Sleep/Wake Probleme debuggen

Beitrag von „Tirom“ vom 8. Juli 2020, 18:14

Hallo Forum!

Ich arbeite gerade an meinem Gigabyte X299X und scheitere leider am Sleep/Wake. Ich weiß, dass das ein schwieriges Thema ist und habe auch echt viel dazu gesucht und ausprobiert.

USB soll ja häufig ein Grund sein. Die XHCIs habe ich mit SSDTs passend nach XHCI, XHC1 und XHC2 umbenannt, danach den Quirk "XhciPortLimit" auf true gesetzt und die USBInjectAll.kext geladen und dann mit dem Hackintool die passende USBPorts.kext erstellt. Danach den Quirk und InectAll wieder entfernt und die USBPortskext in die config.plist eingetragen. In ioregistryexplorer sieht auch alles gut aus. Die Ports funktionieren auch wunderbar.

Leider will der Rechner dennoch nicht aus dem Sleep aufwachen. Er spring kurz an und erzeugt ein Kernel Panic bei schwarzen Bildschirm. Und hier weiß ich nicht weiter.

Daher wollte ich fragen, wie ich das Problem am besten einenge, um eine Lösung zu finden. Der Kernel Panic habe ich mir durchgelesen, aber nicht das Keyword gefunden, dass mich auf den richtigen Weg gebracht hat. Wie kann ich am besten vorgehen? Gibt es noch Logs, in denen ich etwas finden kann? Nutzt es die Debug-Version von OpenCore zu verwenden?

Gebt mir bitte einen kleinen Anstoss, um mich in die richtige Richtung zu bewegen

Vielen Danke

EDIT: Hier der Bericht vom Kernel Panic

Code

1. panic(cpu 0 caller 0xffffff801424aa3a): Kernel trap at 0xffffff8014266d57, type 13=general protection, registers:
2. CR0: 0x0000000080010033, CR2: 0xffffff873bfb9000, CR3: 0x000000001c07a000, CR4: 0x00000000003626e0
3. RAX: 0x000000007e008003, RBX: 0xffffff8014a5dc30, RCX: 0x00000000000000e2, RDX: 0x0000000000000000

4. RSP: 0xffffff874ea83bb0, RBP: 0xffffff874ea83be0, RSI: 0x0000000000000003, RDI: 0xffffff8014a5dbd0
5. R8: 0x0000000000000000, R9: 0xffffff801aab0034, R10: 0x0000000000000003, R11: 0x0000000000000001
6. R12: 0xffffff8014930022, R13: 0x0000000000000001, R14: 0x0000000000000000, R15: 0xffffff8014930008
7. RFL: 0x0000000000010046, RIP: 0xffffff8014266d57, CS: 0x0000000000000008, SS: 0x0000000000000010
8. Fault CR2: 0xffffff873bfb9000, Error code: 0x0000000000000000, Fault CPU: 0x0, PL: 0, VF: 0
9.
10. Backtrace (CPU 0), Frame : Return Address
11. 0xffffff8013f51220 : 0xffffff801411f5cd mach_kernel : _handle_debugger_trap + 0x49d
12. 0xffffff8013f51270 : 0xffffff8014258b05 mach_kernel : _kdp_i386_trap + 0x155
13. 0xffffff8013f512b0 : 0xffffff801424a68e mach_kernel : _kernel_trap + 0x4ee
14. 0xffffff8013f51300 : 0xffffff80140c5a40 mach_kernel : _return_from_trap + 0xe0
15. 0xffffff8013f51320 : 0xffffff801411ec97 mach_kernel : _DebuggerTrapWithState + 0x17
16. 0xffffff8013f51420 : 0xffffff801411f087 mach_kernel : _panic_trap_to_debugger + 0x227
17. 0xffffff8013f51470 : 0xffffff80148c27cc mach_kernel : _panic + 0x54
18. 0xffffff8013f514e0 : 0xffffff801424aa3a mach_kernel : _sync_iss_to_iks + 0x2aa
19. 0xffffff8013f51660 : 0xffffff801424a738 mach_kernel : _kernel_trap + 0x598
20. 0xffffff8013f516b0 : 0xffffff80140c5a40 mach_kernel : _return_from_trap + 0xe0
21. 0xffffff8013f516d0 : 0xffffff8014266d57 mach_kernel : _xcpm_perf_bias_set + 0x1c7
22. 0xffffff874ea83be0 : 0xffffff8014266feb mach_kernel : _xcpm_init + 0xab
23. 0xffffff874ea83c00 : 0xffffff8014257021 mach_kernel : _acpi_sleep_kernel + 0x441
24. 0xffffff874ea83c70 : 0xffffff7f96acec2a com.apple.driver.AppleACPIPlatform : __ZN23AppleACPIPlatformExpert13sleepPlatformEv + 0x204
25. 0xffffff874ea83cc0 : 0xffffff7f96ad2eab com.apple.driver.AppleACPIPlatform : __ZN12AppleACPICPU7haltCPUEv + 0x75
26. 0xffffff874ea83ce0 : 0xffffff801484cc30 mach_kernel : __Z16IOCPUSleepKernelv + 0x290
27. 0xffffff874ea83d40 : 0xffffff8014885e25 mach_kernel : __ZN14IOPMrootDomain15powerChangeDoneEm + 0xac5
28. 0xffffff874ea83de0 : 0xffffff80148173b7 mach_kernel : __ZN9IOService8all_doneEv + 0x767
29. 0xffffff874ea83e50 : 0xffffff801481414c mach_kernel : __ZN9IOService23actionPMWorkQueueInvokeEP11IOPMRequestP13IOPMWorkQueue + 0x86c
30. 0xffffff874ea83ea0 : 0xffffff80148117e0 mach_kernel : __ZN13IOPMWorkQueue17checkRequestQueueEP11queue_entryPb + 0xa0
31. 0xffffff874ea83ef0 : 0xffffff8014811679 mach_kernel : __ZN13IOPMWorkQueue12checkForWorkEv + 0xc9

32. 0xffffff874ea83f30 : 0xffffff801482cede mach_kernel : __ZN10IOWorkLoop15runEventSourcesEv + 0x11e
33. 0xffffff874ea83f70 : 0xffffff801482c4d6 mach_kernel : __ZN10IOWorkLoop10threadMainEv + 0x36
34. 0xffffff874ea83fa0 : 0xffffff80140c513e mach_kernel : _call_continuation + 0x2e
35. Kernel Extensions in backtrace:
36. com.apple.driver.AppleACPIPlatform(6.1)[3A6D8ECD-C39E-39C8-984A-2CC417488A56]@0xffffff7f96ac3000->0xffffff7f96b5dfff
37. dependency: com.apple.iokit.IOACPIFamily(1.4)[0A7D7382-66FE-391B-9F93-97A996256C25]@0xffffff7f95536000
38. dependency: com.apple.iokit.IOPCIFamily(2.9)[BE052F4D-9B80-3FCD-B36D-BACB7DEE0DF2]@0xffffff7f94aee000
39. dependency: com.apple.driver.AppleSMC(3.1.9)[4589419D-7CCC-39A9-9E2F-F73FE42DD902]@0xffffff7f959ff000
40.
41. BSD process name corresponding to current thread: kernel_task
42. Boot args: -v keepsyms=1 debug=0x100 alcid=7 darkwake=0
43.
44. Mac OS version:
45. 19F101
46.
47. Kernel version:
48. Darwin Kernel Version 19.5.0: Tue May 26 20:41:44 PDT 2020; root:xnu-6153.121.2~2/RELEASE_X86_64
49. Kernel UUID: 54F1A78D-6F41-32BD-BFED-4381F9F6E2EF
50. Kernel slide: 0x0000000013e00000
51. Kernel text base: 0xffffff8014000000
52. __HIB text base: 0xffffff8013f00000
53. System model name: MacPro7,1 (Mac-27AD2F918AE68F61)
54. System shutdown begun: NO
55. Panic diags file available: YES (0x0)
56.
57. System uptime in nanoseconds: 679704862822

Alles anzeigen

Beitrag von „kuckkuck“ vom 9. Juli 2020, 20:52

Moin, du machst da einige ACPI Kunststücke auf deiner EFI. Ist dir bewusst was die einzelnen

Eingriffe im Detail tun? Besonders mit SSDT-THUNDERBOLT.aml sehe ich Probleme.

An deiner Stelle würde ich mal ein paar ACPI Eingriffe deaktivieren und beobachten wie sich das auf den Sleep auswirkt.

