Word Juggler IIe v2.9
© 1982-1984, Quark, Inc.

This is a step-by-step guide to crack Word Juggler v2.9. The crack is for the US version of the program. As you own the German one, some addresses may vary and we'll tell where to pay attention and put the right values.

A message from an old man: "Protect your original disk against stupidity, put a sticker on the write notch."

Let's play…

1. Get my world famous copy disk @ http://www.brutaldeluxe.fr/products/apple2/diskcopiers.html and transfer it onto a real floppy thanks to the wonderful ADTPro software by David Schmidt. Then, boot the disk.

THE FIRST MILESTONE IS TO COPY THE ORIGINAL DISK

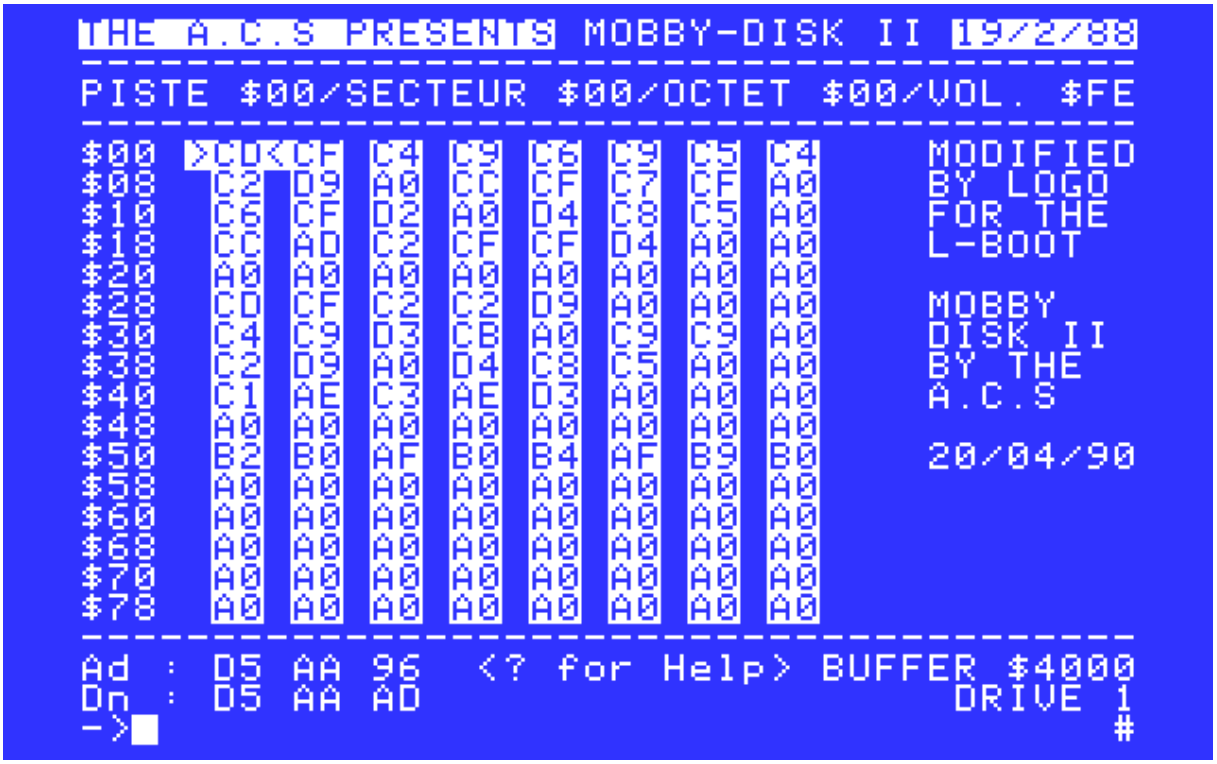
2. Let's make a copy of the original with Locksmith 6.3 F-disk Backup. Press 3, please.

3. If you have one drive only, press 11 then return. Otherwise, go to 4.

4. Your original disk is in drive 1, a blank 5.25” floppy is in drive 2. Perform the copy by pressing return. You’ll see that tracks 1 and 2 show only inverse A characters. The meaning is that Locksmith could not decode the address field of the sectors of these tracks.

