Pirate slayer boot

The loader has two visible parts; A file called "ea" and one called "loader" where the ea one is merely a loader of the actual loader.

Loaded with ",8" is loads to basic and displays a message that will load itself with ",8,1". Loading from ,8,1 is loads to $00F7, over the stack and ends at $0246. It autostarts by having a block of $02 in the higher parts of the stack so at the loaders RTS, it forks off to $0203.

Logic of the "EA" file"

Full of illegal opcodes.

.C:0203 AF 72 01 LAX $0172 ; Load A and X with $4C

.C:0206 BD C5 01 LDA $01C5,X ; Load A with $8A from $0211

.C:0209 AA TAX ; Transfer A to X

.C:020a 8F 3D 03 SAX $033D ; Store A and X to $033D

.C:020d BD 9C 01 LDA $019C,X ; Load A with $FF from $0226

.C:0210 0C 8A 45 NOOP $458A ; NOP over the $8A

.C:0213 AA TAX ; Transfer A to X

.C:0214 8F 3E 03 SAX $033E ; Store A to X

.C:0217 BD 3C 01 LDA $013C,X ; Load $4C from $023B

.C:021a 8D 3C 03 STA $033C ;

.C:021d 20 3C 03 JSR $033C ; The above generated a JMP $FF8A (Restore vectors)

.C:0220 AF E0 01 LAX $01E0 ; Load A and X with $02

.C:0223 CA DEX ; Make X $1

.C:0224 8A TXA ; And save to A

.C:0225 0C FF AF NOOP $AFFF ; NOP over the two bytes

.C:0228 8F 20 D0 SAX $D020 ; Save to $d020

.C:022b 8F 21 D0 SAX $D021 ; and to $d021

.C:022e AF 00 02 LAX $0200 ; Load A and X with $93

.C:0231 20 CA F1 JSR $F1CA ; Print byte ($93 is clear screen)

.C:0234 A9 01 LDA #$01 ;

.C:0236 8F 86 02 SAX $0286 ; Set black as cursor colour

.C:0239 87 9D SAX $9D ; Suppres all kernal messages

.C:023b 4C 3D 01 JMP $013D ; Jump to the loader part

.C:023e 55 49 EOR $49,X

.C:0240 4C 4F 41 JMP $414F ; The text "loader"

.C:0243 44 45 NOOP $45

.C:0245 52 JAM

;=====================================

.C:013d AF 27 02 LAX $0227 ; Load A and X with $AF

.C:0140 9A TXS ; Set stack pointer

.C:0141 A9 0F LDA #$0F

.C:0143 A8 TAY

.C:0144 A2 08 LDX #$08 ; Device

.C:0146 20 BA FF JSR $FFBA ; Open Command channel

.C:0149 A9 02 LDA #$02

.C:014b A2 3E LDX #$3E

.C:014d A0 02 LDY #$02

.C:014f 20 BD FF JSR $FFBD ; Point to two bytes at $023E ($55, $49)

.C:0152 20 C0 FF JSR $FFC0 ; Open

.C:0155 A2 0F LDX #$0F

.C:0157 20 C9 FF JSR $FFC9 ; Channel

.C:015a A9 49 LDA #$49

.C:015c 20 D2 FF JSR $FFD2 ; Output $49 (I for init)

.C:015f 20 CC FF JSR $FFCC ; Reset normal device

.C:0162 A9 0F LDA #$0F

.C:0164 20 C3 FF JSR $FFC3 ; Close

.C:0167 A9 06 LDA #$06

.C:0169 A2 40 LDX #$40

.C:016b A0 02 LDY #$02

.C:016d 20 BD FF JSR $FFBD ; Set filename to the one at $0240 (6 bytes)

.C:0170 A9 00 LDA #$00

.C:0172 4C D5 FF JMP $FFD5 ; Load the loader file -

Jumps then to what's loaded into the stack at the addes of the stack pointer. SP here is $AF.

;==============================

Logic of the "loader" file

Loads to $FFFA

- Still don't know how the file is started... And also not how the file gets where it gets.

I believe it starts at $0034 (At stack: $01b0)

.C:0034 78 SEI

.C:0035 A7 03 LAX $03

.C:0037 87 01 SAX $01

.C:0039 60 RTS ; -> $01e2

.C:003a A7 04 LAX $04

.C:003c 87 01 SAX $01

.C:003e 60 RTS

.C:003f 4C FF FF JMP $FFFF

Then to $01E2 (At stack: $01b2)

.C:01e2 AF 1C F2 LAX $F21C ; $00

.C:01e5 0C 4C 35 NOOP $354C ; Mask a jump for confusion?

.C:01e8 BD B6 F2 LDA $F2B6,X ; Kill carts

.C:01eb 49 34 EOR #$34 ; Mask

.C:01ed 44 AD NOOP $AD ; Mask a store for confusion?

.C:01ef 9D 00 80 STA $8000,X ; Kill carts?

.C:01f2 CA DEX

.C:01f3 D0 F3 BNE $01E8

Q: Is the code after $8014 needed or does it just copy a full page for no reason?

.C:01f5 68 PLA ; A is $22 - at Stack $01B4

.C:01f6 60 RTS ; -> $0210

The result of the $01E2 loading from $F2B6 and masking with #$34 is this:

.C:8009 78 SEI

.C:800a A9 00 LDA #$00

.C:800c 49 02 EOR #$02

.C:800e 8D 20 D0 STA $D020

.C:8011 4C 0C 80 JMP $800C

Then to $0210 (At stack: $01b5)

.C:01f7 68 PLA

.C:01f8 C8 INY

.C:01f9 D0 02 BNE $01FD

.C:01fb E6 30 INC $30

.C:01fd C0 1C CPY #$1C

.C:01ff D0 07 BNE $0208

.C:0201 A5 30 LDA $30

.C:0203 C9 F2 CMP #$F2

.C:0205 D0 01 BNE $0208

.C:0207 60 RTS

.C:0208 AE 1E F2 LDX $F21E

.C:020b 8A TXA

.C:020c 9B 00 C3 SHS $C300,Y

.C:020f 60 RTS

.C:0210 C6 31 DEC $31 ; $31 is $da when entering the first time

.C:0212 C6 02 DEC $02 ; $95 when entering

.C:0214 AF 1C F2 LAX $F21C ; $00 when entering

.C:0217 5D B0 01 EOR $01B0,X

>C:01b0 33 00 e1 01 22 0f 02 8b f2 64 74 f2 06 04 00 aa f2 f5 1c f2 f6 01 c0 67 f1 09 f3 ff 3..."....dt............g....

>C:01cc fe 21 f2 39 00 3e 00 33 00 90 f2 63 19 ff 22 02 09 0c a9 35 4c c2 .!.9.>.3...c.."....5L.....L5

.C:021a E8 INX

.C:021b E0 32 CPX #$32

.C:021d D0 F8 BNE $0217

.C:021f 8D 32 00 STA $0032 ; A is $4b when we get here

.C:0222 60 RTS

Here it goes to $F28C (At stack: $01b7)

.C:f28c A0 00 LDY #$00

.C:f28e 68 PLA ; A turns #$64 (At stack: $01b9)

.C:f28f 60 RTS

Here it goes to $F275 (At stack: $01ba)

.C:f275 68 PLA ; A turns #$06 (At stack: $01bc)

.C:f276 B1 2F LDA ($2F),Y ; $EF00,0 - #$6D

.C:f278 60 RTS

Goes to $0005 (At stack: $01bd)

.C:0005 38 SEC

.C:0006 E5 02 SBC $02 ; A was #$6D and subtracted with #$94, and turned #$D9

.C:0008 60 RTS

Goes to $F2AB (At stack: $01bf)

.C:f2ab 04 69 NOOP $69

.C:f2ad BA TSX ; $C0

.C:f2ae E8 INX

.C:f2af 9A TXS ; $C1 - skips #$F5 which is at $01c1

.C:f2b0 45 31 EOR $31

.C:f2b2 4C 5A F2 JMP $F25A

.C:f25a 8D 54 A0 STA $A054 ; A is #$00 (Just storing away A or checking that no Basic ROM is switched in?)