Beitrag von „Tirom“ vom 9. Juli 2020, 23:45

Hallo kuckkuck!

Danke für den Hinweiß. Leider weiß ich das nicht im Detail. Ich versuche mich in das Thema einzulesen, finde da aber wenig Gutes. Das Video von al6042 kenne ich natürlich. Ich wünschte, jemand würde zum Beispiel die OC ACPI Exampels in einem Video Schritt für Schritt erklären. Ich glaube, dass da einigen Unwissenden mit geholfen wäre

Die Thunderbolt werde ich morgen als erstes mal deaktivieren. Die Datei stammt von CaseySJ. Thunderbolt funktioniert dank modifizierter Firmware und der SSDT auch wunderbar. Sogar mit Hotplug und so weiter.

Welche andere SSDT macht noch Kunststücken? Ich melde mich, nach meinem Test.

Nochmals Danke!

Beitrag von „kuckkuck“ vom 10. Juli 2020, 00:04

SSDT-RTC0 ist ein Ersatz für SSDT-AWAC, wenn STAS nicht existiert;

SSDT XHC wird teilweise durch SSDT-Thunderbolt ausgehebelt, da sich Thunderbolt mit weit mehr als nur Thunderbolt beschäftigt und teilweise Werte für _UPC setzt;

Rename zu XHC1 bewirkt dass Apples Mechanismen auf den Controller matchen, deswegen

benennt man die Controller zu zB XHC um;

SSDT DTGP ist kosmetisch, aber wird wohl von ANS (kosmetisch?) und Thunderbolt referenziert;

Einige Renames sind etwas gewagt oder kosmetischer Natur.

Beitrag von „Tirom“ vom 11. Juli 2020, 15:24

Hallo kuckkuck !

Danke für deine Hilfe!

STAS existiert bei mir, richtig?

Code

```
1. Device (RTC)
2. {
3. Name (_HID, EisaId ("PNP0B00") /* AT Real-Time Clock */) // _HID: Hardware ID
4. Name (_CRS, ResourceTemplate () // _CRS: Current Resource Settings
5. {
6. IO (Decode16,
7. 0x0070, // Range Minimum
8. 0x0070, // Range Maximum
9. 0x01, // Alignment
10. 0x02, // Length
11. )
12. IO (Decode16,
13. 0x0074, // Range Minimum
14. 0x0074, // Range Maximum
15. 0x01, // Alignment
16. 0x04, // Length
17. )
18. IRQNoFlags ()
19. {8}
20. })
21. Method (_STA, 0, NotSerialized) // _STA: Status
22. {
```

```
23. If ((STAS == 0x01))
24. {
25. Return (0x0F)
26. }
27. Else
28. {
29. Return (0x00)
30. }
31. }
32. }
```

Alles anzeigen

Ich habe auch einige SSDTs deaktiviert, aber der Wake funktioniert weiterhin nicht.

- ~~SSDT-ANS.aml~~
- SSDT-AWAC.aml
- ~~SSDT-RTC0.aml~~
- SSDT-PLUG.aml
- ~~SSDT-THUNDERBOLT.aml~~
- ~~SSDT-USBX.aml~~
- SSDT-X299-DTGP.aml
- SSDT-X299-XHC.aml

Die RadeonVII habe ich auch mal gegen eine R9 280X ausgetauscht, um zu schauen ob es an ihr liegt. Kein Unterschied. Ich gehe jetzt noch mal die Darkwakes durch.

Was könnte ich noch ausprobieren?

Vielen Danke

Beitrag von „kuckkuck“ vom 11. Juli 2020, 17:52

> Zitat von Tirom
>
> STAS existiert bei mir, richtig?

Sieht gut aus.

Wie lautet denn der aktuelle Fehler bei Sleep?

Wofür brauchst du SSDT-X299-XHC.aml?

Mein Ziel war es nicht dich langfristig dazu zu bewegen alle diese ACPI Patches wegzulassen, jedoch kannst du probieren die verdächtigen SSDTs zeitweise zu deaktivieren, um herauszufinden welchen Einfluss sie auf den Sleep haben.

Hier noch ein paar interessante Terminal Befehle zu Sleep: Problem mit Sleep/Wake- Update 2018: UI Lag nach Wake

---

Beitrag von „Tirom“ vom 11. Juli 2020, 19:19

Hallo kuckkuck!

Ich bin jetzt alle darkwakes 0, 8, 2, 1, 4, 3 durch und hab weiterhin keinen Erfolg. Ich entferne jetzt noch mal die XHC. Ich dachte, die XHCS müssen korrekt heißen, damit Sleep gut funktioniert. War wohl falsch

Hier der aktuelle Bericht:

Code

1. panic(cpu 0 caller 0xffffff801e44aa3a): Kernel trap at 0xffffff801e466d57, type 13=general protection, registers:
2. CR0: 0x0000000080010033, CR2: 0xffffff8743df1000, CR3: 0x0000000023f90000, CR4: 0x00000000003626e0

3. RAX: 0x000000007e008003, RBX: 0xffffff801ec5dc30, RCX: 0x00000000000000e2, RDX: 0x0000000000000000
4. RSP: 0xffffff87598cbbb0, RBP: 0xffffff87598cbbe0, RSI: 0x0000000000000003, RDI: 0xffffff801ec5dbd0
5. R8: 0x0000000000000000, R9: 0xffffff80229d4034, R10: 0x0000000000000003, R11: 0x0000000000000001
6. R12: 0xffffff801eb30022, R13: 0x0000000000000001, R14: 0x0000000000000000, R15: 0xffffff801eb30008
7. RFL: 0x0000000000010046, RIP: 0xffffff801e466d57, CS: 0x0000000000000008, SS: 0x0000000000000010
8. Fault CR2: 0xffffff8743df1000, Error code: 0x0000000000000000, Fault CPU: 0x0, PL: 0, VF: 0
9.
10. Backtrace (CPU 0), Frame : Return Address
11. 0xffffff801e151220 : 0xffffff801e31f5cd mach_kernel : _handle_debugger_trap + 0x49d
12. 0xffffff801e151270 : 0xffffff801e458b05 mach_kernel : _kdp_i386_trap + 0x155
13. 0xffffff801e1512b0 : 0xffffff801e44a68e mach_kernel : _kernel_trap + 0x4ee
14. 0xffffff801e151300 : 0xffffff801e2c5a40 mach_kernel : _return_from_trap + 0xe0
15. 0xffffff801e151320 : 0xffffff801e31ec97 mach_kernel : _DebuggerTrapWithState + 0x17
16. 0xffffff801e151420 : 0xffffff801e31f087 mach_kernel : _panic_trap_to_debugger + 0x227
17. 0xffffff801e151470 : 0xffffff801eac27cc mach_kernel : _panic + 0x54
18. 0xffffff801e1514e0 : 0xffffff801e44aa3a mach_kernel : _sync_iss_to_iks + 0x2aa
19. 0xffffff801e151660 : 0xffffff801e44a738 mach_kernel : _kernel_trap + 0x598
20. 0xffffff801e1516b0 : 0xffffff801e2c5a40 mach_kernel : _return_from_trap + 0xe0
21. 0xffffff801e1516d0 : 0xffffff801e466d57 mach_kernel : _xcpm_perf_bias_set + 0x1c7
22. 0xffffff87598cbbe0 : 0xffffff801e466feb mach_kernel : _xcpm_init + 0xab
23. 0xffffff87598cbc00 : 0xffffff801e457021 mach_kernel : _acpi_sleep_kernel + 0x441
24. 0xffffff87598cbc70 : 0xffffff7fa02dfc2a com.apple.driver.AppleACPIPlatform : __ZN23AppleACPIPlatformExpert13sleepPlatformEv + 0x204
25. 0xffffff87598cbcc0 : 0xffffff7fa02e3eab com.apple.driver.AppleACPIPlatform : __ZN12AppleACPICPU7haltCPUEv + 0x75
26. 0xffffff87598cbce0 : 0xffffff801ea4cc30 mach_kernel : __Z16IOCPUSleepKernelv + 0x290
27. 0xffffff87598cbd40 : 0xffffff801ea85e25 mach_kernel : __ZN14IOPMrootDomain15powerChangeDoneEm + 0xac5
28. 0xffffff87598cbde0 : 0xffffff801ea173b7 mach_kernel : __ZN9IOService8all_doneEv + 0x767
29. 0xffffff87598cbe50 : 0xffffff801ea1414c mach_kernel : __ZN9IOService23actionPMWorkQueueInvokeEP11IOPMRequestP13IOPMWorkQueue + 0x86c
30. 0xffffff87598cbea0 : 0xffffff801ea117e0 mach_kernel : __ZN13IOPMWorkQueue17checkRequestQueueEP11queue_entryPb + 0xa0