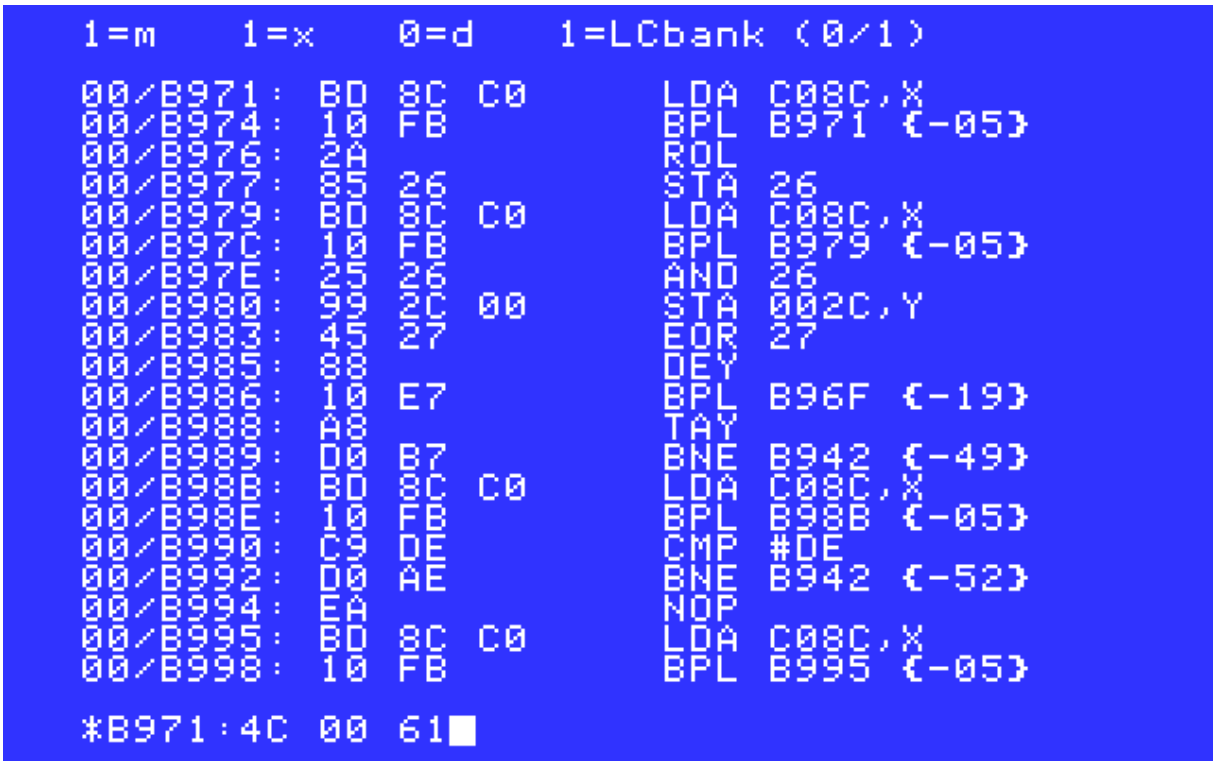
THE SECOND MILESTONE IS TO COPY THE TWO PROTECTED TRACKS

5. Now, press ctrl-openapple-reset to reboot and select Mobby Disk II.

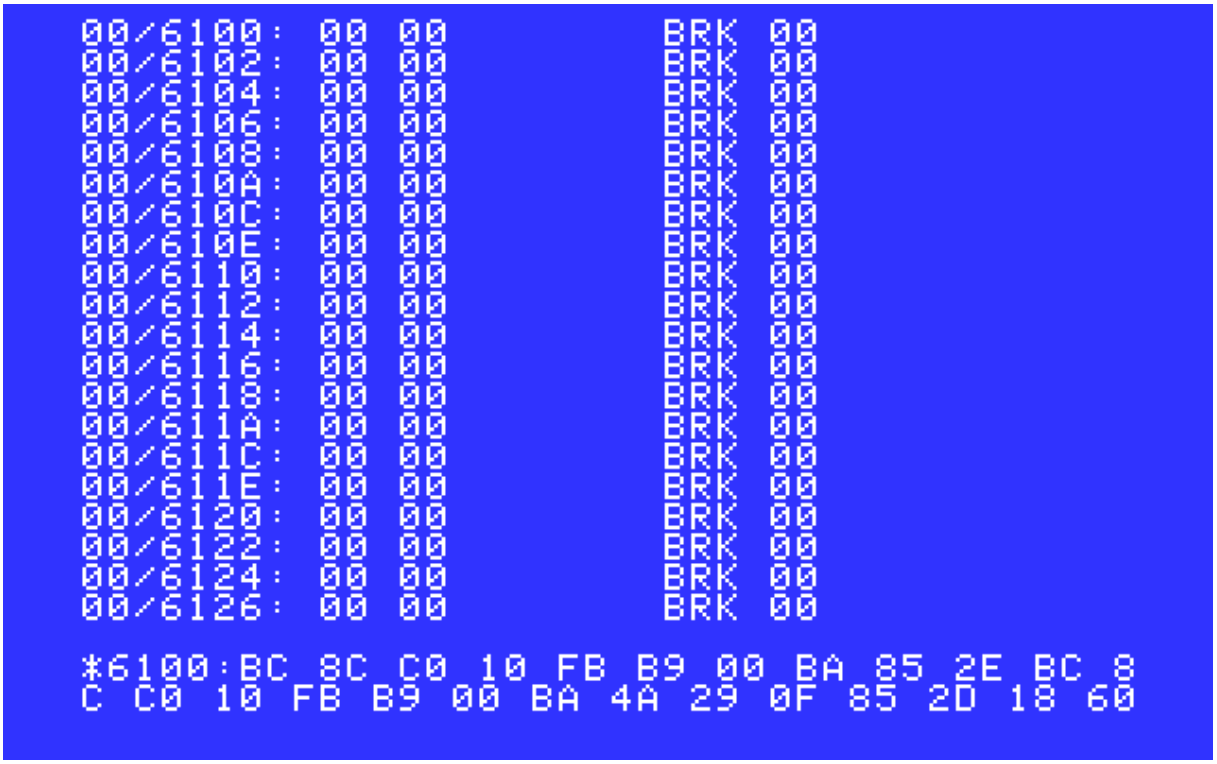


6. Press ctrl-E then space then space to enter the monitor

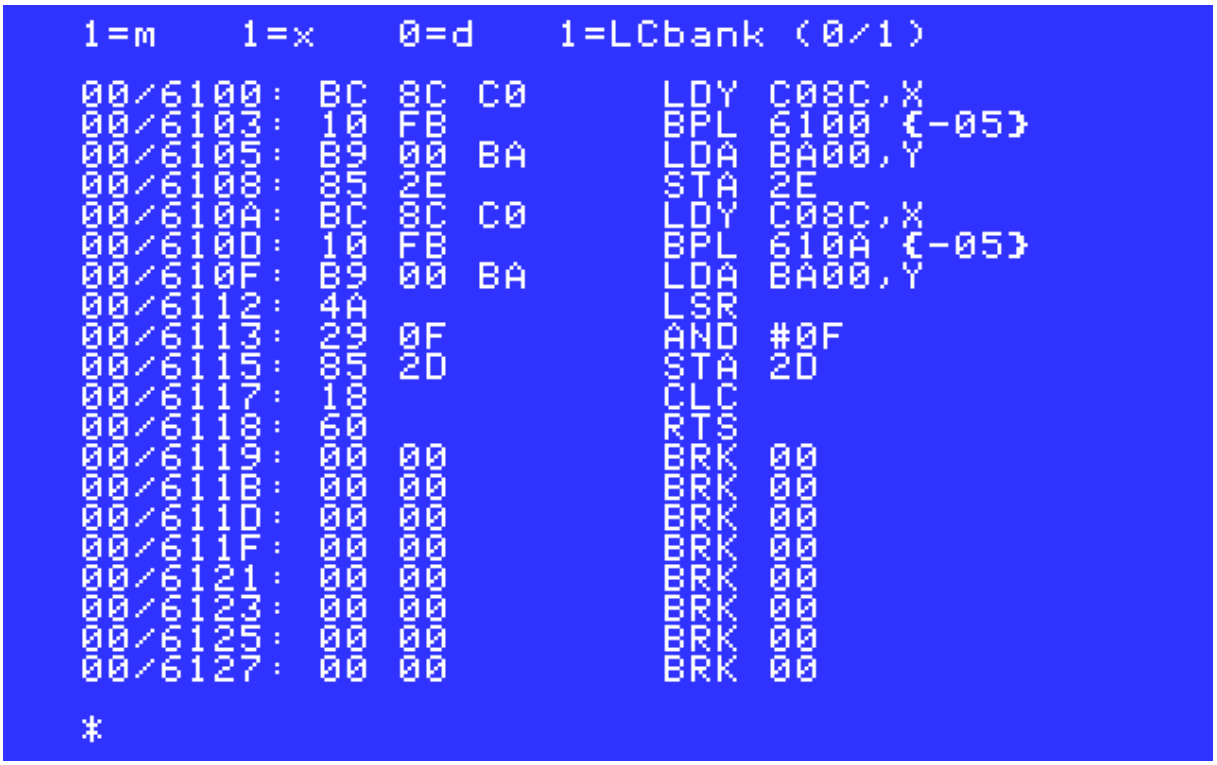
7. Once in the monitor, at $B971 replace BD 8C C0 with 4C 00 61 then press return



8. At $6100, enter the following code and press return