.C:f25d 04 8D NOOP $8D

.C:f25f A9 03 LDA #$03

.C:f261 8D 00 01 STA $0100

.C:f264 A9 08 LDA #$08

.C:f266 AA TAX

.C:f267 8F 01 01 SAX $0101 ; So $0100 has #$03 and $0101 has #$08

.C:f26a AF 54 A0 LAX $A054

.C:f26d 60 RTS

Goes to $F21D (At stack: $01c2)

.C:f21d 44 B9 NOOP $B9 ; A is #$00

.C:f21f 91 2F STA ($2F),Y ; 42f points to $EF00

.C:f221 60 RTS

Goes to $01F7 (At stack: $01c4)

.C:01f7 68 PLA ; A is #$C0 from $01C6 on the stack

.C:01f8 C8 INY ; Y is #$00->#$01 here

.C:01f9 D0 02 BNE $01FD

.C:01fb E6 30 INC $30

.C:01fd C0 1C CPY #$1C

.C:01ff D0 07 BNE $0208 ; Take the branch

.C:0201 A5 30 LDA $30

.C:0203 C9 F2 CMP #$F2

.C:0205 D0 01 BNE $0208

.C:0207 60 RTS ; <- This is where the shit loop ends! A:F2 X:60 Y:1C SP:c6

.C:0208 AE 1E F2 LDX $F21E ; #$b9

.C:020b 8A TXA

.C:020c 9B 00 C3 SHS $C300,Y ; A:B9 X:B9 Y:01 SP:b9

.C:020f 60 RTS

SHS is also called TAS:

TAS {adr} = stores A&X into S and A&X&H into {adr}

Back to $F275 as per above. At $01f8, Y turns #$02 and keep rolling. The loop ends when Y is #$1C in at $01fd and then $30 incs until #$F2

Goes to $F168

.C:f168 A0 00 LDY #$00

.C:f16a AD 11 D0 LDA $D011

.C:f16d 29 EF AND #$EF

.C:f16f 8D 11 D0 STA $D011

.C:f172 B9 00 EF LDA $EF00,Y

.C:f175 49 FF EOR #$FF

.C:f177 99 00 E0 STA $E000,Y

.C:f17a B9 00 F0 LDA $F000,Y

.C:f17d 49 FF EOR #$FF

.C:f17f 99 00 E1 STA $E100,Y

.C:f182 B9 00 F1 LDA $F100,Y

.C:f185 49 FF EOR #$FF

.C:f187 99 00 E2 STA $E200,Y

.C:f18a A9 00 LDA #$00

.C:f18c 99 00 E4 STA $E400,Y

.C:f18f 99 00 E5 STA $E500,Y

.C:f192 99 00 E6 STA $E600,Y

.C:f195 99 00 E7 STA $E700,Y

.C:f198 A9 01 LDA #$01

.C:f19a 99 00 D8 STA $D800,Y

.C:f19d 99 00 D9 STA $D900,Y

.C:f1a0 99 00 DA STA $DA00,Y

.C:f1a3 99 00 DB STA $DB00,Y

.C:f1a6 C8 INY

.C:f1a7 D0 C9 BNE $F172

After this, $E000 to $E300 is populated with the inverse of what was at $EF00 to $F200.

Clear 4E400 and set $01 in the colour background.

.C:f1a9 78 SEI

.C:f1aa A2 07 LDX #$07

.C:f1ac A0 16 LDY #$16

.C:f1ae B9 B0 ED LDA $EDB0,Y

.C:f1b1 99 48 E5 STA $E548,Y

.C:f1b4 88 DEY

.C:f1b5 10 F7 BPL $F1AE

.C:f1b7 AD AF F1 LDA $F1AF

.C:f1ba 18 CLC

.C:f1bb 69 17 ADC #$17

.C:f1bd 8D AF F1 STA $F1AF

.C:f1c0 AD B0 F1 LDA $F1B0

.C:f1c3 69 00 ADC #$00

.C:f1c5 8D B0 F1 STA $F1B0

.C:f1c8 AD B2 F1 LDA $F1B2

.C:f1cb 18 CLC

.C:f1cc 69 28 ADC #$28

.C:f1ce 8D B2 F1 STA $F1B2

.C:f1d1 AD B3 F1 LDA $F1B3

.C:f1d4 69 00 ADC #$00

.C:f1d6 8D B3 F1 STA $F1B3

.C:f1d9 CA DEX

.C:f1da D0 D0 BNE $F1AC

Fiddle with the just cleared $E400 area

.C:f1dc 58 CLI

.C:f1dd A9 01 LDA #$01

.C:f1df 8D 20 D0 STA $D020

.C:f1e2 8D 21 D0 STA $D021

.C:f1e5 AD 80 DD LDA $DD80

.C:f1e8 29 FC AND #$FC

.C:f1ea 8D 80 DD STA $DD80

.C:f1ed A9 98 LDA #$98

.C:f1ef 8D 18 D0 STA $D018

.C:f1f2 AD 16 D0 LDA $D016

.C:f1f5 09 10 ORA #$10

.C:f1f7 8D 16 D0 STA $D016

.C:f1fa A9 01 LDA #$01

.C:f1fc 8D 22 D0 STA $D022

.C:f1ff A9 0A LDA #$0A

.C:f201 8D 23 D0 STA $D023

.C:f204 AF 1C F2 LAX $F21C ; Start with Zero

.C:f207 5D 1D F2 EOR $F21D,X ; Count up till $F2B6

.C:f20a E8 INX

.C:f20b E0 99 CPX #$99

.C:f20d D0 F8 BNE $F207

.C:f20f 8D 33 00 STA $0033 ; Store the result of a mask in $0033 (Used in $F30a routine below)

.C:f212 AF 1C F2 LAX $F21C

.C:f215 9D 68 F1 STA $F168,X ; Remove the init area above ($f168 to $f212) - but doesn't - No branch back!

.C:f218 E8 INX

.C:f219 E0 AA CPX #$AA

.C:f21b 60 RTS ; A:00 X:01 Y:FF SP:c8

From this - save the area $e000 to $E800, 01 to colmem and and the VIC registers as per $F1DD above

Goes to $F30a (At stack: $01c9)

.C:f30a AD 0E DC LDA $DC0E

.C:f30d 48 PHA

.C:f30e A9 00 LDA #$00

.C:f310 8D 0E DC STA $DC0E

.C:f313 78 SEI

.C:f314 A9 0B LDA #$0B

.C:f316 8D 11 D0 STA $D011

.C:f319 AD 12 D0 LDA $D012

.C:f31c C9 80 CMP #$80

.C:f31e D0 F9 BNE $F319

.C:f320 AD 12 D0 LDA $D012

.C:f323 C9 00 CMP #$00

.C:f325 D0 F9 BNE $F320

.C:f327 8D 0E DD STA $DD0E

.C:f32a A5 32 LDA $32

.C:f32c 8D 04 DD STA $DD04

.C:f32f A5 33 LDA $33

.C:f331 8D 05 DD STA $DD05

.C:f334 A9 01 LDA #$01

.C:f336 8D 0E DD STA $DD0E

.C:f339 A0 2B LDY #$2B

.C:f33b AF 04 DD LAX $DD04

.C:f33e 99 0A F3 STA $F30A,Y ; Trashes $F30A to $F335 with $DD04

.C:f341 88 DEY

.C:f342 10 F7 BPL $F33B

.C:f344 A2 F3 LDX #$F3

.C:f346 86 2C STX $2C

.C:f348 A9 00 LDA #$00

.C:f34a 85 2B STA $2B

.C:f34c A0 5B LDY #$5B

.C:f34e B1 2B LDA ($2B),Y

.C:f350 4D 04 DD EOR $DD04

.C:f353 4D 05 DD EOR $DD05

.C:f356 91 2B STA ($2B),Y

.C:f358 C8 INY

.C:f359 D0 F3 BNE $F34E

OK this shit is a loop that uses the timers to decipher the thing from $F35B to $F400

