Erledigt Tutorial für funktionierenden Batteriestatus

Beitrag von „hitman20“ vom 15. Juli 2017, 00:28

Hallo,

da die Frage öfter mal aufkommt, wie man den Batteriestatus korrekt unter OS X anzeigen kann, möchte ich euch hier in dem Tutorial erklären, wie man dafür die DSDT korrekt patched. Ich habe im Forum noch keine Anleitung dafür gefunden.

Ihr braucht dazu auch die ACPIBatteryManager.kext im Ordner /EFI/CLOVER/kexts/Other oder /System/Library/Extensions oder in /Library/Extensions sonst funktioniert es nicht.

Wenn Ihr eure bereits funktionierende DSDT.dsl mit MaciASL öffnet, solltest Ihr zuerst nachdem Wort "EmbeddedControl" suchen. Dort solltest Ihr in der Regel gleich in der "Operation Region" landen.
Wenn unter der "Operation Region" das Feld "Field (ECRM, ByteAcc, Lock, Preserve)" steht seid Ihr auf jedenfall richtig.

Dort sollte nun eine ganze Reihe von Wörtern stehen, die dahinter lauter Zahlen haben wie z. B. 8 ,16, 32 usw. Es kann natürlich sein das nicht alle Zahlen vorkommen.

Dies sollte dann ungefähr so aussehen wie im Spoiler.

Spoiler anzeigen

Wenn Ihr dort dann seid, müsst Ihr nach den Werten suchen die alle Grösser 8 sind. Also dann 16, 32 usw. Die Wörter mit dem Wert 8 müssen nicht bearbeitet werden. Das Wort "CTL1" hat zwar den Wert 16 aber wenn ich weiter in der DSDT suche, kommt dieses nirgends mehr vor. Diese müssen dann nicht bearbeitet werden. Es müssen nur die Wörter bearbeitet werden, die auch dann in der DSDT noch gefunden werden wenn man diese sucht.

Damit die Methode zum Patchen funktioniert, müsst Ihr noch Methoden in die DSDT hinzufügen, damit diese die Aufteilung dann versteht.
Wenn Ihr nur Werte mit 16 habt reicht eine Methode schon aus. Bitte diese im MaciASL über die Patch Methode importieren und einfach in das Textfeld ein kopieren und mit "Patch" bestätigen.
Die Methoden zum Einfügen sind in den Spoiler zu finden.
Methode B1B2:

Spoiler anzeigen

Wenn Ihr Werte mit 32 habt braucht Ihr diese Methode noch zusätzlich:
Methode B1B4:

Spoiler anzeigen

Wenn Ihr die Methode(n) importiert habt und sich die DSDT noch ohne Fehler kompilieren lässt, kann man mit dem Patchen beginnen.

In meinem Beispiel wird dann dann das Wort "CAP0" nochmal gefunden wenn ich weiter danach in der DSDT suche. Dort taucht dann die Zeile auf.

Code

1. "Store (^^PCI0.LPCB.EC0.CAP0, Index (BFB0, 0x02))"

Danach geht man nochmal in die "OperationRegion (ECRM, EmbeddedControl, Zero, 0x0100)" und fügt jetzt zwei neue Zeile hinzu und erstellt z.B. einen Wert "CAPX, 8," und "CAPY, 8," und kann dann den Wert "CAP0, 16," löschen, da diese jetzt in die zwei Werte "CAPX" und "CAPY" aufgeteilt wurden. Die aufgeteilten Wörter müssen immer den Werten der originalen Wörter entsprechen also CAP0 hatte 16 und deshalb muss dieser nur zweimal aufgeteilt werden. Bei einem Wert mit 32 müsste dieser in vier Teile aufgeteilt werden. Nach der Änderung der Wörter kompiliere ich die DSDT nochmal und dann sollte so ein Fehler auftreten "Object not found or not accessible from scope (^^PCI0.LPCB.EC0.CAP0)" Wenn Ihr diesen Fehler anklickt kommt Ihr gleich in die betroffene Zeile und müsst nicht nochmal neu suchen.