9. The disassembled version must look like the following picture:

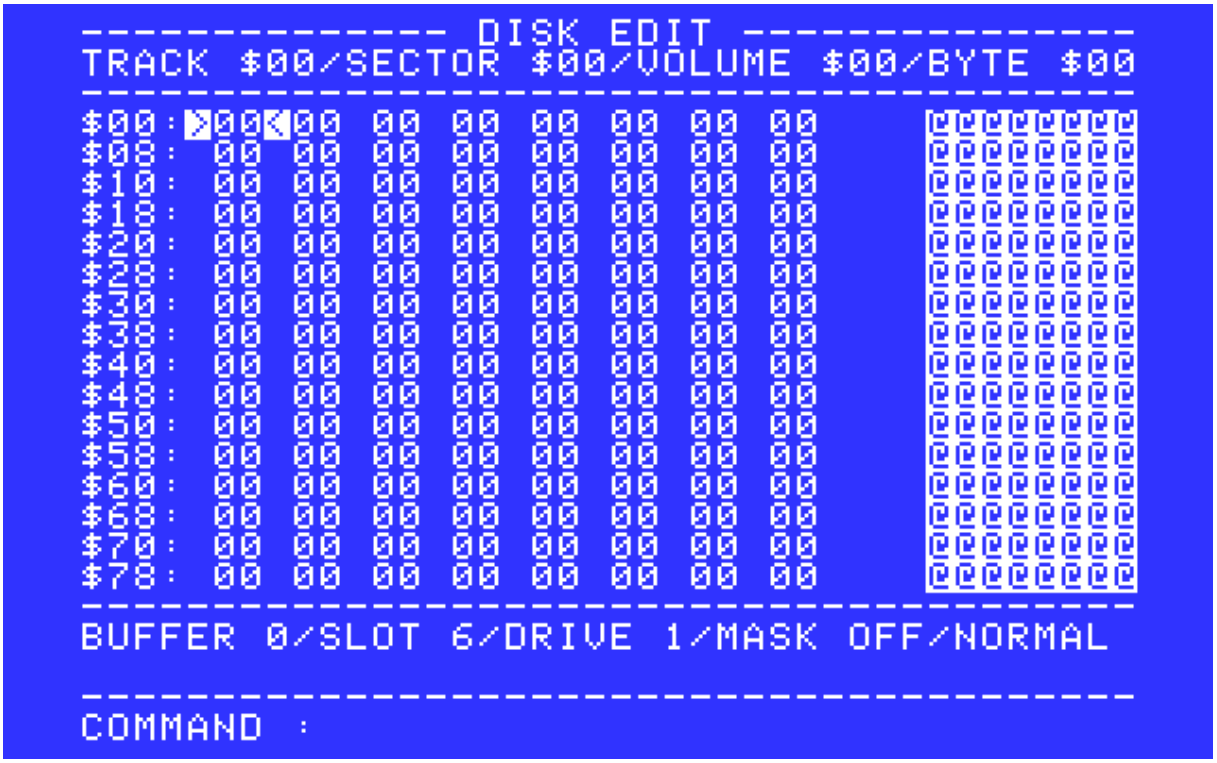


10. Let's read the two protected tracks:
    a. Press ctrl-y to return to Mobby Disk II
    b. Put the original disk in drive 1
    c. Press R to read track 1, sector 0
    d. If OK, press down-arrow then press the letter O to confirm that you want to read the entire track 1
    e. Press down-arrow then press the letter O to confirm that you want to read the entire track 2