.C:f35b E6 2C INC $2C

.C:f35d E8 INX

.C:f35e E0 FF CPX #$FF

.C:f360 D0 EC BNE $F34E ; Keep going until $FFFF

.C:f362 A0 28 LDY #$28

.C:f364 AF 04 DD LAX $DD04

.C:f367 99 39 F3 STA $F339,Y ; Trash above

.C:f36a 88 DEY

.C:f36b 10 F7 BPL $F364

.C:f36d A2 F3 LDX #$F3

.C:f36f 86 2C STX $2C

.C:f371 A9 00 LDA #$00

.C:f373 85 2B STA $2B

.C:f375 A0 87 LDY #$87

.C:f377 B1 2B LDA ($2B),Y

.C:f379 4D 04 DD EOR $DD04

.C:f37c 4D 05 DD EOR $DD05

.C:f37f 4D 04 D0 EOR $D004

.C:f382 91 2B STA ($2B),Y

.C:f384 C8 INY

.C:f385 D0 F0 BNE $F377

OK this shit is a loop that uses the timers to decipher

.C:f387 E6 2C INC $2C

.C:f389 E8 INX

.C:f38a E0 FF CPX #$FF

.C:f38c D0 E9 BNE $F377 ; Keep going until $FFFF

.C:f38e A0 2B LDY #$2B

.C:f390 AF 04 DD LAX $DD04

.C:f393 99 62 F3 STA $F362,Y ; Trash above

.C:f396 88 DEY

.C:f397 10 F7 BPL $F390

.C:f399 A2 F3 LDX #$F3

.C:f39b 86 2C STX $2C

.C:f39d A9 00 LDA #$00

.C:f39f 85 2B STA $2B

.C:f3a1 A0 B2 LDY #$B2

.C:f3a3 B1 2B LDA ($2B),Y

.C:f3a5 4D 04 DD EOR $DD04

.C:f3a8 4D 05 DD EOR $DD05

.C:f3ab 49 45 EOR #$45

.C:f3ad 91 2B STA ($2B),Y

.C:f3af C8 INY

.C:f3b0 D0 F1 BNE $F3A3

.C:f3b2 E6 2C INC $2C

.C:f3b4 E8 INX

.C:f3b5 E0 FF CPX #$FF

.C:f3b7 D0 EA BNE $F3A3 ; Keep going until $FFFF

.C:f3b9 A0 2A LDY #$2A

.C:f3bb AF 04 DD LAX $DD04

.C:f3be 99 8E F3 STA $F38E,Y ; Trash above

.C:f3c1 88 DEY

.C:f3c2 10 F7 BPL $F3BB

.C:f3c4 A2 F3 LDX #$F3

.C:f3c6 86 2C STX $2C

.C:f3c8 A9 00 LDA #$00

.C:f3ca 85 2B STA $2B

.C:f3cc A0 DD LDY #$DD

.C:f3ce B1 2B LDA ($2B),Y

.C:f3d0 4D 04 DD EOR $DD04

.C:f3d3 4D 05 DD EOR $DD05

.C:f3d6 49 4F EOR #$4F

.C:f3d8 91 2B STA ($2B),Y

.C:f3da C8 INY

.C:f3db D0 F1 BNE $F3CE

.C:f3dd E6 2C INC $2C

.C:f3df E8 INX

.C:f3e0 E0 FF CPX #$FF

.C:f3e2 D0 EA BNE $F3CE ; Keep going until $FFFF

.C:f3e4 A0 2A LDY #$2A

.C:f3e6 AF 04 DD LAX $DD04

.C:f3e9 99 B9 F3 STA $F3B9,Y ; Trash above

.C:f3ec 88 DEY

.C:f3ed 10 F7 BPL $F3E6

.C:f3ef A2 F4 LDX #$F4

.C:f3f1 86 2C STX $2C

.C:f3f3 A9 00 LDA #$00

.C:f3f5 85 2B STA $2B

.C:f3f7 A0 08 LDY #$08

.C:f3f9 B1 2B LDA ($2B),Y

.C:f3fb 4D 04 DD EOR $DD04

.C:f3fe 4D 05 DD EOR $DD05

.C:f401 49 41 EOR #$41

.C:f403 91 2B STA ($2B),Y

.C:f405 C8 INY

.C:f406 D0 F1 BNE $F3F9

.C:f408 E6 2C INC $2C

.C:f40a E8 INX

.C:f40b E0 FF CPX #$FF ; Keep going until $FFFF

.C:f40d D0 EA BNE $F3F9

.C:f40f A0 2A LDY #$2A

.C:f411 AF 04 DD LAX $DD04

.C:f414 99 E4 F3 STA $F3E4,Y ; Trash above

.C:f417 88 DEY

.C:f418 10 F7 BPL $F411

.C:f41a A2 F4 LDX #$F4

.C:f41c 86 2C STX $2C

.C:f41e A9 00 LDA #$00

.C:f420 85 2B STA $2B

.C:f422 A0 33 LDY #$33

.C:f424 B1 2B LDA ($2B),Y

.C:f426 4D 04 DD EOR $DD04

.C:f429 4D 05 DD EOR $DD05

.C:f42c 49 21 EOR #$21

.C:f42e 91 2B STA ($2B),Y

.C:f430 C8 INY

.C:f431 D0 F1 BNE $F424

.C:f433 E6 2C INC $2C

.C:f435 E8 INX

.C:f436 E0 FF CPX #$FF ; Keep going until $FFFF

.C:f438 D0 EA BNE $F424

.C:f43a A0 2A LDY #$2A

.C:f43c AF 04 DD LAX $DD04

.C:f43f 99 0F F4 STA $F40F,Y ; Trash above

.C:f442 88 DEY

.C:f443 10 F7 BPL $F43C

.C:f445 A2 F4 LDX #$F4

.C:f447 86 2C STX $2C

.C:f449 A9 00 LDA #$00

.C:f44b 85 2B STA $2B

.C:f44d A0 5E LDY #$5E

.C:f44f B1 2B LDA ($2B),Y

.C:f451 4D 04 DD EOR $DD04

.C:f454 4D 05 DD EOR $DD05

.C:f457 45 32 EOR $32 ; Extra EOR compared to the rest of the loops

.C:f459 91 2B STA ($2B),Y

.C:f45b C8 INY

.C:f45c D0 F1 BNE $F44F

.C:f45e E6 2C INC $2C

.C:f460 E8 INX

.C:f461 E0 FF CPX #$FF ; Keep going until $FFFF

.C:f463 D0 EA BNE $F44F

.C:f465 A0 2A LDY #$2A

.C:f467 AF 04 DD LAX $DD04

.C:f46a 99 3A F4 STA $F43A,Y ; Trash above

.C:f46d 88 DEY

.C:f46e 10 F7 BPL $F467

.C:f470 A2 F4 LDX #$F4

.C:f472 86 2C STX $2C

.C:f474 A9 00 LDA #$00

.C:f476 85 2B STA $2B

.C:f478 A0 89 LDY #$89

.C:f47a B1 2B LDA ($2B),Y

.C:f47c 4D 04 DD EOR $DD04

.C:f47f 4D 05 DD EOR $DD05

.C:f482 45 33 EOR $33 ; Extra EOR compared to the rest of the loops

.C:f484 91 2B STA ($2B),Y

.C:f486 C8 INY

.C:f487 D0 F1 BNE $F47A

.C:f489 E6 2C INC $2C ; Keep going until $FFFF

.C:f48b E8 INX

.C:f48c E0 FF CPX #$FF

.C:f48e D0 EA BNE $F47A

.C:f490 A0 2A LDY #$2A

.C:f492 AF 04 DD LAX $DD04

.C:f495 99 65 F4 STA $F465,Y ; Trash above

.C:f498 88 DEY

.C:f499 10 F7 BPL $F492

.C:f49b A2 F4 LDX #$F4

.C:f49d 86 2C STX $2C

.C:f49f A9 00 LDA #$00

.C:f4a1 85 2B STA $2B

.C:f4a3 A0 B4 LDY #$B4

.C:f4a5 B1 2B LDA ($2B),Y

.C:f4a7 4D 04 DD EOR $DD04

.C:f4aa 4D 05 DD EOR $DD05

.C:f4ad 49 4B EOR #$4B ; #$4b - "k"

.C:f4af 91 2B STA ($2B),Y

.C:f4b1 C8 INY

.C:f4b2 D0 F1 BNE $F4A5

.C:f4b4 E6 2C INC $2C ; Keep going until $FFFF

.C:f4b6 E8 INX

.C:f4b7 E0 FF CPX #$FF

.C:f4b9 D0 EA BNE $F4A5

.C:f4bb A0 2A LDY #$2A

.C:f4bd AF 04 DD LAX $DD04

.C:f4c0 99 90 F4 STA $F490,Y ; Trash above

.C:f4c3 88 DEY

.C:f4c4 10 F7 BPL $F4BD

.C:f4c6 A2 F4 LDX #$F4

.C:f4c8 86 2C STX $2C

.C:f4ca A9 00 LDA #$00

.C:f4cc 85 2B STA $2B

.C:f4ce A0 DF LDY #$DF

.C:f4d0 B1 2B LDA ($2B),Y

.C:f4d2 4D 04 DD EOR $DD04

.C:f4d5 4D 05 DD EOR $DD05

.C:f4d8 49 2E EOR #$2E ; #$2E - "."

.C:f4da 91 2B STA ($2B),Y

.C:f4dc C8 INY

.C:f4dd D0 F1 BNE $F4D0

.C:f4df E6 2C INC $2C ; Keep going until $FFFF

.C:f4e1 E8 INX

.C:f4e2 E0 FF CPX #$FF

.C:f4e4 D0 EA BNE $F4D0

.C:f4e6 A0 2A LDY #$2A

.C:f4e8 AF 04 DD LAX $DD04

.C:f4eb 99 BB F4 STA $F4BB,Y ; Trash above

.C:f4ee 88 DEY

.C:f4ef 10 F7 BPL $F4E8

.C:f4f1 A2 F5 LDX #$F5

.C:f4f3 86 2C STX $2C

.C:f4f5 A9 00 LDA #$00

.C:f4f7 85 2B STA $2B

.C:f4f9 A0 0A LDY #$0A

.C:f4fb B1 2B LDA ($2B),Y

.C:f4fd 4D 04 DD EOR $DD04

.C:f500 4D 05 DD EOR $DD05

.C:f503 49 45 EOR #$45 ; #$45 - "e"

.C:f505 91 2B STA ($2B),Y

.C:f507 C8 INY

.C:f508 D0 F1 BNE $F4FB

.C:f50a E6 2C INC $2C

.C:f50c E8 INX

.C:f50d E0 FF CPX #$FF

.C:f50f D0 EA BNE $F4FB

.C:f511 A0 2A LDY #$2A

.C:f513 AF 04 DD LAX $DD04

.C:f516 99 E6 F4 STA $F4E6,Y ; Trash above

.C:f519 88 DEY

.C:f51a 10 F7 BPL $F513

.C:f51c A2 F5 LDX #$F5

.C:f51e 86 2C STX $2C

.C:f520 A9 00 LDA #$00

.C:f522 85 2B STA $2B

.C:f524 A0 35 LDY #$35

.C:f526 B1 2B LDA ($2B),Y

.C:f528 4D 04 DD EOR $DD04

.C:f52b 4D 05 DD EOR $DD05

.C:f52e 49 2E EOR #$2E ; #$2E - "."