31. 0xffffff87598cbef0 : 0xffffff801ea11679 mach_kernel : __ZN13IOPMWorkQueue12checkForWorkEv + 0xc9
32. 0xffffff87598cbf30 : 0xffffff801ea2cede mach_kernel : __ZN10IOWorkLoop15runEventSourcesEv + 0x11e
33. 0xffffff87598cbf70 : 0xffffff801ea2c4d6 mach_kernel : __ZN10IOWorkLoop10threadMainEv + 0x36
34. 0xffffff87598cbfa0 : 0xffffff801e2c513e mach_kernel : _call_continuation + 0x2e
35. Kernel Extensions in backtrace:
36. com.apple.driver.AppleACPIPlatform(6.1)[3A6D8ECD-C39E-39C8-984A-2CC417488A56]@0xffffff7fa02d4000->0xffffff7fa036efff
37. dependency: com.apple.iokit.IOACPIFamily(1.4)[0A7D7382-66FE-391B-9F93-97A996256C25]@0xffffff7f9f475000
38. dependency: com.apple.iokit.IOPCIFamily(2.9)[BE052F4D-9B80-3FCD-B36D-BACB7DEE0DF2]@0xffffff7f9ecee000
39. dependency: com.apple.driver.AppleSMC(3.1.9)[4589419D-7CCC-39A9-9E2F-F73FE42DD902]@0xffffff7f9f487000
40.
41. BSD process name corresponding to current thread: kernel_task
42. Boot args: -v keepsyms=1 debug=0x100 alcid=7 darkwake=0
43.
44. Mac OS version:
45. 19F101
46.
47. Kernel version:
48. Darwin Kernel Version 19.5.0: Tue May 26 20:41:44 PDT 2020; root:xnu-6153.121.2~2/RELEASE_X86_64
49. Kernel UUID: 54F1A78D-6F41-32BD-BFED-4381F9F6E2EF
50. Kernel slide: 0x000000001e000000
51. Kernel text base: 0xffffff801e200000
52. __HIB text base: 0xffffff801e100000
53. System model name: MacPro7,1 (Mac-27AD2F918AE68F61)
54. System shutdown begun: NO
55. Panic diags file available: YES (0x0)
56.
57. System uptime in nanoseconds: 182190685176

Alles anzeigen

Beitrag von „kuckkuck“ vom 11. Juli 2020, 20:25

Häng mir mal bitte einen IORegistry-Explorer Dump an. Hast du alle BIOS Settings nochmal penibelst abgecheckt?

Beitrag von „Tirom“ vom 11. Juli 2020, 20:41

Hey kuckkuck !

Ich schaue gleich noch mal über die BIOS Settings drüber.

Aktuell habe von MacPro7,1 (war irgendwo empfohlen) wieder auf iMacPro1,1 gewechselt. Mit darkwake=0 geht es nicht. Ich probiere da mal weiter mit rum.

Die ioregistryexplorer Infos habe ich dir angehängt. Gebootet habe ich jetzt mit meinen ganzen Patches:

- PC00 -> PCI0
- LPC0 -> LPCB
- FPU_ -> MATH
- TMR_ -> TIMR
- PIC_ -> IPIC
- PMC1 -> PMCR
- IOTR -> LDRC
- SMBS._ADR -> XSBU.XADR
- CPxx -> PRxx

Hier die Infos aus dem Terminal:

Code

1. log show --style syslog | fgrep "Wake reason"
2.
3. 2020-06-25 23:49:07.943009+0200 localhost powerd[99]: [powerd:sleepWake] Wake reason: "<private>" identity: "<private>"
4. 2020-06-25 23:49:32.056052+0200 localhost kernel[0]: (AppleACPIPlatform) AppleACPIPlatformPower Wake reason: RTC (Alarm)
5. 2020-06-25 23:49:32.056053+0200 localhost kernel[0]: (AppleACPIPlatform) AppleACPIPlatformPower Wake reason: RTC (Alarm)

6. 2020-06-27 19:46:40.589626+0200 localhost powerd[99]: [powerd:sleepWake] Wake reason: "<private>" identity: "<private>"
7. 2020-06-27 23:24:35.281856+0200 localhost powerd[100]: [powerd:sleepWake] Wake reason: "<private>" identity: "<private>"
8. 2020-07-03 17:22:47.286504+0200 localhost powerd[102]: [powerd:sleepWake] Wake reason: "<private>" identity: "<private>"
9. 2020-07-05 23:15:16.944542+0200 localhost powerd[102]: [powerd:sleepWake] Wake reason: "<private>" identity: "<private>"
10. 2020-07-09 19:02:06.401503+0200 localhost powerd[102]: [powerd:sleepWake] Wake reason: "<private>" identity: "<private>"
11. 2020-07-11 12:36:17.732818+0200 localhost powerd[102]: [powerd:sleepWake] Wake reason: "<private>" identity: "<private>"
12. 2020-07-11 15:53:22.652301+0200 localhost powerd[102]: [powerd:sleepWake] Wake reason: "<private>" identity: "<private>"

Alles anzeigen
Code

1. pmset -g
2.
3. System-wide power settings:
4. Currently in use: hibernatemode 0 disksleep 10 powernap 1 sleep 1 Sleep On Power Button 1 ttyskeepawake 1 hibernatefile /var/vm/sleepimage tcpkeepalive 1 autorestart 0 displaysleep 10

Code

1. sudo pmset -g log | tail -n 20
2.
3. 2020-07-11 18:57:20 +0200 : Showing all currently held IOKit power assertions
4. Assertion status system-wide: BackgroundTask 0 ApplePushServiceTask 0 UserIsActive 1 PreventUserIdleDisplaySleep 0 PreventSystemSleep 0 ExternalMedia 1 PreventUserIdleSystemSleep 0 NetworkClientActive 0
5. Listed by owning process: pid 102(powerd): [0x0000001800088000] 00:12:48 ExternalMedia named: "com.apple.powermanagement.externalmediamounted" pid 145(hidd): [0x0000002f000980de] 00:00:00 UserIsActive named: "com.apple.iohideventsystem.queue.tickle serviceID:100000683 name:AppleHIDKeyboardEve product:Apple Keyboard eventType:3" Timeout will fire in 599 secs Action=TimeoutActionRelease
6. Kernel Assertions: 0x4=USB id=502 level=255 0x4=USB mod=01.01.70, 01:00 description=com.apple.usb.externaldevice.14300000 owner=IOUSBHostDevice id=506 level=255 0x4=USB mod=01.01.70, 01:00 description=com.apple.usb.externaldevice.14700000 owner=Mass Storage Device id=507 level=255 0x4=USB mod=01.01.70, 01:00 description=com.apple.usb.externaldevice.14320000 owner=Gaming Mouse G502

id=508 level=255 0x4=USB mod=01.01.70, 01:00 description=com.apple.usb.externaldevice.14330000 owner=Keyboard Hub

7. Idle sleep preventers: IODisplayWrangler

Code

1. sudo pmset -g assertions
2. 2020-07-11 18:57:54 +0200 Assertion status system-wide: BackgroundTask 0 ApplePushServiceTask 0 UserIsActive 1 PreventUserIdleDisplaySleep 0 PreventSystemSleep 0 ExternalMedia 1 PreventUserIdleSystemSleep 0 NetworkClientActive 0
3. Listed by owning process: pid 102(powerd): [0x0000001800088000] 00:13:22 ExternalMedia named: "com.apple.powermanagement.externalmediamounted" pid 145(hidd): [0x0000002f000980de] 00:00:00 UserIsActive named: "com.apple.iohideventsystem.queue.tickle serviceID:100000683 name:AppleHIDKeyboardEve product:Apple Keyboard eventType:3" Timeout will fire in 600 secs Action=TimeoutActionRelease
4. Kernel Assertions: 0x4=USB id=502 level=255 0x4=USB mod=01.01.70, 01:00 description=com.apple.usb.externaldevice.14300000 owner=IOUSBHostDevice id=506 level=255 0x4=USB mod=01.01.70, 01:00 description=com.apple.usb.externaldevice.14700000 owner=Mass Storage Device id=507 level=255 0x4=USB mod=01.01.70, 01:00 description=com.apple.usb.externaldevice.14320000 owner=Gaming Mouse G502 id=508 level=255 0x4=USB mod=01.01.70, 01:00 description=com.apple.usb.externaldevice.14330000 owner=Keyboard Hub
5. Idle sleep preventers: IODisplayWrangler