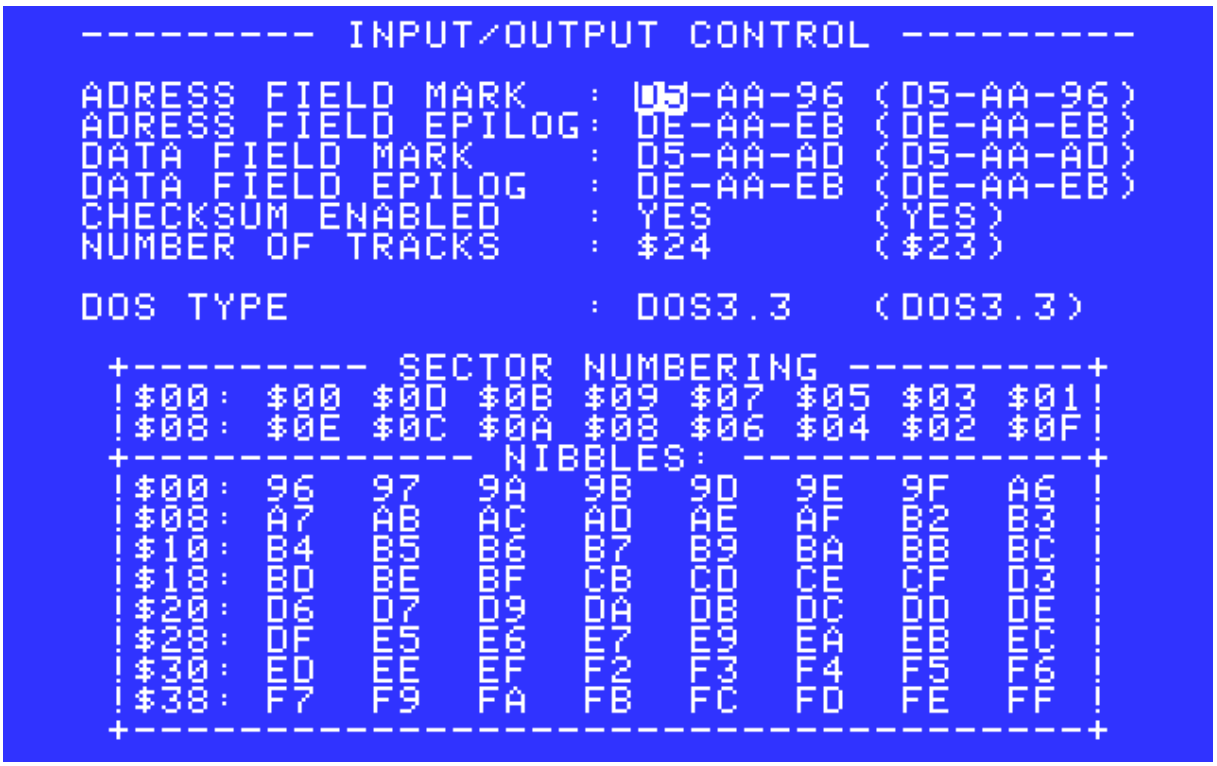
11. Let's write the tracks on our copy:
    a. Press ctrl-e, then space, then space to enter the monitor
    b. At $B971, put BD 8C C0 then return
    c. Press ctrl-y to return to Mobby Disk II
    d. Put the copy disk in drive 1
    e. Press W and write track 1, sector 0
    f. Press ctrl-W then press the letter O to confirm that you want to write the entire buffer $4000..$4FFF to track 1
    g. Press ctrl-W then press the letter O to confirm that you want to write the entire buffer $5000..$5FFF to track 2