.C:f530 91 2B STA ($2B),Y

.C:f532 C8 INY

.C:f533 D0 F1 BNE $F526

.C:f535 E6 2C INC $2C

.C:f537 E8 INX

.C:f538 E0 FF CPX #$FF

.C:f53a D0 EA BNE $F526

.C:f53c A0 2A LDY #$2A

.C:f53e AF 04 DD LAX $DD04

.C:f541 99 11 F5 STA $F511,Y ; Trash above

.C:f544 88 DEY

.C:f545 10 F7 BPL $F53E

.C:f547 A2 F5 LDX #$F5

.C:f549 86 2C STX $2C

.C:f54b A9 00 LDA #$00

.C:f54d 85 2B STA $2B

.C:f54f A0 60 LDY #$60

.C:f551 B1 2B LDA ($2B),Y

.C:f553 4D 04 DD EOR $DD04

.C:f556 4D 05 DD EOR $DD05

.C:f559 49 48 EOR #$48 ; #48 - "h"

.C:f55b 91 2B STA ($2B),Y

.C:f55d C8 INY

.C:f55e D0 F1 BNE $F551

.C:f560 E6 2C INC $2C

.C:f562 E8 INX

.C:f563 E0 FF CPX #$FF

.C:f565 D0 EA BNE $F551

.C:f567 68 PLA

.C:f568 8D 0E DC STA $DC0E

.C:f56b A9 1B LDA #$1B

.C:f56d 8D 11 D0 STA $D011

.C:f570 A9 00 LDA #$00

.C:f572 8D 0E DD STA $DD0E

.C:f575 60 RTS ; Ciphering done!

Goes to $FF00 (At stack: $01cb)

Sets pointer to the $F576 where the actual drive code is.

.C:ff00 AD 00 F3 LDA $F300 ; #$76

.C:ff03 85 2B STA $2B

.C:ff05 AD 01 F3 LDA $F301 ; #$f5

.C:ff08 85 2C STA $2C

.C:ff0a AD 04 F3 LDA $F304 ; #$d4

.C:ff0d 8D 40 00 STA $0040

.C:ff10 AD 05 F3 LDA $F305 ; #$08

.C:ff13 8D 41 00 STA $0041

.C:ff16 60 RTS

Goes to $F222 (At stack: $01cd)

Copy the drive code to $0400 (Source at $F576 and copy until $fade)

.C:f222 A0 00 LDY #$00

.C:f224 B1 2B LDA ($2B),Y ; Load source

.C:f226 91 2D STA ($2D),Y ; Save destination

.C:f228 0C 52 F2 NOOP $F252

.C:f22b 6A ROR A

.C:f22c 91 2B STA ($2B),Y ; Trash source

.C:f22e E6 2B INC $2B

.C:f230 44 8D NOOP $8D

.C:f232 D0 05 BNE $F239

.C:f234 E6 2C INC $2C

.C:f236 0C 35 37 NOOP $3735

.C:f239 E6 2D INC $2D

.C:f23b D0 02 BNE $F23F

.C:f23d E6 2E INC $2E

.C:f23f A7 2B LAX $2B

.C:f241 CD 02 F3 CMP $F302

.C:f244 0C 00 04 NOOP $0400

.C:f247 D0 DB BNE $F224

.C:f249 A7 2C LAX $2C

.C:f24b CD 03 F3 CMP $F303

.C:f24e D0 D4 BNE $F224

.C:f250 60 RTS

Ok, now the driv ecode is at $0400

Goes to $003a (At stack: $01cf)

.C:003a A7 04 LAX $04 ; Switch in RAM with #$07 in $01?

.C:003c 87 01 SAX $01

.C:003e 60 RTS ; The RTS jumps to $003f

.C:003f 4C D4 08 JMP $08D4

; Copy drive code to drive RAM

.C:08d4 AD 00 DD LDA $DD00

.C:08d7 29 03 AND #$03

.C:08d9 8D 00 DD STA $DD00

.C:08dc A2 08 LDX #$08

.C:08de A9 0F LDA #$0F

.C:08e0 A0 0F LDY #$0F

.C:08e2 20 BA FF JSR $FFBA

.C:08e5 A9 00 LDA #$00

.C:08e7 20 BD FF JSR $FFBD

.C:08ea 20 4A F3 JSR $F34A

.C:08ed A9 00 LDA #$00 ; Source is $0400 in computer

.C:08ef 85 04 STA $04

.C:08f1 A9 04 LDA #$04

.C:08f3 85 05 STA $05

.C:08f5 A9 03 LDA #$03 ; Destination is $0300 in drive RAM

.C:08f7 8D 61 09 STA $0961

.C:08fa A9 00 LDA #$00

.C:08fc 8D 60 09 STA $0960

.C:08ff A2 0F LDX #$0F

.C:0901 20 50 F2 JSR $F250

.C:0904 A0 00 LDY #$00

.C:0906 B9 5D 09 LDA $095D,Y

.C:0909 20 CA F1 JSR $F1CA

.C:090c C8 INY

.C:090d C0 06 CPY #$06

.C:090f D0 F5 BNE $0906

.C:0911 A0 00 LDY #$00

.C:0913 B1 04 LDA ($04),Y

.C:0915 20 CA F1 JSR $F1CA ; Write to drive mem

.C:0918 C8 INY

.C:0919 C0 20 CPY #$20

.C:091b D0 F6 BNE $0913

.C:091d 20 33 F3 JSR $F333

.C:0920 AD 60 09 LDA $0960

.C:0923 18 CLC

.C:0924 69 20 ADC #$20

.C:0926 8D 60 09 STA $0960

.C:0929 AD 61 09 LDA $0961

.C:092c 69 00 ADC #$00

.C:092e 8D 61 09 STA $0961

.C:0931 A5 04 LDA $04

.C:0933 18 CLC

.C:0934 69 20 ADC #$20

.C:0936 85 04 STA $04

.C:0938 A5 05 LDA $05

.C:093a 69 00 ADC #$00

.C:093c 85 05 STA $05

.C:093e C9 08 CMP #$08 ; Until $08FF

.C:0940 F0 BD BEQ $08FF

.C:0942 90 BB BCC $08FF

.C:0944 A2 0F LDX #$0F ; And execute the drive code at $0300

.C:0946 20 50 F2 JSR $F250

.C:0949 A0 00 LDY #$00

.C:094b B9 63 09 LDA $0963,Y

.C:094e 20 CA F1 JSR $F1CA

.C:0951 C8 INY

.C:0952 C0 05 CPY #$05

.C:0954 D0 F5 BNE $094B

.C:0956 20 33 F3 JSR $F333

.C:0959 20 2F F3 JSR $F32F

.C:095c 60 RTS

>C:095d 4d 2d 57 00 00 20 4d 2d 45 00 03 00 M-W.. M-E...

0960 and 0961 are the destination bytes for the Memory Write in the drive RAM.

Goes to $0034 (At stack: $01d3)

.C:0034 78 SEI ; Flip out the ROM

.C:0035 A7 03 LAX $03

.C:0037 87 01 SAX $01

.C:0039 60 RTS

Goes to $f291 (At stack: $01d5)

.C:f291 68 PLA ; #$63 at $01d7

.C:f292 AF 99 F2 LAX $F299

.C:f295 18 CLC

.C:f296 69 20 ADC #$20 ; Store #$20 on all of screen - clear the drive code

.C:f298 9D 00 04 STA $0400,X

.C:f29b 9D 00 05 STA $0500,X

.C:f29e 9D 00 06 STA $0600,X

.C:f2a1 9D 00 07 STA $0700,X

.C:f2a4 9D 00 08 STA $0800,X

.C:f2a7 E8 INX

.C:f2a8 D0 EE BNE $F298

.C:f2aa 60 RTS

Goes to $ff1a (At stack: $01d8)

.C:ff1a 6C 08 F3 JMP ($F308) ; $FD77

.C:fd77 20 DE FA JSR $FADE

.C:fd7a A2 04 LDX #$04 ; Delay loop?