Code

1. kextstat | grep -v apple
2.
3. Index Refs Address Size Wired Name (Version) UUID <Linked Against>
4. 41 6 0xffffff7f84053000 0x29000 0x29000 as.vit9696.Lilu (1.4.5) E42CE60E-EC0B-33AE-A513-5383B81BF165 <8 6 5 3 2 1>
5. 42 0 0xffffff7f84095000 0x145000 0x145000 as.vit9696.AppleALC (1.5.0) 2DC43BEC-BE69-32C7-8D49-FE1ED85A87D6 <41 13 8 6 5 3 2 1>
6. 43 0 0xffffff7f841e3000 0xe000 0xe000 meow.IOIIIO.MacProMemoryNotificationDisabler (1.0.0) 1DE82929-5E49-3185-A1BA-1E8CFB131D19 <41 8 6 5 3 2 1>
7. 44 0 0xffffff7f8421a000 0x6f000 0x6f000 as.vit9696.WhateverGreen (1.4.0) 28229092-CBB6-30FD-8954-4E887FAD958D <41 13 8 6 5 3 2 1>
8. 45 2 0xffffff7f8407c000 0x19000 0x19000 as.vit9696.VirtualSMC (1.1.4) 9DD28544-2B81-33EB-B3F1-488D0FBB0947 <41 12 8 6 5 3 2 1>

9. 46 0 0xffffff7f841f1000 0xa000 0xa000 as.vit9696.SMCProcessor (1.1.4) 5FDFDB28-994C-389E-816F-6731F74A6465 <45 41 12 8 6 5 3 2 1>
10. 54 0 0xffffff7f8420e000 0xc000 0xc000 hu.interferenc.TSCAdjustReset (1.1) 0D04744A-4524-37AA-9DA2-094D4DF7DDD8 <8 5 3>
11. 69 0 0xffffff7f841fb000 0x10000 0x10000 ru.joedm.SMCSuperIO (1.1.4) 3EBEC0A6-5D45-3034-A52E-8B63A6FABDA1 <45 41 12 8 6 5 3 2 1>
12. 73 0 0xffffff7f80d2d000 0x2e000 0x2e000 com.SmallTree.driver.SmallTreeIntel8259x (3.5.0) 8E38DEAB-C6D9-3986-926F-F1018AB0C605 <18 13 6 5 3 1>

Alles anzeigen

Beitrag von „kuckkuck“ vom 11. Juli 2020, 20:53

Was genau meinst du mit CPxx -> PRxx? Da sollte doch zumindest eine Zahl stehen, aber auch den Grund des Renames verstehe ich nicht.

Ist SMBS._ADR -> XSBU.XADR mit irgendeiner SSDT verbunden? Ansonsten macht das auch wenig Sinn auf den ersten Blick.

XHCI oder XHC1 solltest du in XHC_ oder dergleichen umbenennen.

Beitrag von „Tirom“ vom 11. Juli 2020, 22:26

Genau, xx steht für die ersten 56 Hexwerte CP00 bis CP37 und benennt diese nach PR00 bis PR55 um. Anscheinend ist das aber quatsch, wenn ich dich richtig verstehe.

Welche Patches soll ich alle deaktivieren/rausschmeißen?

EDIT: Hab jetzt noch folgenden Patch in die config.plist eingetragen:

XHCI -> XHC_

58484349 -> 5848435F

Jetzt erstelle ich eine neue USBPorts.kext mit hackintool. Bis gleich

EDIT2: Ich bin jetzt Darkwake 0 8 2 1 4 3 und ohne durchgegangen. Alle stürzen ab:

Code

1. panic(cpu 0 caller 0xffffff801944aa3a): Kernel trap at 0xffffff8019466d57, type 13=general protection, registers:
2. CR0: 0x0000000080010033, CR2: 0xffffff873eda1000, CR3: 0x000000001ef7a000, CR4: 0x00000000003626e0
3. RAX: 0x000000007e008003, RBX: 0xffffff8019c5dc30, RCX: 0x00000000000000e2, RDX: 0x0000000000000000
4. RSP: 0xffffff8754613bb0, RBP: 0xffffff8754613be0, RSI: 0x0000000000000003, RDI: 0xffffff8019c5dbd0
5. R8: 0x0000000000000000, R9: 0xffffff801d9c4029, R10: 0x0000000000000003, R11: 0x0000000000000001
6. R12: 0xffffff8019b30022, R13: 0x0000000000000001, R14: 0x0000000000000000, R15: 0xffffff8019b30008
7. RFL: 0x0000000000010046, RIP: 0xffffff8019466d57, CS: 0x0000000000000008, SS: 0x0000000000000010
8. Fault CR2: 0xffffff873eda1000, Error code: 0x0000000000000000, Fault CPU: 0x0, PL: 0, VF: 0
9.
10. Backtrace (CPU 0), Frame : Return Address
11. 0xffffff8019151220 : 0xffffff801931f5cd mach_kernel : _handle_debugger_trap + 0x49d
12. 0xffffff8019151270 : 0xffffff8019458b05 mach_kernel : _kdp_i386_trap + 0x155
13. 0xffffff80191512b0 : 0xffffff801944a68e mach_kernel : _kernel_trap + 0x4ee
14. 0xffffff8019151300 : 0xffffff80192c5a40 mach_kernel : _return_from_trap + 0xe0
15. 0xffffff8019151320 : 0xffffff801931ec97 mach_kernel : _DebuggerTrapWithState + 0x17
16. 0xffffff8019151420 : 0xffffff801931f087 mach_kernel : _panic_trap_to_debugger + 0x227
17. 0xffffff8019151470 : 0xffffff8019ac27cc mach_kernel : _panic + 0x54
18. 0xffffff80191514e0 : 0xffffff801944aa3a mach_kernel : _sync_iss_to_iks + 0x2aa
19. 0xffffff8019151660 : 0xffffff801944a738 mach_kernel : _kernel_trap + 0x598
20. 0xffffff80191516b0 : 0xffffff80192c5a40 mach_kernel : _return_from_trap + 0xe0
21. 0xffffff80191516d0 : 0xffffff8019466d57 mach_kernel : _xcpm_perf_bias_set + 0x1c7
22. 0xffffff8754613be0 : 0xffffff8019466feb mach_kernel : _xcpm_init + 0xab
23. 0xffffff8754613c00 : 0xffffff8019457021 mach_kernel : _acpi_sleep_kernel + 0x441
24. 0xffffff8754613c70 : 0xffffff7f9b2dfc2a com.apple.driver.AppleACPIPlatform : __ZN23AppleACPIPlatformExpert13sleepPlatformEv + 0x204