THE THIRD MILESTONE IS TO NORMALIZE THE DISK

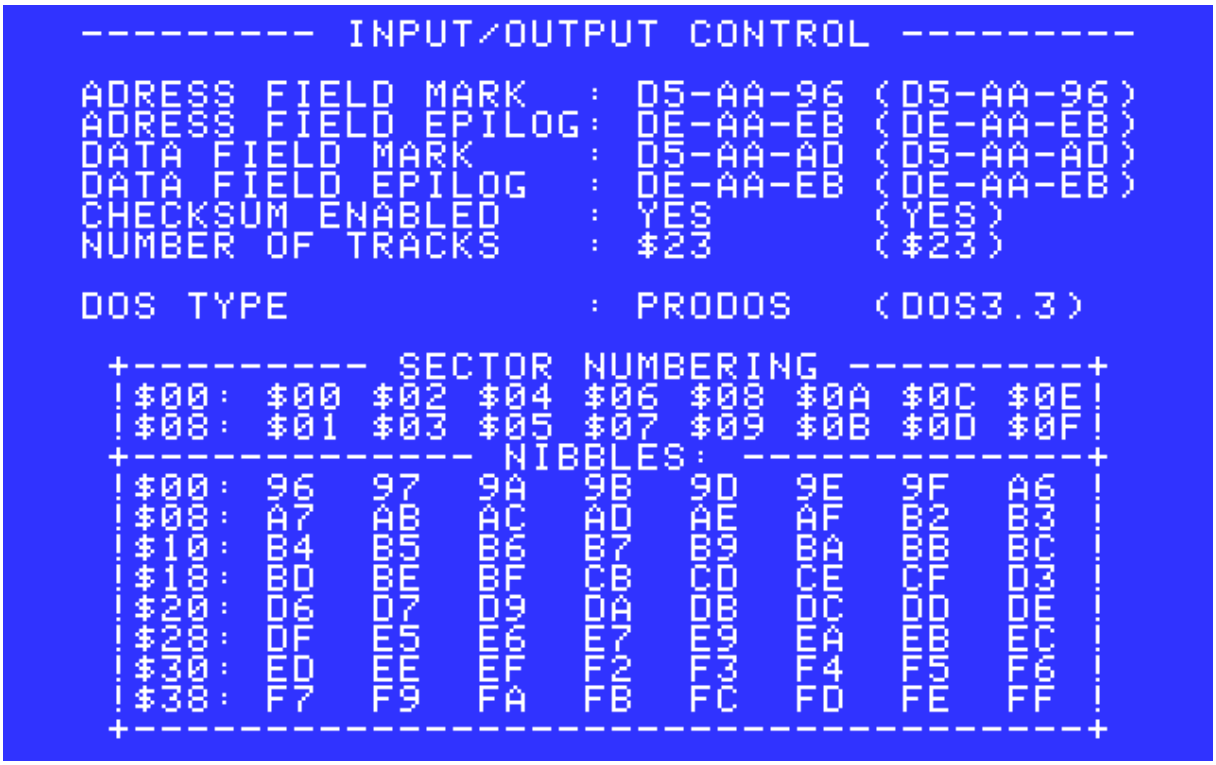
12. Now, press ctrl-openapple-reset to reboot and select Disk Fixer V4.0 (the world's most powerful 5.25" sector editor on Earth)



13. As the Word Juggler disk is ProDOS based, we'll tell Disk Fixer to use the ProDOS interleaving. Press the letter O to enter the Options menu:



14. Now, press return then right arrow to move the cursor to the DOS TYPE entry. Once there, press P (for ProDOS ;-)) then press escape two times (note that I've also changed the number of tracks from $24 to $23 but you do not care, it is useful if you want to use the Find feature of the program)



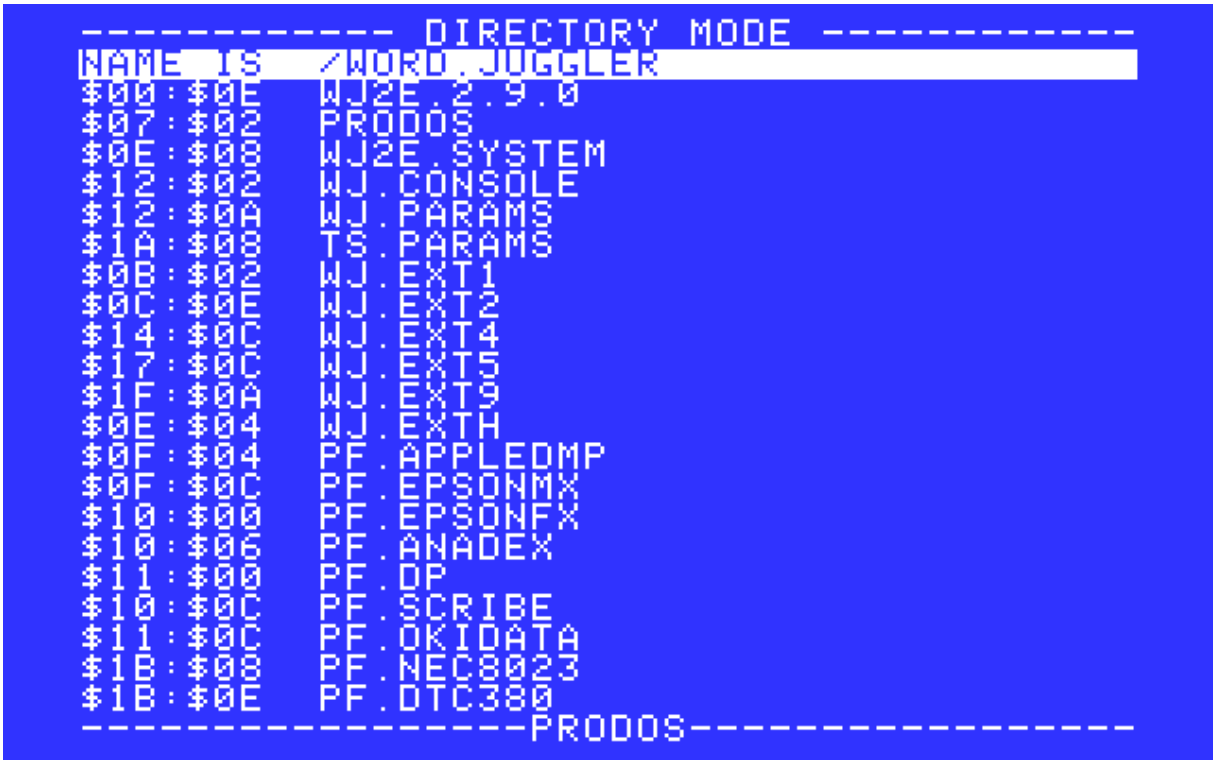
15. We'll make changes to the WJ2E.SYSTEM file. Press D to enter the Directory feature of Disk Fixer. Select WJ2E.SYSTEM and press return.

The WJ2E.SYSTEM file contains strings in English. It is important to recall (I wrote that earlier) that some addresses may change due to your German version.

The WJ2E.SYSTEM file checks the computer the program is launched on, clears the screen, displays some messages and loads the WJ2E.2.9.0 file that is located on the first blocks of the ProDOS disk. The two formerly protected tracks are part of the file (and vice versa).

