 078h, 000h

168 00012FA1 18181818187800 <1>

169 00012FA8 10386CC60000000000- <1> db 010h, 038h, 06ch, 0c6h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh

169 00012FB1 000000000000FF <1>

170 00012FB8 303018000000000000- <1> db 030h, 030h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 078h, 00ch, 07ch, 0cch, 076h, 000h

170 00012FC1 00780C7CCC7600 <1>

171 00012FC8 E060607C6666DC0000- <1> db 0e0h, 060h, 060h, 07ch, 066h, 066h, 0dch, 000h, 000h, 000h, 078h, 0cch, 0c0h, 0cch, 078h, 000h

171 00012FD1 0078CCC0CC7800 <1>

172 00012FD8 1C0C0C7CCCCC760000- <1> db 01ch, 00ch, 00ch, 07ch, 0cch, 0cch, 076h, 000h, 000h, 000h, 078h, 0cch, 0fch, 0c0h, 078h, 000h

172 00012FE1 0078CCFCC07800 <1>

173 00012FE8 386C60F06060F00000- <1> db 038h, 06ch, 060h, 0f0h, 060h, 060h, 0f0h, 000h, 000h, 000h, 076h, 0cch, 0cch, 07ch, 00ch, 0f8h

173 00012FF1 0076CCCC7C0CF8 <1>

174 00012FF8 E0606C766666E60030- <1> db 0e0h, 060h, 06ch, 076h, 066h, 066h, 0e6h, 000h, 030h, 000h, 070h, 030h, 030h, 030h, 078h, 000h

174 00013001 00703030307800 <1>

175 00013008 0C000C0C0CCCCC78E0- <1> db 00ch, 000h, 00ch, 00ch, 00ch, 0cch, 0cch, 078h, 0e0h, 060h, 066h, 06ch, 078h, 06ch, 0e6h, 000h

175 00013011 60666C786CE600 <1>

176 00013018 703030303030780000- <1> db 070h, 030h, 030h, 030h, 030h, 030h, 078h, 000h, 000h, 000h, 0cch, 0feh, 0feh, 0d6h, 0c6h, 000h

176 00013021 00CCFEFED6C600 <1>

177 00013028 0000F8CCCCCCCC0000- <1> db 000h, 000h, 0f8h, 0cch, 0cch, 0cch, 0cch, 000h, 000h, 000h, 078h, 0cch, 0cch, 0cch, 078h, 000h

177 00013031 0078CCCCCC7800 <1>

178 00013038 0000DC66667C60F000- <1> db 000h, 000h, 0dch, 066h, 066h, 07ch, 060h, 0f0h, 000h, 000h, 076h, 0cch, 0cch, 07ch, 00ch, 01eh

178 00013041 0076CCCC7C0C1E <1>

179 00013048 0000DC766660F00000- <1> db 000h, 000h, 0dch, 076h, 066h, 060h, 0f0h, 000h, 000h, 000h, 07ch, 0c0h, 078h, 00ch, 0f8h, 000h

179 00013051 007CC0780CF800 <1>

180 00013058 10307C303034180000- <1> db 010h, 030h, 07ch, 030h, 030h, 034h, 018h, 000h, 000h, 000h, 0cch, 0cch, 0cch, 0cch, 076h, 000h

180 00013061 00CCCCCCCC7600 <1>

181 00013068 0000CCCCCC78300000- <1> db 000h, 000h, 0cch, 0cch, 0cch, 078h, 030h, 000h, 000h, 000h, 0c6h, 0d6h, 0feh, 0feh, 06ch, 000h

181 00013071 00C6D6FEFE6C00 <1>

182 00013078 0000C66C386CC60000- <1> db 000h, 000h, 0c6h, 06ch, 038h, 06ch, 0c6h, 000h, 000h, 000h, 0cch, 0cch, 0cch, 07ch, 00ch, 0f8h

182 00013081 00CCCCCC7C0CF8 <1>

183 00013088 0000FC983064FC001C- <1> db 000h, 000h, 0fch, 098h, 030h, 064h, 0fch, 000h, 01ch, 030h, 030h, 0e0h, 030h, 030h, 01ch, 000h

183 00013091 3030E030301C00 <1>

184 00013098 1818180018181800E0- <1> db 018h, 018h, 018h, 000h, 018h, 018h, 018h, 000h, 0e0h, 030h, 030h, 01ch, 030h, 030h, 0e0h, 000h

184 000130A1 30301C3030E000 <1>

185 000130A8 76DC00000000000000- <1> db 076h, 0dch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 010h, 038h, 06ch, 0c6h, 0c6h, 0feh, 000h

185 000130B1 10386CC6C6FE00 <1>

186 000130B8 78CCC0CC78180C7800- <1> db 078h, 0cch, 0c0h, 0cch, 078h, 018h, 00ch, 078h, 000h, 0cch, 000h, 0cch, 0cch, 0cch, 07eh, 000h

186 000130C1 CC00CCCCCC7E00 <1>

187 000130C8 1C0078CCFCC078007E- <1> db 01ch, 000h, 078h, 0cch, 0fch, 0c0h, 078h, 000h, 07eh, 0c3h, 03ch, 006h, 03eh, 066h, 03fh, 000h

187 000130D1 C33C063E663F00 <1>

188 000130D8 CC00780C7CCC7E00E0- <1> db 0cch, 000h, 078h, 00ch, 07ch, 0cch, 07eh, 000h, 0e0h, 000h, 078h, 00ch, 07ch, 0cch, 07eh, 000h

188 000130E1 00780C7CCC7E00 <1>

189 000130E8 3030780C7CCC7E0000- <1> db 030h, 030h, 078h, 00ch, 07ch, 0cch, 07eh, 000h, 000h, 000h, 078h, 0c0h, 0c0h, 078h, 00ch, 038h

189 000130F1 0078C0C0780C38 <1>

190 000130F8 7EC33C667E603C00CC- <1> db 07eh, 0c3h, 03ch, 066h, 07eh, 060h, 03ch, 000h, 0cch, 000h, 078h, 0cch, 0fch, 0c0h, 078h, 000h

190 00013101 0078CCFCC07800 <1>

191 00013108 E00078CCFCC07800CC- <1> db 0e0h, 000h, 078h, 0cch, 0fch, 0c0h, 078h, 000h, 0cch, 000h, 070h, 030h, 030h, 030h, 078h, 000h

191 00013111 00703030307800 <1>

192 00013118 7CC6381818183C00E0- <1> db 07ch, 0c6h, 038h, 018h, 018h, 018h, 03ch, 000h, 0e0h, 000h, 070h, 030h, 030h, 030h, 078h, 000h

192 00013121 00703030307800 <1>

193 00013128 C6386CC6FEC6C60030- <1> db 0c6h, 038h, 06ch, 0c6h, 0feh, 0c6h, 0c6h, 000h, 030h, 030h, 000h, 078h, 0cch, 0fch, 0cch, 000h

193 00013131 300078CCFCCC00 <1>

194 00013138 1C00FC607860FC0000- <1> db 01ch, 000h, 0fch, 060h, 078h, 060h, 0fch, 000h, 000h, 000h, 07fh, 00ch, 07fh, 0cch, 07fh, 000h

194 00013141 007F0C7FCC7F00 <1>

195 00013148 3E6CCCFECCCCCE0078- <1> db 03eh, 06ch, 0cch, 0feh, 0cch, 0cch, 0ceh, 000h, 078h, 0cch, 000h, 078h, 0cch, 0cch, 078h, 000h

195 00013151 CC0078CCCC7800 <1>

196 00013158 00CC0078CCCC780000- <1> db 000h, 0cch, 000h, 078h, 0cch, 0cch, 078h, 000h, 000h, 0e0h, 000h, 078h, 0cch, 0cch, 078h, 000h

196 00013161 E00078CCCC7800 <1>

197 00013168 78CC00CCCCCC7E0000- <1> db 078h, 0cch, 000h, 0cch, 0cch, 0cch, 07eh, 000h, 000h, 0e0h, 000h, 0cch, 0cch, 0cch, 07eh, 000h

197 00013171 E000CCCCCC7E00 <1>

198 00013178 00CC00CCCC7C0CF8C3- <1> db 000h, 0cch, 000h, 0cch, 0cch, 07ch, 00ch, 0f8h, 0c3h, 018h, 03ch, 066h, 066h, 03ch, 018h, 000h

198 00013181 183C66663C1800 <1>

199 00013188 CC00CCCCCCCC780018- <1> db 0cch, 000h, 0cch, 0cch, 0cch, 0cch, 078h, 000h, 018h, 018h, 07eh, 0c0h, 0c0h, 07eh, 018h, 018h

199 00013191 187EC0C07E1818 <1>

200 00013198 386C64F060E6FC00CC- <1> db 038h, 06ch, 064h, 0f0h, 060h, 0e6h, 0fch, 000h, 0cch, 0cch, 078h, 0fch, 030h, 0fch, 030h, 030h

200 000131A1 CC78FC30FC3030 <1>

201 000131A8 F8CCCCFAC6CFC6C70E- <1> db 0f8h, 0cch, 0cch, 0fah, 0c6h, 0cfh, 0c6h, 0c7h, 00eh, 01bh, 018h, 03ch, 018h, 018h, 0d8h, 070h

201 000131B1 1B183C1818D870 <1>

202 000131B8 1C00780C7CCC7E0038- <1> db 01ch, 000h, 078h, 00ch, 07ch, 0cch, 07eh, 000h, 038h, 000h, 070h, 030h, 030h, 030h, 078h, 000h

202 000131C1 00703030307800 <1>

203 000131C8 001C0078CCCC780000- <1> db 000h, 01ch, 000h, 078h, 0cch, 0cch, 078h, 000h, 000h, 01ch, 000h, 0cch, 0cch, 0cch, 07eh, 000h

203 000131D1 1C00CCCCCC7E00 <1>

204 000131D8 00F800F8CCCCCC00FC- <1> db 000h, 0f8h, 000h, 0f8h, 0cch, 0cch, 0cch, 000h, 0fch, 000h, 0cch, 0ech, 0fch, 0dch, 0cch, 000h

204 000131E1 00CCECFCDCCC00 <1>

205 000131E8 3C6C6C3E007E000038- <1> db 03ch, 06ch, 06ch, 03eh, 000h, 07eh, 000h, 000h, 038h, 06ch, 06ch, 038h, 000h, 07ch, 000h, 000h

205 000131F1 6C6C38007C0000 <1>

206 000131F8 30003060C0CC780000- <1> db 030h, 000h, 030h, 060h, 0c0h, 0cch, 078h, 000h, 000h, 000h, 000h, 0fch, 0c0h, 0c0h, 000h, 000h

206 00013201 0000FCC0C00000 <1>

207 00013208 000000FC0C0C0000C3- <1> db 000h, 000h, 000h, 0fch, 00ch, 00ch, 000h, 000h, 0c3h, 0c6h, 0cch, 0deh, 033h, 066h, 0cch, 00fh

207 00013211 C6CCDE3366CC0F <1>

208 00013218 C3C6CCDB376FCF0318- <1> db 0c3h, 0c6h, 0cch, 0dbh, 037h, 06fh, 0cfh, 003h, 018h, 018h, 000h, 018h, 018h, 018h, 018h, 000h

208 00013221 18001818181800 <1>

209 00013228 003366CC6633000000- <1> db 000h, 033h, 066h, 0cch, 066h, 033h, 000h, 000h, 000h, 0cch, 066h, 033h, 066h, 0cch, 000h, 000h

209 00013231 CC663366CC0000 <1>

210 00013238 228822882288228855- <1> db 022h, 088h, 022h, 088h, 022h, 088h, 022h, 088h, 055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah

210 00013241 AA55AA55AA55AA <1>

211 00013248 DB77DBEEDB77DBEE18- <1> db 0dbh, 077h, 0dbh, 0eeh, 0dbh, 077h, 0dbh, 0eeh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h

211 00013251 18181818181818 <1>

212 00013258 18181818F818181818- <1> db 018h, 018h, 018h, 018h, 0f8h, 018h, 018h, 018h, 018h, 018h, 0f8h, 018h, 0f8h, 018h, 018h, 018h

212 00013261 18F818F8181818 <1>

213 00013268 36363636F636363600- <1> db 036h, 036h, 036h, 036h, 0f6h, 036h, 036h, 036h, 000h, 000h, 000h, 000h, 0feh, 036h, 036h, 036h

213 00013271 000000FE363636 <1>

214 00013278 0000F818F818181836- <1> db 000h, 000h, 0f8h, 018h, 0f8h, 018h, 018h, 018h, 036h, 036h, 0f6h, 006h, 0f6h, 036h, 036h, 036h

214 00013281 36F606F6363636 <1>

215 00013288 363636363636363600- <1> db 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 000h, 000h, 0feh, 006h, 0f6h, 036h, 036h, 036h

215 00013291 00FE06F6363636 <1>

216 00013298 3636F606FE00000036- <1> db 036h, 036h, 0f6h, 006h, 0feh, 000h, 000h, 000h, 036h, 036h, 036h, 036h, 0feh, 000h, 000h, 000h

216 000132A1 363636FE000000 <1>

217 000132A8 1818F818F800000000- <1> db 018h, 018h, 0f8h, 018h, 0f8h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0f8h, 018h, 018h, 018h

217 000132B1 000000F8181818 <1>

218 000132B8 181818181F00000018- <1> db 018h, 018h, 018h, 018h, 01fh, 000h, 000h, 000h, 018h, 018h, 018h, 018h, 0ffh, 000h, 000h, 000h

218 000132C1 181818FF000000 <1>

219 000132C8 00000000FF18181818- <1> db 000h, 000h, 000h, 000h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 01fh, 018h, 018h, 018h

219 000132D1 1818181F181818 <1>

220 000132D8 00000000FF00000018- <1> db 000h, 000h, 000h, 000h, 0ffh, 000h, 000h, 000h, 018h, 018h, 018h, 018h, 0ffh, 018h, 018h, 018h

220 000132E1 181818FF181818 <1>

221 000132E8 18181F181F18181836- <1> db 018h, 018h, 01fh, 018h, 01fh, 018h, 018h, 018h, 036h, 036h, 036h, 036h, 037h, 036h, 036h, 036h

221 000132F1 36363637363636 <1>

222 000132F8 363637303F00000000- <1> db 036h, 036h, 037h, 030h, 03fh, 000h, 000h, 000h, 000h, 000h, 03fh, 030h, 037h, 036h, 036h, 036h

222 00013301 003F3037363636 <1>

223 00013308 3636F700FF00000000- <1> db 036h, 036h, 0f7h, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 0ffh, 000h, 0f7h, 036h, 036h, 036h

223 00013311 00FF00F7363636 <1>

224 00013318 363637303736363600- <1> db 036h, 036h, 037h, 030h, 037h, 036h, 036h, 036h, 000h, 000h, 0ffh, 000h, 0ffh, 000h, 000h, 000h

224 00013321 00FF00FF000000 <1>

225 00013328 3636F700F736363618- <1> db 036h, 036h, 0f7h, 000h, 0f7h, 036h, 036h, 036h, 018h, 018h, 0ffh, 000h, 0ffh, 000h, 000h, 000h

225 00013331 18FF00FF000000 <1>

226 00013338 36363636FF00000000- <1> db 036h, 036h, 036h, 036h, 0ffh, 000h, 000h, 000h, 000h, 000h, 0ffh, 000h, 0ffh, 018h, 018h, 018h

226 00013341 00FF00FF181818 <1>

227 00013348 00000000FF36363636- <1> db 000h, 000h, 000h, 000h, 0ffh, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 03fh, 000h, 000h, 000h

227 00013351 3636363F000000 <1>

228 00013358 18181F181F00000000- <1> db 018h, 018h, 01fh, 018h, 01fh, 000h, 000h, 000h, 000h, 000h, 01fh, 018h, 01fh, 018h, 018h, 018h

228 00013361 001F181F181818 <1>

229 00013368 000000003F36363636- <1> db 000h, 000h, 000h, 000h, 03fh, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 0ffh, 036h, 036h, 036h

229 00013371 363636FF363636 <1>

230 00013378 1818FF18FF18181818- <1> db 018h, 018h, 0ffh, 018h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 0f8h, 000h, 000h, 000h

230 00013381 181818F8000000 <1>

231 00013388 000000001F181818FF- <1> db 000h, 000h, 000h, 000h, 01fh, 018h, 018h, 018h, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh

231 00013391 FFFFFFFFFFFFFF <1>

232 00013398 00000000FFFFFFFFF0- <1> db 000h, 000h, 000h, 000h, 0ffh, 0ffh, 0ffh, 0ffh, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h

232 000133A1 F0F0F0F0F0F0F0 <1>

233 000133A8 0F0F0F0F0F0F0F0FFF- <1> db 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 0ffh, 0ffh, 0ffh, 0ffh, 000h, 000h, 000h, 000h

233 000133B1 FFFFFF00000000 <1>

234 000133B8 000076DCC8DC760000- <1> db 000h, 000h, 076h, 0dch, 0c8h, 0dch, 076h, 000h, 000h, 078h, 0cch, 0f8h, 0cch, 0f8h, 0c0h, 0c0h

234 000133C1 78CCF8CCF8C0C0 <1>

235 000133C8 00FCCCC0C0C0C00000- <1> db 000h, 0fch, 0cch, 0c0h, 0c0h, 0c0h, 0c0h, 000h, 000h, 0feh, 06ch, 06ch, 06ch, 06ch, 06ch, 000h

235 000133D1 FE6C6C6C6C6C00 <1>

236 000133D8 FCCC603060CCFC0000- <1> db 0fch, 0cch, 060h, 030h, 060h, 0cch, 0fch, 000h, 000h, 000h, 07eh, 0d8h, 0d8h, 0d8h, 070h, 000h

236 000133E1 007ED8D8D87000 <1>

237 000133E8 00666666667C60C000- <1> db 000h, 066h, 066h, 066h, 066h, 07ch, 060h, 0c0h, 000h, 076h, 0dch, 018h, 018h, 018h, 018h, 000h

237 000133F1 76DC1818181800 <1>

238 000133F8 FC3078CCCC7830FC38- <1> db 0fch, 030h, 078h, 0cch, 0cch, 078h, 030h, 0fch, 038h, 06ch, 0c6h, 0feh, 0c6h, 06ch, 038h, 000h

238 00013401 6CC6FEC66C3800 <1>

239 00013408 386CC6C66C6CEE001C- <1> db 038h, 06ch, 0c6h, 0c6h, 06ch, 06ch, 0eeh, 000h, 01ch, 030h, 018h, 07ch, 0cch, 0cch, 078h, 000h

239 00013411 30187CCCCC7800 <1>

240 00013418 00007EDBDB7E000006- <1> db 000h, 000h, 07eh, 0dbh, 0dbh, 07eh, 000h, 000h, 006h, 00ch, 07eh, 0dbh, 0dbh, 07eh, 060h, 0c0h

240 00013421 0C7EDBDB7E60C0 <1>

241 00013428 3860C0F8C060380078- <1> db 038h, 060h, 0c0h, 0f8h, 0c0h, 060h, 038h, 000h, 078h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 000h

241 00013431 CCCCCCCCCCCC00 <1>

242 00013438 00FC00FC00FC000030- <1> db 000h, 0fch, 000h, 0fch, 000h, 0fch, 000h, 000h, 030h, 030h, 0fch, 030h, 030h, 000h, 0fch, 000h

242 00013441 30FC303000FC00 <1>

243 00013448 603018306000FC0018- <1> db 060h, 030h, 018h, 030h, 060h, 000h, 0fch, 000h, 018h, 030h, 060h, 030h, 018h, 000h, 0fch, 000h

243 00013451 3060301800FC00 <1>

244 00013458 0E1B1B181818181818- <1> db 00eh, 01bh, 01bh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 0d8h, 0d8h, 070h

244 00013461 18181818D8D870 <1>

245 00013468 303000FC0030300000- <1> db 030h, 030h, 000h, 0fch, 000h, 030h, 030h, 000h, 000h, 076h, 0dch, 000h, 076h, 0dch, 000h, 000h

245 00013471 76DC0076DC0000 <1>

246 00013478 386C6C380000000000- <1> db 038h, 06ch, 06ch, 038h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 000h, 000h, 000h

246 00013481 00001818000000 <1>

247 00013488 00000000180000000F- <1> db 000h, 000h, 000h, 000h, 018h, 000h, 000h, 000h, 00fh, 00ch, 00ch, 00ch, 0ech, 06ch, 03ch, 01ch

247 00013491 0C0C0CEC6C3C1C <1>

248 00013498 786C6C6C6C00000070- <1> db 078h, 06ch, 06ch, 06ch, 06ch, 000h, 000h, 000h, 070h, 018h, 030h, 060h, 078h, 000h, 000h, 000h

248 000134A1 18306078000000 <1>

249 000134A8 00003C3C3C3C000000- <1> db 000h, 000h, 03ch, 03ch, 03ch, 03ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

249 000134B1 00000000000000 <1>

250 <1> vgafont14:

251 000134B8 000000000000000000- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

251 000134C1 00000000000000 <1>

252 000134C8 7E81A58181BD99817E- <1> db 07eh, 081h, 0a5h, 081h, 081h, 0bdh, 099h, 081h, 07eh, 000h, 000h, 000h, 000h, 000h, 07eh, 0ffh

252 000134D1 00000000007EFF <1>

253 000134D8 DBFFFFC3E7FF7E0000- <1> db 0dbh, 0ffh, 0ffh, 0c3h, 0e7h, 0ffh, 07eh, 000h, 000h, 000h, 000h, 000h, 000h, 06ch, 0feh, 0feh

253 000134E1 000000006CFEFE <1>

254 000134E8 FEFE7C381000000000- <1> db 0feh, 0feh, 07ch, 038h, 010h, 000h, 000h, 000h, 000h, 000h, 000h, 010h, 038h, 07ch, 0feh, 07ch

254 000134F1 000010387CFE7C <1>

255 000134F8 381000000000000018- <1> db 038h, 010h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 03ch, 03ch, 0e7h, 0e7h, 0e7h, 018h, 018h

255 00013501 3C3CE7E7E71818 <1>

256 00013508 3C0000000000183C7E- <1> db 03ch, 000h, 000h, 000h, 000h, 000h, 018h, 03ch, 07eh, 0ffh, 0ffh, 07eh, 018h, 018h, 03ch, 000h

256 00013511 FFFF7E18183C00 <1>

257 00013518 00000000000000183C- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 03ch, 03ch, 018h, 000h, 000h, 000h, 000h, 000h

257 00013521 3C180000000000 <1>

258 00013528 FFFFFFFFFFE7C3C3E7- <1> db 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0e7h, 0c3h, 0c3h, 0e7h, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 000h, 000h

258 00013531 FFFFFFFFFF0000 <1>

259 00013538 00003C664242663C00- <1> db 000h, 000h, 03ch, 066h, 042h, 042h, 066h, 03ch, 000h, 000h, 000h, 000h, 0ffh, 0ffh, 0ffh, 0ffh

259 00013541 000000FFFFFFFF <1>

260 00013548 C399BDBD99C3FFFFFF- <1> db 0c3h, 099h, 0bdh, 0bdh, 099h, 0c3h, 0ffh, 0ffh, 0ffh, 0ffh, 000h, 000h, 01eh, 00eh, 01ah, 032h

260 00013551 FF00001E0E1A32 <1>

261 00013558 78CCCCCC7800000000- <1> db 078h, 0cch, 0cch, 0cch, 078h, 000h, 000h, 000h, 000h, 000h, 03ch, 066h, 066h, 066h, 03ch, 018h

261 00013561 003C6666663C18 <1>

262 00013568 7E181800000000003F- <1> db 07eh, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 03fh, 033h, 03fh, 030h, 030h, 030h, 070h, 0f0h

262 00013571 333F30303070F0 <1>

263 00013578 E000000000007F637F- <1> db 0e0h, 000h, 000h, 000h, 000h, 000h, 07fh, 063h, 07fh, 063h, 063h, 063h, 067h, 0e7h, 0e6h, 0c0h

263 00013581 63636367E7E6C0 <1>

264 00013588 000000001818DB3CE7- <1> db 000h, 000h, 000h, 000h, 018h, 018h, 0dbh, 03ch, 0e7h, 03ch, 0dbh, 018h, 018h, 000h, 000h, 000h

264 00013591 3CDB1818000000 <1>

265 00013598 000080C0E0F8FEF8E0- <1> db 000h, 000h, 080h, 0c0h, 0e0h, 0f8h, 0feh, 0f8h, 0e0h, 0c0h, 080h, 000h, 000h, 000h, 000h, 000h

265 000135A1 C0800000000000 <1>

266 000135A8 02060E3EFE3E0E0602- <1> db 002h, 006h, 00eh, 03eh, 0feh, 03eh, 00eh, 006h, 002h, 000h, 000h, 000h, 000h, 000h, 018h, 03ch

266 000135B1 0000000000183C <1>

267 000135B8 7E1818187E3C180000- <1> db 07eh, 018h, 018h, 018h, 07eh, 03ch, 018h, 000h, 000h, 000h, 000h, 000h, 066h, 066h, 066h, 066h

267 000135C1 00000066666666 <1>

268 000135C8 666600666600000000- <1> db 066h, 066h, 000h, 066h, 066h, 000h, 000h, 000h, 000h, 000h, 07fh, 0dbh, 0dbh, 0dbh, 07bh, 01bh

268 000135D1 007FDBDBDB7B1B <1>

269 000135D8 1B1B1B000000007CC6- <1> db 01bh, 01bh, 01bh, 000h, 000h, 000h, 000h, 07ch, 0c6h, 060h, 038h, 06ch, 0c6h, 0c6h, 06ch, 038h

269 000135E1 60386CC6C66C38 <1>

270 000135E8 0CC67C000000000000- <1> db 00ch, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0feh, 0feh, 0feh, 000h

270 000135F1 000000FEFEFE00 <1>

271 000135F8 00000000183C7E1818- <1> db 000h, 000h, 000h, 000h, 018h, 03ch, 07eh, 018h, 018h, 018h, 07eh, 03ch, 018h, 07eh, 000h, 000h

271 00013601 187E3C187E0000 <1>

272 00013608 0000183C7E18181818- <1> db 000h, 000h, 018h, 03ch, 07eh, 018h, 018h, 018h, 018h, 018h, 018h, 000h, 000h, 000h, 000h, 000h

272 00013611 18180000000000 <1>

273 00013618 1818181818187E3C18- <1> db 018h, 018h, 018h, 018h, 018h, 018h, 07eh, 03ch, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

273 00013621 00000000000000 <1>

274 00013628 180CFE0C1800000000- <1> db 018h, 00ch, 0feh, 00ch, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 030h, 060h

274 00013631 00000000003060 <1>

275 00013638 FE6030000000000000- <1> db 0feh, 060h, 030h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0c0h, 0c0h, 0c0h

275 00013641 00000000C0C0C0 <1>

276 00013648 FE0000000000000000- <1> db 0feh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 028h, 06ch, 0feh, 06ch, 028h, 000h