25. 0xffffff8754613cc0 : 0xffffff7f9b2e3eab com.apple.driver.AppleACPIPlatform : __ZN12AppleACPICPU7haltCPUEv + 0x75
26. 0xffffff8754613ce0 : 0xffffff8019a4cc30 mach_kernel : __Z16IOCPUSleepKernelv + 0x290
27. 0xffffff8754613d40 : 0xffffff8019a85e25 mach_kernel : __ZN14IOPMrootDomain15powerChangeDoneEm + 0xac5
28. 0xffffff8754613de0 : 0xffffff8019a173b7 mach_kernel : __ZN9IOService8all_doneEv + 0x767
29. 0xffffff8754613e50 : 0xffffff8019a1414c mach_kernel : __ZN9IOService23actionPMWorkQueueInvokeEP11IOPMRequestP13IOPMWorkQueue + 0x86c
30. 0xffffff8754613ea0 : 0xffffff8019a117e0 mach_kernel : __ZN13IOPMWorkQueue17checkRequestQueueEP11queue_entryPb + 0xa0
31. 0xffffff8754613ef0 : 0xffffff8019a11679 mach_kernel : __ZN13IOPMWorkQueue12checkForWorkEv + 0xc9
32. 0xffffff8754613f30 : 0xffffff8019a2cede mach_kernel : __ZN10IOWorkLoop15runEventSourcesEv + 0x11e
33. 0xffffff8754613f70 : 0xffffff8019a2c4d6 mach_kernel : __ZN10IOWorkLoop10threadMainEv + 0x36
34. 0xffffff8754613fa0 : 0xffffff80192c513e mach_kernel : _call_continuation + 0x2e
35. Kernel Extensions in backtrace:
36. com.apple.driver.AppleACPIPlatform(6.1)[3A6D8ECD-C39E-39C8-984A-2CC417488A56]@0xffffff7f9b2d4000->0xffffff7f9b36efff
37. dependency: com.apple.iokit.IOACPIFamily(1.4)[0A7D7382-66FE-391B-9F93-97A996256C25]@0xffffff7f9a475000
38. dependency: com.apple.iokit.IOPCIFamily(2.9)[BE052F4D-9B80-3FCD-B36D-BACB7DEE0DF2]@0xffffff7f99cee000
39. dependency: com.apple.driver.AppleSMC(3.1.9)[4589419D-7CCC-39A9-9E2F-F73FE42DD902]@0xffffff7f9a487000
40.
41. BSD process name corresponding to current thread: kernel_task
42. Boot args: -v keepsyms=1 debug=0x100 alcid=7
43.
44. Mac OS version:
45. 19F101
46.
47. Kernel version:
48. Darwin Kernel Version 19.5.0: Tue May 26 20:41:44 PDT 2020; root:xnu-6153.121.2~2/RELEASE_X86_64
49. Kernel UUID: 54F1A78D-6F41-32BD-BFED-4381F9F6E2EF
50. Kernel slide: 0x0000000019000000
51. Kernel text base: 0xffffff8019200000
52. __HIB text base: 0xffffff8019100000

53. System model name: iMacPro1,1 (Mac-7BA5B2D9E42DDD94)
54. System shutdown begun: NO
55. Panic diags file available: YES (0x0)
56.
57. System uptime in nanoseconds: 151107116397

Alles anzeigen

Beitrag von „apfelnico“ vom 11. Juli 2020, 23:09

> Zitat von Tirom
>
> Gebootet habe ich jetzt mit meinen ganzen Patches:
>
> PC00 -> PCI0
> LPC0 -> LPCB
> FPU_ -> MATH
> TMR_ -> TIMR
> PIC_ -> IPIC
> PMC1 -> PMCR
> IOTR -> LDRC
> SMBS._ADR -> XSBU.XADR
> CPxx -> PRxx
>
> Alles anzeigen

Nicht einer davon ist wichtig. PC00 ist völlig in Ordnung, das ist uralt. Hatte ich mal als erster gemacht, weil damals sonst nicht Audio ging, weil AppleALC davon ausging, dass "HDEF" eben in "PCI0" zu finden ist und in den ersten ACPIs der iMacPro ebenfalls PCI0 wie bei allen vorherigen Plattformen vorlag.

Den LPC0 umzubenennen ist ebenfalls völlig unwichtig. Auch PMC1 wird so wie es ist erkannt. FPU, TMR und PIC ebenso bzw spielen eh keine Rolle. Und SMBS auszublenden damit SBUS erscheint – ist eh nicht kompatibel und wird auf dieser Plattform auch nicht gebraucht. Die einzelnen Prozessor-Cores umzubenennen ist ebenfalls Tinnef und bringt rein gar nichts. Und wozu den XHCI umbenennen? Auch der läuft nativ, lediglich mittels eigener Kext etwas näher spezifizieren.

Um dein Wake zu erreichen, brauchst du auch nicht mit "Darkwake" experimentieren. Die Holzhammer-Methode – der einzige Patch der ausreichend ist – die Methode "_PRW" in meinetwegen "XPRW" umzubenennen, schon greift die nicht mehr und lässt den Rechner

schlafen wie er soll. Aufwachen dann über kurzes drücken auf den An/Ausschalter.

Beitrag von „Tirom“ vom 11. Juli 2020, 23:54

Hallo apfelnico!

Danke für die ausführliche Erklärung. Die nicht-Apple-ACPI scheint ja inzwischen echt gut aufgefangen zu werden. Das freut mich!

Eine Frage zu der _PRW -> XPRW Umbenennung. Das macht man doch, wenn der Rechner unbeabsichtigt durch USB geweckt wird, richtig? Das ist ja nicht mein Problem: Mein Rechner schläft wie er soll, wenn ich ihn aufwecke stürzt er leider ab. Der Monitor bleibt schwarz und bei der Maus geht die LED nicht an. Die Lüfter laufen aber.

Aber Danke für deine Hilfe, meine Config ist wieder etwas leerer geworden 

Beitrag von „apfelnico“ vom 12. Juli 2020, 00:08

Ah, falsch verstanden. Dachte er geht kurz in den Sleep, um danach gleich wieder aufzuwachen. Kenne leider nicht das Gigabyte, weiß nicht, was da anders läuft. Quirks dafür/dagegen?

Zu den XHCx. Hier ist es schon interessant, die weiteren USB-Controller – nicht "Basis" auf PC00, sondern weitere an RPxx, welche dann einfach nur alle gleich "PXSX" heißen, umzubenennen. Das ist keinesfalls zwingend erforderlich, es läuft auch so. Es hilft nur ungemein, wenn man auch hier die Ports näher mittels Kext spezifizieren möchte, dann kann man diese sehr individuell eben mit XHC2 etc ansprechen. Auch lassen die sich dann in zum Beispiel "Hackintool" besser auseinander halten. Geht aber auch anders.

Beitrag von „kuckkuck“ vom 12. Juli 2020, 00:21

X299 ist wirklich nicht meine Plattform, deswegen ist es sehr gut, dass hier jetzt jemand mit im Boot ist der sich da ein wenig besser auskennt

> Zitat von apfelnico
>
> Und wozu den XHCI umbenennen?

Entweder XHCI oder XHC1 matcht auf manche Port Injectors von Apple SMBios, die in AppleUSBXHCIPCI hinterlegt sind, deswegen ist das der einzige Rename den ich noch nachvollziehen kann.

Beitrag von „Tirom“ vom 12. Juli 2020, 00:26

apfelnico Kein Problem, ist ja leider auch etwas unübersichtlich geworden, bei meinem Problem... Kann ich die Umbenennung der anderen XHCs auch in der config.plist machen oder muss das in einer SSDT geschehen? Die hatte ich mal gehabt, aber inzwischen deaktiviert.

kuckkuck Inzwischen ist nix anderes mehr drin, außer XHCI -> XHC_ mit neu erstellter USBPorts.kext. Dann bin ich noch mal alle darkwakes durchgegangen, aber keiner hat funktioniert.

Beitrag von „kuckkuck“ vom 12. Juli 2020, 00:30

> Zitat von Tirom
>
> Inzwischen ist nix anderes mehr drin, außer XHCI -> XHC_

Es kann sein, dass ich mich hier getäuscht habe und es XHC1 statt XHCI heißen muss. Ich kann das morgen mal nachschauen. Du selber kannst einfach mal im IOReg nachschauen, ob irgendein Controller XHC1 heißt und wenn ja, ob die von dir eingestellten Ports trotzdem richtig

übernommen wurden.

Das Testen der darkwakes ist mMn. aktuell unnötige Arbeit, da wir mit einem viel tiefer greifenden Problem in der Panik zu kämpfen haben. Vielleicht ist das ja ein bekanntes Problem mit X299 und jemand hat einen Fix parat. Mir fallen auf das Fehlerbild leider keine konkreten Vorschläge ein, nur Dinge die man ausprobieren könnte.

Beitrag von „Tirom“ vom 12. Juli 2020, 12:12

Hey kuckkuck!

Danke für die Info bezüglich der darkwakes, das spart eine Menge Zeit

Ich bin jetzt gerade nur etwas verwirrt, wie ich weiter manchen soll. In meinem System habe ich ohne Patches

- PC00.XHCI (Intel USB3)
- PC00.RP05.PXSX (Thunderbolt)
- PC00.RP13.PXSX (ASMedia USB-C)

apfelnico schreibt irgendwo, dass Umbenennungen XHCI -> XHC_ keinen Sinn ergeben, die PXSX -> XHC2/3 aber schon.

Das geht aber nicht über Opencore, sondern nur über eine SSDT, richtig?

Soll ich XHCI nach XHC_ oder XHC0 umbenennen oder als XHCI lassen?