.C:fd7c A0 00 LDY #$00

.C:fd7e 88 DEY

.C:fd7f D0 FD BNE $FD7E

.C:fd81 CA DEX

.C:fd82 D0 FA BNE $FD7E

.C:fd84 60 RTS

Goes to $0223 (At stack: $01dA)

.C:0223 6E 1B FF ROR $FF1B ; Fiddles with jump vector

.C:0226 6E 1C FF ROR $FF1C

.C:0229 A9 06 LDA #$06

.C:022b AA TAX

.C:022c EE 21 D0 INC $D021

.C:022f 8D FE 01 STA $01FE

.C:0232 A9 08 LDA #$08

.C:0234 AA TAX

.C:0235 04 A9 NOOP $A9

.C:0237 8F FF 01 SAX $01FF

.C:023a A2 03 LDX #$03

.C:023c 8A TXA

.C:023d 8F 00 01 SAX $0100

.C:0240 A9 08 LDA #$08

.C:0242 AA TAX

.C:0243 8F 01 01 SAX $0101

.C:0246 A9 17 LDA #$17

; This JSR loads the main screen

.C:0248 20 17 FF JSR $FF17 ; Does JMP ($F306) -> $FC3E (Never returns - good break point)

.C:024b 20 3A 00 JSR $003A ; All RAM

.C:024e 4C 23 02 JMP $0223

>C:0251 out of my code hacker!@@@@@@O@@@@@@@@@@@

=====================================================

Loader ($FC3E) in separate part but usage is:

### #$00 - instruct to load a block to Low (x)/High (y)

### #$10 - load file (loading the main one)

### #$80 - load the actual block to track (y)/Sector (x)

### #$90 - save the actual block to track (y)/Sector (x)

### #$70

### #$f0

## Loading the first part/$7800 part

This is the file that is at Track 03, Sector 00. Normal file chain is used so file can be recreated by a directory entry pointing at this location. File is encrypted with #$EA (Placed in $FB3D at address $FCAA). Uses the $10 parameter while loading from the $FB3D.

A 17, X 08, Y 00 - Biggest loading (7800 - 9F80) - File: “**1.base 7800.prg**”

EOA is flashing - and background is grey

Now it Jumps to the start at $7800 (Capture with break at $0073 or $7800)

## The $7800 part

The $7800 part loads additional data using a track loader. Uses the $00 parameter to pass X/Y loading parameters to the loader and then the $80 parameter to get a full sector.

The actual loaders buffer is $FB3E and the $7800 routine uses $1700 as buffer (always call $FB3D with $00 using 17/00 in X and Y), and then copies it to the final destination.

Freeze points:

* "Watch write 0006" (on Device 8:)
* "watch exec 0200 if .A == $80"
* - 85C1 (LDA #0, LDX $B3, LDY $B4) - Call to 00,00,17 - Sets a number of parameters only. “Break 85c7”
* 85A7 (LDA $85DB, LDX $85CE, LDY $85CD. “Break 85b0”

After loading the block it calls $8539 (“Break 8568” or “Break 8588”)

- picks source from $b3/$b4 and copies this to $FE/$FF

- picks destination from $84c1 and copies this to $F5/$F6

If 84C6 is not #$00 starts #$80 bytes later

Break 85b0 (check the A, X and Y) and 8568 (check 00f5 for the destination - routine start at 8539) or 0200 to capture both

The copy routine at $FB39 uses $B3 and $B4 as source, which is the same $B3 and $B4 that was used to call the drive routine as start. They do contain #$17 and #$00.

850a -> 83e9

8432 - Loads after the menu screen

.C:81bb 20 28 82 JSR $8228 ; Init

.C:81be 20 38 86 JSR $8638

.C:81c1 20 B1 89 JSR $89B1

.C:81c4 A9 03 LDA #$03

.C:81c6 20 16 8E JSR $8E16

.C:81c9 A9 60 LDA #$60

.C:81cb 8D B9 A1 STA $A1B9

.C:81ce 20 E9 83 JSR $83E9 ; Load the first level - show the ugly Greece map

; 4000-6000 (Pic)

; 6c00 Screen

; F1 = Carry clear, F7 = Carry Set

.C:81d1 08 PHP

.C:81d2 20 02 82 JSR $8202

.C:81d5 28 PLP

.C:81d6 90 03 BCC $81DB

.C:81d8 20 00 7E JSR $7E00 ; Manage saves screen (after pressing F1)

.C:81db 20 28 82 JSR $8228 ; Init and load

.C:81de 20 EB 82 JSR $82EB ; Load more and show “The Goal”