276 00013651 00286CFE6C2800 <1>

277 00013658 000000000000001038- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 010h, 038h, 038h, 07ch, 07ch, 0feh, 0feh, 000h, 000h

277 00013661 387C7CFEFE0000 <1>

278 00013668 0000000000FEFE7C7C- <1> db 000h, 000h, 000h, 000h, 000h, 0feh, 0feh, 07ch, 07ch, 038h, 038h, 010h, 000h, 000h, 000h, 000h

278 00013671 38381000000000 <1>

279 00013678 000000000000000000- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

279 00013681 00000000000000 <1>

280 00013688 183C3C3C1818001818- <1> db 018h, 03ch, 03ch, 03ch, 018h, 018h, 000h, 018h, 018h, 000h, 000h, 000h, 000h, 066h, 066h, 066h

280 00013691 00000000666666 <1>

281 00013698 240000000000000000- <1> db 024h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 06ch, 06ch, 0feh, 06ch

281 000136A1 0000006C6CFE6C <1>

282 000136A8 6C6CFE6C6C00000018- <1> db 06ch, 06ch, 0feh, 06ch, 06ch, 000h, 000h, 000h, 018h, 018h, 07ch, 0c6h, 0c2h, 0c0h, 07ch, 006h

282 000136B1 187CC6C2C07C06 <1>

283 000136B8 86C67C181800000000- <1> db 086h, 0c6h, 07ch, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 0c2h, 0c6h, 00ch, 018h, 030h, 066h

283 000136C1 00C2C60C183066 <1>

284 000136C8 C60000000000386C6C- <1> db 0c6h, 000h, 000h, 000h, 000h, 000h, 038h, 06ch, 06ch, 038h, 076h, 0dch, 0cch, 0cch, 076h, 000h

284 000136D1 3876DCCCCC7600 <1>

285 000136D8 000000303030600000- <1> db 000h, 000h, 000h, 030h, 030h, 030h, 060h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

285 000136E1 00000000000000 <1>

286 000136E8 00000C183030303030- <1> db 000h, 000h, 00ch, 018h, 030h, 030h, 030h, 030h, 030h, 018h, 00ch, 000h, 000h, 000h, 000h, 000h

286 000136F1 180C0000000000 <1>

287 000136F8 30180C0C0C0C0C1830- <1> db 030h, 018h, 00ch, 00ch, 00ch, 00ch, 00ch, 018h, 030h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

287 00013701 00000000000000 <1>

288 00013708 663CFF3C6600000000- <1> db 066h, 03ch, 0ffh, 03ch, 066h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h

288 00013711 00000000001818 <1>

289 00013718 7E1818000000000000- <1> db 07eh, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

289 00013721 00000000000000 <1>

290 00013728 181818300000000000- <1> db 018h, 018h, 018h, 030h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0feh, 000h, 000h, 000h

290 00013731 000000FE000000 <1>

291 00013738 000000000000000000- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 000h

291 00013741 00000000181800 <1>

292 00013748 0000000002060C1830- <1> db 000h, 000h, 000h, 000h, 002h, 006h, 00ch, 018h, 030h, 060h, 0c0h, 080h, 000h, 000h, 000h, 000h

292 00013751 60C08000000000 <1>

293 00013758 00007CC6CEDEF6E6C6- <1> db 000h, 000h, 07ch, 0c6h, 0ceh, 0deh, 0f6h, 0e6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h

293 00013761 C67C0000000000 <1>

294 00013768 18387818181818187E- <1> db 018h, 038h, 078h, 018h, 018h, 018h, 018h, 018h, 07eh, 000h, 000h, 000h, 000h, 000h, 07ch, 0c6h

294 00013771 00000000007CC6 <1>

295 00013778 060C183060C6FE0000- <1> db 006h, 00ch, 018h, 030h, 060h, 0c6h, 0feh, 000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 006h, 006h

295 00013781 0000007CC60606 <1>

296 00013788 3C0606C67C00000000- <1> db 03ch, 006h, 006h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 00ch, 01ch, 03ch, 06ch, 0cch, 0feh

296 00013791 000C1C3C6CCCFE <1>

297 00013798 0C0C1E0000000000FE- <1> db 00ch, 00ch, 01eh, 000h, 000h, 000h, 000h, 000h, 0feh, 0c0h, 0c0h, 0c0h, 0fch, 006h, 006h, 0c6h

297 000137A1 C0C0C0FC0606C6 <1>

298 000137A8 7C00000000003860C0- <1> db 07ch, 000h, 000h, 000h, 000h, 000h, 038h, 060h, 0c0h, 0c0h, 0fch, 0c6h, 0c6h, 0c6h, 07ch, 000h

298 000137B1 C0FCC6C6C67C00 <1>

299 000137B8 00000000FEC6060C18- <1> db 000h, 000h, 000h, 000h, 0feh, 0c6h, 006h, 00ch, 018h, 030h, 030h, 030h, 030h, 000h, 000h, 000h

299 000137C1 30303030000000 <1>

300 000137C8 00007CC6C6C67CC6C6- <1> db 000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 07ch, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h

300 000137D1 C67C0000000000 <1>

301 000137D8 7CC6C6C67E06060C78- <1> db 07ch, 0c6h, 0c6h, 0c6h, 07eh, 006h, 006h, 00ch, 078h, 000h, 000h, 000h, 000h, 000h, 000h, 018h

301 000137E1 00000000000018 <1>

302 000137E8 180000001818000000- <1> db 018h, 000h, 000h, 000h, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 000h

302 000137F1 00000000181800 <1>

303 000137F8 000018183000000000- <1> db 000h, 000h, 018h, 018h, 030h, 000h, 000h, 000h, 000h, 000h, 006h, 00ch, 018h, 030h, 060h, 030h

303 00013801 00060C18306030 <1>

304 00013808 180C06000000000000- <1> db 018h, 00ch, 006h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07eh, 000h, 000h, 07eh, 000h

304 00013811 00007E00007E00 <1>

305 00013818 000000000000603018- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 060h, 030h, 018h, 00ch, 006h, 00ch, 018h, 030h, 060h, 000h

305 00013821 0C060C18306000 <1>

306 00013828 000000007CC6C60C18- <1> db 000h, 000h, 000h, 000h, 07ch, 0c6h, 0c6h, 00ch, 018h, 018h, 000h, 018h, 018h, 000h, 000h, 000h

306 00013831 18001818000000 <1>

307 00013838 00007CC6C6DEDEDEDC- <1> db 000h, 000h, 07ch, 0c6h, 0c6h, 0deh, 0deh, 0deh, 0dch, 0c0h, 07ch, 000h, 000h, 000h, 000h, 000h

307 00013841 C07C0000000000 <1>

308 00013848 10386CC6C6FEC6C6C6- <1> db 010h, 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h, 000h, 0fch, 066h

308 00013851 0000000000FC66 <1>

309 00013858 66667C666666FC0000- <1> db 066h, 066h, 07ch, 066h, 066h, 066h, 0fch, 000h, 000h, 000h, 000h, 000h, 03ch, 066h, 0c2h, 0c0h

309 00013861 0000003C66C2C0 <1>

310 00013868 C0C0C2663C00000000- <1> db 0c0h, 0c0h, 0c2h, 066h, 03ch, 000h, 000h, 000h, 000h, 000h, 0f8h, 06ch, 066h, 066h, 066h, 066h

310 00013871 00F86C66666666 <1>

311 00013878 666CF80000000000FE- <1> db 066h, 06ch, 0f8h, 000h, 000h, 000h, 000h, 000h, 0feh, 066h, 062h, 068h, 078h, 068h, 062h, 066h

311 00013881 66626878686266 <1>

312 00013888 FE0000000000FE6662- <1> db 0feh, 000h, 000h, 000h, 000h, 000h, 0feh, 066h, 062h, 068h, 078h, 068h, 060h, 060h, 0f0h, 000h

312 00013891 6878686060F000 <1>

313 00013898 000000003C66C2C0C0- <1> db 000h, 000h, 000h, 000h, 03ch, 066h, 0c2h, 0c0h, 0c0h, 0deh, 0c6h, 066h, 03ah, 000h, 000h, 000h

313 000138A1 DEC6663A000000 <1>

314 000138A8 0000C6C6C6C6FEC6C6- <1> db 000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h, 000h

314 000138B1 C6C60000000000 <1>

315 000138B8 3C181818181818183C- <1> db 03ch, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h, 000h, 01eh, 00ch

315 000138C1 00000000001E0C <1>

316 000138C8 0C0C0C0CCCCC780000- <1> db 00ch, 00ch, 00ch, 00ch, 0cch, 0cch, 078h, 000h, 000h, 000h, 000h, 000h, 0e6h, 066h, 06ch, 06ch

316 000138D1 000000E6666C6C <1>

317 000138D8 786C6C66E600000000- <1> db 078h, 06ch, 06ch, 066h, 0e6h, 000h, 000h, 000h, 000h, 000h, 0f0h, 060h, 060h, 060h, 060h, 060h

317 000138E1 00F06060606060 <1>

318 000138E8 6266FE0000000000C6- <1> db 062h, 066h, 0feh, 000h, 000h, 000h, 000h, 000h, 0c6h, 0eeh, 0feh, 0feh, 0d6h, 0c6h, 0c6h, 0c6h

318 000138F1 EEFEFED6C6C6C6 <1>

319 000138F8 C60000000000C6E6F6- <1> db 0c6h, 000h, 000h, 000h, 000h, 000h, 0c6h, 0e6h, 0f6h, 0feh, 0deh, 0ceh, 0c6h, 0c6h, 0c6h, 000h

319 00013901 FEDECEC6C6C600 <1>

320 00013908 00000000386CC6C6C6- <1> db 000h, 000h, 000h, 000h, 038h, 06ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 06ch, 038h, 000h, 000h, 000h

320 00013911 C6C66C38000000 <1>

321 00013918 0000FC6666667C6060- <1> db 000h, 000h, 0fch, 066h, 066h, 066h, 07ch, 060h, 060h, 060h, 0f0h, 000h, 000h, 000h, 000h, 000h

321 00013921 60F00000000000 <1>

322 00013928 7CC6C6C6C6D6DE7C0C- <1> db 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0d6h, 0deh, 07ch, 00ch, 00eh, 000h, 000h, 000h, 000h, 0fch, 066h

322 00013931 0E00000000FC66 <1>

323 00013938 66667C6C6666E60000- <1> db 066h, 066h, 07ch, 06ch, 066h, 066h, 0e6h, 000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 0c6h, 060h

323 00013941 0000007CC6C660 <1>

324 00013948 380CC6C67C00000000- <1> db 038h, 00ch, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 07eh, 07eh, 05ah, 018h, 018h, 018h

324 00013951 007E7E5A181818 <1>

325 00013958 18183C0000000000C6- <1> db 018h, 018h, 03ch, 000h, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h

325 00013961 C6C6C6C6C6C6C6 <1>

326 00013968 7C0000000000C6C6C6- <1> db 07ch, 000h, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 06ch, 038h, 010h, 000h

326 00013971 C6C6C66C381000 <1>

327 00013978 00000000C6C6C6C6D6- <1> db 000h, 000h, 000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0d6h, 0d6h, 0feh, 07ch, 06ch, 000h, 000h, 000h

327 00013981 D6FE7C6C000000 <1>

328 00013988 0000C6C66C3838386C- <1> db 000h, 000h, 0c6h, 0c6h, 06ch, 038h, 038h, 038h, 06ch, 0c6h, 0c6h, 000h, 000h, 000h, 000h, 000h

328 00013991 C6C60000000000 <1>

329 00013998 666666663C1818183C- <1> db 066h, 066h, 066h, 066h, 03ch, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h, 000h, 0feh, 0c6h

329 000139A1 0000000000FEC6 <1>

330 000139A8 8C183060C2C6FE0000- <1> db 08ch, 018h, 030h, 060h, 0c2h, 0c6h, 0feh, 000h, 000h, 000h, 000h, 000h, 03ch, 030h, 030h, 030h

330 000139B1 0000003C303030 <1>

331 000139B8 303030303C00000000- <1> db 030h, 030h, 030h, 030h, 03ch, 000h, 000h, 000h, 000h, 000h, 080h, 0c0h, 0e0h, 070h, 038h, 01ch

331 000139C1 0080C0E070381C <1>

332 000139C8 0E060200000000003C- <1> db 00eh, 006h, 002h, 000h, 000h, 000h, 000h, 000h, 03ch, 00ch, 00ch, 00ch, 00ch, 00ch, 00ch, 00ch

332 000139D1 0C0C0C0C0C0C0C <1>

333 000139D8 3C00000010386CC600- <1> db 03ch, 000h, 000h, 000h, 010h, 038h, 06ch, 0c6h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

333 000139E1 00000000000000 <1>

334 000139E8 000000000000000000- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh, 000h

334 000139F1 0000000000FF00 <1>

335 000139F8 303018000000000000- <1> db 030h, 030h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

335 00013A01 00000000000000 <1>

336 00013A08 000000780C7CCCCC76- <1> db 000h, 000h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h, 000h, 0e0h, 060h

336 00013A11 0000000000E060 <1>

337 00013A18 60786C6666667C0000- <1> db 060h, 078h, 06ch, 066h, 066h, 066h, 07ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07ch

337 00013A21 0000000000007C <1>

338 00013A28 C6C0C0C67C00000000- <1> db 0c6h, 0c0h, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 01ch, 00ch, 00ch, 03ch, 06ch, 0cch

338 00013A31 001C0C0C3C6CCC <1>

339 00013A38 CCCC76000000000000- <1> db 0cch, 0cch, 076h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c6h

339 00013A41 00007CC6FEC0C6 <1>

340 00013A48 7C0000000000386C64- <1> db 07ch, 000h, 000h, 000h, 000h, 000h, 038h, 06ch, 064h, 060h, 0f0h, 060h, 060h, 060h, 0f0h, 000h

340 00013A51 60F0606060F000 <1>

341 00013A58 0000000000000076CC- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 076h, 0cch, 0cch, 0cch, 07ch, 00ch, 0cch, 078h, 000h

341 00013A61 CCCC7C0CCC7800 <1>

342 00013A68 0000E060606C766666- <1> db 000h, 000h, 0e0h, 060h, 060h, 06ch, 076h, 066h, 066h, 066h, 0e6h, 000h, 000h, 000h, 000h, 000h

342 00013A71 66E60000000000 <1>

343 00013A78 18180038181818183C- <1> db 018h, 018h, 000h, 038h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h, 000h, 006h, 006h

343 00013A81 00000000000606 <1>

344 00013A88 000E0606060666663C- <1> db 000h, 00eh, 006h, 006h, 006h, 006h, 066h, 066h, 03ch, 000h, 000h, 000h, 0e0h, 060h, 060h, 066h

344 00013A91 000000E0606066 <1>

345 00013A98 6C786C66E600000000- <1> db 06ch, 078h, 06ch, 066h, 0e6h, 000h, 000h, 000h, 000h, 000h, 038h, 018h, 018h, 018h, 018h, 018h

345 00013AA1 00381818181818 <1>

346 00013AA8 18183C000000000000- <1> db 018h, 018h, 03ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ech, 0feh, 0d6h, 0d6h, 0d6h

346 00013AB1 0000ECFED6D6D6 <1>

347 00013AB8 C60000000000000000- <1> db 0c6h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0dch, 066h, 066h, 066h, 066h, 066h, 000h

347 00013AC1 DC666666666600 <1>

348 00013AC8 000000000000007CC6- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h

348 00013AD1 C6C6C67C000000 <1>

349 00013AD8 0000000000DC666666- <1> db 000h, 000h, 000h, 000h, 000h, 0dch, 066h, 066h, 066h, 07ch, 060h, 060h, 0f0h, 000h, 000h, 000h

349 00013AE1 7C6060F0000000 <1>

350 00013AE8 00000076CCCCCC7C0C- <1> db 000h, 000h, 000h, 076h, 0cch, 0cch, 0cch, 07ch, 00ch, 00ch, 01eh, 000h, 000h, 000h, 000h, 000h

350 00013AF1 0C1E0000000000 <1>

351 00013AF8 00DC76666060F00000- <1> db 000h, 0dch, 076h, 066h, 060h, 060h, 0f0h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07ch

351 00013B01 0000000000007C <1>

352 00013B08 C6701CC67C00000000- <1> db 0c6h, 070h, 01ch, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 010h, 030h, 030h, 0fch, 030h, 030h

352 00013B11 00103030FC3030 <1>

353 00013B18 30361C000000000000- <1> db 030h, 036h, 01ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0cch, 0cch, 0cch, 0cch, 0cch

353 00013B21 0000CCCCCCCCCC <1>

354 00013B28 760000000000000000- <1> db 076h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 066h, 066h, 066h, 066h, 03ch, 018h, 000h

354 00013B31 666666663C1800 <1>

355 00013B38 00000000000000C6C6- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 0d6h, 0d6h, 0feh, 06ch, 000h, 000h, 000h

355 00013B41 D6D6FE6C000000 <1>

356 00013B48 0000000000C66C3838- <1> db 000h, 000h, 000h, 000h, 000h, 0c6h, 06ch, 038h, 038h, 06ch, 0c6h, 000h, 000h, 000h, 000h, 000h

356 00013B51 6CC60000000000 <1>

357 00013B58 000000C6C6C6C67E06- <1> db 000h, 000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 07eh, 006h, 00ch, 0f8h, 000h, 000h, 000h, 000h, 000h

357 00013B61 0CF80000000000 <1>

358 00013B68 00FECC183066FE0000- <1> db 000h, 0feh, 0cch, 018h, 030h, 066h, 0feh, 000h, 000h, 000h, 000h, 000h, 00eh, 018h, 018h, 018h

358 00013B71 0000000E181818 <1>

359 00013B78 701818180E00000000- <1> db 070h, 018h, 018h, 018h, 00eh, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 018h, 018h, 000h, 018h

359 00013B81 00181818180018 <1>

360 00013B88 181818000000000070- <1> db 018h, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 070h, 018h, 018h, 018h, 00eh, 018h, 018h, 018h

360 00013B91 1818180E181818 <1>

361 00013B98 70000000000076DC00- <1> db 070h, 000h, 000h, 000h, 000h, 000h, 076h, 0dch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

361 00013BA1 00000000000000 <1>

362 00013BA8 00000000000010386C- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 010h, 038h, 06ch, 0c6h, 0c6h, 0feh, 000h, 000h, 000h, 000h

362 00013BB1 C6C6FE00000000 <1>

363 00013BB8 00003C66C2C0C0C266- <1> db 000h, 000h, 03ch, 066h, 0c2h, 0c0h, 0c0h, 0c2h, 066h, 03ch, 00ch, 006h, 07ch, 000h, 000h, 000h

363 00013BC1 3C0C067C000000 <1>

364 00013BC8 CCCC00CCCCCCCCCC76- <1> db 0cch, 0cch, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h, 00ch, 018h, 030h

364 00013BD1 000000000C1830 <1>

365 00013BD8 007CC6FEC0C67C0000- <1> db 000h, 07ch, 0c6h, 0feh, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 010h, 038h, 06ch, 000h, 078h

365 00013BE1 000010386C0078 <1>

366 00013BE8 0C7CCCCC7600000000- <1> db 00ch, 07ch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h, 000h, 0cch, 0cch, 000h, 078h, 00ch, 07ch

366 00013BF1 00CCCC00780C7C <1>

367 00013BF8 CCCC76000000006030- <1> db 0cch, 0cch, 076h, 000h, 000h, 000h, 000h, 060h, 030h, 018h, 000h, 078h, 00ch, 07ch, 0cch, 0cch

367 00013C01 1800780C7CCCCC <1>

368 00013C08 7600000000386C3800- <1> db 076h, 000h, 000h, 000h, 000h, 038h, 06ch, 038h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 076h, 000h

368 00013C11 780C7CCCCC7600 <1>

369 00013C18 0000000000003C6660- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 03ch, 066h, 060h, 066h, 03ch, 00ch, 006h, 03ch, 000h, 000h

369 00013C21 663C0C063C0000 <1>

370 00013C28 0010386C007CC6FEC0- <1> db 000h, 010h, 038h, 06ch, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h

370 00013C31 C67C0000000000 <1>

371 00013C38 CCCC007CC6FEC0C67C- <1> db 0cch, 0cch, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 060h, 030h, 018h

371 00013C41 00000000603018 <1>

372 00013C48 007CC6FEC0C67C0000- <1> db 000h, 07ch, 0c6h, 0feh, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 066h, 066h, 000h, 038h

372 00013C51 00000066660038 <1>

373 00013C58 181818183C00000000- <1> db 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h, 018h, 03ch, 066h, 000h, 038h, 018h, 018h

373 00013C61 183C6600381818 <1>

374 00013C68 18183C000000006030- <1> db 018h, 018h, 03ch, 000h, 000h, 000h, 000h, 060h, 030h, 018h, 000h, 038h, 018h, 018h, 018h, 018h

374 00013C71 18003818181818 <1>

375 00013C78 3C00000000C6C61038- <1> db 03ch, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 010h, 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 000h

375 00013C81 6CC6C6FEC6C600 <1>

376 00013C88 0000386C3800386CC6- <1> db 000h, 000h, 038h, 06ch, 038h, 000h, 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 000h, 000h, 000h

376 00013C91 C6FEC6C6000000 <1>

377 00013C98 18306000FE66607C60- <1> db 018h, 030h, 060h, 000h, 0feh, 066h, 060h, 07ch, 060h, 066h, 0feh, 000h, 000h, 000h, 000h, 000h

377 00013CA1 66FE0000000000 <1>

378 00013CA8 0000CC76367ED8D86E- <1> db 000h, 000h, 0cch, 076h, 036h, 07eh, 0d8h, 0d8h, 06eh, 000h, 000h, 000h, 000h, 000h, 03eh, 06ch

378 00013CB1 00000000003E6C <1>

379 00013CB8 CCCCFECCCCCCCE0000- <1> db 0cch, 0cch, 0feh, 0cch, 0cch, 0cch, 0ceh, 000h, 000h, 000h, 000h, 010h, 038h, 06ch, 000h, 07ch

379 00013CC1 000010386C007C <1>

380 00013CC8 C6C6C6C67C00000000- <1> db 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 000h, 07ch, 0c6h, 0c6h

380 00013CD1 00C6C6007CC6C6 <1>

381 00013CD8 C6C67C000000006030- <1> db 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 060h, 030h, 018h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h

381 00013CE1 18007CC6C6C6C6 <1>

382 00013CE8 7C000000003078CC00- <1> db 07ch, 000h, 000h, 000h, 000h, 030h, 078h, 0cch, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h

382 00013CF1 CCCCCCCCCC7600 <1>

383 00013CF8 00000060301800CCCC- <1> db 000h, 000h, 000h, 060h, 030h, 018h, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h

383 00013D01 CCCCCC76000000 <1>

384 00013D08 0000C6C600C6C6C6C6- <1> db 000h, 000h, 0c6h, 0c6h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 07eh, 006h, 00ch, 078h, 000h, 000h, 0c6h

384 00013D11 7E060C780000C6 <1>

385 00013D18 C6386CC6C6C6C66C38- <1> db 0c6h, 038h, 06ch, 0c6h, 0c6h, 0c6h, 0c6h, 06ch, 038h, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 000h

385 00013D21 00000000C6C600 <1>

386 00013D28 C6C6C6C6C6C67C0000- <1> db 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 018h, 018h, 03ch, 066h, 060h

386 00013D31 000018183C6660 <1>

387 00013D38 60663C181800000000- <1> db 060h, 066h, 03ch, 018h, 018h, 000h, 000h, 000h, 000h, 038h, 06ch, 064h, 060h, 0f0h, 060h, 060h

387 00013D41 386C6460F06060 <1>

388 00013D48 60E6FC000000000066- <1> db 060h, 0e6h, 0fch, 000h, 000h, 000h, 000h, 000h, 066h, 066h, 03ch, 018h, 07eh, 018h, 07eh, 018h

388 00013D51 663C187E187E18 <1>

389 00013D58 1800000000F8CCCCF8- <1> db 018h, 000h, 000h, 000h, 000h, 0f8h, 0cch, 0cch, 0f8h, 0c4h, 0cch, 0deh, 0cch, 0cch, 0c6h, 000h

389 00013D61 C4CCDECCCCC600 <1>

390 00013D68 0000000E1B1818187E- <1> db 000h, 000h, 000h, 00eh, 01bh, 018h, 018h, 018h, 07eh, 018h, 018h, 018h, 018h, 0d8h, 070h, 000h

390 00013D71 18181818D87000 <1>

391 00013D78 0018306000780C7CCC- <1> db 000h, 018h, 030h, 060h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h, 00ch

391 00013D81 CC76000000000C <1>

392 00013D88 18300038181818183C- <1> db 018h, 030h, 000h, 038h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h, 018h, 030h, 060h

392 00013D91 00000000183060 <1>

393 00013D98 007CC6C6C6C67C0000- <1> db 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 018h, 030h, 060h, 000h, 0cch

393 00013DA1 000018306000CC <1>

394 00013DA8 CCCCCCCC7600000000- <1> db 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h, 000h, 076h, 0dch, 000h, 0dch, 066h, 066h

394 00013DB1 0076DC00DC6666 <1>

395 00013DB8 66666600000076DC00- <1> db 066h, 066h, 066h, 000h, 000h, 000h, 076h, 0dch, 000h, 0c6h, 0e6h, 0f6h, 0feh, 0deh, 0ceh, 0c6h

395 00013DC1 C6E6F6FEDECEC6 <1>

396 00013DC8 C6000000003C6C6C3E- <1> db 0c6h, 000h, 000h, 000h, 000h, 03ch, 06ch, 06ch, 03eh, 000h, 07eh, 000h, 000h, 000h, 000h, 000h

396 00013DD1 007E0000000000 <1>

397 00013DD8 000000386C6C38007C- <1> db 000h, 000h, 000h, 038h, 06ch, 06ch, 038h, 000h, 07ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h

397 00013DE1 00000000000000 <1>

398 00013DE8 0000303000303060C6- <1> db 000h, 000h, 030h, 030h, 000h, 030h, 030h, 060h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h

398 00013DF1 C67C0000000000 <1>

399 00013DF8 00000000FEC0C0C000- <1> db 000h, 000h, 000h, 000h, 0feh, 0c0h, 0c0h, 0c0h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

399 00013E01 00000000000000 <1>

400 00013E08 0000FE060606000000- <1> db 000h, 000h, 0feh, 006h, 006h, 006h, 000h, 000h, 000h, 000h, 000h, 0c0h, 0c0h, 0c6h, 0cch, 0d8h

400 00013E11 0000C0C0C6CCD8 <1>

401 00013E18 3060DC860C183E0000- <1> db 030h, 060h, 0dch, 086h, 00ch, 018h, 03eh, 000h, 000h, 0c0h, 0c0h, 0c6h, 0cch, 0d8h, 030h, 066h

401 00013E21 C0C0C6CCD83066 <1>

402 00013E28 CE9E3E060600000018- <1> db 0ceh, 09eh, 03eh, 006h, 006h, 000h, 000h, 000h, 018h, 018h, 000h, 018h, 018h, 03ch, 03ch, 03ch