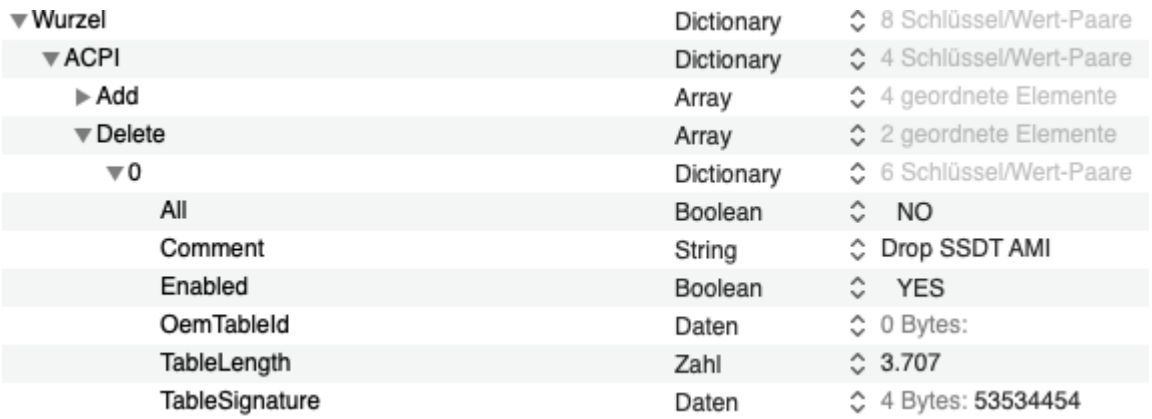
Kann ich sonst irgendwie helfen? Ich habe richtig Lust mich da rein zu knien, weiß aber nicht, was ich wirklich tun kann. Bringt es etwas, die DSDT und die IOReg von einem echten iMacPro zu suchen und mit meinen zu vergleichen? Gebt mir eine Aufgabe

Danke!

Beitrag von „apfelnico“ vom 12. Juli 2020, 13:14

Tirom

XHCI bleibt. Die weiteren USB-Controller an den PCIe Root Ports (RPxx) sind nur allgemein als "PXSX" benannt. Diese würde ich umbenennen. Nicht aber mit XHC1 anfangen, dieser ist durch Apple anders definiert. Also weiter mit XHC2 und höher. Das geht nur mit einer SSDT. Nicht unbedingt eine neue SSDT dafür anlegen, sondern in der vorhandenen ACPI nach der USB-Deklaration suchen. Bei mir heisst diese "SSDT AMI" und hat eine "TableLength" von "3707". Steht im Header. Damit kannst du diese exakt aus der ACPI entfernen mittels OpenCore:



Somit kannst du die modifizierte SSDT problemlos einbinden und bekommst keine Probleme. Der Vorteil dadurch ist, dass du später mittels Hackintool alle USB-Controller inklusive deren Ports individuell bestimmen kannst um eine perfekte USB-Kext zu generieren.

Edit:

Den Thunderbolt an RP05.PXSX würde ich erstmal davon unberührt lassen. Dessen USB-C Controller ist etwas tiefer versteckt. Dafür gibt es eine eigene SSDT:

Beitrag von „Tirom“ vom 12. Juli 2020, 21:19

Hallo apfelnico !

Danke für die gute Erklärung. Ich fasse noch mal kurz zusammen:

Ich Suche in meiner DSDT den Abschnitt _SB.PC00.RP13 und kopiere den gesamten "Scope" in eine neue .aml. In dieser nenne ich das "Device (PXSX)" nach "Device (XHC2)" um. Das sieht bei mir so aus:

Code

```
Scope (RP13)
{
Method (_INI, 0, NotSerialized) // _INI: Initialize
{
LTRN = LTRD /* \LTRD */
LMSL = PMLD /* \PMLD */
LNSL = PNLD /* \PNLD */
OBFN = OBFD /* \OBFD */
}

OperationRegion (PXCS, PCI_Config, 0x00, 0x0480)
Field (PXCS, AnyAcc, NoLock, Preserve)
{
VDID, 32,
Offset (0x50),
L0SE, 1,
, 3,
LDIS, 1,
Offset (0x51),
Offset (0x52),
, 13,
LASX, 1,
Offset (0x5A),
ABPX, 1,
, 2,
PDCX, 1,
, 2,
PDSX, 1,
Offset (0x5B),
Offset (0x60),
Offset (0x62),
PSPX, 1,
PMEP, 1,
Offset (0xA4),
D3HT, 2,
```

```
Offset (0xD8),
, 30,
HPEX, 1,
PMEX, 1,
Offset (0xE2),
, 2,
L23E, 1,
L23R, 1,
Offset (0x324),
, 3,
LEDM, 1,
Offset (0x420),
, 30,
DPGE, 1
}

Field (PXCS, AnyAcc, NoLock, WriteAsZeros)
{
Offset (0xDC),
, 30,
HPSX, 1,
PMSX, 1
}

Name (LTRV, Package (0x04)
{
0x00,
0x00,
0x00,
0x00
})
Method (_DSM, 4, Serialized) // _DSM: Device-Specific Method
{
If ((Arg0 == ToUUID ("e5c937d0-3553-4d7a-9117-ea4d19c3434d") /* Device Labeling Interface */))
{
Switch (ToInteger (Arg2))
{
Case (0x00)
{
Name (OPTS, Buffer (0x02)
```

```
76. {
77. 0x00, 0x00 // ..
78. })
79. CreateBitField (OPTS, 0x00, FUN0)
80. CreateBitField (OPTS, 0x04, FUN4)
81. CreateBitField (OPTS, 0x06, FUN6)
82. CreateBitField (OPTS, 0x08, FUN8)
83. CreateBitField (OPTS, 0x09, FUN9)
84. If ((Arg1 >= 0x02))
85. {
86. FUN0 = 0x01
87. If (LTRE)
88. {
89. FUN6 = 0x01
90. }
91.
92. If (OBFF)
93. {
94. FUN4 = 0x01
95. }
96.
97. If ((ECR1 == 0x01))
98. {
99. If ((Arg1 >= 0x03))
100. {
101. FUN8 = 0x01
102. FUN9 = 0x01
103. }
104. }
105. }
106.
107. Return (OPTS) /* \_SB_.PC00.RP13._DSM.OPTS */
108. }
109. Case (0x04)
110. {
111. If ((Arg1 >= 0x02))
112. {
113. If (OBFN)
114. {
115. Return (Buffer (0x10)
116. {
```

```
117. /* 0000 */ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, // ........
118. /* 0008 */ 0x00, 0x00, 0x00, 0x08, 0x00, 0x00, 0x00, 0x00 // ........
119. })
120. }
121. Else
122. {
123. Return (Buffer (0x10)
124. {
125. /* 0000 */ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, // ........
126. /* 0008 */ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 // ........
127. })
128. }
129. }
130. }
131. Case (0x06)
132. {
133. If ((Arg1 >= 0x02))
134. {
135. If (LTRN)
136. {
137. If (((LMSL == 0x00) || (LNSL == 0x00)))
138. {
139. If ((PCHS == SPTH))
140. {
141. LMSL = 0x0846
142. LNSL = 0x0846
143. }
144. ElseIf ((PCHS == SPTL))
145. {
146. LMSL = 0x1003
147. LNSL = 0x1003
148. }
149. }
150.
151. LTRV [0x00] = ((LMSL >> 0x0A) & 0x07)
152. LTRV [0x01] = (LMSL & 0x03FF)
153. LTRV [0x02] = ((LNSL >> 0x0A) & 0x07)
154. LTRV [0x03] = (LNSL & 0x03FF)
155. Return (LTRV) /* \_SB_.PC00.RP13.LTRV */
156. }
157. Else
```

```
158. {
159. Return (Buffer (0x01)
160. {
161. 0x00 // .
162. })
163. }
164. }
165. }
166. Case (0x08)
167. {
168. If ((ECR1 == 0x01))
169. {
170. If ((Arg1 >= 0x03))
171. {
172. Return (0x01)
173. }
174. }
175. }
176. Case (0x09)
177. {
178. If ((ECR1 == 0x01))
179. {
180. If ((Arg1 >= 0x03))
181. {
182. Return (Package (0x05)
183. {
184. 0xC350,
185. Ones,
186. Ones,
187. 0xC350,
188. Ones
189. })
190. }
191. }
192. }
193.
194. }
195. }
196.
197. Return (Buffer (0x01)
198. {
```

```
0x00 // .
})
}

Device (XHC2)
{
Name (_ADR, 0x00) // _ADR: Address
Method (_PRW, 0, NotSerialized) // _PRW: Power Resources for Wake
{
Return (GPRW (0x69, 0x04))
}
}

Method (HPME, 0, Serialized)
{
If (((VDID != 0xFFFFFFFF) && (PMSX == 0x01)))
{
Notify (PXSX, 0x02) // Device Wake
PMSX = 0x01
PSPX = 0x01
}
}

Method (_PRW, 0, NotSerialized) // _PRW: Power Resources for Wake
{
Return (GPRW (0x69, 0x04))
}
}
```

Alles anzeigen

Jetzt habe ich aber Fragen:

Welchen Header muss ich noch hinzufügen? Nehme ich auch den aus meiner DSDT?