We must change some bytes in the WJ2E.SYSTEM file because it still contains the code to read protected tracks and our tracks are now readable. We must tell the program!

A ProDOS-based title has the advantage of (normally) being hard-disk-drive installable. That is not the case here, where tracks and sectors are read as if we were DOS 3.3-based. *Boo Quark*

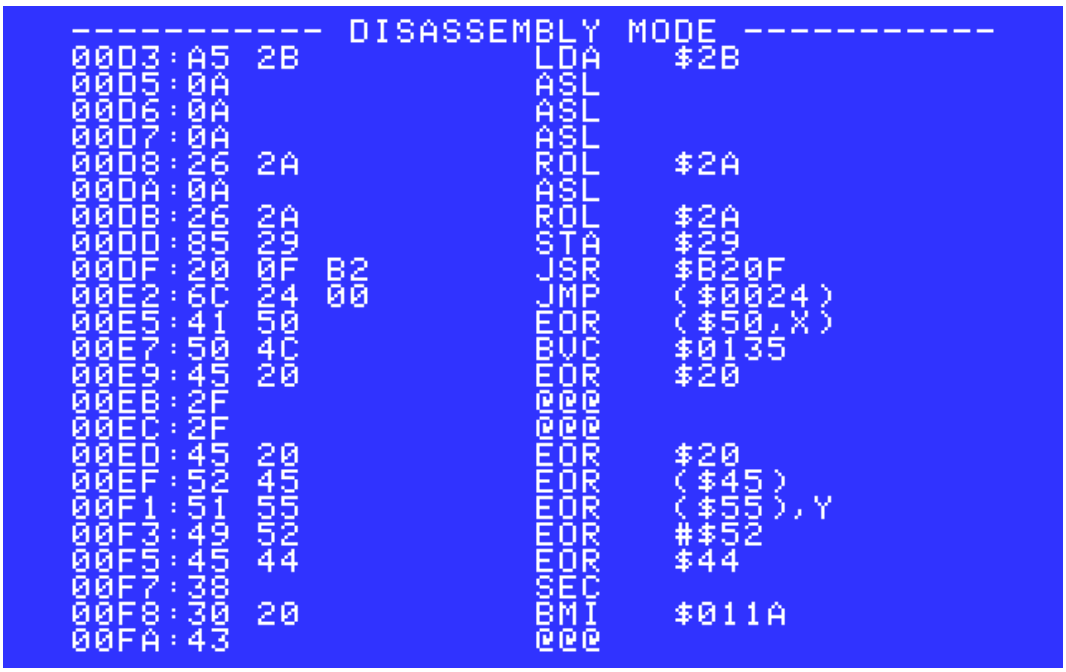


16. This is the first block of the WJ2E.SYSTEM file (note that I've pressed Y to remove the inverse characters on the right side of the window)

17. The first change: remove the call to the read sector routine.

This is on the second half of the first block. So press right-arrow once to reach that part. Then, press M to move to offset $DF:

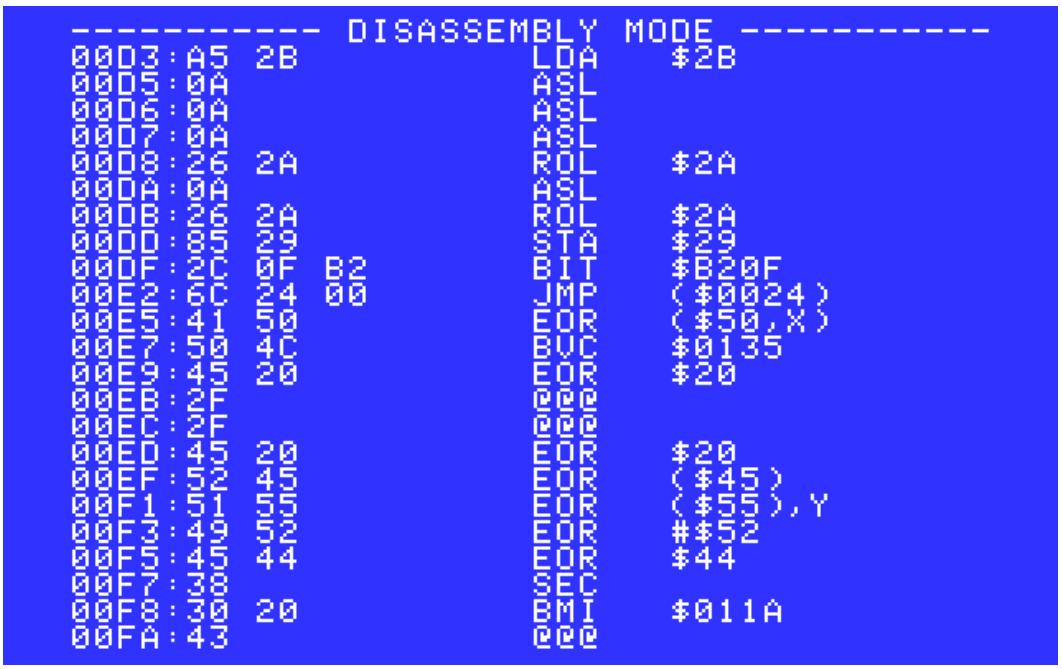
The same sector as viewed from the Disassembly mode (press L). You can see that at offset $DF we have the 20 0F B2 (or JSR $B20F or call subroutine located at $B20F)



Warning: the offset may vary on your German version. If so, search for the code shown at $D3 above (A5 2B 0A 0A 0A 26 2A 0A 26 2A 85 29) and put a 2C instead of a 20 one byte after.