402 00013E31 180018183C3C3C <1>

403 00013E38 180000000000000036- <1> db 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 036h, 06ch, 0d8h, 06ch, 036h, 000h, 000h, 000h

403 00013E41 6CD86C36000000 <1>

404 00013E48 000000000000D86C36- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 0d8h, 06ch, 036h, 06ch, 0d8h, 000h, 000h, 000h, 000h, 000h

404 00013E51 6CD80000000000 <1>

405 00013E58 114411441144114411- <1> db 011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h, 055h, 0aah

405 00013E61 441144114455AA <1>

406 00013E68 55AA55AA55AA55AA55- <1> db 055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah, 0ddh, 077h, 0ddh, 077h

406 00013E71 AA55AADD77DD77 <1>

407 00013E78 DD77DD77DD77DD77DD- <1> db 0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 018h, 018h, 018h, 018h, 018h, 018h

407 00013E81 77181818181818 <1>

408 00013E88 181818181818181818- <1> db 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 0f8h

408 00013E91 181818181818F8 <1>

409 00013E98 181818181818181818- <1> db 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 0f8h, 018h, 0f8h, 018h, 018h

409 00013EA1 1818F818F81818 <1>

410 00013EA8 181818183636363636- <1> db 018h, 018h, 018h, 018h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 0f6h, 036h, 036h, 036h, 036h

410 00013EB1 3636F636363636 <1>

411 00013EB8 363600000000000000- <1> db 036h, 036h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0feh, 036h, 036h, 036h, 036h, 036h, 036h

411 00013EC1 FE363636363636 <1>

412 00013EC8 0000000000F818F818- <1> db 000h, 000h, 000h, 000h, 000h, 0f8h, 018h, 0f8h, 018h, 018h, 018h, 018h, 018h, 018h, 036h, 036h

412 00013ED1 18181818183636 <1>

413 00013ED8 363636F606F6363636- <1> db 036h, 036h, 036h, 0f6h, 006h, 0f6h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h

413 00013EE1 36363636363636 <1>

414 00013EE8 363636363636363636- <1> db 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 000h, 000h, 000h, 000h, 000h, 0feh

414 00013EF1 360000000000FE <1>

415 00013EF8 06F636363636363636- <1> db 006h, 0f6h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 0f6h, 006h, 0feh

415 00013F01 36363636F606FE <1>

416 00013F08 000000000000363636- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 0feh, 000h, 000h

416 00013F11 36363636FE0000 <1>

417 00013F18 000000001818181818- <1> db 000h, 000h, 000h, 000h, 018h, 018h, 018h, 018h, 018h, 0f8h, 018h, 0f8h, 000h, 000h, 000h, 000h

417 00013F21 F818F800000000 <1>

418 00013F28 000000000000000000- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0f8h, 018h, 018h, 018h, 018h, 018h, 018h

418 00013F31 F8181818181818 <1>

419 00013F38 181818181818181F00- <1> db 018h, 018h, 018h, 018h, 018h, 018h, 018h, 01fh, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h

419 00013F41 00000000001818 <1>

420 00013F48 1818181818FF000000- <1> db 018h, 018h, 018h, 018h, 018h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

420 00013F51 00000000000000 <1>

421 00013F58 000000FF1818181818- <1> db 000h, 000h, 000h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h

421 00013F61 18181818181818 <1>

422 00013F68 181F18181818181800- <1> db 018h, 01fh, 018h, 018h, 018h, 018h, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh

422 00013F71 000000000000FF <1>

423 00013F78 000000000000181818- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 0ffh, 018h, 018h

423 00013F81 18181818FF1818 <1>

424 00013F88 181818181818181818- <1> db 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 01fh, 018h, 01fh, 018h, 018h, 018h, 018h

424 00013F91 1F181F18181818 <1>

425 00013F98 181836363636363636- <1> db 018h, 018h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 037h, 036h, 036h, 036h, 036h, 036h, 036h

425 00013FA1 37363636363636 <1>

426 00013FA8 363636363637303F00- <1> db 036h, 036h, 036h, 036h, 036h, 037h, 030h, 03fh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

426 00013FB1 00000000000000 <1>

427 00013FB8 0000003F3037363636- <1> db 000h, 000h, 000h, 03fh, 030h, 037h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h

427 00013FC1 36363636363636 <1>

428 00013FC8 36F700FF0000000000- <1> db 036h, 0f7h, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh

428 00013FD1 000000000000FF <1>

429 00013FD8 00F736363636363636- <1> db 000h, 0f7h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 037h, 030h, 037h

429 00013FE1 36363636373037 <1>

430 00013FE8 363636363636000000- <1> db 036h, 036h, 036h, 036h, 036h, 036h, 000h, 000h, 000h, 000h, 000h, 0ffh, 000h, 0ffh, 000h, 000h

430 00013FF1 0000FF00FF0000 <1>

431 00013FF8 000000003636363636- <1> db 000h, 000h, 000h, 000h, 036h, 036h, 036h, 036h, 036h, 0f7h, 000h, 0f7h, 036h, 036h, 036h, 036h

431 00014001 F700F736363636 <1>

432 00014008 36361818181818FF00- <1> db 036h, 036h, 018h, 018h, 018h, 018h, 018h, 0ffh, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h

432 00014011 FF000000000000 <1>

433 00014018 36363636363636FF00- <1> db 036h, 036h, 036h, 036h, 036h, 036h, 036h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

433 00014021 00000000000000 <1>

434 00014028 000000FF00FF181818- <1> db 000h, 000h, 000h, 0ffh, 000h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 000h, 000h, 000h, 000h

434 00014031 18181800000000 <1>

435 00014038 000000FF3636363636- <1> db 000h, 000h, 000h, 0ffh, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h

435 00014041 36363636363636 <1>

436 00014048 363F00000000000018- <1> db 036h, 03fh, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 018h, 018h, 018h, 01fh, 018h, 01fh

436 00014051 181818181F181F <1>

437 00014058 000000000000000000- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 01fh, 018h, 01fh, 018h, 018h

437 00014061 00001F181F1818 <1>

438 00014068 181818180000000000- <1> db 018h, 018h, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 03fh, 036h, 036h, 036h, 036h

438 00014071 00003F36363636 <1>

439 00014078 363636363636363636- <1> db 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 0ffh, 036h, 036h, 036h, 036h, 036h, 036h

439 00014081 FF363636363636 <1>

440 00014088 1818181818FF18FF18- <1> db 018h, 018h, 018h, 018h, 018h, 0ffh, 018h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h

440 00014091 18181818181818 <1>

441 00014098 1818181818F8000000- <1> db 018h, 018h, 018h, 018h, 018h, 0f8h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

441 000140A1 00000000000000 <1>

442 000140A8 0000001F1818181818- <1> db 000h, 000h, 000h, 01fh, 018h, 018h, 018h, 018h, 018h, 018h, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh

442 000140B1 18FFFFFFFFFFFF <1>

443 000140B8 FFFFFFFFFFFFFFFF00- <1> db 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh

443 000140C1 000000000000FF <1>

444 000140C8 FFFFFFFFFFFFF0F0F0- <1> db 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h

444 000140D1 F0F0F0F0F0F0F0 <1>

445 000140D8 F0F0F0F00F0F0F0F0F- <1> db 0f0h, 0f0h, 0f0h, 0f0h, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh

445 000140E1 0F0F0F0F0F0F0F <1>

446 000140E8 0F0FFFFFFFFFFFFFFF- <1> db 00fh, 00fh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h

446 000140F1 00000000000000 <1>

447 000140F8 000000000076DCD8D8- <1> db 000h, 000h, 000h, 000h, 000h, 076h, 0dch, 0d8h, 0d8h, 0dch, 076h, 000h, 000h, 000h, 000h, 000h

447 00014101 DC760000000000 <1>

448 00014108 00007CC6FCC6C6FCC0- <1> db 000h, 000h, 07ch, 0c6h, 0fch, 0c6h, 0c6h, 0fch, 0c0h, 0c0h, 040h, 000h, 000h, 000h, 0feh, 0c6h

448 00014111 C040000000FEC6 <1>

449 00014118 C6C0C0C0C0C0C00000- <1> db 0c6h, 0c0h, 0c0h, 0c0h, 0c0h, 0c0h, 0c0h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0feh, 06ch

449 00014121 0000000000FE6C <1>

450 00014128 6C6C6C6C6C00000000- <1> db 06ch, 06ch, 06ch, 06ch, 06ch, 000h, 000h, 000h, 000h, 000h, 0feh, 0c6h, 060h, 030h, 018h, 030h

450 00014131 00FEC660301830 <1>

451 00014138 60C6FE000000000000- <1> db 060h, 0c6h, 0feh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07eh, 0d8h, 0d8h, 0d8h, 0d8h

451 00014141 00007ED8D8D8D8 <1>

452 00014148 700000000000000066- <1> db 070h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 066h, 066h, 066h, 066h, 07ch, 060h, 060h, 0c0h

452 00014151 6666667C6060C0 <1>

453 00014158 00000000000076DC18- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 076h, 0dch, 018h, 018h, 018h, 018h, 018h, 000h, 000h, 000h

453 00014161 18181818000000 <1>

454 00014168 00007E183C6666663C- <1> db 000h, 000h, 07eh, 018h, 03ch, 066h, 066h, 066h, 03ch, 018h, 07eh, 000h, 000h, 000h, 000h, 000h

454 00014171 187E0000000000 <1>

455 00014178 386CC6C6FEC6C66C38- <1> db 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 06ch, 038h, 000h, 000h, 000h, 000h, 000h, 038h, 06ch

455 00014181 0000000000386C <1>

456 00014188 C6C6C66C6C6CEE0000- <1> db 0c6h, 0c6h, 0c6h, 06ch, 06ch, 06ch, 0eeh, 000h, 000h, 000h, 000h, 000h, 01eh, 030h, 018h, 00ch

456 00014191 0000001E30180C <1>

457 00014198 3E6666663C00000000- <1> db 03eh, 066h, 066h, 066h, 03ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07eh, 0dbh, 0dbh

457 000141A1 000000007EDBDB <1>

458 000141A8 7E0000000000000003- <1> db 07eh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 003h, 006h, 07eh, 0dbh, 0dbh, 0f3h, 07eh, 060h

458 000141B1 067EDBDBF37E60 <1>

459 000141B8 C000000000001C3060- <1> db 0c0h, 000h, 000h, 000h, 000h, 000h, 01ch, 030h, 060h, 060h, 07ch, 060h, 060h, 030h, 01ch, 000h

459 000141C1 607C6060301C00 <1>

460 000141C8 00000000007CC6C6C6- <1> db 000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h

460 000141D1 C6C6C6C6000000 <1>

461 000141D8 000000FE0000FE0000- <1> db 000h, 000h, 000h, 0feh, 000h, 000h, 0feh, 000h, 000h, 0feh, 000h, 000h, 000h, 000h, 000h, 000h

461 000141E1 FE000000000000 <1>

462 000141E8 0018187E18180000FF- <1> db 000h, 018h, 018h, 07eh, 018h, 018h, 000h, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 030h, 018h

462 000141F1 00000000003018 <1>

463 000141F8 0C060C1830007E0000- <1> db 00ch, 006h, 00ch, 018h, 030h, 000h, 07eh, 000h, 000h, 000h, 000h, 000h, 00ch, 018h, 030h, 060h

463 00014201 0000000C183060 <1>

464 00014208 30180C007E00000000- <1> db 030h, 018h, 00ch, 000h, 07eh, 000h, 000h, 000h, 000h, 000h, 00eh, 01bh, 01bh, 018h, 018h, 018h

464 00014211 000E1B1B181818 <1>

465 00014218 181818181818181818- <1> db 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 0d8h, 0d8h

465 00014221 1818181818D8D8 <1>

466 00014228 700000000000001818- <1> db 070h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 000h, 07eh, 000h, 018h, 018h, 000h, 000h

466 00014231 007E0018180000 <1>

467 00014238 00000000000076DC00- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 076h, 0dch, 000h, 076h, 0dch, 000h, 000h, 000h, 000h, 000h

467 00014241 76DC0000000000 <1>

468 00014248 00386C6C3800000000- <1> db 000h, 038h, 06ch, 06ch, 038h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

468 00014251 00000000000000 <1>

469 00014258 000000001818000000- <1> db 000h, 000h, 000h, 000h, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

469 00014261 00000000000000 <1>

470 00014268 000000180000000000- <1> db 000h, 000h, 000h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 00fh, 00ch, 00ch, 00ch, 00ch

470 00014271 00000F0C0C0C0C <1>

471 00014278 0CEC6C3C1C00000000- <1> db 00ch, 0ech, 06ch, 03ch, 01ch, 000h, 000h, 000h, 000h, 0d8h, 06ch, 06ch, 06ch, 06ch, 06ch, 000h

471 00014281 D86C6C6C6C6C00 <1>

472 00014288 0000000000000070D8- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 070h, 0d8h, 030h, 060h, 0c8h, 0f8h, 000h, 000h, 000h

472 00014291 3060C8F8000000 <1>

473 00014298 00000000000000007C- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07ch, 07ch, 07ch, 07ch, 07ch, 07ch, 000h, 000h

473 000142A1 7C7C7C7C7C0000 <1>

474 000142A8 000000000000000000- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

474 000142B1 00000000000000 <1>

475 <1> vgafont16:

476 000142B8 000000000000000000- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

476 000142C1 00000000000000 <1>

477 000142C8 00007E81A58181BD99- <1> db 000h, 000h, 07eh, 081h, 0a5h, 081h, 081h, 0bdh, 099h, 081h, 081h, 07eh, 000h, 000h, 000h, 000h

477 000142D1 81817E00000000 <1>

478 000142D8 00007EFFDBFFFFC3E7- <1> db 000h, 000h, 07eh, 0ffh, 0dbh, 0ffh, 0ffh, 0c3h, 0e7h, 0ffh, 0ffh, 07eh, 000h, 000h, 000h, 000h

478 000142E1 FFFF7E00000000 <1>

479 000142E8 000000006CFEFEFEFE- <1> db 000h, 000h, 000h, 000h, 06ch, 0feh, 0feh, 0feh, 0feh, 07ch, 038h, 010h, 000h, 000h, 000h, 000h

479 000142F1 7C381000000000 <1>

480 000142F8 0000000010387CFE7C- <1> db 000h, 000h, 000h, 000h, 010h, 038h, 07ch, 0feh, 07ch, 038h, 010h, 000h, 000h, 000h, 000h, 000h

480 00014301 38100000000000 <1>

481 00014308 000000183C3CE7E7E7- <1> db 000h, 000h, 000h, 018h, 03ch, 03ch, 0e7h, 0e7h, 0e7h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h

481 00014311 18183C00000000 <1>

482 00014318 000000183C7EFFFF7E- <1> db 000h, 000h, 000h, 018h, 03ch, 07eh, 0ffh, 0ffh, 07eh, 018h, 018h, 03ch, 000h, 000h, 000h, 000h

482 00014321 18183C00000000 <1>

483 00014328 000000000000183C3C- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 018h, 03ch, 03ch, 018h, 000h, 000h, 000h, 000h, 000h, 000h

483 00014331 18000000000000 <1>

484 00014338 FFFFFFFFFFFFE7C3C3- <1> db 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0e7h, 0c3h, 0c3h, 0e7h, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh

484 00014341 E7FFFFFFFFFFFF <1>

485 00014348 00000000003C664242- <1> db 000h, 000h, 000h, 000h, 000h, 03ch, 066h, 042h, 042h, 066h, 03ch, 000h, 000h, 000h, 000h, 000h

485 00014351 663C0000000000 <1>

486 00014358 FFFFFFFFFFC399BDBD- <1> db 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0c3h, 099h, 0bdh, 0bdh, 099h, 0c3h, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh

486 00014361 99C3FFFFFFFFFF <1>

487 00014368 00001E0E1A3278CCCC- <1> db 000h, 000h, 01eh, 00eh, 01ah, 032h, 078h, 0cch, 0cch, 0cch, 0cch, 078h, 000h, 000h, 000h, 000h

487 00014371 CCCC7800000000 <1>

488 00014378 00003C666666663C18- <1> db 000h, 000h, 03ch, 066h, 066h, 066h, 066h, 03ch, 018h, 07eh, 018h, 018h, 000h, 000h, 000h, 000h

488 00014381 7E181800000000 <1>

489 00014388 00003F333F30303030- <1> db 000h, 000h, 03fh, 033h, 03fh, 030h, 030h, 030h, 030h, 070h, 0f0h, 0e0h, 000h, 000h, 000h, 000h

489 00014391 70F0E000000000 <1>

490 00014398 00007F637F63636363- <1> db 000h, 000h, 07fh, 063h, 07fh, 063h, 063h, 063h, 063h, 067h, 0e7h, 0e6h, 0c0h, 000h, 000h, 000h

490 000143A1 67E7E6C0000000 <1>

491 000143A8 0000001818DB3CE73C- <1> db 000h, 000h, 000h, 018h, 018h, 0dbh, 03ch, 0e7h, 03ch, 0dbh, 018h, 018h, 000h, 000h, 000h, 000h

491 000143B1 DB181800000000 <1>

492 000143B8 0080C0E0F0F8FEF8F0- <1> db 000h, 080h, 0c0h, 0e0h, 0f0h, 0f8h, 0feh, 0f8h, 0f0h, 0e0h, 0c0h, 080h, 000h, 000h, 000h, 000h

492 000143C1 E0C08000000000 <1>

493 000143C8 0002060E1E3EFE3E1E- <1> db 000h, 002h, 006h, 00eh, 01eh, 03eh, 0feh, 03eh, 01eh, 00eh, 006h, 002h, 000h, 000h, 000h, 000h

493 000143D1 0E060200000000 <1>

494 000143D8 0000183C7E1818187E- <1> db 000h, 000h, 018h, 03ch, 07eh, 018h, 018h, 018h, 07eh, 03ch, 018h, 000h, 000h, 000h, 000h, 000h

494 000143E1 3C180000000000 <1>

495 000143E8 000066666666666666- <1> db 000h, 000h, 066h, 066h, 066h, 066h, 066h, 066h, 066h, 000h, 066h, 066h, 000h, 000h, 000h, 000h

495 000143F1 00666600000000 <1>

496 000143F8 00007FDBDBDB7B1B1B- <1> db 000h, 000h, 07fh, 0dbh, 0dbh, 0dbh, 07bh, 01bh, 01bh, 01bh, 01bh, 01bh, 000h, 000h, 000h, 000h

496 00014401 1B1B1B00000000 <1>

497 00014408 007CC660386CC6C66C- <1> db 000h, 07ch, 0c6h, 060h, 038h, 06ch, 0c6h, 0c6h, 06ch, 038h, 00ch, 0c6h, 07ch, 000h, 000h, 000h

497 00014411 380CC67C000000 <1>

498 00014418 0000000000000000FE- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0feh, 0feh, 0feh, 0feh, 000h, 000h, 000h, 000h

498 00014421 FEFEFE00000000 <1>

499 00014428 0000183C7E1818187E- <1> db 000h, 000h, 018h, 03ch, 07eh, 018h, 018h, 018h, 07eh, 03ch, 018h, 07eh, 000h, 000h, 000h, 000h

499 00014431 3C187E00000000 <1>

500 00014438 0000183C7E18181818- <1> db 000h, 000h, 018h, 03ch, 07eh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 000h, 000h, 000h, 000h

500 00014441 18181800000000 <1>

501 00014448 000018181818181818- <1> db 000h, 000h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 07eh, 03ch, 018h, 000h, 000h, 000h, 000h

501 00014451 7E3C1800000000 <1>

502 00014458 0000000000180CFE0C- <1> db 000h, 000h, 000h, 000h, 000h, 018h, 00ch, 0feh, 00ch, 018h, 000h, 000h, 000h, 000h, 000h, 000h

502 00014461 18000000000000 <1>

503 00014468 00000000003060FE60- <1> db 000h, 000h, 000h, 000h, 000h, 030h, 060h, 0feh, 060h, 030h, 000h, 000h, 000h, 000h, 000h, 000h

503 00014471 30000000000000 <1>

504 00014478 000000000000C0C0C0- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 0c0h, 0c0h, 0c0h, 0feh, 000h, 000h, 000h, 000h, 000h, 000h

504 00014481 FE000000000000 <1>

505 00014488 00000000002466FF66- <1> db 000h, 000h, 000h, 000h, 000h, 024h, 066h, 0ffh, 066h, 024h, 000h, 000h, 000h, 000h, 000h, 000h

505 00014491 24000000000000 <1>

506 00014498 000000001038387C7C- <1> db 000h, 000h, 000h, 000h, 010h, 038h, 038h, 07ch, 07ch, 0feh, 0feh, 000h, 000h, 000h, 000h, 000h

506 000144A1 FEFE0000000000 <1>

507 000144A8 00000000FEFE7C7C38- <1> db 000h, 000h, 000h, 000h, 0feh, 0feh, 07ch, 07ch, 038h, 038h, 010h, 000h, 000h, 000h, 000h, 000h

507 000144B1 38100000000000 <1>

508 000144B8 000000000000000000- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

508 000144C1 00000000000000 <1>

509 000144C8 0000183C3C3C181818- <1> db 000h, 000h, 018h, 03ch, 03ch, 03ch, 018h, 018h, 018h, 000h, 018h, 018h, 000h, 000h, 000h, 000h

509 000144D1 00181800000000 <1>

510 000144D8 006666662400000000- <1> db 000h, 066h, 066h, 066h, 024h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

510 000144E1 00000000000000 <1>

511 000144E8 0000006C6CFE6C6C6C- <1> db 000h, 000h, 000h, 06ch, 06ch, 0feh, 06ch, 06ch, 06ch, 0feh, 06ch, 06ch, 000h, 000h, 000h, 000h

511 000144F1 FE6C6C00000000 <1>

512 000144F8 18187CC6C2C07C0606- <1> db 018h, 018h, 07ch, 0c6h, 0c2h, 0c0h, 07ch, 006h, 006h, 086h, 0c6h, 07ch, 018h, 018h, 000h, 000h

512 00014501 86C67C18180000 <1>

513 00014508 00000000C2C60C1830- <1> db 000h, 000h, 000h, 000h, 0c2h, 0c6h, 00ch, 018h, 030h, 060h, 0c6h, 086h, 000h, 000h, 000h, 000h

513 00014511 60C68600000000 <1>

514 00014518 0000386C6C3876DCCC- <1> db 000h, 000h, 038h, 06ch, 06ch, 038h, 076h, 0dch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h

514 00014521 CCCC7600000000 <1>

515 00014528 003030306000000000- <1> db 000h, 030h, 030h, 030h, 060h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

515 00014531 00000000000000 <1>

516 00014538 00000C183030303030- <1> db 000h, 000h, 00ch, 018h, 030h, 030h, 030h, 030h, 030h, 030h, 018h, 00ch, 000h, 000h, 000h, 000h

516 00014541 30180C00000000 <1>

517 00014548 000030180C0C0C0C0C- <1> db 000h, 000h, 030h, 018h, 00ch, 00ch, 00ch, 00ch, 00ch, 00ch, 018h, 030h, 000h, 000h, 000h, 000h

517 00014551 0C183000000000 <1>

518 00014558 0000000000663CFF3C- <1> db 000h, 000h, 000h, 000h, 000h, 066h, 03ch, 0ffh, 03ch, 066h, 000h, 000h, 000h, 000h, 000h, 000h

518 00014561 66000000000000 <1>

519 00014568 000000000018187E18- <1> db 000h, 000h, 000h, 000h, 000h, 018h, 018h, 07eh, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 000h

519 00014571 18000000000000 <1>

520 00014578 000000000000000000- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 018h, 030h, 000h, 000h, 000h

520 00014581 18181830000000 <1>

521 00014588 00000000000000FE00- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0feh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

521 00014591 00000000000000 <1>

522 00014598 000000000000000000- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 000h, 000h, 000h, 000h

522 000145A1 00181800000000 <1>

523 000145A8 0000000002060C1830- <1> db 000h, 000h, 000h, 000h, 002h, 006h, 00ch, 018h, 030h, 060h, 0c0h, 080h, 000h, 000h, 000h, 000h

523 000145B1 60C08000000000 <1>

524 000145B8 00003C66C3C3DBDBC3- <1> db 000h, 000h, 03ch, 066h, 0c3h, 0c3h, 0dbh, 0dbh, 0c3h, 0c3h, 066h, 03ch, 000h, 000h, 000h, 000h

524 000145C1 C3663C00000000 <1>

525 000145C8 000018387818181818- <1> db 000h, 000h, 018h, 038h, 078h, 018h, 018h, 018h, 018h, 018h, 018h, 07eh, 000h, 000h, 000h, 000h

525 000145D1 18187E00000000 <1>

526 000145D8 00007CC6060C183060- <1> db 000h, 000h, 07ch, 0c6h, 006h, 00ch, 018h, 030h, 060h, 0c0h, 0c6h, 0feh, 000h, 000h, 000h, 000h

526 000145E1 C0C6FE00000000 <1>

527 000145E8 00007CC606063C0606- <1> db 000h, 000h, 07ch, 0c6h, 006h, 006h, 03ch, 006h, 006h, 006h, 0c6h, 07ch, 000h, 000h, 000h, 000h

527 000145F1 06C67C00000000 <1>

528 000145F8 00000C1C3C6CCCFE0C- <1> db 000h, 000h, 00ch, 01ch, 03ch, 06ch, 0cch, 0feh, 00ch, 00ch, 00ch, 01eh, 000h, 000h, 000h, 000h

528 00014601 0C0C1E00000000 <1>

529 00014608 0000FEC0C0C0FC0606- <1> db 000h, 000h, 0feh, 0c0h, 0c0h, 0c0h, 0fch, 006h, 006h, 006h, 0c6h, 07ch, 000h, 000h, 000h, 000h

529 00014611 06C67C00000000 <1>

530 00014618 00003860C0C0FCC6C6- <1> db 000h, 000h, 038h, 060h, 0c0h, 0c0h, 0fch, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h

530 00014621 C6C67C00000000 <1>

531 00014628 0000FEC606060C1830- <1> db 000h, 000h, 0feh, 0c6h, 006h, 006h, 00ch, 018h, 030h, 030h, 030h, 030h, 000h, 000h, 000h, 000h

531 00014631 30303000000000 <1>

532 00014638 00007CC6C6C67CC6C6- <1> db 000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h

532 00014641 C6C67C00000000 <1>

533 00014648 00007CC6C6C67E0606- <1> db 000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 07eh, 006h, 006h, 006h, 00ch, 078h, 000h, 000h, 000h, 000h

533 00014651 060C7800000000 <1>

534 00014658 000000001818000000- <1> db 000h, 000h, 000h, 000h, 018h, 018h, 000h, 000h, 000h, 018h, 018h, 000h, 000h, 000h, 000h, 000h