Wenn dann diese SSDT hinzugefügt wird, muss man schauen, dass die Info nicht doppelt vorhanden ist, richtig? Also muss der Bereich in der original DSDT entfernt werden. Daher der ACPI Delete in der config.plist.

Aber woher weiß ich, was gelöscht werden muss. 53534454 bedeutet SSDT, aber woher bekomme ich die TableLength. Und woher weiß OC, wann der Abschnitt beginnt.

Sorry, da stehe ich etwas auf dem Schlauch. Gibt es irgendwo eine Einführung: SSDTs für Dummies?

Beitrag von „apfelnico“ vom 12. Juli 2020, 21:54

Nein, ganz anders. Du hast es nicht richtig gelesen. Es geht nicht um die DSDT. Lade mal deine komplette ACPI hoch, dann zeige ich dir was ich meine.

Beitrag von „Tirom“ vom 12. Juli 2020, 22:15

Hey apfelnico !

So tief war ich dem Thema noch nicht drin. Hab auch in die anderen ACPIs reingeschaut, aber noch nix gefunden gehabt. War aber auch nicht sicher, ob ich am richtigen Ort suche.

Hier mein ganzes ACPI. Danke für deine Unterstützung! 

Beitrag von „apfelnico“ vom 13. Juli 2020, 00:09

Tirom

Bitte noch ein IORegistryExplorer-File und einmal in den Systembericht gehen (Apfel-Menü, Über diesen Mac, Systembericht ...) und auch hier sichern. Beide Files zurück und ich habe einen umfassenden Überblick.

Beitrag von „Tirom“ vom 13. Juli 2020, 20:04

Hallo apfelnico !

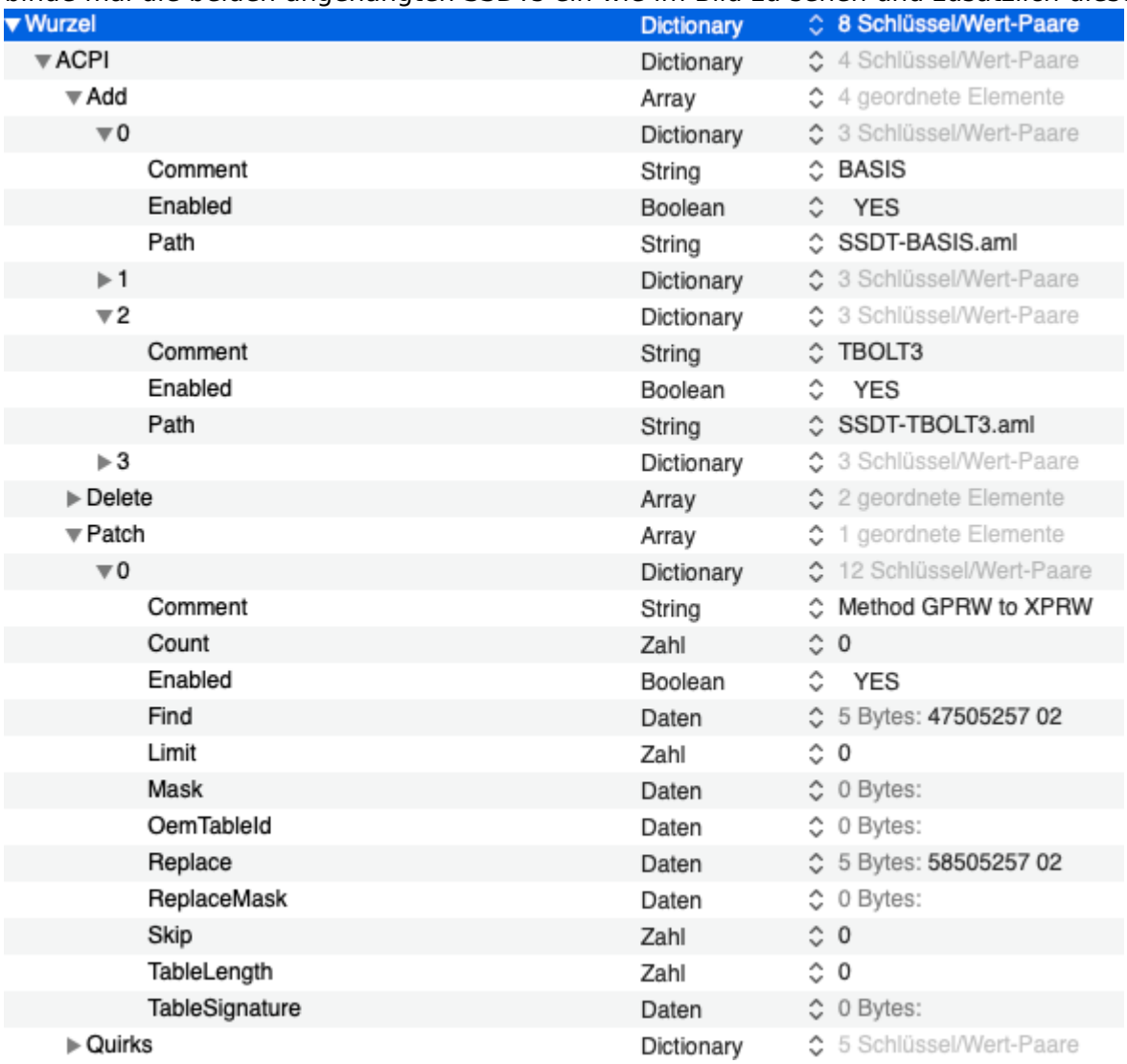
Hier sind die Dateien. Wenn du willst, dann nimm mich doch gleich mit. Wir können gerne über Discord oder TeamViewer sprechen. Würde mich total freuen

---

Beitrag von „apfelnico“ vom 13. Juli 2020, 22:49

Tirom

binde mal die beiden angehängten SSDTs ein wie im Bild zu sehen und zusätzlich diesen Patch:



▼ Wurzel	Dictionary	8 Schlüssel/Wert-Paare
▼ ACPI	Dictionary	4 Schlüssel/Wert-Paare
▼ Add	Array	4 geordnete Elemente
▼ 0	Dictionary	3 Schlüssel/Wert-Paare
Comment	String	BASIS
Enabled	Boolean	YES
Path	String	SSDT-BASIS.aml
▶ 1	Dictionary	3 Schlüssel/Wert-Paare
▼ 2	Dictionary	3 Schlüssel/Wert-Paare
Comment	String	TBOLT3
Enabled	Boolean	YES
Path	String	SSDT-TBOLT3.aml
▶ 3	Dictionary	3 Schlüssel/Wert-Paare
▶ Delete	Array	2 geordnete Elemente
▼ Patch	Array	1 geordnete Elemente
▼ 0	Dictionary	12 Schlüssel/Wert-Paare
Comment	String	Method GPRW to XPRW
Count	Zahl	0
Enabled	Boolean	YES
Find	Daten	5 Bytes: 47505257 02
Limit	Zahl	0
Mask	Daten	0 Bytes:
OemTableId	Daten	0 Bytes:
Replace	Daten	5 Bytes: 58505257 02
ReplaceMask	Daten	0 Bytes:
Skip	Zahl	0
TableLength	Zahl	0
TableSignature	Daten	0 Bytes:
▶ Quirks	Dictionary	5 Schlüssel/Wert-Paare

Sollten weitere ACPI-Patches vorhanden sein, raus damit. Auch weitere SSDTs komplett entfernen (AWAC Zeugs etc).

Es fehlen noch einige Devices, aber erst mal schauen wie es läuft.

Beitrag von „Tirom“ vom 14. Juli 2020, 10:45

Hallo apfelnico !

Vielen Dank für die SSDTs! 

Ich hab alle SSDTs auf disabled gestellt und nur deine aktiviert, den Patch hinzugefügt und per Hackintool (via USBInjectAll und XhciPortLimit) eine neue USBPorts.kext erstellt und eingebunden (sowie die beiden Helfer wieder entfernt).