.C:81e1 20 0C 82 JSR $820C ; Load more

.C:81e4 20 38 86 JSR $8638

.C:81e7 20 16 82 JSR $8216

.C:81ea 20 39 82 JSR $8239

.C:81ed 20 28 82 JSR $8228 ; Init

.C:81f0 20 52 82 JSR $8252

.C:81f3 20 20 82 JSR $8220

.C:81f6 4C B2 84 JMP $84B2

.C:84b2 20 62 84 JSR .do\_load

.C:84b5 20 32 86 JSR $8632

.C:84b8 20 3E 86 JSR $863E

.C:84bb 20 17 88 JSR $8817

.C:84be 4C 00 E4 JMP $E400 ; Start the game

; Loading the Greece map

.C:83e9 A9 00 LDA #$00

.C:83eb 20 3D 9D JSR $9D3D ; Load all the $9f80 files

.C:83ee 20 A8 8D JSR $8DA8

.C:83f1 C9 85 CMP #$85

.C:83f3 F0 05 BEQ $83FA

.C:83f5 20 FF 83 JSR $83FF

.C:83f8 18 CLC

.C:83f9 60 RTS

.C:83fa 20 FF 83 JSR $83FF

.C:83fd 38 SEC

.C:83fe 60 RTS

.C:9d3d 8D 3C 9D STA $9D3C

.C:9d40 48 PHA

.C:9d41 C9 01 CMP #$01

.C:9d43 90 04 BCC $9D49

.C:9d45 A9 07 LDA #$07

.C:9d47 D0 02 BNE $9D4B

.C:9d49 A9 03 LDA #$03

.C:9d4b 48 PHA

.C:9d4c 20 AD 9D JSR $9DAD ; Load the $9F80 files

.C:9d4f 68 PLA

.C:9d50 A8 TAY

.C:9d51 B9 D8 8C LDA $8CD8,Y

.C:9d54 48 PHA

.C:9d55 88 DEY

.C:9d56 B9 D8 8C LDA $8CD8,Y

.C:9d59 48 PHA

.C:9d5a 88 DEY

.C:9d5b B9 D8 8C LDA $8CD8,Y

.C:9d5e 8D 21 D0 STA $D021

.C:9d61 88 DEY

.C:9d62 B9 D8 8C LDA $8CD8,Y

.C:9d65 8D 20 D0 STA $D020

.C:9d68 68 PLA

.C:9d69 85 20 STA $20

.C:9d6b A9 00 LDA #$00

.C:9d6d 85 FE STA $FE

.C:9d6f A9 DB LDA #$DB

.C:9d71 85 FF STA $FF

.C:9d73 A5 20 LDA $20

.C:9d75 A2 03 LDX #$03

.C:9d77 A0 E8 LDY #$E8

.C:9d79 88 DEY

.C:9d7a 91 FE STA ($FE),Y

.C:9d7c D0 FB BNE $9D79

.C:9d7e C6 FF DEC $FF

.C:9d80 CA DEX

.C:9d81 10 F6 BPL $9D79

.C:9d83 68 PLA

.C:9d84 85 20 STA $20

.C:9d86 A9 00 LDA #$00

.C:9d88 85 FE STA $FE

.C:9d8a A9 6F LDA #$6F

.C:9d8c 85 FF STA $FF

.C:9d8e A5 20 LDA $20

.C:9d90 A2 03 LDX #$03

.C:9d92 A0 E8 LDY #$E8

.C:9d94 88 DEY

.C:9d95 91 FE STA ($FE),Y

.C:9d97 D0 FB BNE $9D94

.C:9d99 C6 FF DEC $FF

.C:9d9b CA DEX

.C:9d9c 10 F6 BPL $9D94

.C:9d9e 68 PLA

.C:9d9f AA TAX

.C:9da0 A9 00 LDA #$00

.C:9da2 20 16 8E JSR $8E16

.C:9da5 A2 28 LDX #$28

.C:9da7 A0 04 LDY #$04

.C:9da9 20 02 91 JSR $9102

.C:9dac 60 RTS

.C:9dad A2 F0 LDX #$F0

.C:9daf A0 03 LDY #$03

.C:9db1 AD 3C 9D LDA $9D3C

.C:9db4 F0 04 BEQ $9DBA

.C:9db6 A2 12 LDX #$12

.C:9db8 A0 04 LDY #$04

.C:9dba 8E C3 84 STX .dest\_adr\_hi+1

.C:9dbd 8C C4 84 STY $84C4

.C:9dc0 A9 00 LDA #$00

.C:9dc2 85 20 STA $20

.C:9dc4 A9 40 LDA #$40

.C:9dc6 85 21 STA $21

.C:9dc8 A9 28 LDA #$28

.C:9dca 8D E6 9E STA $9EE6

.C:9dcd A9 00 LDA #$00

.C:9dcf 8D E7 9E STA $9EE7

.C:9dd2 A9 80 LDA #$80

.C:9dd4 8D C1 84 STA .dest\_adr\_lo

.C:9dd7 85 22 STA $22

.C:9dd9 A9 03 LDA #$03

.C:9ddb 8D C2 84 STA .dest\_adr\_hi

.C:9dde 85 23 STA $23

.C:9de0 20 C9 84 JSR .sector\_compare\_adr+1

.C:9de3 10 03 BPL $9DE8

.C:9de5 4C 85 84 JMP $8485

.C:9de8 AD 81 03 LDA $0381

.C:9deb 0A ASL A

.C:9dec 8D C5 84 STA $84C5

.C:9def AD 80 03 LDA $0380

.C:9df2 10 03 BPL $9DF7

.C:9df4 EE C5 84 INC $84C5

.C:9df7 09 80 ORA #$80

.C:9df9 85 30 STA $30

.C:9dfb A0 02 LDY #$02

.C:9dfd B1 22 LDA ($22),Y

.C:9dff 85 31 STA $31

.C:9e01 18 CLC

.C:9e02 A5 22 LDA $22

.C:9e04 69 02 ADC #$02

.C:9e06 85 2A STA $2A

.C:9e08 A5 23 LDA $23

.C:9e0a 69 00 ADC #$00

.C:9e0c 85 2B STA $2B

.C:9e0e 18 CLC

.C:9e0f A5 2A LDA $2A

.C:9e11 65 31 ADC $31

.C:9e13 69 01 ADC #$01

.C:9e15 85 22 STA $22

.C:9e17 A5 2B LDA $2B

.C:9e19 69 00 ADC #$00

.C:9e1b 85 23 STA $23

.C:9e1d AD 3C 9D LDA $9D3C

.C:9e20 F0 03 BEQ $9E25

.C:9e22 20 7B 9E JSR $9E7B

.C:9e25 A0 00 LDY #$00

.C:9e27 A5 22 LDA $22

.C:9e29 C5 30 CMP $30

.C:9e2b 90 05 BCC $9E32

.C:9e2d AD C5 84 LDA $84C5

.C:9e30 F0 3D BEQ $9E6F

.C:9e32 A0 00 LDY #$00

.C:9e34 B1 22 LDA ($22),Y

.C:9e36 85 28 STA $28

.C:9e38 20 8D 9E JSR $9E8D

.C:9e3b A4 31 LDY $31

.C:9e3d F0 09 BEQ $9E48

.C:9e3f B1 2A LDA ($2A),Y

.C:9e41 C5 28 CMP $28

.C:9e43 F0 0E BEQ $9E53

.C:9e45 88 DEY

.C:9e46 D0 F7 BNE $9E3F

.C:9e48 A5 28 LDA $28

.C:9e4a 20 AC 9E JSR $9EAC

.C:9e4d 20 B2 9E JSR $9EB2

.C:9e50 4C 27 9E JMP $9E27

.C:9e27 A5 22 LDA $22

.C:9e29 C5 30 CMP $30

.C:9e2b 90 05 BCC $9E32

.C:9e2d AD C5 84 LDA $84C5

.C:9e30 F0 3D BEQ $9E6F

.C:9e32 A0 00 LDY #$00

.C:9e34 B1 22 LDA ($22),Y

.C:9e36 85 28 STA $28

.C:9e38 20 8D 9E JSR $9E8D

.C:9e3b A4 31 LDY $31

.C:9e3d F0 09 BEQ $9E48

.C:9e3f B1 2A LDA ($2A),Y

.C:9e41 C5 28 CMP $28

.C:9e43 F0 0E BEQ $9E53

.C:9e45 88 DEY

.C:9e46 D0 F7 BNE $9E3F

.C:9e48 A5 28 LDA $28

.C:9e4a 20 AC 9E JSR $9EAC

.C:9e4d 20 B2 9E JSR $9EB2

.C:9e50 4C 27 9E JMP $9E27

.C:9e53 A0 00 LDY #$00

.C:9e55 B1 22 LDA ($22),Y

.C:9e57 48 PHA

.C:9e58 20 8D 9E JSR $9E8D

.C:9e5b 68 PLA

.C:9e5c AA TAX

.C:9e5d 20 63 9E JSR $9E63

.C:9e60 4C 27 9E JMP $9E27

.C:9e63 A5 28 LDA $28

.C:9e65 20 AC 9E JSR $9EAC

.C:9e68 20 B2 9E JSR $9EB2

.C:9e6b CA DEX

.C:9e6c D0 F5 BNE $9E63

.C:9e6e 60 RTS

.C:9e6f AD 3C 9D LDA $9D3C

.C:9e72 F0 03 BEQ $9E77

.C:9e74 20 7B 9E JSR $9E7B

.C:9e77 20 7E 9E JSR $9E7E

.C:9e7a 60 RTS

.C:9e7b 20 7E 9E JSR $9E7E

.C:9e7e A9 00 LDA #$00

.C:9e80 85 28 STA $28

.C:9e82 A2 A0 LDX #$A0

.C:9e84 20 63 9E JSR $9E63

.C:9e87 A2 A0 LDX #$A0

.C:9e89 20 63 9E JSR $9E63

.C:9e8c 60 RTS

.C:9e8d E6 22 INC $22

.C:9e8f F0 01 BEQ $9E92

.C:9e91 60 RTS

.C:9e92 20 7D 85 JSR $857D

.C:9e95 A9 80 LDA #$80

.C:9e97 8D C1 84 STA .dest\_adr\_lo

.C:9e9a 85 22 STA $22

.C:9e9c A9 9F LDA #$9F

.C:9e9e 8D C2 84 STA .dest\_adr\_hi

.C:9ea1 85 23 STA $23

.C:9ea3 20 C9 84 JSR .sector\_compare\_adr+1

.C:9ea6 10 03 BPL $9EAB

.C:9ea8 4C 85 84 JMP $8485

.C:9eab 60 RTS

$7800 Loading before the Greece map

A 00, X 00, Y 17

A 80, X 08, Y 1F (1f,08 - 0380) - Turn RED (File: **2a1.0380.prg and 2a2.9f80.prg**)

A 00, X 00, Y 17

A 80, X 09, Y 1F (1F,09 - 9F80) - b

A 00, X 00, Y 17

A 80, X 0A, Y 1F (1F,0A - 9F80) - c

A 00, X 00, Y 17

A 80, X 0B, Y 1F (1F,0B - 9F80) - d

A 00, X 00, Y 17

A 80, X 0C, Y 1F (1F,0C - 9F80) - e

A 00, X 00, Y 17

A 80, X 0D, Y 1F (1F,0D - 9F80) - f

A 00, X 00, Y 17

A 80, X 0E, Y 1F (1F,0E - 9F80) - g

A 00, X 00, Y 17

A 80, X 0F, Y 1F (1F,0F - 9F80) - h

A 00, X 00, Y 17

A 80, X 00, Y 20 (20,00 - 9F80) - i

A 00, X 00, Y 17

A 80, X 01, Y 20 (20,01 - 9F80) - j

A 00, X 00, Y 17

A 80, X 02, Y 20 (20,02 - 9F80) - k

A 00, X 00, Y 17

A 80, X 03, Y 20 (20,03 - 9F80) - l

A 00, X 00, Y 17

A 80, X 04, Y 20 (20,04 - 9F80) - m

A 00, X 00, Y 17

A 80, X 05, Y 20 (20,05 - 9F80) - n

A 00, X 00, Y 17

A 80, X 06, Y 20 (20,06 - 9F80) - o

A 00, X 00, Y 17

A 80, X 07, Y 20 (20,07 - 9F80) - p

A 00, X 00, Y 17

A 80, X 08, Y 20 (20,08 - 9F80) - q

Launch Intro screen - to continue you need to press F1 to manage Saves and F7 to continue directly to the game.