534 00014661 18180000000000 <1>

535 00014668 000000001818000000- <1> db 000h, 000h, 000h, 000h, 018h, 018h, 000h, 000h, 000h, 018h, 018h, 030h, 000h, 000h, 000h, 000h

535 00014671 18183000000000 <1>

536 00014678 000000060C18306030- <1> db 000h, 000h, 000h, 006h, 00ch, 018h, 030h, 060h, 030h, 018h, 00ch, 006h, 000h, 000h, 000h, 000h

536 00014681 180C0600000000 <1>

537 00014688 00000000007E00007E- <1> db 000h, 000h, 000h, 000h, 000h, 07eh, 000h, 000h, 07eh, 000h, 000h, 000h, 000h, 000h, 000h, 000h

537 00014691 00000000000000 <1>

538 00014698 0000006030180C060C- <1> db 000h, 000h, 000h, 060h, 030h, 018h, 00ch, 006h, 00ch, 018h, 030h, 060h, 000h, 000h, 000h, 000h

538 000146A1 18306000000000 <1>

539 000146A8 00007CC6C60C181818- <1> db 000h, 000h, 07ch, 0c6h, 0c6h, 00ch, 018h, 018h, 018h, 000h, 018h, 018h, 000h, 000h, 000h, 000h

539 000146B1 00181800000000 <1>

540 000146B8 0000007CC6C6DEDEDE- <1> db 000h, 000h, 000h, 07ch, 0c6h, 0c6h, 0deh, 0deh, 0deh, 0dch, 0c0h, 07ch, 000h, 000h, 000h, 000h

540 000146C1 DCC07C00000000 <1>

541 000146C8 000010386CC6C6FEC6- <1> db 000h, 000h, 010h, 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h

541 000146D1 C6C6C600000000 <1>

542 000146D8 0000FC6666667C6666- <1> db 000h, 000h, 0fch, 066h, 066h, 066h, 07ch, 066h, 066h, 066h, 066h, 0fch, 000h, 000h, 000h, 000h

542 000146E1 6666FC00000000 <1>

543 000146E8 00003C66C2C0C0C0C0- <1> db 000h, 000h, 03ch, 066h, 0c2h, 0c0h, 0c0h, 0c0h, 0c0h, 0c2h, 066h, 03ch, 000h, 000h, 000h, 000h

543 000146F1 C2663C00000000 <1>

544 000146F8 0000F86C6666666666- <1> db 000h, 000h, 0f8h, 06ch, 066h, 066h, 066h, 066h, 066h, 066h, 06ch, 0f8h, 000h, 000h, 000h, 000h

544 00014701 666CF800000000 <1>

545 00014708 0000FE666268786860- <1> db 000h, 000h, 0feh, 066h, 062h, 068h, 078h, 068h, 060h, 062h, 066h, 0feh, 000h, 000h, 000h, 000h

545 00014711 6266FE00000000 <1>

546 00014718 0000FE666268786860- <1> db 000h, 000h, 0feh, 066h, 062h, 068h, 078h, 068h, 060h, 060h, 060h, 0f0h, 000h, 000h, 000h, 000h

546 00014721 6060F000000000 <1>

547 00014728 00003C66C2C0C0DEC6- <1> db 000h, 000h, 03ch, 066h, 0c2h, 0c0h, 0c0h, 0deh, 0c6h, 0c6h, 066h, 03ah, 000h, 000h, 000h, 000h

547 00014731 C6663A00000000 <1>

548 00014738 0000C6C6C6C6FEC6C6- <1> db 000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h

548 00014741 C6C6C600000000 <1>

549 00014748 00003C181818181818- <1> db 000h, 000h, 03ch, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h

549 00014751 18183C00000000 <1>

550 00014758 00001E0C0C0C0C0CCC- <1> db 000h, 000h, 01eh, 00ch, 00ch, 00ch, 00ch, 00ch, 0cch, 0cch, 0cch, 078h, 000h, 000h, 000h, 000h

550 00014761 CCCC7800000000 <1>

551 00014768 0000E666666C78786C- <1> db 000h, 000h, 0e6h, 066h, 066h, 06ch, 078h, 078h, 06ch, 066h, 066h, 0e6h, 000h, 000h, 000h, 000h

551 00014771 6666E600000000 <1>

552 00014778 0000F0606060606060- <1> db 000h, 000h, 0f0h, 060h, 060h, 060h, 060h, 060h, 060h, 062h, 066h, 0feh, 000h, 000h, 000h, 000h

552 00014781 6266FE00000000 <1>

553 00014788 0000C3E7FFFFDBC3C3- <1> db 000h, 000h, 0c3h, 0e7h, 0ffh, 0ffh, 0dbh, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 000h, 000h, 000h, 000h

553 00014791 C3C3C300000000 <1>

554 00014798 0000C6E6F6FEDECEC6- <1> db 000h, 000h, 0c6h, 0e6h, 0f6h, 0feh, 0deh, 0ceh, 0c6h, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h

554 000147A1 C6C6C600000000 <1>

555 000147A8 00007CC6C6C6C6C6C6- <1> db 000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h

555 000147B1 C6C67C00000000 <1>

556 000147B8 0000FC6666667C6060- <1> db 000h, 000h, 0fch, 066h, 066h, 066h, 07ch, 060h, 060h, 060h, 060h, 0f0h, 000h, 000h, 000h, 000h

556 000147C1 6060F000000000 <1>

557 000147C8 00007CC6C6C6C6C6C6- <1> db 000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0d6h, 0deh, 07ch, 00ch, 00eh, 000h, 000h

557 000147D1 D6DE7C0C0E0000 <1>

558 000147D8 0000FC6666667C6C66- <1> db 000h, 000h, 0fch, 066h, 066h, 066h, 07ch, 06ch, 066h, 066h, 066h, 0e6h, 000h, 000h, 000h, 000h

558 000147E1 6666E600000000 <1>

559 000147E8 00007CC6C660380C06- <1> db 000h, 000h, 07ch, 0c6h, 0c6h, 060h, 038h, 00ch, 006h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h

559 000147F1 C6C67C00000000 <1>

560 000147F8 0000FFDB9918181818- <1> db 000h, 000h, 0ffh, 0dbh, 099h, 018h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h

560 00014801 18183C00000000 <1>

561 00014808 0000C6C6C6C6C6C6C6- <1> db 000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h

561 00014811 C6C67C00000000 <1>

562 00014818 0000C3C3C3C3C3C3C3- <1> db 000h, 000h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 066h, 03ch, 018h, 000h, 000h, 000h, 000h

562 00014821 663C1800000000 <1>

563 00014828 0000C3C3C3C3C3DBDB- <1> db 000h, 000h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 0dbh, 0dbh, 0ffh, 066h, 066h, 000h, 000h, 000h, 000h

563 00014831 FF666600000000 <1>

564 00014838 0000C3C3663C18183C- <1> db 000h, 000h, 0c3h, 0c3h, 066h, 03ch, 018h, 018h, 03ch, 066h, 0c3h, 0c3h, 000h, 000h, 000h, 000h

564 00014841 66C3C300000000 <1>

565 00014848 0000C3C3C3663C1818- <1> db 000h, 000h, 0c3h, 0c3h, 0c3h, 066h, 03ch, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h

565 00014851 18183C00000000 <1>

566 00014858 0000FFC3860C183060- <1> db 000h, 000h, 0ffh, 0c3h, 086h, 00ch, 018h, 030h, 060h, 0c1h, 0c3h, 0ffh, 000h, 000h, 000h, 000h

566 00014861 C1C3FF00000000 <1>

567 00014868 00003C303030303030- <1> db 000h, 000h, 03ch, 030h, 030h, 030h, 030h, 030h, 030h, 030h, 030h, 03ch, 000h, 000h, 000h, 000h

567 00014871 30303C00000000 <1>

568 00014878 00000080C0E070381C- <1> db 000h, 000h, 000h, 080h, 0c0h, 0e0h, 070h, 038h, 01ch, 00eh, 006h, 002h, 000h, 000h, 000h, 000h

568 00014881 0E060200000000 <1>

569 00014888 00003C0C0C0C0C0C0C- <1> db 000h, 000h, 03ch, 00ch, 00ch, 00ch, 00ch, 00ch, 00ch, 00ch, 00ch, 03ch, 000h, 000h, 000h, 000h

569 00014891 0C0C3C00000000 <1>

570 00014898 10386CC60000000000- <1> db 010h, 038h, 06ch, 0c6h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

570 000148A1 00000000000000 <1>

571 000148A8 000000000000000000- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh, 000h, 000h

571 000148B1 00000000FF0000 <1>

572 000148B8 303018000000000000- <1> db 030h, 030h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

572 000148C1 00000000000000 <1>

573 000148C8 0000000000780C7CCC- <1> db 000h, 000h, 000h, 000h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h

573 000148D1 CCCC7600000000 <1>

574 000148D8 0000E06060786C6666- <1> db 000h, 000h, 0e0h, 060h, 060h, 078h, 06ch, 066h, 066h, 066h, 066h, 07ch, 000h, 000h, 000h, 000h

574 000148E1 66667C00000000 <1>

575 000148E8 00000000007CC6C0C0- <1> db 000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 0c0h, 0c0h, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h

575 000148F1 C0C67C00000000 <1>

576 000148F8 00001C0C0C3C6CCCCC- <1> db 000h, 000h, 01ch, 00ch, 00ch, 03ch, 06ch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h

576 00014901 CCCC7600000000 <1>

577 00014908 00000000007CC6FEC0- <1> db 000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h

577 00014911 C0C67C00000000 <1>

578 00014918 0000386C6460F06060- <1> db 000h, 000h, 038h, 06ch, 064h, 060h, 0f0h, 060h, 060h, 060h, 060h, 0f0h, 000h, 000h, 000h, 000h

578 00014921 6060F000000000 <1>

579 00014928 000000000076CCCCCC- <1> db 000h, 000h, 000h, 000h, 000h, 076h, 0cch, 0cch, 0cch, 0cch, 0cch, 07ch, 00ch, 0cch, 078h, 000h

579 00014931 CCCC7C0CCC7800 <1>

580 00014938 0000E060606C766666- <1> db 000h, 000h, 0e0h, 060h, 060h, 06ch, 076h, 066h, 066h, 066h, 066h, 0e6h, 000h, 000h, 000h, 000h

580 00014941 6666E600000000 <1>

581 00014948 000018180038181818- <1> db 000h, 000h, 018h, 018h, 000h, 038h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h

581 00014951 18183C00000000 <1>

582 00014958 00000606000E060606- <1> db 000h, 000h, 006h, 006h, 000h, 00eh, 006h, 006h, 006h, 006h, 006h, 006h, 066h, 066h, 03ch, 000h

582 00014961 06060666663C00 <1>

583 00014968 0000E06060666C7878- <1> db 000h, 000h, 0e0h, 060h, 060h, 066h, 06ch, 078h, 078h, 06ch, 066h, 0e6h, 000h, 000h, 000h, 000h

583 00014971 6C66E600000000 <1>

584 00014978 000038181818181818- <1> db 000h, 000h, 038h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h

584 00014981 18183C00000000 <1>

585 00014988 0000000000E6FFDBDB- <1> db 000h, 000h, 000h, 000h, 000h, 0e6h, 0ffh, 0dbh, 0dbh, 0dbh, 0dbh, 0dbh, 000h, 000h, 000h, 000h

585 00014991 DBDBDB00000000 <1>

586 00014998 0000000000DC666666- <1> db 000h, 000h, 000h, 000h, 000h, 0dch, 066h, 066h, 066h, 066h, 066h, 066h, 000h, 000h, 000h, 000h

586 000149A1 66666600000000 <1>

587 000149A8 00000000007CC6C6C6- <1> db 000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h

587 000149B1 C6C67C00000000 <1>

588 000149B8 0000000000DC666666- <1> db 000h, 000h, 000h, 000h, 000h, 0dch, 066h, 066h, 066h, 066h, 066h, 07ch, 060h, 060h, 0f0h, 000h

588 000149C1 66667C6060F000 <1>

589 000149C8 000000000076CCCCCC- <1> db 000h, 000h, 000h, 000h, 000h, 076h, 0cch, 0cch, 0cch, 0cch, 0cch, 07ch, 00ch, 00ch, 01eh, 000h

589 000149D1 CCCC7C0C0C1E00 <1>

590 000149D8 0000000000DC766660- <1> db 000h, 000h, 000h, 000h, 000h, 0dch, 076h, 066h, 060h, 060h, 060h, 0f0h, 000h, 000h, 000h, 000h

590 000149E1 6060F000000000 <1>

591 000149E8 00000000007CC66038- <1> db 000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 060h, 038h, 00ch, 0c6h, 07ch, 000h, 000h, 000h, 000h

591 000149F1 0CC67C00000000 <1>

592 000149F8 0000103030FC303030- <1> db 000h, 000h, 010h, 030h, 030h, 0fch, 030h, 030h, 030h, 030h, 036h, 01ch, 000h, 000h, 000h, 000h

592 00014A01 30361C00000000 <1>

593 00014A08 0000000000CCCCCCCC- <1> db 000h, 000h, 000h, 000h, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h

593 00014A11 CCCC7600000000 <1>

594 00014A18 0000000000C3C3C3C3- <1> db 000h, 000h, 000h, 000h, 000h, 0c3h, 0c3h, 0c3h, 0c3h, 066h, 03ch, 018h, 000h, 000h, 000h, 000h

594 00014A21 663C1800000000 <1>

595 00014A28 0000000000C3C3C3DB- <1> db 000h, 000h, 000h, 000h, 000h, 0c3h, 0c3h, 0c3h, 0dbh, 0dbh, 0ffh, 066h, 000h, 000h, 000h, 000h

595 00014A31 DBFF6600000000 <1>

596 00014A38 0000000000C3663C18- <1> db 000h, 000h, 000h, 000h, 000h, 0c3h, 066h, 03ch, 018h, 03ch, 066h, 0c3h, 000h, 000h, 000h, 000h

596 00014A41 3C66C300000000 <1>

597 00014A48 0000000000C6C6C6C6- <1> db 000h, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07eh, 006h, 00ch, 0f8h, 000h

597 00014A51 C6C67E060CF800 <1>

598 00014A58 0000000000FECC1830- <1> db 000h, 000h, 000h, 000h, 000h, 0feh, 0cch, 018h, 030h, 060h, 0c6h, 0feh, 000h, 000h, 000h, 000h

598 00014A61 60C6FE00000000 <1>

599 00014A68 00000E181818701818- <1> db 000h, 000h, 00eh, 018h, 018h, 018h, 070h, 018h, 018h, 018h, 018h, 00eh, 000h, 000h, 000h, 000h

599 00014A71 18180E00000000 <1>

600 00014A78 000018181818001818- <1> db 000h, 000h, 018h, 018h, 018h, 018h, 000h, 018h, 018h, 018h, 018h, 018h, 000h, 000h, 000h, 000h

600 00014A81 18181800000000 <1>

601 00014A88 0000701818180E1818- <1> db 000h, 000h, 070h, 018h, 018h, 018h, 00eh, 018h, 018h, 018h, 018h, 070h, 000h, 000h, 000h, 000h

601 00014A91 18187000000000 <1>

602 00014A98 000076DC0000000000- <1> db 000h, 000h, 076h, 0dch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

602 00014AA1 00000000000000 <1>

603 00014AA8 0000000010386CC6C6- <1> db 000h, 000h, 000h, 000h, 010h, 038h, 06ch, 0c6h, 0c6h, 0c6h, 0feh, 000h, 000h, 000h, 000h, 000h

603 00014AB1 C6FE0000000000 <1>

604 00014AB8 00003C66C2C0C0C0C2- <1> db 000h, 000h, 03ch, 066h, 0c2h, 0c0h, 0c0h, 0c0h, 0c2h, 066h, 03ch, 00ch, 006h, 07ch, 000h, 000h

604 00014AC1 663C0C067C0000 <1>

605 00014AC8 0000CC0000CCCCCCCC- <1> db 000h, 000h, 0cch, 000h, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h

605 00014AD1 CCCC7600000000 <1>

606 00014AD8 000C1830007CC6FEC0- <1> db 000h, 00ch, 018h, 030h, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h

606 00014AE1 C0C67C00000000 <1>

607 00014AE8 0010386C00780C7CCC- <1> db 000h, 010h, 038h, 06ch, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h

607 00014AF1 CCCC7600000000 <1>

608 00014AF8 0000CC0000780C7CCC- <1> db 000h, 000h, 0cch, 000h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h

608 00014B01 CCCC7600000000 <1>

609 00014B08 0060301800780C7CCC- <1> db 000h, 060h, 030h, 018h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h

609 00014B11 CCCC7600000000 <1>

610 00014B18 00386C3800780C7CCC- <1> db 000h, 038h, 06ch, 038h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h

610 00014B21 CCCC7600000000 <1>

611 00014B28 000000003C66606066- <1> db 000h, 000h, 000h, 000h, 03ch, 066h, 060h, 060h, 066h, 03ch, 00ch, 006h, 03ch, 000h, 000h, 000h

611 00014B31 3C0C063C000000 <1>

612 00014B38 0010386C007CC6FEC0- <1> db 000h, 010h, 038h, 06ch, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h

612 00014B41 C0C67C00000000 <1>

613 00014B48 0000C600007CC6FEC0- <1> db 000h, 000h, 0c6h, 000h, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h

613 00014B51 C0C67C00000000 <1>

614 00014B58 00603018007CC6FEC0- <1> db 000h, 060h, 030h, 018h, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h

614 00014B61 C0C67C00000000 <1>

615 00014B68 000066000038181818- <1> db 000h, 000h, 066h, 000h, 000h, 038h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h

615 00014B71 18183C00000000 <1>

616 00014B78 00183C660038181818- <1> db 000h, 018h, 03ch, 066h, 000h, 038h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h

616 00014B81 18183C00000000 <1>

617 00014B88 006030180038181818- <1> db 000h, 060h, 030h, 018h, 000h, 038h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h

617 00014B91 18183C00000000 <1>

618 00014B98 00C60010386CC6C6FE- <1> db 000h, 0c6h, 000h, 010h, 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h

618 00014BA1 C6C6C600000000 <1>

619 00014BA8 386C3800386CC6C6FE- <1> db 038h, 06ch, 038h, 000h, 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h

619 00014BB1 C6C6C600000000 <1>

620 00014BB8 18306000FE66607C60- <1> db 018h, 030h, 060h, 000h, 0feh, 066h, 060h, 07ch, 060h, 060h, 066h, 0feh, 000h, 000h, 000h, 000h

620 00014BC1 6066FE00000000 <1>

621 00014BC8 00000000006E3B1B7E- <1> db 000h, 000h, 000h, 000h, 000h, 06eh, 03bh, 01bh, 07eh, 0d8h, 0dch, 077h, 000h, 000h, 000h, 000h

621 00014BD1 D8DC7700000000 <1>

622 00014BD8 00003E6CCCCCFECCCC- <1> db 000h, 000h, 03eh, 06ch, 0cch, 0cch, 0feh, 0cch, 0cch, 0cch, 0cch, 0ceh, 000h, 000h, 000h, 000h

622 00014BE1 CCCCCE00000000 <1>

623 00014BE8 0010386C007CC6C6C6- <1> db 000h, 010h, 038h, 06ch, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h

623 00014BF1 C6C67C00000000 <1>

624 00014BF8 0000C600007CC6C6C6- <1> db 000h, 000h, 0c6h, 000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h

624 00014C01 C6C67C00000000 <1>

625 00014C08 00603018007CC6C6C6- <1> db 000h, 060h, 030h, 018h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h

625 00014C11 C6C67C00000000 <1>

626 00014C18 003078CC00CCCCCCCC- <1> db 000h, 030h, 078h, 0cch, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h

626 00014C21 CCCC7600000000 <1>

627 00014C28 0060301800CCCCCCCC- <1> db 000h, 060h, 030h, 018h, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h

627 00014C31 CCCC7600000000 <1>

628 00014C38 0000C60000C6C6C6C6- <1> db 000h, 000h, 0c6h, 000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07eh, 006h, 00ch, 078h, 000h

628 00014C41 C6C67E060C7800 <1>

629 00014C48 00C6007CC6C6C6C6C6- <1> db 000h, 0c6h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h

629 00014C51 C6C67C00000000 <1>

630 00014C58 00C600C6C6C6C6C6C6- <1> db 000h, 0c6h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h

630 00014C61 C6C67C00000000 <1>

631 00014C68 0018187EC3C0C0C0C3- <1> db 000h, 018h, 018h, 07eh, 0c3h, 0c0h, 0c0h, 0c0h, 0c3h, 07eh, 018h, 018h, 000h, 000h, 000h, 000h

631 00014C71 7E181800000000 <1>

632 00014C78 00386C6460F0606060- <1> db 000h, 038h, 06ch, 064h, 060h, 0f0h, 060h, 060h, 060h, 060h, 0e6h, 0fch, 000h, 000h, 000h, 000h

632 00014C81 60E6FC00000000 <1>

633 00014C88 0000C3663C18FF18FF- <1> db 000h, 000h, 0c3h, 066h, 03ch, 018h, 0ffh, 018h, 0ffh, 018h, 018h, 018h, 000h, 000h, 000h, 000h

633 00014C91 18181800000000 <1>

634 00014C98 00FC66667C62666F66- <1> db 000h, 0fch, 066h, 066h, 07ch, 062h, 066h, 06fh, 066h, 066h, 066h, 0f3h, 000h, 000h, 000h, 000h

634 00014CA1 6666F300000000 <1>

635 00014CA8 000E1B1818187E1818- <1> db 000h, 00eh, 01bh, 018h, 018h, 018h, 07eh, 018h, 018h, 018h, 018h, 018h, 0d8h, 070h, 000h, 000h

635 00014CB1 181818D8700000 <1>

636 00014CB8 0018306000780C7CCC- <1> db 000h, 018h, 030h, 060h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h

636 00014CC1 CCCC7600000000 <1>

637 00014CC8 000C18300038181818- <1> db 000h, 00ch, 018h, 030h, 000h, 038h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h

637 00014CD1 18183C00000000 <1>

638 00014CD8 00183060007CC6C6C6- <1> db 000h, 018h, 030h, 060h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h

638 00014CE1 C6C67C00000000 <1>

639 00014CE8 0018306000CCCCCCCC- <1> db 000h, 018h, 030h, 060h, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h

639 00014CF1 CCCC7600000000 <1>

640 00014CF8 000076DC00DC666666- <1> db 000h, 000h, 076h, 0dch, 000h, 0dch, 066h, 066h, 066h, 066h, 066h, 066h, 000h, 000h, 000h, 000h

640 00014D01 66666600000000 <1>

641 00014D08 76DC00C6E6F6FEDECE- <1> db 076h, 0dch, 000h, 0c6h, 0e6h, 0f6h, 0feh, 0deh, 0ceh, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h

641 00014D11 C6C6C600000000 <1>

642 00014D18 003C6C6C3E007E0000- <1> db 000h, 03ch, 06ch, 06ch, 03eh, 000h, 07eh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

642 00014D21 00000000000000 <1>

643 00014D28 00386C6C38007C0000- <1> db 000h, 038h, 06ch, 06ch, 038h, 000h, 07ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

643 00014D31 00000000000000 <1>

644 00014D38 0000303000303060C0- <1> db 000h, 000h, 030h, 030h, 000h, 030h, 030h, 060h, 0c0h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h

644 00014D41 C6C67C00000000 <1>

645 00014D48 000000000000FEC0C0- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 0feh, 0c0h, 0c0h, 0c0h, 0c0h, 000h, 000h, 000h, 000h, 000h

645 00014D51 C0C00000000000 <1>

646 00014D58 000000000000FE0606- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 0feh, 006h, 006h, 006h, 006h, 000h, 000h, 000h, 000h, 000h

646 00014D61 06060000000000 <1>

647 00014D68 00C0C0C2C6CC183060- <1> db 000h, 0c0h, 0c0h, 0c2h, 0c6h, 0cch, 018h, 030h, 060h, 0ceh, 09bh, 006h, 00ch, 01fh, 000h, 000h

647 00014D71 CE9B060C1F0000 <1>

648 00014D78 00C0C0C2C6CC183066- <1> db 000h, 0c0h, 0c0h, 0c2h, 0c6h, 0cch, 018h, 030h, 066h, 0ceh, 096h, 03eh, 006h, 006h, 000h, 000h

648 00014D81 CE963E06060000 <1>

649 00014D88 00001818001818183C- <1> db 000h, 000h, 018h, 018h, 000h, 018h, 018h, 018h, 03ch, 03ch, 03ch, 018h, 000h, 000h, 000h, 000h

649 00014D91 3C3C1800000000 <1>

650 00014D98 0000000000366CD86C- <1> db 000h, 000h, 000h, 000h, 000h, 036h, 06ch, 0d8h, 06ch, 036h, 000h, 000h, 000h, 000h, 000h, 000h

650 00014DA1 36000000000000 <1>

651 00014DA8 0000000000D86C366C- <1> db 000h, 000h, 000h, 000h, 000h, 0d8h, 06ch, 036h, 06ch, 0d8h, 000h, 000h, 000h, 000h, 000h, 000h

651 00014DB1 D8000000000000 <1>

652 00014DB8 114411441144114411- <1> db 011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h

652 00014DC1 44114411441144 <1>

653 00014DC8 55AA55AA55AA55AA55- <1> db 055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah

653 00014DD1 AA55AA55AA55AA <1>

654 00014DD8 DD77DD77DD77DD77DD- <1> db 0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 0ddh, 077h

654 00014DE1 77DD77DD77DD77 <1>

655 00014DE8 181818181818181818- <1> db 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h

655 00014DF1 18181818181818 <1>

656 00014DF8 18181818181818F818- <1> db 018h, 018h, 018h, 018h, 018h, 018h, 018h, 0f8h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h

656 00014E01 18181818181818 <1>

657 00014E08 1818181818F818F818- <1> db 018h, 018h, 018h, 018h, 018h, 0f8h, 018h, 0f8h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h

657 00014E11 18181818181818 <1>

658 00014E18 36363636363636F636- <1> db 036h, 036h, 036h, 036h, 036h, 036h, 036h, 0f6h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h

658 00014E21 36363636363636 <1>

659 00014E28 00000000000000FE36- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0feh, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h

659 00014E31 36363636363636 <1>

660 00014E38 0000000000F818F818- <1> db 000h, 000h, 000h, 000h, 000h, 0f8h, 018h, 0f8h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h

660 00014E41 18181818181818 <1>

661 00014E48 3636363636F606F636- <1> db 036h, 036h, 036h, 036h, 036h, 0f6h, 006h, 0f6h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h

661 00014E51 36363636363636 <1>

662 00014E58 363636363636363636- <1> db 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h

662 00014E61 36363636363636 <1>

663 00014E68 0000000000FE06F636- <1> db 000h, 000h, 000h, 000h, 000h, 0feh, 006h, 0f6h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h

663 00014E71 36363636363636 <1>

664 00014E78 3636363636F606FE00- <1> db 036h, 036h, 036h, 036h, 036h, 0f6h, 006h, 0feh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

