HD-Park-Switch

(or How to Patch a CMD-HD to your own needs)

- By Ninja/The Dreams
- Originally published in Domination #17
- Converted to ascii by Jazzcat/Onslaught

One thing I really like about the C64 nowadays is that it is a quiet computer. No fans or similar, just wonderful! Unfortunately, the SCSI-HDD inside my CMD-HD is the opposite. When it is running, it sounds like an aeroplane. Being a curious programmer, I tried to get rid of this annoyance. I realised that I never use the 'Write protect' button. So, maybe I could abuse it to park/unpark my HDD?

Well, the forthcoming project might not be too useful for most of you. Nevertheless, it might give you an idea how the CMD-HD works and how to apply own patches to the CMD-HD-ROM. You never know when you might need that!

How to do

First of all, the HD-ROM is not really ROM but in fact RAM which can be protected from storing data to it. This makes sense, as the HD-KERNAL has to be loaded from the system partition when the HD boots up. Furthermore, upgrading the HD-DOS does not require hardware modifications. Ofcourse, it also means that applying patches is pretty easy: unprotect RAM, modify KERNAL, protect RAM. If the 1541 had such capabilities...

The patch itself is quite simple. Install a backpack to that point where the 'Write protect'-flag was toggled. From there, send the corresponding SCSI-jobcode to park/unpark the HD-mechanism. Finally go back to the standard procedure.

ROM-versions 1.86, 1.90 and 1.92 are handled (are there any more though?), though only 1.92 was tested. As we do just easy stuff, I do not expect many problems with those older versions. For the rest, I will let the source-code speak (I assume you know a little about sending and executing drive code. All necessary information was re-engineered (and that was the main work) by me or Doc Bacardi/The Dreams

Enjoy and comments are welcome.

The code

```
; HD-Park-Switch V1.0 by Ninja/The Dreams in 2002

org $0801

binclude "help/hdpshead.prg",2
; include BASIC-header, which contains
```

```
; some information and will start the
; following routines.
; Works in 64 and 128-mode!

align 256    ; start at beginning of a page

jmp in:

lda #$0f     ; channel #15
ldx $ba       ; use current device
tay              ; use command channel
jsr $ffba    ; set file-parameters

lda $fff6
cmp #$ff    ; check platform
bne c64_found    ; c64, then jump

lda #$0f
tax              ; set memconfig for channel
jsr $ff68    ; in C128-mode
ldy $2e        ; get C128-BASIC-start
byt $2c       ; skip next opcode
c64_found:

ldy $2c       ; get C64-BASIC-start
iny              ; increment to point to this
                   ; page

ldx #lo (mw-command)  ; lobyte of
                                          ; command
lda #hd_code_len+6   ; we send all
                                         ; bytes at once
jsr $ffbd   ; set up memory-write-
                    ; command
jsr $ffc0   ; send command

ldx #$0f
jsr $ffc6   ; set channel as input
jsr $ffcf    ; get char
cmp #'0'    ; "0" from OK-string?
bne drive_err   ; if not, skip execution
                          ; will probably be a non
                          ; CMD-HD-drive
                          ; complaining about the
                          ; long command string

ldx #$0f
jsr $ffc9        ; channel as output
lda #'U'
jsr $ffd2
```

```
lda #'3'
jsr $ffd2        ; send "U3", executes at
                       ; $0500


drive]err:

jsr $ffcc        ; restore input/output
lda #$0f
jmp $ffc3       ; close channel and go
                       ; back

mw]command:
byt "M-W",0,5,hd_code_len

hd]code:
phase $0500   ; HD-code is at $0500

sei                   ; no interrupts
ldy #2             ; check 3 ROM versions
                       ; (1.86, 1.90, 1.92)

next_rom:
ldx rom_ofs,y  ; get version-specific
                          ; offset into x
lda #$4c           ; $4c = JMP opcode
cmp $f28a,x     ; present in ROM?
bne wrong_rom  ; no, then next version
sta bp_mod+1    ; store address]lo into
                            ; our backpack


lda rom_jmplo,y  ; get version-specific
                               ; offset into x
lda #$4c               ; $4c = JMP opcode
cmp $f28b,x         ; present in ROM?
bne wrong_rom   ; no, then next version
sta bp_mod+1    ; store address]lo into
                            ; our backpack

lda_rom_jmphi,y  ; get version-specific
                                ; address]hi
cmp $f28c,x           ; present in ROM?
beq rom_found      ; yes, then go patch

wrong_rom:
dey                      ; try next version
bpl next_rom     ; still one left?
cli
rts                      ; no, then goodbye
                            ; without changes
```

```
rom_found:
sta bp]mod+2    ; store address]hi into
                            ; our backpack
lda $8f00
ora #20
sta $8f00            ; unprotect RAM
ldy #bp_len-1

copy_bp:
lda backpack,y
sta $ff60,y
dey
bpl copy]bp     ; copy backpack to $ff60

lda #$60
sta $f28b,x
lda #$ff            ; apply JMP $ff60 to
                          ; version-specific
sta $f28c,x       ; address

lda $8f00
and #$df
sta $8f00          ; protect RAM
cli
rts                     ; goodbye

backpack:
lda #$f8     ; SCSI_Jobcode 'Start
                   ; Device'
bit $49        ; Check for 'Write Protect'
bpl wp_disabled   ; disabled, then skip
                                ; next opcode
lda #$fa  ; SCSI-Jobcode 'Stop Device'

wp_disabled:
sta $20  ; into Native-Job-Que
bp]mod:
jmp $ffff ; back to original routine self-
                 ; modified from above
bp_len     = *-backpack


;           V1.86            V1.90          V1.92

roms_ofs:
byt $f28a-f28a,$f2f6-f28a,$f331-f28a
; where to patch

rom_jmplo:
byt        $be      ,        $2a      ,      $65
```

```
rom_jmphi:
byt        $f2      ,        $f3       ,      $f3
; what to patch

dephase
hd_code_len   = *-hd_code
end $0801
```