$7800 Loading after Greece map but before the menu - pressing F7:

14,00-07 - Loading to 7000

08,08-0F - Loading to A001

09,00-0F

0A,00-05

15,00-0F - Loading to 1800

16,00-0F

17,00-04 (Yellow)

19,01

14,00 - 7000 (8 blocks - File: **3a.7000.prg**)

14,01 - 7100

14,02 - 7200

14,03 - 7300

14,04 - 7400

14,05 - 7500

14,06 - 7600

14,07 - 7700

08,08 - A001 (30 block - File: **3b.A001.prg - included the A000 to be safe**)

08,09 - A101

08,0A - A201

08,0B - A301

08,0C - A401

08,0D - A501

08,0E - A601

08,0F - A701

09,00 - A801

09,01 - A901

09,02 - AA01

09,03 - AB01

09,04 - AC01

09,05 - AD01

09,06 - AE01

09,07 - AF01

09,08 - B001

09,09 - B101

09,0A - B201

09,0B - B301

09,0C - B401

09,0D - B501

09,0E - B601

09,0F - B701

0A,00 - B801

0A,01 - B901

0A,02 - BA01

0A,03 - BB01

0A,04 - BC01

0A,05 - BD01

15,00 - 1800 (37 block File: **3c.1800.prg**)

15,01 - 1900

15,02 - 1A00

15,03 - 1B00

15,04 - 1C00

15,05 - 1D00

15,06 - 1E00

15,07 - 1F00

15,08 - 2000

15,09 - 2100

15,0A - 2200

15,0B - 2300

15,0C - 2400

15,0D - 2500

15,0E - 2600

15,0F - 2700

16,00 - 2800

16,01 - 2900

16,02 - 2A00

16,03 - 2B00

16,04 - 2C00

16,05 - 2D00

16,06 - 2E00

16,07 - 2F00

16,08 - 3000

16,09 - 3100

16,0A - 3200

16,0B - 3300

16,0C - 3400

16,0D - 3500

16,0E - 3600

16,0F - 3700

17,00 - 3800

17,01 - 3900

17,02 - 3A00

17,03 - 3B00

17,04 - 3C00 - Yellow

19,01 - 9f80 (File: **3d.9f80.prg)**

(First A00, X00, Y17 and then A80, X and Y according to the table above)

$7800 Loading after menu but before the game starts:

Load new infotexts:

1D,0B - 9F80 - a

1F,01 - 9F80 - b

1F,02 - 9F80 - c

1E,0F - 9F80 - d

05,0F - E400 (22 block File: **4e.e400.prg**)

06,00 - E500

06,01 - E600

06,02 - E700

06,03 - E800

06,04 - E900

06,05 - EA00

06,06 - EB00

06,07 - EC00

06,08 - ED00

06,09 - EE00

06,0A - EF00

06,0B - F000

06,0C - F100

06,0D - F200

06,0E - F300

06,0F - F400

07,00 - F500

07,01 - F600

07,02 - F700

07,03 - F800

07,04 - F900

14,08 - 6000 (4 blocks File: **4f.6000.prg**)

14,09 - 6100

14,0A - 6200

14,0B - 6300

14,0C - 6800 (4 blocks File: **4g.6800.prg**)

14,0D - 6900

14,0E - 6A00

14,0E - 6B00

07,05 - 0400 (19 blocks File: **4h.0400.prg**)

07,06 - 0500

07,07 - 0600

07,08 - 0700

07,09 - 0800

07,0A - 0900

07,0B - 0A00

07,0C - 0B00

07,0D - 0C00

07,0E - 0D00

07,0F - 0E00

08,00 - 0F00

08,01 - 1000

08,02 - 1100

08,03 - 1200

08,04 - 1300

08,05 - 1400

08,06 - 1500

08,07 - 1600

17,05 - 3d00 (19 block file File: **4i.3d00.prg**)

17,06 - 3e00

17,07 - 3f00

17,08 - 4000

17,09 - 4100

17,0A - 4200

17,0B - 4300

17,0C - 4400

17,0D - 4500

17,0E - 4600

17,0F - 4700

18,00 - 4800

18,01 - 4900

18,02 - 4A00

18,03 - 4B00

18,04 - 4C00

18,05 - 4D00

18,06 - 4E00

18,07 - 4F00

05,08 - 7e00 (6 block File: **4j.7e00.prg**)

05,09 - 7f00

05,0A - 8000

05,0B - 8100

05,0C - 8200

05,0D - 8300

Game starts at $E400. Called every round.

=============================

Save game (Call with 90):

14,00 - 1700 - Load (presumably loading to check which disk it is)

826d is an indexpointer to "saved games"

If index == 0, a default game is loaded from track,sector 21,00 (dec)

02a0 ->

sector = $a0 / 2 & $0f => 00

track = hibyte $02a0 \* 8 => 21 (dec)

(Calculated at the routine at $850a)

Table is located at 9d28 and can be translated to:

$0000 : Index 0 is not allowed, then game is loaded from 21,00 ($15,$00)

$0180 : => 12,00 (dec)

$01e0 : => 15,00 (dec) ($0f,$00)

$0260 : => 19,00 (dec) ($13,$00)

$02c0 : => 22,00 (dec) ($16,$00)

$00c0 : => 06,00 (dec) ($06,$00)

$0120 : => 09,00 (dec) ($09,$00)

$0320 : => 25,00 (dec) ($19,$00)

$0060 : => 03,00 (dec) ($03,$00)

$0380 : => 28,00 (dec) ($1c,$00)

Games are loaded $1800-$3CFF (37 sectors)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Index | Filename (given) | Length | Sector/Track (ingame value) | Real T/S | Start |
| 0 |  | 01 | bc,00 |  | 0340 |
| 1 | 3a | 10 | 80,02 | 14,00 | 7000 |
| 2 | 3b | 3b | 10,01 | 08,00 | a001 |
| 3 | 4e | 2c | be,00 | 05,0f | e400 |
| 4 | 4f | 08 | 90,02 | 14,08 | 6000 |
| 5 | 4g | 08 | 98,02 | 14,0c | 6800 |
| 6 | 4h | 25 | ea,00 | 07,05 | 0400 |
| 7 | 4i | 26 | ea,02 | 17,05 | 3d00 |
| 8 | 4j | 0c | b0,00 | 05,08 | 7e00 |

(The arrow in the header shows that the freeze at 7800 happens before the calling of the routines in the column)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| After which point | <- 7800 | <- Greece | <- F1 menu | <- F7 Info | Game |
| 833a |  |  | 8) Once |  |  |
| 8492 |  | 3) Counter in A A=10, A=3B (3a/b/c filerna) | 9)Counter in A  A=4a | 5) Counter in A  A= 2c, A=8, A=8, A=25x, A=26, A=0c |  |
| 860b |  |  | 7) Once  10) After being asked to put Heracles disk into drive - goto 5)  11)3 times |  |  |
| 9c62 |  |  |  | 4) Loading text | 6)Loading text |
| 9de0 | 1) Once |  |  |  |  |
| 9ea3 | 2) 33 times |  |  |  |  |

1+2) 9de0+9ea3 - Loads additional data to Greece map.

3+5) 9492 - Capture to load all 3\* and 4Ä files

4+6) 9c62 Capture loading to $1700

7+8+9+10) Loading savegame

11) Saving savegame 860b - request to put save disk in drive, saving, load disk into drive

### Text blocks

First called from 83a4 ->**99dd** with 93 in A (masked to 13). 9d21 is 0 for the first sector, and then 1 for the rest.

If the sector is the last of the string, it ends with FF. If there is more, there is a marker (F0) followed by a pointer to the next string. The load routine is called with the new pointer in the end of A.

Routine at 9c44 (Load text block) is called from 99fc and 9a34.