664 00014E81 00000000000000 <1>

665 00014E88 36363636363636FE00- <1> db 036h, 036h, 036h, 036h, 036h, 036h, 036h, 0feh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

665 00014E91 00000000000000 <1>

666 00014E98 1818181818F818F800- <1> db 018h, 018h, 018h, 018h, 018h, 0f8h, 018h, 0f8h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

666 00014EA1 00000000000000 <1>

667 00014EA8 00000000000000F818- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0f8h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h

667 00014EB1 18181818181818 <1>

668 00014EB8 181818181818181F00- <1> db 018h, 018h, 018h, 018h, 018h, 018h, 018h, 01fh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

668 00014EC1 00000000000000 <1>

669 00014EC8 18181818181818FF00- <1> db 018h, 018h, 018h, 018h, 018h, 018h, 018h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

669 00014ED1 00000000000000 <1>

670 00014ED8 00000000000000FF18- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h

670 00014EE1 18181818181818 <1>

671 00014EE8 181818181818181F18- <1> db 018h, 018h, 018h, 018h, 018h, 018h, 018h, 01fh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h

671 00014EF1 18181818181818 <1>

672 00014EF8 00000000000000FF00- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

672 00014F01 00000000000000 <1>

673 00014F08 18181818181818FF18- <1> db 018h, 018h, 018h, 018h, 018h, 018h, 018h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h

673 00014F11 18181818181818 <1>

674 00014F18 18181818181F181F18- <1> db 018h, 018h, 018h, 018h, 018h, 01fh, 018h, 01fh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h

674 00014F21 18181818181818 <1>

675 00014F28 363636363636363736- <1> db 036h, 036h, 036h, 036h, 036h, 036h, 036h, 037h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h

675 00014F31 36363636363636 <1>

676 00014F38 363636363637303F00- <1> db 036h, 036h, 036h, 036h, 036h, 037h, 030h, 03fh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

676 00014F41 00000000000000 <1>

677 00014F48 00000000003F303736- <1> db 000h, 000h, 000h, 000h, 000h, 03fh, 030h, 037h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h

677 00014F51 36363636363636 <1>

678 00014F58 3636363636F700FF00- <1> db 036h, 036h, 036h, 036h, 036h, 0f7h, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

678 00014F61 00000000000000 <1>

679 00014F68 0000000000FF00F736- <1> db 000h, 000h, 000h, 000h, 000h, 0ffh, 000h, 0f7h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h

679 00014F71 36363636363636 <1>

680 00014F78 363636363637303736- <1> db 036h, 036h, 036h, 036h, 036h, 037h, 030h, 037h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h

680 00014F81 36363636363636 <1>

681 00014F88 0000000000FF00FF00- <1> db 000h, 000h, 000h, 000h, 000h, 0ffh, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

681 00014F91 00000000000000 <1>

682 00014F98 3636363636F700F736- <1> db 036h, 036h, 036h, 036h, 036h, 0f7h, 000h, 0f7h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h

682 00014FA1 36363636363636 <1>

683 00014FA8 1818181818FF00FF00- <1> db 018h, 018h, 018h, 018h, 018h, 0ffh, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

683 00014FB1 00000000000000 <1>

684 00014FB8 36363636363636FF00- <1> db 036h, 036h, 036h, 036h, 036h, 036h, 036h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

684 00014FC1 00000000000000 <1>

685 00014FC8 0000000000FF00FF18- <1> db 000h, 000h, 000h, 000h, 000h, 0ffh, 000h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h

685 00014FD1 18181818181818 <1>

686 00014FD8 00000000000000FF36- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h

686 00014FE1 36363636363636 <1>

687 00014FE8 363636363636363F00- <1> db 036h, 036h, 036h, 036h, 036h, 036h, 036h, 03fh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

687 00014FF1 00000000000000 <1>

688 00014FF8 18181818181F181F00- <1> db 018h, 018h, 018h, 018h, 018h, 01fh, 018h, 01fh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

688 00015001 00000000000000 <1>

689 00015008 00000000001F181F18- <1> db 000h, 000h, 000h, 000h, 000h, 01fh, 018h, 01fh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h

689 00015011 18181818181818 <1>

690 00015018 000000000000003F36- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 03fh, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h

690 00015021 36363636363636 <1>

691 00015028 36363636363636FF36- <1> db 036h, 036h, 036h, 036h, 036h, 036h, 036h, 0ffh, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h

691 00015031 36363636363636 <1>

692 00015038 1818181818FF18FF18- <1> db 018h, 018h, 018h, 018h, 018h, 0ffh, 018h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h

692 00015041 18181818181818 <1>

693 00015048 18181818181818F800- <1> db 018h, 018h, 018h, 018h, 018h, 018h, 018h, 0f8h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

693 00015051 00000000000000 <1>

694 00015058 000000000000001F18- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 01fh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h

694 00015061 18181818181818 <1>

695 00015068 FFFFFFFFFFFFFFFFFF- <1> db 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh

695 00015071 FFFFFFFFFFFFFF <1>

696 00015078 00000000000000FFFF- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh

696 00015081 FFFFFFFFFFFFFF <1>

697 00015088 F0F0F0F0F0F0F0F0F0- <1> db 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h

697 00015091 F0F0F0F0F0F0F0 <1>

698 00015098 0F0F0F0F0F0F0F0F0F- <1> db 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh

698 000150A1 0F0F0F0F0F0F0F <1>

699 000150A8 FFFFFFFFFFFFFF0000- <1> db 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

699 000150B1 00000000000000 <1>

700 000150B8 000000000076DCD8D8- <1> db 000h, 000h, 000h, 000h, 000h, 076h, 0dch, 0d8h, 0d8h, 0d8h, 0dch, 076h, 000h, 000h, 000h, 000h

700 000150C1 D8DC7600000000 <1>

701 000150C8 000078CCCCCCD8CCC6- <1> db 000h, 000h, 078h, 0cch, 0cch, 0cch, 0d8h, 0cch, 0c6h, 0c6h, 0c6h, 0cch, 000h, 000h, 000h, 000h

701 000150D1 C6C6CC00000000 <1>

702 000150D8 0000FEC6C6C0C0C0C0- <1> db 000h, 000h, 0feh, 0c6h, 0c6h, 0c0h, 0c0h, 0c0h, 0c0h, 0c0h, 0c0h, 0c0h, 000h, 000h, 000h, 000h

702 000150E1 C0C0C000000000 <1>

703 000150E8 00000000FE6C6C6C6C- <1> db 000h, 000h, 000h, 000h, 0feh, 06ch, 06ch, 06ch, 06ch, 06ch, 06ch, 06ch, 000h, 000h, 000h, 000h

703 000150F1 6C6C6C00000000 <1>

704 000150F8 000000FEC660301830- <1> db 000h, 000h, 000h, 0feh, 0c6h, 060h, 030h, 018h, 030h, 060h, 0c6h, 0feh, 000h, 000h, 000h, 000h

704 00015101 60C6FE00000000 <1>

705 00015108 00000000007ED8D8D8- <1> db 000h, 000h, 000h, 000h, 000h, 07eh, 0d8h, 0d8h, 0d8h, 0d8h, 0d8h, 070h, 000h, 000h, 000h, 000h

705 00015111 D8D87000000000 <1>

706 00015118 000000006666666666- <1> db 000h, 000h, 000h, 000h, 066h, 066h, 066h, 066h, 066h, 07ch, 060h, 060h, 0c0h, 000h, 000h, 000h

706 00015121 7C6060C0000000 <1>

707 00015128 0000000076DC181818- <1> db 000h, 000h, 000h, 000h, 076h, 0dch, 018h, 018h, 018h, 018h, 018h, 018h, 000h, 000h, 000h, 000h

707 00015131 18181800000000 <1>

708 00015138 0000007E183C666666- <1> db 000h, 000h, 000h, 07eh, 018h, 03ch, 066h, 066h, 066h, 03ch, 018h, 07eh, 000h, 000h, 000h, 000h

708 00015141 3C187E00000000 <1>

709 00015148 000000386CC6C6FEC6- <1> db 000h, 000h, 000h, 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 06ch, 038h, 000h, 000h, 000h, 000h

709 00015151 C66C3800000000 <1>

710 00015158 0000386CC6C6C66C6C- <1> db 000h, 000h, 038h, 06ch, 0c6h, 0c6h, 0c6h, 06ch, 06ch, 06ch, 06ch, 0eeh, 000h, 000h, 000h, 000h

710 00015161 6C6CEE00000000 <1>

711 00015168 00001E30180C3E6666- <1> db 000h, 000h, 01eh, 030h, 018h, 00ch, 03eh, 066h, 066h, 066h, 066h, 03ch, 000h, 000h, 000h, 000h

711 00015171 66663C00000000 <1>

712 00015178 00000000007EDBDBDB- <1> db 000h, 000h, 000h, 000h, 000h, 07eh, 0dbh, 0dbh, 0dbh, 07eh, 000h, 000h, 000h, 000h, 000h, 000h

712 00015181 7E000000000000 <1>

713 00015188 00000003067EDBDBF3- <1> db 000h, 000h, 000h, 003h, 006h, 07eh, 0dbh, 0dbh, 0f3h, 07eh, 060h, 0c0h, 000h, 000h, 000h, 000h

713 00015191 7E60C000000000 <1>

714 00015198 00001C3060607C6060- <1> db 000h, 000h, 01ch, 030h, 060h, 060h, 07ch, 060h, 060h, 060h, 030h, 01ch, 000h, 000h, 000h, 000h

714 000151A1 60301C00000000 <1>

715 000151A8 0000007CC6C6C6C6C6- <1> db 000h, 000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h

715 000151B1 C6C6C600000000 <1>

716 000151B8 00000000FE0000FE00- <1> db 000h, 000h, 000h, 000h, 0feh, 000h, 000h, 0feh, 000h, 000h, 0feh, 000h, 000h, 000h, 000h, 000h

716 000151C1 00FE0000000000 <1>

717 000151C8 0000000018187E1818- <1> db 000h, 000h, 000h, 000h, 018h, 018h, 07eh, 018h, 018h, 000h, 000h, 0ffh, 000h, 000h, 000h, 000h

717 000151D1 0000FF00000000 <1>

718 000151D8 00000030180C060C18- <1> db 000h, 000h, 000h, 030h, 018h, 00ch, 006h, 00ch, 018h, 030h, 000h, 07eh, 000h, 000h, 000h, 000h

718 000151E1 30007E00000000 <1>

719 000151E8 0000000C1830603018- <1> db 000h, 000h, 000h, 00ch, 018h, 030h, 060h, 030h, 018h, 00ch, 000h, 07eh, 000h, 000h, 000h, 000h

719 000151F1 0C007E00000000 <1>

720 000151F8 00000E1B1B18181818- <1> db 000h, 000h, 00eh, 01bh, 01bh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h

720 00015201 18181818181818 <1>

721 00015208 1818181818181818D8- <1> db 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 0d8h, 0d8h, 0d8h, 070h, 000h, 000h, 000h, 000h

721 00015211 D8D87000000000 <1>

722 00015218 000000001818007E00- <1> db 000h, 000h, 000h, 000h, 018h, 018h, 000h, 07eh, 000h, 018h, 018h, 000h, 000h, 000h, 000h, 000h

722 00015221 18180000000000 <1>

723 00015228 000000000076DC0076- <1> db 000h, 000h, 000h, 000h, 000h, 076h, 0dch, 000h, 076h, 0dch, 000h, 000h, 000h, 000h, 000h, 000h

723 00015231 DC000000000000 <1>

724 00015238 00386C6C3800000000- <1> db 000h, 038h, 06ch, 06ch, 038h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

724 00015241 00000000000000 <1>

725 00015248 000000000000001818- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

725 00015251 00000000000000 <1>

726 00015258 000000000000000018- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

726 00015261 00000000000000 <1>

727 00015268 000F0C0C0C0C0CEC6C- <1> db 000h, 00fh, 00ch, 00ch, 00ch, 00ch, 00ch, 0ech, 06ch, 06ch, 03ch, 01ch, 000h, 000h, 000h, 000h

727 00015271 6C3C1C00000000 <1>

728 00015278 00D86C6C6C6C6C0000- <1> db 000h, 0d8h, 06ch, 06ch, 06ch, 06ch, 06ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

728 00015281 00000000000000 <1>

729 00015288 0070D83060C8F80000- <1> db 000h, 070h, 0d8h, 030h, 060h, 0c8h, 0f8h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

729 00015291 00000000000000 <1>

730 00015298 000000007C7C7C7C7C- <1> db 000h, 000h, 000h, 000h, 07ch, 07ch, 07ch, 07ch, 07ch, 07ch, 07ch, 000h, 000h, 000h, 000h, 000h

730 000152A1 7C7C0000000000 <1>

731 000152A8 000000000000000000- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h

731 000152B1 00000000000000 <1>

732 <1> vgafont14alt:

733 000152B8 1D000000002466FF66- <1> db 01dh, 000h, 000h, 000h, 000h, 024h, 066h, 0ffh, 066h, 024h, 000h, 000h, 000h, 000h, 000h, 022h

733 000152C1 24000000000022 <1>

734 000152C8 006363632200000000- <1> db 000h, 063h, 063h, 063h, 022h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 02bh, 000h

734 000152D1 00000000002B00 <1>

735 000152D8 0000181818FF181818- <1> db 000h, 000h, 018h, 018h, 018h, 0ffh, 018h, 018h, 018h, 000h, 000h, 000h, 000h, 02dh, 000h, 000h

735 000152E1 000000002D0000 <1>

736 000152E8 00000000FF00000000- <1> db 000h, 000h, 000h, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 04dh, 000h, 000h, 0c3h

736 000152F1 0000004D0000C3 <1>

737 000152F8 E7FFDBC3C3C3C3C300- <1> db 0e7h, 0ffh, 0dbh, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 000h, 000h, 000h, 054h, 000h, 000h, 0ffh, 0dbh

737 00015301 0000540000FFDB <1>

738 00015308 9918181818183C0000- <1> db 099h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 056h, 000h, 000h, 0c3h, 0c3h, 0c3h

738 00015311 00560000C3C3C3 <1>

739 00015318 C3C3C3663C18000000- <1> db 0c3h, 0c3h, 0c3h, 066h, 03ch, 018h, 000h, 000h, 000h, 057h, 000h, 000h, 0c3h, 0c3h, 0c3h, 0c3h

739 00015321 570000C3C3C3C3 <1>

740 00015328 DBDBFF666600000058- <1> db 0dbh, 0dbh, 0ffh, 066h, 066h, 000h, 000h, 000h, 058h, 000h, 000h, 0c3h, 0c3h, 066h, 03ch, 018h

740 00015331 0000C3C3663C18 <1>

741 00015338 3C66C3C30000005900- <1> db 03ch, 066h, 0c3h, 0c3h, 000h, 000h, 000h, 059h, 000h, 000h, 0c3h, 0c3h, 0c3h, 066h, 03ch, 018h

741 00015341 00C3C3C3663C18 <1>

742 00015348 18183C0000005A0000- <1> db 018h, 018h, 03ch, 000h, 000h, 000h, 05ah, 000h, 000h, 0ffh, 0c3h, 086h, 00ch, 018h, 030h, 061h

742 00015351 FFC3860C183061 <1>

743 00015358 C3FF0000006D000000- <1> db 0c3h, 0ffh, 000h, 000h, 000h, 06dh, 000h, 000h, 000h, 000h, 000h, 0e6h, 0ffh, 0dbh, 0dbh, 0dbh

743 00015361 0000E6FFDBDBDB <1>

744 00015368 DB0000007600000000- <1> db 0dbh, 000h, 000h, 000h, 076h, 000h, 000h, 000h, 000h, 000h, 0c3h, 0c3h, 0c3h, 066h, 03ch, 018h

744 00015371 00C3C3C3663C18 <1>

745 00015378 000000770000000000- <1> db 000h, 000h, 000h, 077h, 000h, 000h, 000h, 000h, 000h, 0c3h, 0c3h, 0dbh, 0dbh, 0ffh, 066h, 000h

745 00015381 C3C3DBDBFF6600 <1>

746 00015388 000091000000006E3B- <1> db 000h, 000h, 091h, 000h, 000h, 000h, 000h, 06eh, 03bh, 01bh, 07eh, 0d8h, 0dch, 077h, 000h, 000h

746 00015391 1B7ED8DC770000 <1>

747 00015398 009B0018187EC3C0C0- <1> db 000h, 09bh, 000h, 018h, 018h, 07eh, 0c3h, 0c0h, 0c0h, 0c3h, 07eh, 018h, 018h, 000h, 000h, 000h

747 000153A1 C37E1818000000 <1>

748 000153A8 9D0000C3663C18FF18- <1> db 09dh, 000h, 000h, 0c3h, 066h, 03ch, 018h, 0ffh, 018h, 0ffh, 018h, 018h, 000h, 000h, 000h, 09eh

748 000153B1 FF18180000009E <1>

749 000153B8 00FC66667C62666F66- <1> db 000h, 0fch, 066h, 066h, 07ch, 062h, 066h, 06fh, 066h, 066h, 0f3h, 000h, 000h, 000h, 0f1h, 000h

749 000153C1 66F3000000F100 <1>

750 000153C8 00181818FF18181800- <1> db 000h, 018h, 018h, 018h, 0ffh, 018h, 018h, 018h, 000h, 0ffh, 000h, 000h, 000h, 0f6h, 000h, 000h

750 000153D1 FF000000F60000 <1>

751 000153D8 18180000FF00001818- <1> db 018h, 018h, 000h, 000h, 0ffh, 000h, 000h, 018h, 018h, 000h, 000h, 000h, 000h

751 000153E1 00000000 <1>

752 <1> vgafont16alt:

753 000153E5 1D00000000002466FF- <1> db 01dh, 000h, 000h, 000h, 000h, 000h, 024h, 066h, 0ffh, 066h, 024h, 000h, 000h, 000h, 000h, 000h

753 000153EE 66240000000000 <1>

754 000153F5 003000003C66C3C3DB- <1> db 000h, 030h, 000h, 000h, 03ch, 066h, 0c3h, 0c3h, 0dbh, 0dbh, 0c3h, 0c3h, 066h, 03ch, 000h, 000h

754 000153FE DBC3C3663C0000 <1>

755 00015405 00004D0000C3E7FFFF- <1> db 000h, 000h, 04dh, 000h, 000h, 0c3h, 0e7h, 0ffh, 0ffh, 0dbh, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 000h

755 0001540E DBC3C3C3C3C300 <1>

756 00015415 000000540000FFDB99- <1> db 000h, 000h, 000h, 054h, 000h, 000h, 0ffh, 0dbh, 099h, 018h, 018h, 018h, 018h, 018h, 018h, 03ch

756 0001541E 1818181818183C <1>

757 00015425 00000000560000C3C3- <1> db 000h, 000h, 000h, 000h, 056h, 000h, 000h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 066h, 03ch

757 0001542E C3C3C3C3C3663C <1>

758 00015435 1800000000570000C3- <1> db 018h, 000h, 000h, 000h, 000h, 057h, 000h, 000h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 0dbh, 0dbh, 0ffh

758 0001543E C3C3C3C3DBDBFF <1>

759 00015445 666600000000580000- <1> db 066h, 066h, 000h, 000h, 000h, 000h, 058h, 000h, 000h, 0c3h, 0c3h, 066h, 03ch, 018h, 018h, 03ch

759 0001544E C3C3663C18183C <1>

760 00015455 66C3C3000000005900- <1> db 066h, 0c3h, 0c3h, 000h, 000h, 000h, 000h, 059h, 000h, 000h, 0c3h, 0c3h, 0c3h, 066h, 03ch, 018h

760 0001545E 00C3C3C3663C18 <1>

761 00015465 1818183C000000005A- <1> db 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h, 05ah, 000h, 000h, 0ffh, 0c3h, 086h, 00ch, 018h

761 0001546E 0000FFC3860C18 <1>

762 00015475 3060C1C3FF00000000- <1> db 030h, 060h, 0c1h, 0c3h, 0ffh, 000h, 000h, 000h, 000h, 06dh, 000h, 000h, 000h, 000h, 000h, 0e6h

762 0001547E 6D0000000000E6 <1>

763 00015485 FFDBDBDBDBDB000000- <1> db 0ffh, 0dbh, 0dbh, 0dbh, 0dbh, 0dbh, 000h, 000h, 000h, 000h, 076h, 000h, 000h, 000h, 000h, 000h

763 0001548E 00760000000000 <1>

764 00015495 C3C3C3C3663C180000- <1> db 0c3h, 0c3h, 0c3h, 0c3h, 066h, 03ch, 018h, 000h, 000h, 000h, 000h, 077h, 000h, 000h, 000h, 000h

764 0001549E 00007700000000 <1>

765 000154A5 00C3C3C3DBDBFF6600- <1> db 000h, 0c3h, 0c3h, 0c3h, 0dbh, 0dbh, 0ffh, 066h, 000h, 000h, 000h, 000h, 078h, 000h, 000h, 000h

765 000154AE 00000078000000 <1>

766 000154B5 0000C3663C183C66C3- <1> db 000h, 000h, 0c3h, 066h, 03ch, 018h, 03ch, 066h, 0c3h, 000h, 000h, 000h, 000h, 091h, 000h, 000h

766 000154BE 00000000910000 <1>

767 000154C5 0000006E3B1B7ED8DC- <1> db 000h, 000h, 000h, 06eh, 03bh, 01bh, 07eh, 0d8h, 0dch, 077h, 000h, 000h, 000h, 000h, 09bh, 000h

767 000154CE 77000000009B00 <1>

768 000154D5 18187EC3C0C0C0C37E- <1> db 018h, 018h, 07eh, 0c3h, 0c0h, 0c0h, 0c0h, 0c3h, 07eh, 018h, 018h, 000h, 000h, 000h, 000h, 09dh

768 000154DE 1818000000009D <1>

769 000154E5 0000C3663C18FF18FF- <1> db 000h, 000h, 0c3h, 066h, 03ch, 018h, 0ffh, 018h, 0ffh, 018h, 018h, 018h, 000h, 000h, 000h, 000h

769 000154EE 18181800000000 <1>

770 000154F5 9E00FC66667C62666F- <1> db 09eh, 000h, 0fch, 066h, 066h, 07ch, 062h, 066h, 06fh, 066h, 066h, 066h, 0f3h, 000h, 000h, 000h

770 000154FE 666666F3000000 <1>

771 00015505 00AB00C0C0C2C6CC18- <1> db 000h, 0abh, 000h, 0c0h, 0c0h, 0c2h, 0c6h, 0cch, 018h, 030h, 060h, 0ceh, 09bh, 006h, 00ch, 01fh

771 0001550E 3060CE9B060C1F <1>

772 00015515 0000AC00C0C0C2C6CC- <1> db 000h, 000h, 0ach, 000h, 0c0h, 0c0h, 0c2h, 0c6h, 0cch, 018h, 030h, 066h, 0ceh, 096h, 03eh, 006h

772 0001551E 183066CE963E06 <1>

773 00015525 06000000 <1> db 006h, 000h, 000h, 000h

2646

2647 00015529 90 align 2

2648

2649 ; EPOCH Variables

2650 ; 13/04/2015 - Retro UNIX 386 v1 Beginning

2651 ; 09/04/2013 epoch variables

2652 ; Retro UNIX 8086 v1 Prototype: UNIXCOPY.ASM, 10/03/2013

2653 ;

2654 0001552A B207 year: dw 1970

2655 0001552C 0100 month: dw 1

2656 0001552E 0100 day: dw 1

2657 00015530 0000 hour: dw 0

2658 00015532 0000 minute: dw 0

2659 00015534 0000 second: dw 0

2660

2661 DMonth:

2662 00015536 0000 dw 0

2663 00015538 1F00 dw 31

2664 0001553A 3B00 dw 59

2665 0001553C 5A00 dw 90

2666 0001553E 7800 dw 120

2667 00015540 9700 dw 151

2668 00015542 B500 dw 181

2669 00015544 D400 dw 212

2670 00015546 F300 dw 243

2671 00015548 1101 dw 273

2672 0001554A 3001 dw 304

2673 0001554C 4E01 dw 334

2674

2675 ; 20/02/2017

2676 KERNELFSIZE equ $ ; 04/07/2016

2677

2678 bss\_start:

2679

2680 ABSOLUTE bss\_start

2681

2682 0001554E <res 00000002> alignb 8 ; 25/12/2016

2683

2684 ; 15/04/2016

2685 ; TRDOS 386 (TRDOS v2.0)

2686 ; 80 interrupts

2687 ; 11/03/2015

2688 ; Interrupt Descriptor Table (20/08/2014)

2689 idt:

2690 ;resb 64\*8 ; INT 0 to INT 3Fh

2691 ; 15/04/2016

2692 00015550 <res 00000280> resb 80\*8 ; INT 0 to INT 4Fh

2693

2694 idt\_end:

2695

2696 ;alignb 4

2697

2698 task\_state\_segment:

2699 ; 24/03/2015

2700 000157D0 <res 00000002> tss.link: resw 1

2701 000157D2 <res 00000002> resw 1

2702 ; tss offset 4

2703 000157D4 <res 00000004> tss.esp0: resd 1

2704 000157D8 <res 00000002> tss.ss0: resw 1

2705 000157DA <res 00000002> resw 1

2706 000157DC <res 00000004> tss.esp1: resd 1

2707 000157E0 <res 00000002> tss.ss1: resw 1

2708 000157E2 <res 00000002> resw 1

2709 000157E4 <res 00000004> tss.esp2: resd 1

2710 000157E8 <res 00000002> tss.ss2: resw 1

2711 000157EA <res 00000002> resw 1

2712 ; tss offset 28

2713 000157EC <res 00000004> tss.CR3: resd 1

2714 000157F0 <res 00000004> tss.eip: resd 1

2715 000157F4 <res 00000004> tss.eflags: resd 1

2716 ; tss offset 40

2717 000157F8 <res 00000004> tss.eax: resd 1

2718 000157FC <res 00000004> tss.ecx: resd 1

2719 00015800 <res 00000004> tss.edx: resd 1

2720 00015804 <res 00000004> tss.ebx: resd 1

2721 00015808 <res 00000004> tss.esp: resd 1

2722 0001580C <res 00000004> tss.ebp: resd 1

2723 00015810 <res 00000004> tss.esi: resd 1

2724 00015814 <res 00000004> tss.edi: resd 1

2725 ; tss offset 72

2726 00015818 <res 00000002> tss.ES: resw 1

2727 0001581A <res 00000002> resw 1

2728 0001581C <res 00000002> tss.CS: resw 1

2729 0001581E <res 00000002> resw 1

2730 00015820 <res 00000002> tss.SS: resw 1

2731 00015822 <res 00000002> resw 1

2732 00015824 <res 00000002> tss.DS: resw 1

2733 00015826 <res 00000002> resw 1

2734 00015828 <res 00000002> tss.FS: resw 1

2735 0001582A <res 00000002> resw 1

2736 0001582C <res 00000002> tss.GS: resw 1

2737 0001582E <res 00000002> resw 1

2738 00015830 <res 00000002> tss.LDTR: resw 1

2739 00015832 <res 00000002> resw 1

2740 ; tss offset 100

2741 00015834 <res 00000002> resw 1

2742 00015836 <res 00000002> tss.IOPB: resw 1

2743 ; tss offset 104

2744 tss\_end:

2745

2746 00015838 <res 00000004> k\_page\_dir: resd 1 ; Kernel's (System) Page Directory address

2747 ; (Physical address = Virtual address)

2748 0001583C <res 00000004> memory\_size: resd 1 ; memory size in pages

2749 00015840 <res 00000004> free\_pages: resd 1 ; number of free pages

2750 00015844 <res 00000004> next\_page: resd 1 ; offset value in M.A.T. for

2751 ; first free page search

2752 00015848 <res 00000004> last\_page: resd 1 ; offset value in M.A.T. which

2753 ; next free page search will be

2754 ; stopped after it. (end of M.A.T.)