Leider stürzt es weiterhin ab. Hier der Bericht:

Code

1. panic(cpu 0 caller 0xffffff801084aa3a): Kernel trap at 0xffffff8010866d57, type 13=general protection, registers:
2. CR0: 0x0000000080010033, CR2: 0xffffff87360c9000, CR3: 0x000000001637a000, CR4: 0x00000000003626e0
3. RAX: 0x000000007e008003, RBX: 0xffffff801105dc30, RCX: 0x00000000000000e2, RDX: 0x0000000000000000
4. RSP: 0xffffff874b903bb0, RBP: 0xffffff874b903be0, RSI: 0x0000000000000003, RDI: 0xffffff801105dbd0
5. R8: 0x0000000000000000, R9: 0xffffff8014dc4029, R10: 0x0000000000000003, R11: 0x0000000000000001
6. R12: 0xffffff8010f30022, R13: 0x0000000000000001, R14: 0x0000000000000000, R15: 0xffffff8010f30008
7. RFL: 0x0000000000010046, RIP: 0xffffff8010866d57, CS: 0x0000000000000008, SS: 0x0000000000000010
8. Fault CR2: 0xffffff87360c9000, Error code: 0x0000000000000000, Fault CPU: 0x0, PL: 0, VF: 0
9.
10. Backtrace (CPU 0), Frame : Return Address
11. 0xffffff8010551220 : 0xffffff801071f5cd mach_kernel : _handle_debugger_trap + 0x49d
12. 0xffffff8010551270 : 0xffffff8010858b05 mach_kernel : _kdp_i386_trap + 0x155
13. 0xffffff80105512b0 : 0xffffff801084a68e mach_kernel : _kernel_trap + 0x4ee
14. 0xffffff8010551300 : 0xffffff80106c5a40 mach_kernel : _return_from_trap + 0xe0
15. 0xffffff8010551320 : 0xffffff801071ec97 mach_kernel : _DebuggerTrapWithState + 0x17
16. 0xffffff8010551420 : 0xffffff801071f087 mach_kernel : _panic_trap_to_debugger + 0x227
17. 0xffffff8010551470 : 0xffffff8010ec27cc mach_kernel : _panic + 0x54
18. 0xffffff80105514e0 : 0xffffff801084aa3a mach_kernel : _sync_iss_to_iks + 0x2aa
19. 0xffffff8010551660 : 0xffffff801084a738 mach_kernel : _kernel_trap + 0x598
20. 0xffffff80105516b0 : 0xffffff80106c5a40 mach_kernel : _return_from_trap + 0xe0
21. 0xffffff80105516d0 : 0xffffff8010866d57 mach_kernel : _xcpm_perf_bias_set + 0x1c7
22. 0xffffff874b903be0 : 0xffffff8010866feb mach_kernel : _xcpm_init + 0xab

23. 0xffffff874b903c00 : 0xffffff8010857021 mach_kernel : _acpi_sleep_kernel + 0x441
24. 0xffffff874b903c70 : 0xffffff7f926dfc2a com.apple.driver.AppleACPIPlatform : __ZN23AppleACPIPlatformExpert13sleepPlatformEv + 0x204
25. 0xffffff874b903cc0 : 0xffffff7f926e3eab com.apple.driver.AppleACPIPlatform : __ZN12AppleACPICPU7haltCPUEv + 0x75
26. 0xffffff874b903ce0 : 0xffffff8010e4cc30 mach_kernel : __Z16IOCPUSleepKernelv + 0x290
27. 0xffffff874b903d40 : 0xffffff8010e85e25 mach_kernel : __ZN14IOPMrootDomain15powerChangeDoneEm + 0xac5
28. 0xffffff874b903de0 : 0xffffff8010e173b7 mach_kernel : __ZN9IOService8all_doneEv + 0x767
29. 0xffffff874b903e50 : 0xffffff8010e1414c mach_kernel : __ZN9IOService23actionPMWorkQueueInvokeEP11IOPMRequestP13IOPMWorkQueue + 0x86c
30. 0xffffff874b903ea0 : 0xffffff8010e117e0 mach_kernel : __ZN13IOPMWorkQueue17checkRequestQueueEP11queue_entryPb + 0xa0
31. 0xffffff874b903ef0 : 0xffffff8010e11679 mach_kernel : __ZN13IOPMWorkQueue12checkForWorkEv + 0xc9
32. 0xffffff874b903f30 : 0xffffff8010e2cede mach_kernel : __ZN10IOWorkLoop15runEventSourcesEv + 0x11e
33. 0xffffff874b903f70 : 0xffffff8010e2c4d6 mach_kernel : __ZN10IOWorkLoop10threadMainEv + 0x36
34. 0xffffff874b903fa0 : 0xffffff80106c513e mach_kernel : _call_continuation + 0x2e
35. Kernel Extensions in backtrace:
36. com.apple.driver.AppleACPIPlatform(6.1)[3A6D8ECD-C39E-39C8-984A-2CC417488A56]@0xffffff7f926d4000->0xffffff7f9276efff
37. dependency: com.apple.iokit.IOACPIFamily(1.4)[0A7D7382-66FE-391B-9F93-97A996256C25]@0xffffff7f91875000
38. dependency: com.apple.iokit.IOPCIFamily(2.9)[BE052F4D-9B80-3FCD-B36D-BACB7DEE0DF2]@0xffffff7f910ee000
39. dependency: com.apple.driver.AppleSMC(3.1.9)[4589419D-7CCC-39A9-9E2F-F73FE42DD902]@0xffffff7f91887000
40.
41. BSD process name corresponding to current thread: kernel_task
42. Boot args: -v keepsyms=1 debug=0x100 alcid=7
43.
44. Mac OS version:
45. 19F101
46.
47. Kernel version:
48. Darwin Kernel Version 19.5.0: Tue May 26 20:41:44 PDT 2020; root:xnu-6153.121.2~2/RELEASE_X86_64
49. Kernel UUID: 54F1A78D-6F41-32BD-BFED-4381F9F6E2EF

50. Kernel slide: 0x0000000010400000
51. Kernel text base: 0xffffff8010600000
52. __HIB text base: 0xffffff8010500000
53. System model name: iMacPro1,1 (Mac-7BA5B2D9E42DDD94)
54. System shutdown begun: NO
55. Panic diags file available: YES (0x0)
56.
57. System uptime in nanoseconds: 250872113825

Alles anzeigen

Beitrag von „apfelnico“ vom 14. Juli 2020, 10:53

Dann würde ich gern mal deinen EFI-Ordner sehen.

Beitrag von „Tirom“ vom 14. Juli 2020, 21:59

Hallo apfelnico !

Hier der gesamte EFI-Ordner. Nicht wundern, eine -amls und .kexte liegen noch im Ordner, sind aber deaktiviert.

Viele Dank und viele Grüße

Tirom

Beitrag von „hackschlafosx“ vom 16. Juli 2020, 23:43

Hallo,

ich klinke mich mal hier rein, ich hoffe, das ist ok. Finde das Thema spannend und hänge an

einem ähnlichen Punkt: Bei mir funktioniert zwar das Einschlafen, aber beim Aufwecken wirft Mojave alle USB-Geräte aus.

Hackintool zeigt für meine USB-Devices XHCI und SLT1 (für Asmedia) an. Im IoRegistry-Explorer gibt es unter PC00@0 direkt einen Eintrag mit XHCI und PC00@0 -> RP01@1c -> IOPP -> SLT1 -> SLT1 sind die Asmedia Ports.

Eigentlich läuft sonst alles - bis eben auf das Auswerfen der USB-Geräte.

In der Open-Core Anleitung heißt es ja, dass man bei einem ImacPro XHC1 zu SHCI per Patch in der config.plist umbenennen soll - ist das Quatsch? Oder gilt das nur für Mainboards, die ihre Intel-Ports eben XHC1 benennen? Entspricht dann XHC1 meinem SLT1 oder dem XHCI? Warum kann/darf man das nicht über einen Patch machen?

Gibt es irgendwo eine gute Anleitung zu diesem Komplex? Doktore da schon mehrere Wochen herum und bin ein wenig am verzweifeln...

Sorry, will den Thread nicht kapern, mich würde vor allem interessieren, wie du, apfelnico die AML mit den XHCI-renames "gebaut" hast (SSDT-BASIS.aml), damit ich das für mein System "nachbauen" kann...