18. We must change the 20 (JSR) with a 2C (BIT) to bypass that call. Exit the disassembly mode by pressing the escape key. Move with I J K M to offset $DF, press return and type 2C. Then press the escape key to exit the edit mode. Press W and press return two times to write the sector back onto disk.

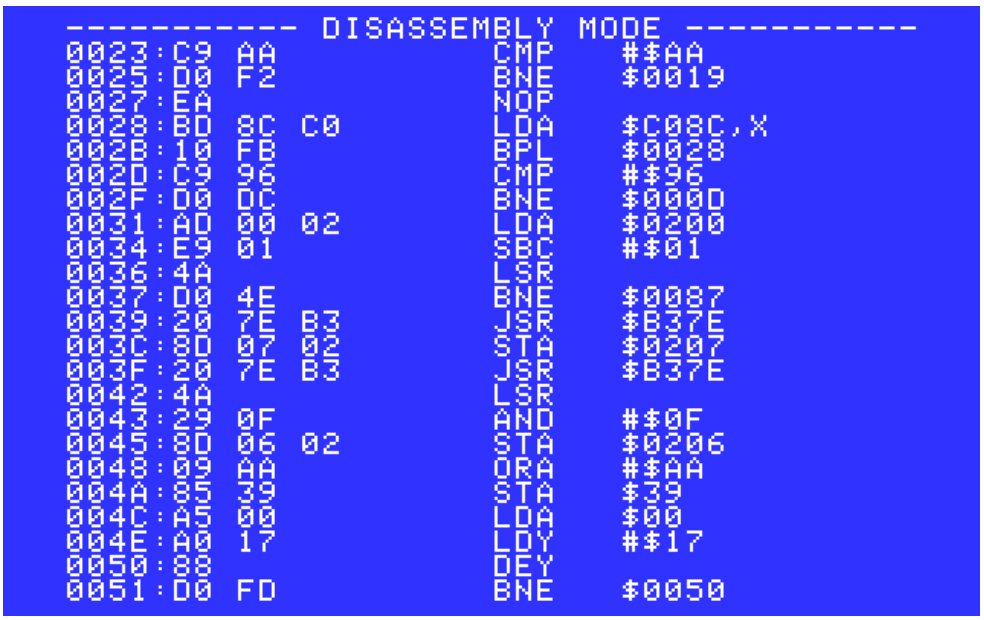
This is the disassembled view of the modified sector. At offset $DF, the JSR was changed to a BIT:



19. The second change: force the address field read routine to always read a standard address field. Ahem! See the last part of the guide to understand!

Press right-arrow to move to the next block (see in the lower right corner of the window, "/$01" means that we are on block 1 of the WJ2E.SYSTEM file). Move to offset $23 and press L to view the disassembled listing.

At offset $31, the code reads the address $200, subtracts 1 to it, divide it by 2 and if the value is non-zero, it branches to offset $87, otherwise it continues at offset $39.

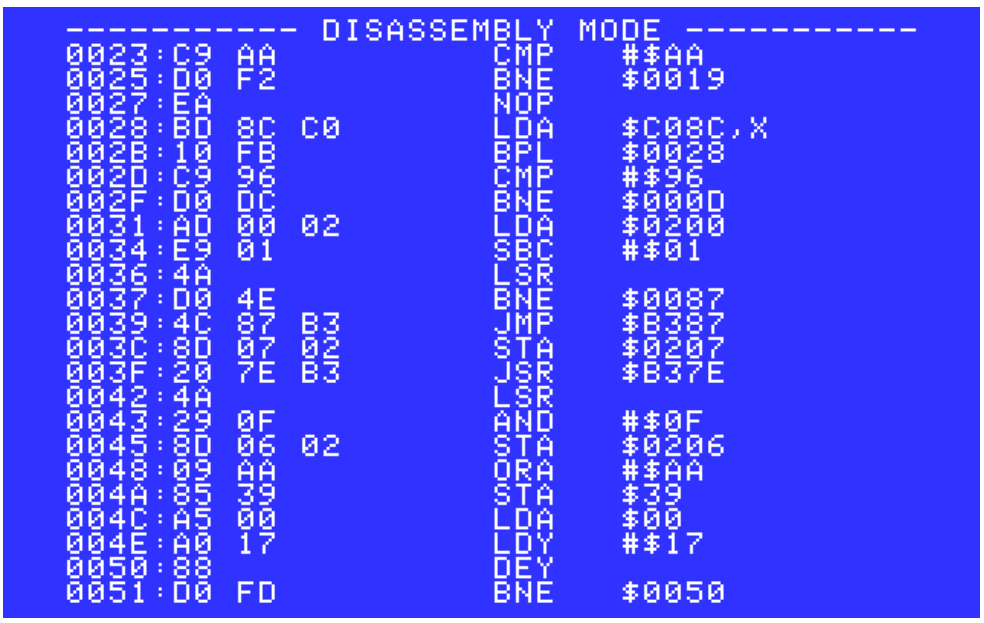


Warning: the offset may vary on your version. If so, search for the code shown at $31 (AD 00 02 E9 01 4A D0 4E) and put a 4C xx one byte after. Xx is the value shown to the right of offset $37. On the US version, it shows "BNE $0087" so xx is 87. If your version shows "BNE $0089" then xx is 89.

20. We must make the change after the code shown above to always call the standard address field read routine instead of the modified one for tracks 1 and 2.

Move to offset $39 and replace the 20 7E values with 4C 87 and write the sector back onto disk. To change the values, press return once at offset $39 to enter the edit mode, type 4C 87 then press escape to exit the edit mode. Press W then return two times to write the sector.