2755 0001584C <res 00000004> first\_page: resd 1 ; offset value in M.A.T. which

2756 ; first free page search

2757 ; will be started on it. (for user)

2758 00015850 <res 00000004> mat\_size: resd 1 ; Memory Allocation Table size in pages

2759

2760 ; 02/09/2014 (Retro UNIX 386 v1)

2761 ; 04/12/2013 (Retro UNIX 8086 v1)

2762 00015854 <res 00000002> CRT\_START: resw 1 ; starting address in regen buffer

2763 ; NOTE: active page only

2764 00015856 <res 00000010> CURSOR\_POSN: resw 8 ; cursor positions for video pages

2765 ACTIVE\_PAGE:

2766 00015866 <res 00000001> ptty: resb 1 ; current tty

2767 ; 01/07/2015 - 29/01/2016

2768 00015867 <res 00000001> ccolor: resb 1 ; current color attribute

2769 ; 26/10/2015

2770 ; 07/09/2014

2771 00015868 <res 00000014> ttychr: resw ntty+2 ; Character buffer (multiscreen)

2772

2773 ; 18/05/2015 (03/06/2013 - Retro UNIX 8086 v1 feature only!)

2774 0001587C <res 00000004> p\_time: resd 1 ; present time (for systime & sysmdate)

2775

2776 ; 18/05/2015 (16/08/2013 - Retro UNIX 8086 v1 feature only !)

2777 ; (open mode locks for pseudo TTYs)

2778 ; [ major tty locks (return error in any conflicts) ]

2779 00015880 <res 00000014> ttyl: resw ntty+2 ; opening locks for TTYs.

2780

2781 ; 15/04/2015 (Retro UNIX 386 v1)

2782 ; 22/09/2013 (Retro UNIX 8086 v1)

2783 00015894 <res 0000000A> wlist: resb ntty+2 ; wait channel list (0 to 9 for TTYs)

2784 ; 15/04/2015 (Retro UNIX 386 v1)

2785 ;; 12/07/2014 -> sp\_init set comm. parameters as 0E3h

2786 ;; 0 means serial port is not available

2787 ;;comprm: ; 25/06/2014

2788 0001589E <res 00000001> com1p: resb 1 ;;0E3h

2789 0001589F <res 00000001> com2p: resb 1 ;;0E3h

2790

2791 ; 17/11/2015

2792 ; request for response (from the terminal)

2793 000158A0 <res 00000002> req\_resp: resw 1

2794 ; 07/11/2015

2795 000158A2 <res 00000001> ccomport: resb 1 ; current COM (serial) port

2796 ; (0= COM1, 1= COM2)

2797 ; 09/11/2015

2798 000158A3 <res 00000001> comqr: resb 1 ; 'query or response' sign (u9.s, 'sndc')

2799 ; 07/11/2015

2800 000158A4 <res 00000002> rchar: resw 1 ; last received char for COM 1 and COM 2

2801 000158A6 <res 00000002> schar: resw 1 ; last sent char for COM 1 and COM 2

2802

2803 ; 22/08/2014 (RTC)

2804 ; (Packed BCD)

2805 000158A8 <res 00000001> time\_seconds: resb 1

2806 000158A9 <res 00000001> time\_minutes: resb 1

2807 000158AA <res 00000001> time\_hours: resb 1

2808 000158AB <res 00000001> date\_wday: resb 1

2809 000158AC <res 00000001> date\_day: resb 1

2810 000158AD <res 00000001> date\_month: resb 1

2811 000158AE <res 00000001> date\_year: resb 1

2812 000158AF <res 00000001> date\_century: resb 1

2813

2814 ; 24/01/2016

2815 000158B0 <res 00000004> RTC\_LH: resd 1

2816 000158B4 <res 00000001> RTC\_WAIT\_FLAG: resb 1

2817 000158B5 <res 00000001> USER\_FLAG: resb 1

2818 ; 19/05/2016

2819 ;RTC\_second:

2820 000158B6 <res 00000001> RTC\_2Hz: resb 1 ; from 2Hz interrupt to 1Hz timer event function

2821

2822 %include 'diskbss.s' ; UNINITIALIZED DISK (BIOS) DATA

1 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2 <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0 - diskbss.s

3 <1> ; ----------------------------------------------------------------------------

4 <1> ; Last Update: 24/01/2016

5 <1> ; ----------------------------------------------------------------------------

6 <1> ; Beginning: 24/01/2016

7 <1> ; ----------------------------------------------------------------------------

8 <1> ; Assembler: NASM version 2.11 (trdos386.s)

9 <1> ; ----------------------------------------------------------------------------

10 <1> ; Turkish Rational DOS

11 <1> ; Operating System Project v2.0 by ERDOGAN TAN (Beginning: 04/01/2016)

12 <1> ;

13 <1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan

14 <1> ; diskbss.inc (10/07/2015)

15 <1> ;

16 <1> ; Derived from 'IBM PC-XT-286' BIOS source code (1986)

17 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

18 <1>

19 <1> ; Retro UNIX 386 v1 Kernel - DISKBSS.INC

20 <1> ; Last Modification: 10/07/2015

21 <1> ; (Unnitialized Disk Parameters Data section for 'DISKIO.INC')

22 <1>

23 000158B7 <res 00000001> <1> alignb 2

24 <1>

25 <1> ;----------------------------------------

26 <1> ; TIMER DATA AREA :

27 <1> ;----------------------------------------

28 <1>

29 <1> TIMER\_LH: ; 16/02/205

30 000158B8 <res 00000002> <1> TIMER\_LOW: resw 1 ; LOW WORD OF TIMER COUNT

31 000158BA <res 00000002> <1> TIMER\_HIGH: resw 1 ; HIGH WORD OF TIMER COUNT

32 000158BC <res 00000001> <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 000158BD <res 00000001> <1> SEEK\_STATUS: resb 1

39 000158BE <res 00000001> <1> MOTOR\_STATUS: resb 1

40 000158BF <res 00000001> <1> MOTOR\_COUNT: resb 1

41 000158C0 <res 00000001> <1> DSKETTE\_STATUS: resb 1

42 000158C1 <res 00000007> <1> NEC\_STATUS: resb 7

43 <1>

44 <1> ;----------------------------------------

45 <1> ; ADDITIONAL MEDIA DATA :

46 <1> ;----------------------------------------

47 <1>

48 000158C8 <res 00000001> <1> LASTRATE: resb 1

49 000158C9 <res 00000001> <1> HF\_STATUS: resb 1

50 000158CA <res 00000001> <1> HF\_ERROR: resb 1

51 000158CB <res 00000001> <1> HF\_INT\_FLAG: resb 1

52 000158CC <res 00000001> <1> HF\_CNTRL: resb 1

53 000158CD <res 00000004> <1> DSK\_STATE: resb 4

54 000158D1 <res 00000002> <1> DSK\_TRK: resb 2

55 <1>

56 <1> ;----------------------------------------

57 <1> ; FIXED DISK DATA AREAS :

58 <1> ;----------------------------------------

59 <1>

60 000158D3 <res 00000001> <1> DISK\_STATUS1: resb 1 ; FIXED DISK STATUS

61 000158D4 <res 00000001> <1> HF\_NUM: resb 1 ; COUNT OF FIXED DISK DRIVES

62 000158D5 <res 00000001> <1> CONTROL\_BYTE: resb 1 ; HEAD CONTROL BYTE

63 <1> ;@PORT\_OFF resb 1 ; RESERVED (PORT OFFSET)

64 <1> ;port1\_off resb 1 ; Hard disk controller 1 - port offset

65 <1> ;port2\_off resb 1 ; Hard idsk controller 2 - port offset

66 <1>

67 000158D6 <res 00000002> <1> alignb 4

68 <1>

69 <1> ;HF\_TBL\_VEC: resd 1 ; Primary master disk param. tbl. pointer

70 <1> ;HF1\_TBL\_VEC: resd 1 ; Primary slave disk param. tbl. pointer

71 <1> HF\_TBL\_VEC: ; 22/12/2014

72 000158D8 <res 00000004> <1> HDPM\_TBL\_VEC: resd 1 ; Primary master disk param. tbl. pointer

73 000158DC <res 00000004> <1> HDPS\_TBL\_VEC: resd 1 ; Primary slave disk param. tbl. pointer

74 000158E0 <res 00000004> <1> HDSM\_TBL\_VEC: resd 1 ; Secondary master disk param. tbl. pointer

75 000158E4 <res 00000004> <1> HDSS\_TBL\_VEC: resd 1 ; Secondary slave disk param. tbl. pointer

76 <1>

77 <1> ; 03/01/2015

78 000158E8 <res 00000001> <1> LBAMode: resb 1

79 <1>

80 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2823

2824 ;;; Real Mode Data (10/07/2015 - BSS)

2825

2826 ;alignb 2

2827

2828 ; 10/01/2016

2829 %include 'trdoskx.s' ; UNINITIALIZED KERNEL (Logical Drive & FS) DATA

1 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2 <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.0) - UNINITIALIZED DATA : trdoskx.s

3 <1> ; ----------------------------------------------------------------------------

4 <1> ; Last Update: 28/08/2017

5 <1> ; ----------------------------------------------------------------------------

6 <1> ; Beginning: 04/01/2016

7 <1> ; ----------------------------------------------------------------------------

8 <1> ; Assembler: NASM version 2.11 (trdos386.s)

9 <1> ; ----------------------------------------------------------------------------

10 <1> ; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan

11 <1> ; TRDOS2.ASM (09/11/2011)

12 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

13 <1> ; DRV\_INIT.ASM [26/09/2009] Last Update: 07/08/2011

14 <1> ; MAINPROG.ASM [17/01/2004] Last Update: 09/11/2011

15 <1> ; DIR.ASM [17/01/2004] Last Update: 09/10/2011

16 <1> ; CMD\_INTR.ASM [29/01/2005] Last update: 09/11/2011

17 <1> ; DRV\_FAT.ASM [07/07/2009] Last update: 21/08/2011

18 <1>

19 000158E9 <res 00000003> <1> alignb 4

20 <1>

21 <1> ; MAINPROG.ASM

22 000158EC <res 00000004> <1> MainProgCfg\_FileSize: resd 1 ; 14/04/2016

23 000158F0 <res 00000004> <1> MainProgCfg\_LineOffset: resd 1 ; 14/04/2016

24 <1>

25 000158F4 <res 00000004> <1> Current\_VolSerial: resd 1

26 <1>

27 000158F8 <res 00000004> <1> Current\_Dir\_FCluster: resd 1

28 <1>

29 000158FC <res 00000001> <1> Current\_Dir\_Level: resb 1

30 000158FD <res 00000001> <1> Current\_FATType: resb 1

31 000158FE <res 00000001> <1> Current\_Drv: resb 1

32 000158FF <res 00000001> <1> Current\_Dir\_Drv: resb 1 ; '?'

33 00015900 <res 00000001> <1> resb 1 ; ':'

34 00015901 <res 00000001> <1> Current\_Dir\_Root: resb 1 ; '/'

35 00015902 <res 0000005A> <1> Current\_Directory: resb 90

36 0001595C <res 00000001> <1> End\_Of\_Current\_Dir\_Str: resb 1

37 0001595D <res 00000001> <1> Current\_Dir\_StrLen: resb 1

38 <1>

39 0001595E <res 00000001> <1> CursorColumn: resb 1

40 0001595F <res 00000001> <1> CmdArgStart: resb 1

41 <1>

42 <1> ; 03/02/2016

43 00015960 <res 0000004E> <1> Remark: resb 78

44 <1>

45 000159AE <res 00000050> <1> CommandBuffer: resb 80

46 <1>

47 000159FE <res 00000100> <1> TextBuffer: resb 256

48 <1>

49 <1> MasterBootBuff:

50 00015AFE <res 000001BE> <1> MasterBootCode: resb 1BEh

51 00015CBC <res 00000040> <1> PartitionTable: resb 64

52 00015CFC <res 00000002> <1> MBIDCode: resw 1

53 <1>

54 <1> PTable\_Buffer:

55 00015CFE <res 00000040> <1> PTable\_hd0: resb 64

56 00015D3E <res 00000040> <1> PTable\_hd1: resb 64

57 00015D7E <res 00000040> <1> PTable\_hd2: resb 64

58 00015DBE <res 00000040> <1> PTable\_hd3: resb 64

59 00015DFE <res 00000040> <1> PTable\_ep0: resb 64

60 00015E3E <res 00000040> <1> PTable\_ep1: resb 64

61 00015E7E <res 00000040> <1> PTable\_ep2: resb 64

62 00015EBE <res 00000040> <1> PTable\_ep3: resb 64

63 <1>

64 00015EFE <res 00000001> <1> scount: resb 1 ; 16/05/2016 (diskio.s, 'int33h:')

65 00015EFF <res 00000001> <1> HD\_LBA\_yes: resb 1

66 00015F00 <res 00000001> <1> PP\_Counter: resb 1

67 00015F01 <res 00000001> <1> EP\_Counter: resb 1

68 <1>

69 00015F02 <res 00000004> <1> EP\_StartSector: resd 1

70 00015F06 <res 00000004> <1> resd 1

71 00015F0A <res 00000004> <1> resd 1

72 00015F0E <res 00000004> <1> resd 1

73 <1>

74 00015F12 <res 00000200> <1> DOSBootSectorBuff: resb 512

75 <1>

76 <1> FAT\_BuffDescriptor:

77 00016112 <res 00000004> <1> FAT\_CurrentCluster: resd 1

78 00016116 <res 00000001> <1> FAT\_BuffValidData: resb 1

79 00016117 <res 00000001> <1> FAT\_BuffDrvName: resb 1

80 00016118 <res 00000002> <1> FAT\_BuffOffset: resw 1

81 0001611A <res 00000004> <1> FAT\_BuffSector: resd 1

82 <1>

83 0001611E <res 00000004> <1> FAT\_ClusterCounter: resd 1

84 00016122 <res 00000004> <1> LastCluster: resd 1

85 <1>

86 <1> ; 16/05/2016

87 <1> ;; 18/03/2016 (TRDOS v2.0)

88 <1> ;ClusterBuffer\_Valid: resb 1

89 <1>

90 <1> Dir\_BuffDescriptor:

91 00016126 <res 00000001> <1> DirBuff\_DRV: resb 1

92 00016127 <res 00000001> <1> DirBuff\_FATType: resb 1

93 00016128 <res 00000001> <1> DirBuff\_ValidData: resb 1

94 00016129 <res 00000002> <1> DirBuff\_CurrentEntry: resw 1

95 0001612B <res 00000002> <1> DirBuff\_LastEntry: resw 1

96 0001612D <res 00000004> <1> DirBuff\_Cluster: resd 1

97 00016131 <res 00000002> <1> DirBuffer\_Size: resw 1

98 <1> ;DirBuff\_EntryCounter: resw 1

99 <1>

100 <1> ; 01/02/2016

101 <1> ; these are on (real mode) segment 8000h and later

102 <1> ; FAT\_Buffer: resb 1536 ; 3 sectors

103 <1> ; Dir\_Buffer: resb 512\*32

104 <1> ; Logical\_DOSDisks: resb 6656 ; 26 \* 256 bytes

105 <1>

106 <1> ; 18/01/2016

107 <1>

108 00016133 <res 00000004> <1> FreeClusterCount: resd 1

109 <1>

110 00016137 <res 00000004> <1> VolSize\_Unit1: resd 1

111 0001613B <res 00000004> <1> VolSize\_Unit2: resd 1

112 <1>

113 0001613F <res 00000004> <1> Vol\_Tot\_Sec\_Str\_Start: resd 1

114 00016143 <res 0000000A> <1> Vol\_Tot\_Sec\_Str: resb 10

115 0001614D <res 00000001> <1> Vol\_Tot\_Sec\_Str\_End: resb 1

116 0001614E <res 00000001> <1> resb 1

117 0001614F <res 00000004> <1> Vol\_Free\_Sectors\_Str\_Start: resd 1

118 00016153 <res 0000000A> <1> Vol\_Free\_Sectors\_Str: resb 10

119 0001615D <res 00000001> <1> Vol\_Free\_Sectors\_Str\_End: resb 1

120 <1>

121 <1> ; 10/02/2016

122 0001615E <res 00000001> <1> RUN\_CDRV: resb 1 ; CMD\_INTR.ASM ; 09/11/2011

123 <1>

124 <1> ; 24/01/2016

125 0001615F <res 00000080> <1> PATH\_Array: resb 128 ; DIR.ASM ; 09/10/2011

126 <1> ; 06/02/2016

127 000161DF <res 00000004> <1> CCD\_DriveDT: resd 1 ; DIR.ASM ; (word)

128 000161E3 <res 00000001> <1> CCD\_Level: resb 1 ; DIR.ASM

129 000161E4 <res 00000001> <1> Last\_Dir\_Level: resb 1 ; DIR.ASM

130 <1> ;

131 000161E5 <res 00000002> <1> CDLF\_AttributesMask: resw 1 ; DIR.ASM

132 000161E7 <res 00000004> <1> CDLF\_FNAddress: resd 1 ; DIR.ASM (word)

133 000161EB <res 00000002> <1> CDLF\_DEType: resw 1 ; DIR.ASM

134 <1> ;

135 000161ED <res 00000001> <1> CD\_COMMAND: resb 1 ; DIR.ASM

136 <1>

137 000161EE <res 00000002> <1> alignb 4

138 <1>

139 <1> ; 29/01/2016

140 000161F0 <res 00000001> <1> Program\_Exit: resb 1 ; CMD\_INTR.ASM ; 09/11/2011

141 <1>

142 <1> ;alignb 4

143 <1> ; 23/02/2016

144 000161F1 <res 00000001> <1> disk\_rw\_op: resb 1 ; 0 = disk read, 1 = disk write

145 <1> ;disk\_rw\_spt: resb 1 ; sectors per track (<= 63) /// (<256)

146 <1> ; 31/01/2016

147 000161F2 <res 00000001> <1> retry\_count: resb 1 ; DISK\_IO.ASM ; 20/07/2011 (CHS\_RetryCount)

148 000161F3 <res 00000001> <1> disk\_rw\_err: resb 1 ; DISK\_IO.ASM ; (Disk\_IO\_err\_code)

149 000161F4 <res 00000004> <1> sector\_count: resd 1 ; DISK\_IO.ASM ; (Disk\_RW\_SectorCount)

150 <1>

151 <1> ; 06/02/2016 (long name)

152 000161F8 <res 00000002> <1> FDE\_AttrMask: resw 1 ; DIR.ASM

153 000161FA <res 00000002> <1> AmbiguousFileName: resw 1 ; DIR.ASM

154 000161FC <res 00000001> <1> PreviousAttr: resb 1 ; DIR.ASM

155 <1> ;

156 000161FD <res 00000001> <1> LongNameFound: resb 1 ; DIR.ASM

157 000161FE <res 00000001> <1> LFN\_EntryLength: resb 1 ; DIR.ASM

158 000161FF <res 00000001> <1> LFN\_CheckSum: resb 1 ; DIR.ASM

159 00016200 <res 00000084> <1> LongFileName: resb 132 ; DIR.ASM

160 <1>

161 <1> ;PATH\_Array\_Ptr: resw 1 ; DIR.ASM

162 00016284 <res 00000001> <1> PATH\_CDLevel: resb 1 ; DIR.ASM

163 00016285 <res 00000001> <1> PATH\_Level: resb 1 ; DIR.ASM

164 <1>

165 <1> ; 07/02/2016

166 00016286 <res 0000000D> <1> Dir\_File\_Name: resb 13 ; DIR.ASM ; 09/10/2011

167 <1>

168 <1> ; 10/02/2016

169 00016293 <res 0000000D> <1> Dir\_Entry\_Name: resb 13 ; DIR.ASM

170 <1>

171 <1> alignb 2

172 <1>

173 000162A0 <res 00000002> <1> AttributesMask: resw 1 ; CMD\_INTR.ASM ; 09/11/2011

174 <1>

175 <1> ; 10/02/2016 (128 bytes -> 126 bytes)

176 <1> ; 08/02/2016

177 <1> ;FFF Structure (128 bytes) ; DIR.ASM ; 09/10/2011

178 000162A2 <res 00000001> <1> FindFile\_Drv: resb 1

179 000162A3 <res 00000041> <1> FindFile\_Directory: resb 65

180 000162E4 <res 0000000D> <1> FindFile\_Name: resb 13

181 <1> FindFile\_LongNameEntryLength:

182 000162F1 <res 00000001> <1> FindFile\_LongNameYes: resb 1 ; Sign for longname procedures

183 <1> ;Above 80 bytes form

184 <1> ;TR-DOS Source/Destination File FullName Format/Structure

185 000162F2 <res 00000002> <1> FindFile\_AttributesMask: resw 1

186 000162F4 <res 00000020> <1> FindFile\_DirEntry: resb 32

187 00016314 <res 00000004> <1> FindFile\_DirFirstCluster: resd 1

188 00016318 <res 00000004> <1> FindFile\_DirCluster: resd 1

189 0001631C <res 00000002> <1> FindFile\_DirEntryNumber: resw 1

190 0001631E <res 00000002> <1> FindFile\_MatchCounter: resw 1

191 00016320 <res 00000002> <1> FindFile\_Reserved: resw 1 ; 06/03/2016

192 <1>

193 00016322 <res 00000004> <1> First\_Path\_Pos: resd 1 ; DIR.ASM ; 09/10/2011

194 00016326 <res 00000004> <1> Last\_Slash\_Pos: resd 1 ; DIR.ASM

195 <1>

196 <1> ; 10/02/2016

197 0001632A <res 00000002> <1> File\_Count: resw 1 ; DIR.ASM ; 09/10/2011

198 0001632C <res 00000002> <1> Dir\_Count: resw 1

199 0001632E <res 00000004> <1> Total\_FSize: resd 1

200 00016332 <res 00000004> <1> TFS\_Dec\_Begin: resd 1

201 00016336 <res 0000000A> <1> resb 10

202 00016340 <res 00000001> <1> TFS\_Dec\_End: resb 1

203 <1>

204 00016341 <res 00000001> <1> PrintDir\_RowCounter: resb 1

205 <1>

206 00016342 <res 00000002> <1> alignb 4

207 <1> ; 15/02/2015 ('show' command variables)

208 00016344 <res 00000004> <1> Show\_FDT: resd 1

209 00016348 <res 00000004> <1> Show\_LDDDT: resd 1

210 0001634C <res 00000004> <1> Show\_Cluster: resd 1

211 00016350 <res 00000004> <1> Show\_FileSize: resd 1

212 00016354 <res 00000004> <1> Show\_FilePointer: resd 1

213 00016358 <res 00000002> <1> Show\_ClusterPointer: resw 1

214 0001635A <res 00000002> <1> Show\_ClusterSize: resw 1

215 0001635C <res 00000001> <1> Show\_RowCount: resb 1

216 <1>

217 0001635D <res 00000003> <1> alignb 4

218 <1> ; 21/02/2016

219 00016360 <res 00000004> <1> DelFile\_FNPointer: resd 1 ; ; CMD\_INTR.ASM (word) ; 09/11/2011

220 <1> ; 27/02/2016

221 <1> ; DIR.ASM (09/10/2011)

222 00016364 <res 00000004> <1> DelFile\_FCluster: resd 1

223 00016368 <res 00000002> <1> DelFile\_EntryCounter: resw 1

224 0001636A <res 00000001> <1> DelFile\_LNEL: resb 1

225 0001636B <res 00000001> <1> resb 1

226 <1>

227 <1> ; DIR.ASM

228 0001636C <res 00000004> <1> mkdir\_DirName\_Offset: resd 1

229 00016370 <res 00000004> <1> mkdir\_FFCluster: resd 1

230 00016374 <res 00000004> <1> mkdir\_LastDirCluster: resd 1

231 00016378 <res 00000004> <1> mkdir\_FreeSectors: resd 1

232 0001637C <res 00000002> <1> mkdir\_attrib: resw 1

233 0001637E <res 00000001> <1> mkdir\_SecPerClust: resb 1

234 0001637F <res 00000001> <1> mkdir\_add\_new\_cluster: resb 1

235 00016380 <res 0000000D> <1> mkdir\_Name: resb 13

236 0001638D <res 00000002> <1> resw 1 ; 01/03/2016

237 <1> ; 27/02/2016

238 0001638F <res 00000001> <1> RmDir\_MultiClusters: resb 1

239 00016390 <res 00000004> <1> RmDir\_DirEntryOffset: resd 1 ; 01/03/2016 (word -> dword)

240 00016394 <res 00000004> <1> RmDir\_ParentDirCluster: resd 1

241 00016398 <res 00000004> <1> RmDir\_DirLastCluster: resd 1

242 0001639C <res 00000004> <1> RmDir\_PreviousCluster: resd 1

243 <1> ; 22/02/2016

244 000163A0 <res 00000001> <1> UPDLMDT\_CDirLevel: resb 1

245 000163A1 <res 00000004> <1> UPDLMDT\_CDirFCluster: resd 1

246 <1>

247 000163A5 <res 00000003> <1> alignb 4

248 <1> ; DRV\_FAT.ASM ; 21/08/2011

249 000163A8 <res 00000004> <1> gffc\_next\_free\_cluster: resd 1

250 000163AC <res 00000004> <1> gffc\_first\_free\_cluster: resd 1

251 000163B0 <res 00000004> <1> gffc\_last\_free\_cluster: resd 1

252 <1>

253 <1> ;29/04/2016

254 <1> Cluster\_Index: ; resd 1

255 <1> ; 22/02/2016

256 000163B4 <res 00000004> <1> ClusterValue: resd 1

257 <1> ; 04/03/2016

258 000163B8 <res 00000001> <1> Attributes: resb 1

259 <1> ;;CFS\_error: resb 1 ;; 01/03/2016

260 000163B9 <res 00000001> <1> resb 1

261 000163BA <res 00000001> <1> CFS\_OPType: resb 1

262 000163BB <res 00000001> <1> CFS\_Drv: resb 1

263 000163BC <res 00000004> <1> CFS\_CC: resd 1

264 000163C0 <res 00000004> <1> CFS\_FAT32FSINFOSEC: resd 1

265 000163C4 <res 00000004> <1> CFS\_FAT32FC: resd 1

266 <1>

267 <1> ; 27/02/2016

268 <1> ;alignb 4

269 000163C8 <res 00000004> <1> glc\_prevcluster: resd 1 ; DRV\_FAT.ASM (21/08/2011)