Calls in the actual game:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Address | Value | More | Track/Sector | 9d1f | Text |
| 04a5 | 24 |  | 19,0a; 1d,05 | 24 (->21) | Achilles wounded in heel |
| 04f1 | 29 |  | 19,0c; 1d,08 | 20 | Poseidon, no friend of the trojans |
| 04fa | 2a |  | 19,0d; 1d,08 1d,09 | 2a | Apollo protects the body |
| 051a | 2c |  | 19,0e; 1d,0a | 2c | Talos the bull-head |
| 05bf | Variable (1a30,y + 40) |  |  |  |  |
| 05e5 | 87 |  | 18,0b;1c,05  1c,09 | 07 | A ferocious gale of hot air (Failed last task) |
| 060f | 90 |  | 19,00;1c,0a | 10 | The earth trembles |
| 0633 | 0e |  | 18,0f;  1c,08;1c,09 | 0e | Hephaestus the smith-god |
| 0640 | 0b |  | 18,0d | 0b | Athene is pleased |
| 064f | 0c |  | 18,0e 1c,07 | 0c | Prometheus, the creator of mankind |
| 0671 | 0d |  | 18,0e; 1c,08 | 0d | Poseidon, ruler of seas |
| 0676 | 0f |  | 18,0f;1c,0a | 0f | No mere mortal may enter the home of the gods. |
| 0685 | 08 |  | 18,0c; 1c,06 | 08 | Cheiron the centaur |
| 068a | 12 |  | 19,01  1c,0b  1c,0c | 12 | The Sphinx requests (Here you are requested type an answer to the riddle) |
| 06a1 | d5 |  | 1b,02  1d,0d | 55 | M A N, crawls on all four  (Correct answer to sphinxe’s question is “man”) |
| 06b0 | a1 |  | 19,08;1d,03 | 21 | The sphinx, mortified by your correct answer |
| 07f7 | 30 |  | 1a,00  1d,0c  1d,0d | 30 | Congratulations, Palaemon. You have survived |
| 0822 | 39 |  | 1a,04 | 39 | Trading vessels frequently |
| 0858 | 31 |  | 1a,00 1d,0d 1d,0e | 31 | You have now fully grown to an awesome man. |
| 0887 | 3a |  | 1a,05 | 3a | Those who offer nothing receives nothing |
| 08b8 | variable + 3b |  |  |  |  |
| 08e5 | 4c + content of $20 |  |  |  |  |
| 08fe | f5 |  | 1c,02  1e,0c 1c,07 | 75 | “I see strong omens ... |
| 0914 | f6 |  | 1c,03  1c,07 | 76 | “I have read for you all the omens ... |
| 092c | content of 9f32 |  |  |  |  |
| 095b | 64 |  | 1b,0a  1e,02  1e,03 | 64 | The ground under your feet feels soft |
| 0972 | 65 |  | 1b,0a  1e,0a | 65 | Medea falls in love with you |
| 099a | 66 | Low portion of 1b,0b | 1b,0b  1e,0a | 66 | Dryope and her sister nymphs |
| 09ba | 67 | High portion of 1b,0b | 1b,0b | 67 | The gates of troy are thrown open. |
| 09c4 | e8 |  | 1b,0c;1e,0f  1e,00; | 68 | You have fought a long hard war. |
| 0a06 | 2f |  | 19,0f;1c,05 | 2f | The trojans have pulled the great wooden horse |
| 0a0e | 69 |  | 1b,0c  1e,02 | 69 | Cassandra speaks |
| 0a18 | ea |  | 1b,0d  1e,08 | 6a | Penelope thanks you |
| 0a50 | 6b |  | 1b,0d | 6b | You are irresistibly swept into a whirlpool. |
| 0a92 | 6c |  | 1b,0e; 1e,09 | 6c | Circe serves you a mess of cheese. |
| 0abe | 6d |  | 1b,0e;1f,03  1f,04  1f,05 | 6d | On the way in, you were greeted |
| 0b2b | 25 |  | 19,0a;1d,05 | 25 | Pausing for breath |
| 0b46 | 2d |  | 19,0e  1d,0a | 2d | You fall asleep in a small cave |
| 0b59 | ee |  | 1b,0f | 6e | The treasure of Stymphalus is yours! |
| 0b72 | 6f |  | It doesn’t load anything! | 6f | A magnificent sword. |
| 0b84 | f0 |  | 1c,00  1e,0b | 70 | You stealthily unfasten the fleece |
| 0bb1 | 71 |  | 1c,00; 1f,05  1f,06 | 71 | You find a goodly lot of jewels |
| 0bcc | 72 |  | 1c,01; 1f,06  1f,07  1d,03 | 72 | Before you is a wholesome looking closter of sweet |
| 0bde | f3 |  | 1c,01 | 73 | Praises to Zeus |
| 0bfa | f4 |  | 1c,02 | 74 | The marvellous cattle of Geyron |
| 0ddf | a6 |  | 19,0b; 1d,06 | 26 | You sacrifice the slain serpent |
| 0dfa | a7 |  | 19,0b;1d,07 | 27 | The hide of the lion |
| 0e14 | f8 |  | 1c,04 | 78 | The wild mares of Thrace are defeated! |
| 0e39 | ab |  | 19,0d  1d,09 | 2b | To the victor belongs the spoils |
| 0e7c | 11 |  | 19,00  1e,08; 1e,09 | 11 | “Thank you” says Phineas. |
| 7e16 | 32 + content of 83e7 |  |  |  |  |
| 7e30 | 0a |  | 18,0d | 0a | “I refuse to serve a breast such as you. Begone! ” |
| 7e43 | 77 |  | 1c,03 | 77 | You have such a powerful blend ... |
| 833c | 00 |  | 18,08; 1d,04 | 00 |  |
| 83a9 | 03 |  | 18,09 | 03 | I have never seen anyone as clumsy as you |
| a9bb | 2e |  | 19,0f  1d,0b | 2e | The ‘eternal life’ granted by Zeus |
| b4b4 | 84 |  | 18,0a  1c,04  1c,05 | 04 | Congratulations! You have completed all of the twelve tasks. |
| e5e6 | 23 |  | 19,09; 1d,04 | 23 | The people of Athens place you on trial |
| e737 | a2 |  | 19,09 | 22 | The age of heroes is over. |
| e83d | Very special |  |  |  |  |
| ed12 | 09 |  | 18,0c; 1c,07 | 09 | Janus, jealous god of gateways |
| f032 | 1A + something |  |  |  |  |
| f08d | 20 |  | 19,08 | 20 | The fates are with you. |
| f7a6 | 02 |  | 18,09 | 02 | A weak link in the victims defence |
| f7c2 | 01 |  | 18,08 | 01 | The weapon is vehemently impaled. |
| f7e7 | Special. |  |  |  |  |

In all, the sectors in question are the ones from 18,08 to 1f,07 (sector up to f only) so 112 disk sectors in total, but twice the internal references as each disk sector makes up two $9f80 sectors.

Solution:

Capture the call at 9c62 and make a JSR to this:

jsr $850a ; set buffer pointers

jsr $84e4 ; last sector?

bcc a1

jsr $85cf ; disable sprites - optional?

// Logic for converting internal pointer to the file name

lda $84c3

clc

sbc #$10

lsr ; divide by 2 - file number is half of $80 block number we look for.

// Now A contains a value that is the same as the index of sectors counted from 18,08 to 1f,07

// (up to f sectors per track only)

jsr LoadAndDepackSector

jsr $85db ; enable sprites - optional?

jsr $8539

lda #$00 ; Status of loading. Carry signals error and $85cb contains type of error

sta $85cb

clc

a1 rts

## Trainers

When it’s your turn, $2e/2f points to your character (2ce2 and onwards)

m 002e

>C:002e 22 2d (lo/hi to your character)

00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13

2d22 e8 46 00 21 1f 1f 5f 9f 39 0f 06 1f 9f 45 04 03 ef 46 07 3f

00,01 Lo/hi name

02 ?

03 ?

04 Cursed beast (max 1f)???

05 Power (max 1f)

06 Health (max 3f)

Bit 0-5 - Health (up to $3f)

Bit 6 - Badly wounded

07 Sword type (Lo nibble is the sword type and high is the skill)

Bil 0-2 - Sword type

Bit 3 - Zeus blessed sword

Bit 4 - Skilled fencer

Bit 5 -?

Bit 6 - Poisoned sword

Bit 7 - Fine sword

08 Armor (max 38)

Bit 0-2 - Armour

Bit 4 Enchanted heavy armour.

09 Close fighting skills/knife

Bil 0-2 - Knifetype

Bit 3 - Zeus blessed knife

Bit 4 - Skilled wrestler

Bit 5 - Nothing ?

Bit 6 - Poisoned dagger

Bit 7 - Fine dagger

0a ?

0b Masterfully Dextrous (max 1f)

Bit 0-5 Dexterity

Bit 6 Adept at doging foes

0c Speed - Turns (max 1f)

Bit 0-3 Speed

Bit 5-7 Adds some sort of speed (swift as an arrow)

0d,0e Lo/hi money

Moves are decreased at $ea3b

d ea3b

.C:ea3b CE 28 9F DEC $9F28

This is also the place where the computers moves are decreased to NOP:in this will not work.

b8bf contains the number of completed tasks. Up to $0c. jmp $b4b2 - loads the Zeus picture showing you completed the game.

Notes for trainer:

Several of the on-screen options have a string and then a reference to the string that + the address for the function.

Desert - string 13b4

Know Thyself 1448 (1047 -> JSR $E5A9)

JSR $SADFC - Screen turns red

JSR $8DA8 - Shows the data

JSR $AE0E

jmp $E8F7

Other options - 1435 (1051 -> JSR $11c2)

Menu of other options - 157d

JSR $0ee2 prints the menu

We have chosen to make a trainer that is MIGHTY kewl.

We hijack the “Know thyself” menu (jsr $0334 at e5af). The main routine for the trainer is at $0334. This is the part that prints the screen and handles the initial keypresses. A set of utility routines is placed at $0200. This is the ones that print a hex value and an On/off value. All subscreens are place in the IFFL file. They are loaded using the routine that prints information on the screen which find the text in the top part of the 256 bytes loaded. The rest contains the trainer code associated with the text loaded. So it’s fair to say that the trainer is based on a plugin system.

One less nice thing is that I didn’t find place to print the current value, so it’s first printed when you change it. Sorry about that.