The same sector once modified in the disassembled mode (press L):



You can see at offset $39 that the JSR was replaced with the JMP to the same address as offset $39. It is the same address as that code runs at address $B300. So “BNE $0087” and “JMP $B3B7” go to the same address.

CONGRATULATIONS! YOU'VE CRACKED YOUR COPY

Let's share some information about the protection type. A standard 5.25" 16-sector floppy disk is divided into 35 tracks of 16 sectors and the data stored on disk is stored in a different way than the bytes in memory. We call them nibbles.

Each track/sector information is stored in an address field and a sector data is stored in a data field. Both in a format that is well designed and known.

A standard address field looks like this:

- header markers: D5 AA 96
- volume information: AA AA (coded in 4*4 format, only 4 bits of each nibble contain valid data)
- track information: AA AA (ditto)
- sector information: AA AA (ditto)
- checksum: AA AA (ditto)
- epilog markers: DE AA (with a final EB but due to a bug, it is not written)

What Quark did for tracks 1 and 2 is a change of the address field:

- header marker: D5 AA 96
- track information on one nibble
- sector information on one nibble
- some bits to desync and lose time
- sector information in AA format on one nibble
- checksum on one nibble
- epilog markers: DE AA

The second change we did was to force the usage of the standard address field read routine whatever the track we are in.

Reboot and… enjoy,

LoGo
5/2017

AS USUAL, SOME CODE

```
0031:AD 00 02       LDA   $0200         ; get track
0034:E9 01          SBC   #$01          ; -1
0036:4A             LSR                 ; /2
0037:D0 4E          BNE   $0087         ; branch if >0

* Read a modified address field

0039:20 7E B3       JSR   $B37E         ; read nibble
003C:8D 07 02       STA   $0207         ; save as track
003F:20 7E B3       JSR   $B37E         ; read nibble
0042:4A             LSR                 ; /2
0043:29 0F          AND   #$0F          ; keep lower 4 bits
0045:8D 06 02       STA   $0206         ; save as sector
0048:09 AA          ORA   #$AA          ; make it 4*4
004A:85 39          STA   $39           ; save it
004C:A5 00          LDA   $00           ; waste time
004E:A0 17          LDY   #$17          ; ..
0050:88             DEY                 ; ..
0051:D0 FD          BNE   $0050         ; ..
0053:BD 8C C0       LDA   $C08C,X       ; read nibble
0056:10 FB          BPL   $0053
0058:C5 39          CMP   $39           ; same as requested?
005A:D0 20          BNE   $007C         ; nope, error
005C:20 7E B3       JSR   $B37E         ; read nibble
005F:4D 06 02       EOR   $0206         ; checksum
0062:4D 07 02       EOR   $0207         ; checksum
0065:D0 15          BNE   $007C         ; error, branch
0067:BD 8C C0       LDA   $C08C,X       ; get marker
006A:10 FB          BPL   $0067
006C:C9 DE          CMP   #$DE
006E:D0 0C          BNE   $007C
0070:18             CLC
0071:BD 8C C0       LDA   $C08C,X       ; and 2nd marker
0074:10 FB          BPL   $0071
0076:49 AA          EOR   #$AA
0078:85 2C          STA   $2C           ; save it
007A:F0 01          BEQ   $007D
007C:38             SEC                 ; error
007D:60             RTS                 ; or no error if c=0

007E:BC 8C C0       LDY   $C08C,X       ; read a nibble
0081:10 FB          BPL   $007E         ; thanks to the denibblize
0083:B9 00 B5       LDA   $B500,Y       ; table, transform it
0086:60             RTS

* Read a standard address field

0087:A0 03          LDY   #$03          ; the standard address
0089:A9 00          LDA   #$00          ; field read routine
008B:38             SEC
008C:85 39          STA   $39
008E:BD 8C C0       LDA   $C08C,X
0091:10 FB          BPL   $008E
```

```
0093:2A             ROL
0094:85 3A          STA   $3A
0096:BD 8C C0       LDA   $C08C,X
0099:10 FB          BPL   $0096         ; $205: checksum
009B:25 3A          AND   $3A           ; $206: sector
009D:99 05 02       STA   $0205,Y       ; $207: track
00A0:45 39          EOR   $39           ; $208: volume
00A2:88             DEY
00A3:10 E7          BPL   $008C
00A5:A8             TAY
00A6:F0 BF          BEQ   $0067
00A8:D0 D2          BNE   $007C         ; until here

*--- The denibblize table

00/B590:00 00 00 00 00 00 00 01-........
00/B598:98 99 02 03 9C 04 05 06-........
00/B5A0:A0 A1 A2 A3 A4 A5 07 08- !"#$%..
00/B5A8:A8 A9 AA 09 0A 0B 0C 0D-()*.....
00/B5B0:B0 B1 0E 0F 10 11 12 13-01......
00/B5B8:B8 14 15 16 17 18 19 1A-8.......
00/B5C0:C0 C1 C2 C3 C4 C5 C6 C7-@ABCDEFG
00/B5C8:C8 C9 CA 1B CC 1C 1D 1E-HIJ.L...
00/B5D0:D0 D1 D2 1F D4 D5 20 21-PQR.TU !
00/B5D8:D8 22 23 24 25 26 27 28-X"#$%&'(
00/B5E0:E0 E1 E2 E3 E4 29 2A 2B-`!"#$)*+
00/B5E8:E8 2C 2D 2E 2F 30 31 32-(,-./012
00/B5F0:F0 F1 33 34 35 36 37 38-01345678
00/B5F8:F8 39 3A 3B 3C 3D 3E 3F-89:;<=>?
```