270 <1> ; 22/10/2016

271 000163CC <res 00000004> <1> glc\_index: resd 1 ; Last Cluster Index (22/10/2016)

272 <1>

273 <1> ; DIR.ASM

274 000163D0 <res 00000002> <1> DLN\_EntryNumber: resw 1

275 000163D2 <res 00000001> <1> DLN\_40h: resb 1

276 <1> ; 28/02/2016

277 000163D3 <res 00000001> <1> TCC\_FATErr: resb 1 ; DRV\_FAT.ASM

278 <1>

279 <1> alignb 4

280 <1> ; DIR.ASM (09/10/2011)

281 000163D4 <res 00000002> <1> LCDE\_EntryIndex: resw 1 ; LCDE\_EntryOffset

282 000163D6 <res 00000002> <1> LCDE\_ClusterSN: resw 1

283 000163D8 <res 00000004> <1> LCDE\_Cluster: resd 1

284 000163DC <res 00000004> <1> LCDE\_ByteOffset: resd 1

285 <1>

286 <1> ;alignb4

287 <1> ; 06/03/2016 (word -> dword)

288 <1> ; CMD\_INTR.ASM (01/08/2010)

289 000163E0 <res 00000004> <1> SourceFilePath: resd 1

290 000163E4 <res 00000004> <1> DestinationFilePath: resd 1

291 <1>

292 <1> ;alignb 4

293 <1> ; 06/03/2016

294 <1> ; FILE.ASM (09/10/2011)

295 <1> ;Source File Structure (same with 'Find File' Structure)

296 000163E8 <res 00000001> <1> SourceFile\_Drv: resb 1

297 000163E9 <res 00000041> <1> SourceFile\_Directory: resb 65

298 0001642A <res 0000000D> <1> SourceFile\_Name: resb 13

299 <1> SourceFile\_LongNameEntryLength:

300 00016437 <res 00000001> <1> SourceFile\_LongNameYes: resb 1 ; Sign for longname procedures

301 <1> ;Above 80 bytes

302 <1> ;is TR-DOS Source File FullName Format/Structure

303 00016438 <res 00000002> <1> SourceFile\_AttributesMask: resw 1

304 0001643A <res 00000020> <1> SourceFile\_DirEntry: resb 32

305 0001645A <res 00000004> <1> SourceFile\_DirFirstCluster: resd 1

306 0001645E <res 00000004> <1> SourceFile\_DirCluster: resd 1

307 00016462 <res 00000002> <1> SourceFile\_DirEntryNumber: resw 1

308 00016464 <res 00000002> <1> SourceFile\_MatchCounter: resw 1

309 <1> ; 16/03/2016

310 00016466 <res 00000001> <1> SourceFile\_SecPerClust: resb 1

311 00016467 <res 00000001> <1> SourceFile\_Reserved: resb 1

312 <1> ; Above is 128 bytes

313 <1>

314 <1> ;Destination File Structure (same with 'Find File' Structure)

315 00016468 <res 00000001> <1> DestinationFile\_Drv: resb 1

316 00016469 <res 00000041> <1> DestinationFile\_Directory: resb 65

317 000164AA <res 0000000D> <1> DestinationFile\_Name: resb 13

318 <1> DestinationFile\_LongNameEntryLength:

319 000164B7 <res 00000001> <1> DestinationFile\_LongNameYes: resb 1 ; Sign for longname procedures

320 <1> ;Above 80 bytes

321 <1> ;is TR-DOS Destination File FullName Format/Structure

322 000164B8 <res 00000002> <1> DestinationFile\_AttributesMask: resw 1

323 000164BA <res 00000020> <1> DestinationFile\_DirEntry: resb 32

324 000164DA <res 00000004> <1> DestinationFile\_DirFirstCluster: resd 1

325 000164DE <res 00000004> <1> DestinationFile\_DirCluster: resd 1

326 000164E2 <res 00000002> <1> DestinationFile\_DirEntryNumber: resw 1

327 000164E4 <res 00000002> <1> DestinationFile\_MatchCounter: resw 1

328 <1> ; 16/03/2016

329 000164E6 <res 00000001> <1> DestinationFile\_SecPerClust: resb 1

330 000164E7 <res 00000001> <1> DestinationFile\_Reserved: resb 1

331 <1> ; Above is 128 bytes

332 <1>

333 <1> ; 24/04/2016

334 000164E8 <res 00000002> <1> resw 1

335 <1>

336 <1> ; 10/03/2016

337 <1> ; FILE.ASM

338 000164EA <res 00000001> <1> move\_cmd\_phase: resb 1

339 000164EB <res 00000001> <1> msftdf\_sf\_df\_drv: resb 1

340 000164EC <res 00000004> <1> msftdf\_drv\_offset: resd 1

341 <1>

342 <1> ; 11/03/2016

343 <1> ; DRV\_FAT.ASM (21/08/2011)

344 000164F0 <res 00000004> <1> FAT\_anc\_LCluster: resd 1

345 000164F4 <res 00000004> <1> FAT\_anc\_FFCluster: resd 1

346 <1>

347 <1> ;alignb 4

348 <1>

349 <1> ; 14/03/2016

350 <1> ; TRDOS 386 = TRDOS v2.0 feature only !

351 <1> ; 'allocate\_memory\_block' in 'memory.s'

352 000164F8 <res 00000004> <1> mem\_ipg\_count: resd 1 ; page count (for contiguous allocation)

353 000164FC <res 00000004> <1> mem\_pg\_count: resd 1 ; page count (for count down)

354 00016500 <res 00000004> <1> mem\_aperture: resd 1 ; contiguous free pages (current)

355 00016504 <res 00000004> <1> mem\_max\_aperture: resd 1 ; maximum value of contiguous free pages

356 00016508 <res 00000004> <1> mem\_pg\_pos: resd 1 ; mem. position (page #) of current aperture

357 0001650C <res 00000004> <1> mem\_max\_pg\_pos: resd 1 ; mem. position (page #) of max. aperture

358 <1>

359 <1> ; 15/03/2016

360 <1> ; FILE.ASM ('copy\_source\_file\_to\_destination\_file')

361 00016510 <res 00000001> <1> copy\_cmd\_phase: resb 1

362 00016511 <res 00000001> <1> csftdf\_rw\_err: resb 1

363 00016512 <res 00000001> <1> DestinationFileFound: resb 1

364 00016513 <res 00000001> <1> csftdf\_cdrv: resb 1

365 00016514 <res 00000004> <1> csftdf\_filesize: resd 1

366 <1> ; TRDOS386 (TRDOS v2.0)

367 00016518 <res 00000004> <1> csftdf\_sf\_mem\_addr: resd 1

368 0001651C <res 00000004> <1> csftdf\_sf\_mem\_bsize: resd 1

369 <1> ;

370 <1>

371 00016520 <res 00000004> <1> csftdf\_sf\_cluster: resd 1 ; 16/03/2016

372 00016524 <res 00000004> <1> csftdf\_df\_cluster: resd 1

373 <1> ; 16/03/2016

374 00016528 <res 00000004> <1> csftdf\_r\_size: resd 1

375 0001652C <res 00000004> <1> csftdf\_w\_size: resd 1

376 00016530 <res 00000004> <1> csftdf\_sf\_rbytes: resd 1

377 00016534 <res 00000004> <1> csftdf\_df\_wbytes: resd 1

378 00016538 <res 00000001> <1> csftdf\_percentage: resb 1

379 <1> ; 17/03/2016

380 00016539 <res 00000001> <1> csftdf\_videopage: resb 1

381 0001653A <res 00000002> <1> csftdf\_cursorpos: resw 1

382 0001653C <res 00000004> <1> csftdf\_sf\_drv\_dt: resd 1

383 00016540 <res 00000004> <1> csftdf\_df\_drv\_dt: resd 1

384 <1>

385 <1> ; 21/03/2016

386 <1> ; 20/03/2016

387 <1> ; FILE.ASM

388 00016544 <res 00000004> <1> createfile\_Name\_Offset: resd 1

389 00016548 <res 00000004> <1> createfile\_FreeSectors: resd 1

390 0001654C <res 00000004> <1> createfile\_size: resd 1

391 00016550 <res 00000004> <1> createfile\_FFCluster: resd 1 ; 11/03/2016

392 00016554 <res 00000004> <1> createfile\_LastDirCluster: resd 1

393 00016558 <res 00000004> <1> createfile\_Cluster: resd 1

394 0001655C <res 00000004> <1> createfile\_PCluster: resd 1

395 00016560 <res 00000001> <1> createfile\_attrib: resb 1

396 00016561 <res 00000001> <1> createfile\_SecPerClust: resb 1

397 00016562 <res 00000002> <1> createfile\_DirIndex: resw 1

398 00016564 <res 00000004> <1> createfile\_CCount: resd 1

399 00016568 <res 00000002> <1> createfile\_BytesPerSec: resw 1 ; 23/03/2016

400 0001656A <res 00000001> <1> createfile\_wfc: resb 1

401 0001656B <res 00000001> <1> createfile\_UpdatePDir: resb 1 ; 31/03/2016

402 <1>

403 <1> ;alignb 4

404 <1>

405 <1> ; 11/04/2016

406 0001656C <res 00000002> <1> env\_var\_length: resw 1

407 <1>

408 0001656E <res 00000002> <1> alignb 4

409 <1>

410 <1> ; 25/04/2016

411 00016570 <res 00000001> <1> readi.valid: resb 1 ; valid data (>0 = valid for readi)

412 00016571 <res 00000001> <1> readi.drv: resb 1 ; drive number (0, 1,2,3,4..)

413 00016572 <res 00000001> <1> readi.spc: resb 1 ; sectors per cluster for 'readi' drive

414 00016573 <res 00000001> <1> readi.s\_index: resb 1 ; sector index in current cluster (buffer)

415 00016574 <res 00000004> <1> readi.sector: resd 1 ; current disk sector

416 00016578 <res 00000002> <1> readi.bpc: resw 1 ; bytes per cluster - 1

417 0001657A <res 00000002> <1> readi.offset: resw 1 ; byte offset in cluster buffer

418 0001657C <res 00000004> <1> readi.cluster: resd 1 ; current cluster number

419 00016580 <res 00000004> <1> readi.c\_index: resd 1 ; cluster index of the current cluster (0,1,2,3..)

420 00016584 <res 00000004> <1> readi.fclust: resd 1 ; first cluster of the current cluster

421 00016588 <res 00000004> <1> readi.fs\_index: resd 1 ; sector index in disk/file section (for Singlix FS)

422 <1> ;readi.buffer: resd 1 ; readi sector buffer address

423 <1>

424 <1> ;alignb 4

425 <1>

426 0001658C <res 00000001> <1> writei.valid: resb 1 ; valid data (>0 = valid for writei)

427 0001658D <res 00000001> <1> writei.drv: resb 1 ; drive number (0, 1,2,3,4..)

428 0001658E <res 00000001> <1> writei.spc: resb 1 ; sectors per cluster for 'writei' drive

429 0001658F <res 00000001> <1> writei.s\_index: resb 1 ; sector index in current cluster (buffer)

430 00016590 <res 00000004> <1> writei.sector: resd 1 ; current disk sector

431 00016594 <res 00000002> <1> writei.bpc: resw 1 ; bytes per cluster - 1

432 00016596 <res 00000002> <1> writei.offset: resw 1 ; byte offset in cluster buffer

433 00016598 <res 00000004> <1> writei.cluster: resd 1 ; current cluster number

434 0001659C <res 00000004> <1> writei.c\_index: resd 1 ; cluster index of the current cluster (0,1,2,3..)

435 000165A0 <res 00000004> <1> writei.fclust: resd 1 ; first cluster of the current cluster

436 000165A4 <res 00000004> <1> writei.fs\_index: resd 1 ; sector index in disk/file section (for Singlix FS)

437 <1> ;writei.buffer: resd 1 ; writei sector buffer address

438 000165A8 <res 00000004> <1> writei.lclust: resd 1 ; writei last cluster (mget\_w) ; 23/10/2016

439 000165AC <res 00000004> <1> writei.l\_index: resd 1 ; writei last cluster index (mget\_w) ; 23/10/2016

440 000165B0 <res 00000001> <1> writei.ofn: resb 1 ; open file number (to be written) ; 23/10/2016

441 <1>

442 000165B1 <res 00000003> <1> alignb 4

443 <1>

444 <1> ; 29/04/2016

445 000165B4 <res 00000004> <1> Run\_CDirFC: resd 1

446 000165B8 <res 00000001> <1> Run\_Auto\_Path: resb 1

447 000165B9 <res 00000001> <1> Run\_Manual\_Path: resb 1 ; 0 -> auto path sequence needed

448 000165BA <res 00000001> <1> EXE\_ID: resb 1

449 000165BB <res 00000001> <1> EXE\_dot: resb 1

450 <1>

451 <1> ; 06/05/2016

452 000165BC <res 00000004> <1> mainprog\_return\_addr: resd 1

453 000165C0 <res 00000004> <1> last\_error: resd 1 ; this will be used to return error code to MainProg

454 <1> ; 'lasterror' keyword will be used later to get the

455 <1> ; last error code/number/status.

456 <1> ; 12/05/2016

457 000165C4 <res 00000004> <1> video\_eax: resd 1 ; eax return value of video function

458 <1>

459 <1> ; 01/06/2016

460 000165C8 <res 00000004> <1> user\_buffer: resd 1 ; 'diskio.s' (INT 33h, Function 08h, floppy disk type)

461 <1>

462 <1> ; 21/05/2016 - TRDOS 386 ('swap/switch', 'rswap', [u.pri])

463 000165CC <res 00000001> <1> priority: resb 1 ; running priority level of process (0,1,2)

464 <1> ; (run queue which is process comes from)

465 <1> ; 22/05/2016 - TRDOS 386 ('set\_run\_sequence', 'rtc\_int', 'u\_timer')

466 000165CD <res 00000001> <1> p\_change: resb 1 ; process change status (for timer events)

467 <1> ; 23/05/2016 - TRDOS 386 ('clock')

468 000165CE <res 00000001> <1> multi\_tasking: resb 1 ; Multi Tasking status (0 = disabled, >0 = enabled)

469 <1> ; (EBX will return with user buffer addr or disk type)

470 <1> ; 07/06/2016

471 000165CF <res 00000001> <1> timer\_events: resb 1 ; number of (active) timer events, <= 16

472 <1>

473 <1> ; 24/06/2016

474 000165D0 <res 00000001> <1> w\_str\_cmd: resb 1 ; WRITE\_STRING command (0,1,2,3) ; video.s

475 000165D1 <res 00000001> <1> p\_crt\_mode: resb 1 ; previous video mode (=3 or 0), backup mark/sign

476 <1> ; 26/06/2016

477 000165D2 <res 00000001> <1> p\_crt\_page: resb 1 ; previous active page (for 'set\_mode')

478 <1> ; 04/07/2016

479 000165D3 <res 00000001> <1> noclearmem: resb 1 ; if set, 'SET MODE' (INT 31h) function (AH = 4)

480 <1> ; will not clear the video memory

481 <1> ; (usable for graphics modes only)

482 <1> alignb 2

483 000165D4 <res 00000002> <1> CRT\_LEN: resw 1 ; length of regen buffer in bytes

484 000165D6 <res 00000010> <1> cursor\_pposn: resw 8 ; cursor positions backup

485 <1>

486 <1> ; 10/07/2016 ('VGA\_FONT\_SETUP', INT 43H address for x86 real mode bios)

487 000165E6 <res 00000004> <1> VGA\_INT43H: resd 1 ; 0 = default (not configured by user)

488 <1> ; 0FFFFFFFFh = user defined fonts

489 <1> ; address:

490 <1> ; vgafont8

491 <1> ; vgafont16

492 <1> ; vgafont14

493 <1>

494 <1> ; 25/07/2016

495 000165EA <res 00000001> <1> VGA\_MTYPE: resb 1 ; 0=CTEXT,1=MTEXT,2=CGA,3=PLANAR1,4=PLANAR4,5=LINEAR

496 <1>

497 <1> ; 23/10/2016

498 000165EB <res 00000001> <1> setfmod resb 1 ; update last modification date&time sign (if >0)

499 <1> ; (it is Open File Number + 1, if > 0)

500 <1> alignb 4

501 <1>

502 <1> ; 16/10/2016

503 000165EC <res 00000004> <1> FFF\_UBuffer: resd 1 ; User's buffer address for FFF & FNF system calls

504 <1> ; 15/10/2016

505 000165F0 <res 00000001> <1> FFF\_Valid: resb 1 ; Find First File Structure validation byte

506 <1> ; 0 = invalid (Find Next File can't use FFF struct)

507 <1> ; >0 = valid, return type for FFF and Find Next File

508 <1> ; 24 = basic parameters, 24 bytes

509 <1> ; 128 = entire FFF structure/table, 128 bytes

510 <1> ; 16/10/2016 (FFF\_Attrib: resw 1)

511 000165F1 <res 00000001> <1> FFF\_Attrib: resb 1 ; Find First File attributes for Find Next File (LB)

512 000165F2 <res 00000001> <1> FFF\_RType: resb 1 ; FFF return type (0 = Basic, >0 = complete) (HB)

513 <1> ; 16/10/2016 - 05/10/2016 (Set Working Path)

514 000165F3 <res 00000001> <1> SWP\_inv\_fname: resb 1 ; Set Working Path - Invalid File Name

515 000165F4 <res 00000002> <1> SWP\_Mode: resw 1 ; Set Working Path - Mode

516 000165F6 <res 00000001> <1> SWP\_DRV: resb 1 ; Set Working Path - Drive

517 000165F7 <res 00000001> <1> SWP\_DRV\_chg: resb 1 ; Set Working Path - Drive Change

518 <1>

519 <1> ; 27/02/2017

520 000165F8 <res 00000001> <1> fpready: resb 1 ; '80387 fpu is ready' flag

521 <1>

522 <1> ; 08/10/2016

523 000165F9 <res 00000009> <1> device\_name: resb 9 ; capitalized (and zero padded) device canem

524 <1> ; (example: "TTY0",0,0,0,0,0")

525 <1>

526 00016602 <res 00000002> <1> alignb 4

527 <1>

528 <1> ; 08/10/2016

529 <1> ; 07/10/2016

530 <1> ; Table of kernel devices (which do not use installable device drivers)

531 <1> ; has been coded into KERNEL (trdosk9.s)

532 <1> ; 07/10/2016

533 <1> ; 8 installable device drivers available to install (NUMIDEV)

534 00016604 <res 00000020> <1> IDEV\_PGDIR: resd NUMIDEV

535 <1> ; Page directories of installable device drivers

536 <1> ;

537 <1> ; Note: Virtual start address is always 400000h

538 <1> ; (end of the 1st 4MB). [org 400000h]

539 <1> ; Segments: KCODE, KDATA

540 <1> ; Method: call 400000h (after changing page dir)

541 <1> ; Query code located at the start (400000h).

542 <1> ; Query code returns with

543 <1> ; eax = device type and driver version

544 <1> ; AL = Device Type minor

545 <1> ; AH = Device Type major

546 <1> ; Byte 16-23 : Version minor

547 <1> ; Byte 24-31 : Version major - 1

548 <1> ; (0:0 -> 1.0)

549 <1> ; ebx = initialization code address

550 <1> ; ecx = configuration table address

551 <1> ; edx = description table address

552 <1> ; esi = device (default) name address (ASCIIZ)

553 <1> ; (name has "/DEV/" prefix)

554 <1> ; edi = dispatch table address

555 <1> ; (for calling kernel-device functions)

556 <1> ; ebp = address table address

557 <1> ; Initialization code returns with

558 <1> ; eax = open code address

559 <1> ; ecx = close code address

560 <1> ; ebx = read code address

561 <1> ; edx = write code address

562 <1> ; esi = IOCTL code address

563 <1> ; edi = dispatch table address

564 <1> ; ebp = address table address

565 <1> ; Address Table:

566 <1> ; Offset 0 : open code address

567 <1> ; Offset 4 : read code address

568 <1> ; Offset 8 : write code address

569 <1> ; Offset 12 : close code address

570 <1> ; Offset 16 : IOCTL code address

571 <1> ; Offset 20 : initialization code address

572 <1> ; Offset 24 : description table address

573 <1> ; Offset 28 : configuration table address

574 <1> ; Offset 32 : device name address

575 <1> ; Offset 36 : dispatch table address

576 <1> ; (for calling kernel-device functions)

577 <1>

578 00016624 <res 00000040> <1> IDEV\_NAME: resb 8\*NUMIDEV

579 <1> ; 8 byte names of installable device drivers

580 <1>

581 00016664 <res 00000008> <1> IDEV\_TYPE: resb NUMIDEV ; Driver type of installable device drivers

582 0001666C <res 00000008> <1> IDEV\_FLAGS: resb NUMIDEV ; Device access parameters for installable

583 <1> ; device drivers (These values are set while

584 <1> ; the device driver is being loaded.)

585 00016674 <res 00000020> <1> IDEV\_OADDR: resd NUMIDEV ; open function addr for installable dev driver

586 00016694 <res 00000020> <1> IDEV\_CADDR: resd NUMIDEV ; close function addr for installable dev driver

587 000166B4 <res 00000020> <1> IDEV\_RADDR: resd NUMIDEV ; read function addr for installable dev driver

588 000166D4 <res 00000020> <1> IDEV\_WADDR: resd NUMIDEV ; write function addr for installable dev driver

589 <1>

590 <1> ; 08/10/2016

591 <1> ; 07/10/2016

592 <1> ; Device Open and Access parameters

593 000166F4 <res 0000001E> <1> DEV\_ACCESS: resb NUMOFDEVICES ; bit 0 = accessable by normal users

594 <1> ; bit 1 = read access permission

595 <1> ; bit 2 = write access permission

596 <1> ; bit 3 = IOCTL permission to users

597 <1> ; bit 4 = block device if it is set

598 <1> ; bit 5 = 16 bit or 1024 byte data

599 <1> ; bit 6 = 32 bit or 2048 byte data

600 <1> ; bit 7 = installable device driver

601 00016712 <res 0000001E> <1> DEV\_R\_OWNER: resb NUMOFDEVICES ; Reading owner no (u.uid) of devices

602 00016730 <res 0000001E> <1> DEV\_R\_OPENCOUNT: resb NUMOFDEVICES ; Reading open count

603 0001674E <res 0000001E> <1> DEV\_W\_OWNER: resb NUMOFDEVICES ; Writing owner no (u.uid) of devices

604 0001676C <res 0000001E> <1> DEV\_W\_OPENCOUNT: resb NUMOFDEVICES ; Writing open count

605 0001678A <res 0000001E> <1> DEV\_DRIVER: resb NUMOFDEVICES ; device driver number (1 to 7Fh)

606 <1> ; \*if bit 7 is set (80 to FFh)

607 <1> ; \*if it is installable device driver

608 <1> ; \*index (0 to 7Fh)

609 <1> ; otherwise it is kernel device index

610 000167A8 <res 0000001E> <1> DEV\_OPENMODE: resb NUMOFDEVICES ; 1 = read mode

611 <1> ; 2 = write mode

612 <1> ; 3 = read & write

613 <1> ; 0 = not open (free)

614 000167C6 <res 00000078> <1> DEV\_NAME\_PTR: resd NUMOFDEVICES ; pointers to name addresses of drivers

615 <1> ; Address base: KDEV\_NAME+

616 <1> ; or IDEV\_NAME+

617 0001683E <res 00000078> <1> DEV\_R\_POINTER: resd NUMOFDEVICES ; reading pointer, writing pointer

618 000168B6 <res 00000078> <1> DEV\_W\_POINTER: resd NUMOFDEVICES ; sector number if block device

619 <1> ; character offset if char device

620 0001692E <res 00000002> <1> alignb 4

621 <1>

622 <1> ; 06/10/2016

623 <1> ; Open File Parameters

624 00016930 <res 00000028> <1> OF\_FCLUSTER: resd OPENFILES ; First clusters of open files

625 00016958 <res 0000000A> <1> OF\_DRIVE: resb OPENFILES ; Logical DOS drive numbers of open files

626 00016962 <res 0000000A> <1> OF\_MODE: resb OPENFILES ; Open mode (1 = read, 2 = write, 3 = r&w)

627 0001696C <res 0000000A> <1> OF\_STATUS: resb OPENFILES ; (bit 0 = read, bit 1 = write)

628 00016976 <res 0000000A> <1> OF\_OPENCOUNT: resb OPENFILES ; Open counts of open files

629 00016980 <res 00000028> <1> OF\_POINTER: resd OPENFILES ; File seek/read/write pointer

630 000169A8 <res 00000028> <1> OF\_SIZE: resd OPENFILES ; File sizes of open files (in bytes)

631 000169D0 <res 00000028> <1> OF\_DIRFCLUSTER: resd OPENFILES ; Directory First Clusters of open files

632 000169F8 <res 00000028> <1> OF\_DIRCLUSTER: resd OPENFILES ; Directory (Entry) Clusters of open files

633 00016A20 <res 00000028> <1> OF\_VOLUMEID: resd OPENFILES ; Vol ID for removable drives of open files

634 00016A48 <res 00000028> <1> OF\_CCLUSTER: resd OPENFILES ; Current clusters of open files

635 00016A70 <res 00000028> <1> OF\_CCINDEX: resd OPENFILES ; Cluster index numbers of current clusters

636 <1> ; 24/10/2016

637 00016A98 <res 00000014> <1> OF\_DIRENTRY: resw OPENFILES ; Directory entry index no. in dir cluster

638 <1> ; Sector index = entry index / 16

639 <1> ;alignb 2

640 <1>

641 00016AAC <res 00000060> <1> DTA: resd 24 ; Find First File data transfer area

642 <1>

643 <1> ; 19/12/2016

644 00016B0C <res 00000001> <1> tcallback: resb 1 ; Timer callback method flag for 'systimer'

645 00016B0D <res 00000001> <1> trtc: resb 1 ; Timer interrupt type flag for 'systimer'

646 <1> ; 20/02/2017

647 00016B0E <res 00000001> <1> no\_page\_swap: resb 1 ; Swap lock for Signal Response Byte pages

648 <1> ;;15/01/2017

649 <1> ; 02/01/2017

650 <1> ;;intflg: resb 1 ; software interrupt in progress signal

651 <1> ; (for timer interrupt)

652 <1>

653 00016B0F <res 00000001> <1> alignb 4

654 <1> ; 13/04/2017

655 00016B10 <res 0000001E> <1> DEV\_INTR: resb NUMOFDEVICES ; Device Interrupt (IRQ) number + 1

656 <1> ; (0= not available, 1= IRQ 0, 16= IRQ 15)

657 00016B2E <res 00000040> <1> DEV\_INT\_HNDLR: resd 16 ; Device Interrupt Handler addr, if > 0

658 <1>

659 <1>

660 <1> ;alignb 4

661 <1>

662 <1> ; 26/02/2017 ; IRQ Callback parameters ('syscalbac')

663 <1> ;Index: ; 0 to 8

664 <1> ; 0 = IRQ3, 1 = IRQ4, 2 = IRQ5, 3 = IRQ7

665 <1> ; 4 = IRQ9, 5 = IRQ10, 6 = IRQ11, 7 = IRQ12, 8 = IRQ13

666 00016B6E <res 00000009> <1> IRQ.owner: resb 9 ; owner, 0 = free, >0 = [u.uno]

667 00016B77 <res 00000009> <1> IRQ.dev: resb 9 ; 0 = default/kernel, >0 = device number

668 00016B80 <res 00000009> <1> IRQ.method: resb 9 ; 0 = Signal Response Byte, 1 = Callback

669 00016B89 <res 00000009> <1> IRQ.srb: resb 9 ; Signal Response/Return Byte value

670 00016B92 <res 00000024> <1> IRQ.addr: resd 9 ; Rignal Response Byte address (physical)

671 <1> ; or Callback service address (virtual)

672 <1> ; 28/02/2017

673 00016BB6 <res 00000004> <1> IRQ\_cr3: resd 1 ; for saving cr3 register in IRQ handler

674 00016BBA <res 00000001> <1> IRQnum: resb 1 ; IRQ number for IRQ handler (trdosk8.s)

675 <1>

676 <1> ; 10/04/2017

677 <1> ; 03/04/2017

678 <1> ; UNINITIALIZED AUDIO DATA

679 00016BBB <res 00000001> <1> alignb 4

680 00016BBC <res 00000001> <1> audio\_pci: resb 1

681 00016BBD <res 00000001> <1> audio\_device: resb 1

682 00016BBE <res 00000001> <1> audio\_mode: resb 1

683 00016BBF <res 00000001> <1> audio\_intr: resb 1

684 00016BC0 <res 00000001> <1> audio\_busy: resb 1 ; Busy flag for audio irq ; 21/04/2017

685 00016BC1 <res 00000001> <1> audio\_reserved: resb 1

686 00016BC2 <res 00000002> <1> audio\_io\_base: resw 1 ; Base I/O address of audio device

687 00016BC4 <res 00000004> <1> audio\_dev\_id: resd 1 ; BUS/DEV/FN ; 00000000BBBBBBBBDDDDDFFF00000000

688 00016BC8 <res 00000004> <1> audio\_vendor: resd 1

689 00016BCC <res 00000004> <1> audio\_stats\_cmd: resd 1

690 <1> ;

691 00016BD0 <res 00000004> <1> audio\_buffer: resd 1 ; virtual address of user's audio buffer

692 00016BD4 <res 00000004> <1> audio\_p\_buffer: resd 1 ; Physical address of user's audio buffer

693 00016BD8 <res 00000004> <1> audio\_buff\_size: resd 1 ; user's audio buffer size (half buffer size)

694 00016BDC <res 00000004> <1> audio\_dma\_buff: resd 1 ; dma buffer address

695 00016BE0 <res 00000004> <1> audio\_dmabuff\_size: resd 1 ; dma buffer size (2 \* half buffer size)

696 00016BE4 <res 00000001> <1> audio\_flag: resb 1 ; dma buffer flag (1st half = 0, 2nd half = 1)

697 00016BE5 <res 00000001> <1> audio\_user: resb 1 ; user number of the owner

698 00016BE6 <res 00000001> <1> audio\_cb\_mode: resb 1 ; 0 = signal response byte method

699 <1> ; 1 = callback method

700 <1> ; 2 = s.r.b. method with auto increment

701 00016BE7 <res 00000001> <1> audio\_srb: resb 1 ; signal response byte value

702 00016BE8 <res 00000004> <1> audio\_cb\_addr: resd 1 ; callback service address or s.r.b. address

703 <1> ; (s.r.b. addr is physical, cbs addr is virtual)

704 <1>

705 00016BEC <res 00000001> <1> audio\_bps: resb 1 ; selected mode: 8 bit, 16 bit

706 00016BED <res 00000001> <1> audio\_stmo: resb 1 ; selected mode: mono /stereo

707 00016BEE <res 00000002> <1> audio\_freq: resw 1 ; sampling rate

708 <1>

709 <1> ; 21/04/2017

710 00016BF0 <res 00000001> <1> audio\_play\_cmd: resb 1 ; Play/Stop command (1 = play, 0 = stop)

711 <1> audio\_civ: ; 28/05/2017 ; Current Buffer Index (AC'97)

712 00016BF1 <res 00000001> <1> audio\_flag\_eol: resb 1 ; End of Link status (vt8233, EOL/FLAG)

713 <1>

714 <1> audio\_master\_volume:

715 00016BF2 <res 00000001> <1> audio\_master\_volume\_l: resb 1 ; sound volume (lineout) left channel

716 00016BF3 <res 00000001> <1> audio\_master\_volume\_r: resb 1 ; sound volume (lineout) right channel

717 <1>

718 <1> alignb 4

719 <1> ; 28/05/2017

720 <1> ; AC'97 Audio Controller Base Adress Registers

721 00016BF4 <res 00000002> <1> NAMBAR: resw 1 ; Native Audio Mixer Base Address

722 00016BF6 <res 00000002> <1> NABMBAR: resw 1 ; Native Audio Bus Mastering Base Address

723 <1>

724 <1> ;alignb 4

725 <1> ; 21/04/2017

726 00016BF8 <res 00000400> <1> audio\_bdl\_buff: resd 32\*8 ; VT8233 (AC97) BDL Buffer Size

727 <1> ; 12/05/2017

728 00016FF8 <res 00000004> <1> base\_addr: resd 1 ; 'direct\_memory\_access' (memory.s)

729 <1>

730 <1> ; 28/08/2017

731 <1> ; 20/08/2017

732 00016FFC <res 00000001> <1> resb 1 ;

733 00016FFD <res 00000001> <1> dma\_user: resb 1 ; user number for sysdma

734 00016FFE <res 00000001> <1> dma\_channel: resb 1 ; dma channel for sysdma

735 00016FFF <res 00000001> <1> dma\_mode: resb 1 ; dma mode for sysdma

736 00017000 <res 00000004> <1> dma\_addr: resd 1 ; dma buffer physical addr for sysdma

737 00017004 <res 00000004> <1> dma\_size: resd 1 ; dma buffer size (in bytes) for sysdma

738 00017008 <res 00000004> <1> dma\_start: resd 1 ; dma start address for sysdma

739 0001700C <res 00000004> <1> dma\_count: resd 1 ; dma count (in bytes) for sysdma

740 <1>

741 00017010 <res 00008FF0> <1> alignb 65536

742 <1> ; 09/08/2017

743 <1> ; 12/05/2017

744 00020000 <res 00010000> <1> sb16\_dma\_buffer: resb 65536 ; DMA buffer for sb16 audio playing.

2830 ; 24/01/2016

2831 %include 'ubss.s' ; UNINITIALIZED KERNEL (USER) DATA

1 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2 <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.0) - UNINITIALIZED USER DATA : ubss.s

3 <1> ; ----------------------------------------------------------------------------

4 <1> ; Last Update: 28/02/2017

5 <1> ; ----------------------------------------------------------------------------

6 <1> ; Beginning: 24/01/2016

7 <1> ; ----------------------------------------------------------------------------

8 <1> ; Assembler: NASM version 2.11 (trdos386.s)

9 <1> ; ----------------------------------------------------------------------------

10 <1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan

11 <1> ; ux.s (04/12/2015)

12 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

13 <1>

14 <1> ; Retro UNIX 386 v1 Kernel - ux.s

15 <1> ; Last Modification: 04/12/2015

16 <1> ;

17 <1> ; ///////// RETRO UNIX 386 V1 SYSTEM DEFINITIONS ///////////////

18 <1> ; (Modified from

19 <1> ; Retro UNIX 8086 v1 system definitions in 'UNIX.ASM', 01/09/2014)

20 <1> ; ((UNIX.ASM (RETRO UNIX 8086 V1 Kernel), 11/03/2013 - 01/09/2014))

21 <1> ; ----------------------------------------------------------------------------

22 <1> ; Derived from UNIX Operating System (v1.0 for PDP-11)

23 <1> ; (Original) Source Code by Ken Thompson (1971-1972)

24 <1> ; <Bell Laboratories (17/3/1972)>

25 <1> ; <Preliminary Release of UNIX Implementation Document>

26 <1> ; (Section E10 (17/3/1972) - ux.s)

27 <1> ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

28 <1>

29 <1> alignb 2

30 <1>

31 <1> inode:

32 <1> ; 11/03/2013.

33 <1> ;Derived from UNIX v1 source code 'inode' structure (ux).

34 <1> ;i.

35 <1>

36 00030000 <res 00000002> <1> i.flgs: resw 1

37 00030002 <res 00000001> <1> i.nlks: resb 1

38 00030003 <res 00000001> <1> i.uid: resb 1

39 <1> ;i.size: resw 1 ; size

40 00030004 <res 00000002> <1> resw 1 ; 29/04/2016

41 00030006 <res 00000010> <1> i.dskp: resw 8 ; 16 bytes

42 00030016 <res 00000004> <1> i.ctim: resd 1

43 0003001A <res 00000004> <1> i.mtim: resd 1

44 0003001E <res 00000002> <1> i.rsvd: resw 1 ; Reserved (ZERO/Undefined word for UNIX v1.)

45 <1>

46 <1> I\_SIZE equ $ - inode

47 <1>

48 <1> process:

49 <1> ; 19/12/2016

50 <1> ; 21/05/2016

51 <1> ; 19/05/2016 - TRDOS 386 (TRDOS v2.0)

52 <1> ; 06/05/2015 - Retro UNIX 386 v1

53 <1> ; 11/03/2013 - 05/02/2014 (Retro UNIX 8086 v1)

54 <1> ;Derived from UNIX v1 source code 'proc' structure (ux).

55 <1> ;p.

56 <1>

57 00030020 <res 00000020> <1> p.pid: resw nproc

58 00030040 <res 00000020> <1> p.ppid: resw nproc

59 00030060 <res 00000020> <1> p.break: resw nproc

60 00030080 <res 00000010> <1> p.ttyc: resb nproc ; console tty in Retro UNIX 8086 v1.

61 00030090 <res 00000010> <1> p.waitc: resb nproc ; waiting channel in Retro UNIX 8086 v1.

62 000300A0 <res 00000010> <1> p.link: resb nproc

63 000300B0 <res 00000010> <1> p.stat: resb nproc

64 <1>

65 <1> ; 06/05/2015 (Retro UNIX 386 v1 feature only !)

66 000300C0 <res 00000040> <1> p.upage: resd nproc ; Physical address of the process's

67 <1> ; 'user' structure

68 <1> ; 21/05/2016

69 <1> ; 19/05/2016 (TRDOS 386 feature only!)

70 00030100 <res 00000010> <1> p.timer: resb nproc ; number of timer events of the processs

71 <1>

72 <1> ; 19/12/2016

73 00030110 <res 00000040> <1> p.tcb: resd nproc ; timer callback service address (if > 0)

74 <1>

75 <1> P\_SIZE equ $ - process

76 <1>

77 <1> ; fsp table (original UNIX v1)

78 <1> ;

79 <1> ;Entry

80 <1> ; 15 0

81 <1> ; 1 |---|---------------------------------------|

82 <1> ; |r/w| i-number of open file |

83 <1> ; |---|---------------------------------------|

84 <1> ; | device number |

85 <1> ; |-------------------------------------------|

86 <1> ; (\*) | offset pointer, i.e., r/w pointer to file |

87 <1> ; |-------------------------------------------|

88 <1> ; | flag that says | number of processes |

89 <1> ; | file deleted | that have file open |

90 <1> ; |-------------------------------------------|

91 <1> ; 2 | |

92 <1> ; |-------------------------------------------|

93 <1> ; | |

94 <1> ; |-------------------------------------------|

95 <1> ; | |

96 <1> ; |-------------------------------------------|

97 <1> ; | |

98 <1> ; |-------------------------------------------|

99 <1> ; 3 | |

100 <1> ; | |

101 <1> ;

102 <1> ; (\*) Retro UNIX 386 v1 modification: 32 bit offset pointer

103 <1>

104 <1>

105 <1> ; 15/04/2015

106 00030150 <res 000001F4> <1> fsp: resb nfiles \* 10 ; 11/05/2015 (8 -> 10)

107 00030344 <res 00000002> <1> idev: resw 1 ; device number is 1 byte in Retro UNIX 8086 v1 !

108 00030346 <res 00000002> <1> cdev: resw 1 ; device number is 1 byte in Retro UNIX 8086 v1 !

109 <1> ; 18/05/2015

110 <1> ; 26/04/2013 device/drive parameters (Retro UNIX 8086 v1 feature only!)

111 <1> ; 'UNIX' device numbers (as in 'cdev' and 'u.cdrv')

112 <1> ; 0 -> root device (which has Retro UNIX 8086 v1 file system)

113 <1> ; 1 -> mounted device (which has Retro UNIX 8086 v1 file system)

114 <1> ; 'Retro UNIX 8086 v1' device numbers: (for disk I/O procedures)

115 <1> ; 0 -> fd0 (physical drive, floppy disk 1), physical drive number = 0

116 <1> ; 1 -> fd1 (physical drive, floppy disk 2), physical drive number = 1

117 <1> ; 2 -> hd0 (physical drive, hard disk 1), physical drive number = 80h

118 <1> ; 3 -> hd1 (physical drive, hard disk 2), physical drive number = 81h

119 <1> ; 4 -> hd2 (physical drive, hard disk 3), physical drive number = 82h

120 <1> ; 5 -> hd3 (physical drive, hard disk 4), physical drive number = 83h

121 00030348 <res 00000001> <1> rdev: resb 1 ; root device number ; Retro UNIX 8086 v1 feature only!

122 <1> ; as above, for physical drives numbers in following table

123 00030349 <res 00000001> <1> mdev: resb 1 ; mounted device number ; Retro UNIX 8086 v1 feature only!

124 <1> ; 15/04/2015

125 0003034A <res 00000001> <1> active: resb 1

126 0003034B <res 00000001> <1> resb 1 ; 09/06/2015

127 0003034C <res 00000002> <1> mnti: resw 1

128 0003034E <res 00000002> <1> mpid: resw 1

129 00030350 <res 00000002> <1> rootdir: resw 1

130 <1>

131 <1> ; 21/05/2016 - TRDOS 386 (TRDOS v2.0) - priority levels, 3 run queues

132 <1> runq:

133 00030352 <res 00000002> <1> runq\_event: resw 1 ; high priority, 'run for event' ; 2

134 00030354 <res 00000002> <1> runq\_normal: resw 1 ; normal/regular priority, 'run as reqular' ; 1

135 00030356 <res 00000002> <1> runq\_background: resw 1 ; low priority, 'run on background' ; 0

136 <1> ;

137 00030358 <res 00000001> <1> imod: resb 1

138 00030359 <res 00000001> <1> smod: resb 1

139 0003035A <res 00000001> <1> mmod: resb 1

140 0003035B <res 00000001> <1> sysflg: resb 1

141 <1>

142 <1> alignb 4

143 <1>

144 <1> user:

145 <1> ; 13/01/2017

146 <1> ; 19/12/2016

147 <1> ; 21/05/2016 - TRDOS 386 (TRDOS v2.0)

148 <1> ; [u.pri] usage method modification

149 <1> ; 04/12/2015

150 <1> ; 18/10/2015

151 <1> ; 12/10/2015

152 <1> ; 21/09/2015

153 <1> ; 24/07/2015

154 <1> ; 16/06/2015

155 <1> ; 09/06/2015

156 <1> ; 11/05/2015

157 <1> ; 16/04/2015 (Retro UNIX 386 v1 - 32 bit modifications)

158 <1> ; 10/10/2013

159 <1> ; 11/03/2013.

160 <1> ;Derived from UNIX v1 source code 'user' structure (ux).

161 <1> ;u.

162 <1>

163 0003035C <res 00000004> <1> u.sp: resd 1 ; esp (kernel stack at the beginning of 'sysent')

164 00030360 <res 00000004> <1> u.usp: resd 1 ; esp (kernel stack points to user's registers)

165 00030364 <res 00000004> <1> u.r0: resd 1 ; eax

166 00030368 <res 00000002> <1> u.cdir: resw 1

167 0003036A <res 0000000A> <1> u.fp: resb 10

168 00030374 <res 00000004> <1> u.fofp: resd 1

169 00030378 <res 00000004> <1> u.dirp: resd 1

170 0003037C <res 00000004> <1> u.namep: resd 1

171 00030380 <res 00000004> <1> u.off: resd 1

172 00030384 <res 00000004> <1> u.base: resd 1

173 00030388 <res 00000004> <1> u.count: resd 1

174 0003038C <res 00000004> <1> u.nread: resd 1

175 00030390 <res 00000004> <1> u.break: resd 1 ; break

176 00030394 <res 00000002> <1> u.ttyp: resw 1

177 <1> ; 10/01/2017 (TRDOS 386, relocation and dword alignment)

178 <1> ; tty number (rtty, rcvt, wtty)

179 00030396 <res 00000001> <1> u.ttyn: resb 1 ; 28/07/2013 - Retro Unix 8086 v1 feature only !

180 00030397 <res 00000001> <1> u.resb: resb 1 ; 10/01/2017 (TRDOS 386, temporary)

181 00030398 <res 00000010> <1> u.dirbuf: resb 16 ; 04/12/2015 (10 -> 16)

182 <1> ;u.pri: resw 1 ; 14/02/2014

183 000303A8 <res 00000001> <1> u.quant: resb 1 ; Retro UNIX 8086 v1 Feature only ! (uquant)

184 000303A9 <res 00000001> <1> u.pri: resb 1 ; Modification: 21/05/2016 (priority levels: 0, 1, 2)

185 000303AA <res 00000002> <1> u.intr: resw 1

186 000303AC <res 00000002> <1> u.quit: resw 1

187 <1> ;u.emt: resw 1 ; 10/10/2013

188 <1> ;u.ilgins: resw 1 ; 10/01/2017

189 000303AE <res 00000002> <1> u.cdrv: resw 1 ; cdev

190 000303B0 <res 00000001> <1> u.uid: resb 1 ; uid

191 000303B1 <res 00000001> <1> u.ruid: resb 1

192 000303B2 <res 00000001> <1> u.bsys: resb 1

193 000303B3 <res 00000001> <1> u.uno: resb 1

194 000303B4 <res 00000004> <1> u.upage: resd 1 ; 16/04/2015 - Retro Unix 386 v1 feature only !

195 000303B8 <res 00000004> <1> u.pgdir: resd 1 ; 09/03/2015 (page dir addr of process)

196 000303BC <res 00000004> <1> u.ppgdir: resd 1 ; 06/05/2015 (page dir addr of the parent process)

197 000303C0 <res 00000004> <1> u.pbase: resd 1 ; 20/05/2015 (physical base/transfer address)

198 000303C4 <res 00000002> <1> u.pcount: resw 1 ; 20/05/2015 (byte -transfer- count for page)

199 <1> ;u.pncount: resw 1

200 <1> ; 16/06/2015 (byte -transfer- count for page, 'namei', 'mkdir')

201 <1> ;u.pnbase: resd 1

202 <1> ; 16/06/2015 (physical base/transfer address, 'namei', 'mkdir')

203 <1> ; 09/06/2015

204 000303C6 <res 00000001> <1> u.kcall: resb 1 ; The caller is 'namei' (dskr) or 'mkdir' (dskw) sign

205 000303C7 <res 00000001> <1> u.brwdev: resb 1 ; Block device number for direct I/O (bread & bwrite)

206 <1> ; 24/07/2015 - 24/06/2015

207 <1> ;u.args: resd 1 ; arguments list (line) offset from start of [u.upage]

208 <1> ; (arg list/line is from offset [u.args] to 4096 in [u.upage])

209 <1> ; ([u.args] points to argument count -argc- address offset)

210 <1> ; 24/06/2015

211 <1> ;u.core: resd 1 ; physical start address of user's memory space (for sys exec)

212 <1> ;u.ecore: resd 1 ; physical end address of user's memory space (for sys exec)

213 <1> ; last error number

214 000303C8 <res 00000004> <1> u.error: resd 1 ; 28/07/2013 - 09/03/2015

215 <1> ; Retro UNIX 8086/386 v1 feature only!

216 <1> ; 21/09/2015 (debugging - page fault analyze)

217 000303CC <res 00000004> <1> u.pfcount: resd 1 ; page fault count for (this) process (for sys geterr)

218 <1> ; 19/12/2016 (TRDOS 386)

219 000303D0 <res 00000004> <1> u.tcb: resd 1 ; Timer callback address/flag which will be used by timer int

220 <1> ; 13/01/2017 (TRDOS 386)

221 000303D4 <res 00000001> <1> u.t\_lock: resb 1 ; Timer interrupt (callback) lock (unlocked by 'sysrele')

222 000303D5 <res 00000001> <1> u.t\_mode: resb 1 ; running mode during timer interrupt (0= system, 0FFh= user)

223 <1> ; 26/02/2017 (TRDOS 386)

224 000303D6 <res 00000001> <1> u.irqc: resb 1 ; Count of IRQ callback services (IRQs in use)

225 <1> ; 28/02/2017 (TRDOS 386)

226 000303D7 <res 00000001> <1> u.irqwait: resb 1 ; IRQ waiting for callback service flag (IRQ number, If > 0)

227 000303D8 <res 00000001> <1> u.r\_lock: resb 1 ; 'IRQ callback service is in progress' flag (IRQ lock)

228 000303D9 <res 00000001> <1> u.r\_mode: resb 1 ; running mode during hadware interrupt

229 <1> ; 27/02/2017 (TRDOS 386)

230 000303DA <res 00000001> <1> u.fpsave: resb 1 ; TRDOS 386, 'save/restore FPU registers' flag

231 000303DB <res 00000001> <1> alignb 4

232 000303DC <res 0000005E> <1> u.fpregs: resb 94 ; 94 byte area for saving and restoring FPU registers

233 <1>

234 0003043A <res 00000002> <1> alignb 4

235 <1>

236 <1> U\_SIZE equ $ - user

237 <1>

238 <1> ; 18/10/2015 - Retro UNIX 386 v1 (local variables for 'namei' and 'sysexec')

239 0003043C <res 00000004> <1> pcore: resd 1 ; physical start address of user's memory space (for sys exec)

240 00030440 <res 00000004> <1> ecore: resd 1 ; physical address of user's stack/last page (for sys exec)

241 00030444 <res 00000004> <1> nbase: resd 1 ; physical base address for 'namei' & 'sysexec'

242 00030448 <res 00000002> <1> ncount: resw 1 ; remain byte count in page for 'namei' & 'sysexec'

243 0003044A <res 00000002> <1> argc: resw 1 ; argument count for 'sysexec'

244 0003044C <res 00000004> <1> argv: resd 1 ; argument list (recent) address for 'sysexec'

245 <1>

246 <1> ; 03/06/2015 - Retro UNIX 386 v1 Beginning

247 <1> ; 07/04/2013 - 31/07/2013 - Retro UNIX 8086 v1

248 00030450 <res 00000001> <1> rw: resb 1 ;; Read/Write sign (iget)

249 <1>

250 <1> ;alignb 4

251 <1>

252 <1> ; 24/04/2016

253 00030451 <res 00000004> <1> ii: resd 1 ; first cluster of the program file

254 00030455 <res 00000004> <1> i.size: resd 1 ; size of the program file

2832

2833 00030459 <res 00000003> alignb 4

2834

2835 ; 23/05/2016 (TRDOS 386)

2836 ; 14/10/2015 (Retro UNIX 386 v1, 'unix386.s')

2837 0003045C <res 00000004> cr3reg: resd 1 ; cr3 register content at the beginning of the timer

2838 ; (or RTC) interrupt handler.

2839

2840 ; 10/12/2016 (callback)

2841 ; 10/06/2016

2842 ; 19/05/2016

2843 ; 18/05/2016 - TRDOS 386 feature only !

2844 00030460 <res 00000100> timer\_set: resd 16\*4 ; 256 bytes memory space for 16 timer events

2845 ; Timer Event Structure: (max. 16 timer events, 16\*16 bytes)

2846 ; Owner: resb 1 ; 0 = free

2847 ; ;>0 = process number (u.uno)

2848 ; Callback: resb 1 ; 0 = response byte address (phy)

2849 ; 1 = callback address (virtual)

2850 ; Interrupt: resb 1 ; 0 = Timer interrupt (or none)

2851 ; ; 1 = Real Time Clock interrupt

2852 ; Response: resb 1 ; 0 to 255, signal return value

2853 ; Count Limit: resd 1 ; count of ticks (total/set)

2854 ; Current Count: resd 1 ; count of ticks (current)

2855 ; Response Addr: resd 1 ; response byte (pointer) address

2856 ; ; (or callback -user service- address)

2857

2858 ;; Memory (swap) Data (11/03/2015)

2859 ; 09/03/2015

2860 00030560 <res 00000002> swpq\_count: resw 1 ; count of pages on the swap queue

2861 00030562 <res 00000004> swp\_drv: resd 1 ; logical drive description table address of the swap drive/disk

2862 00030566 <res 00000004> swpd\_size: resd 1 ; size of swap drive/disk (volume) in sectors (512 bytes).

2863 0003056A <res 00000004> swpd\_free: resd 1 ; free page blocks (4096 bytes) on swap disk/drive (logical)

2864 0003056E <res 00000004> swpd\_next: resd 1 ; next free page block

2865 00030572 <res 00000004> swpd\_last: resd 1 ; last swap page block

2866

2867 00030576 <res 00000002> alignb 4

2868

2869 ; 10/07/2015

2870 ; 28/08/2014

2871 00030578 <res 00000004> error\_code: resd 1

2872 ; 29/08/2014

2873 0003057C <res 00000004> FaultOffset: resd 1

2874 ; 21/09/2015

2875 00030580 <res 00000004> PF\_Count: resd 1 ; total page fault count

2876 ; (for debugging - page fault analyze)

2877 ; 'page\_fault\_handler' (memory.inc)

2878 ; 'sysgeterr' (u9.s)

2879

2880 ; 29/04/2016 (TRDOS 386 = TRDOS v2.0)

2881 ; 22/08/2015 (Retro UNIX 386 v1)

2882 buffer:

2883 00030584 <res 00000008> resb 8

2884 readi\_buffer:

2885 0003058C <res 00000200> resb 512

2886 0003078C <res 00000008> resb 8

2887 writei\_buffer:

2888 00030794 <res 00000200> resb 512

2889 ; 24/10/2016

2890 00030994 <res 00000008> resb 8

2891 rw\_buffer:

2892 0003099C <res 00000800> resb 2048 ; general purposed, r/w sector buffer

2893

2894 bss\_end:

2895

2896 ; 27/12/2013

2897 \_end: ; end of kernel code