Für Werte die nur 16 haben kommt die Methode B1B2 zum Einsatz und für Werte die 32 haben

kommt B1B4 zum EInsatz die wir vorher importiert haben.

Da für das Wort "CAP0" diese Zeile noch gefunden wurde

Code

1. "Store (^^PCI0.LPCB.EC0.CAP0, Index (BFB0, 0x02))"

müssen wir diese jetzt auf unsere geänderten Werte CAPX und CAPY abändern. Die neue Zeile würde jetzt so aussehen

Code

1. Store (B1B2(^^PCI0.LPCB.EC0.CAPX,^^PCI0.LPCB.EC0.CAPY), Index (BFB0, 0x02))

Wenn Ihr das richtig angepasst habt, sollte sich die DSDT ohne Fehler kompilieren lassen, am besten die DSDT nach jeder Änderung einmal kompilieren falls Ihr euch vielleicht irgendwo vertan habt. Dieses vorgehen macht Ihr nun für alle Werte die grösser 8 sind und auch in der DSDT noch gefunden werden. Wenn Ihr alles richtig gemacht habt, solltet Ihr dann einen korrekten Batterie Status haben.

Hätte das Feld CAP0 den Wert 32 gehabt, hätte ich dieses so aufgeteilt

Spoiler anzeigen

Die Store Methode in der der Wert "CAP0" vorgekommen ist, müsste dann so aussehen:

Code

1. Store (B1B4(^^PCI0.LPCB.EC0.CAPV, ^^PCI0.LPCB.EC0.CAPW, ^^PCI0.LPCB.EC0.CAPX,^^PCI0.LPCB.EC0.CAPY), Index (BFB0, 0x02)

Es kann auch sein das bei euch Felder noch vorkommen die den Wert 64 oder grösser haben. Mit diesen hatte ich noch nichts zu tun und kann diese leider dort hier nicht beschreiben, wie mit diesen vorgegangen wird.

Ich habe meine unbearbeitete DSDT Datei mit Namen DSDT.dsl hochgeladen und die bearbeitete mit dem Namen DSDT_bat.dsl, so das man die Änderungen nachschauen und auch nachvollziehen kann, wenn man dies an seiner eigenen DSDT probiert.

Ich hoffe das war soweit verständlich erklärt.

Gruss

Beitrag von „Noir0SX“ vom 15. Juli 2017, 00:49

Kann das sein das da ein Fehler ist

> Zitat
>
> Hätte das Feld CAP0 den Wert 32 gehabt, hätte ich dieses so aufgeteilt

sollte das letzte im Spoiler nicht y sein ?

Beitrag von „hitman20“ vom 15. Juli 2017, 00:52

Danke @BlackOSX das war ein Tippfehler von mir. Ich habe es in meinem Beitrag korrigiert.

Beitrag von „Noir0SX“ vom 15. Juli 2017, 12:33

@hitman20 Nach was kann man suchen wenn z.B. EC(EmbeddedControl) gar nicht vorkommt. Unter _BIF etc. könnten diese Infos bei mir liegen, da sieht es aber dann ganz anders aus.

EDIT
Auch in deiner fertigen Datei ist der TippFehler

Store (B1B2(^^PCI0.LPCB.EC0.CRTX, ^^PCI0.LPCB.EC0.CRTX), Local0)

Beitrag von „hitman20“ vom 15. Juli 2017, 13:46

@BlackOSX Habe die DSDT_bat.aml ausgetauscht. Den Tippfehler habe ich behoben. Dieser ist mir wahrscheinlich nicht aufgefallen weil beim Kompilieren kein Error kam. Danke.

Hast Du mal versucht nur nach "EmbeddedControl" zu suchen oder mal einzeln nach den Worten "ByteAcc oder Lock oder Preserve" ob dort was kommt? Vielleicht klappt es auch wenn Du in deiner DSDT mal nach

Code

1. Method (_BIF, 0, NotSerialized) // _BIF: Battery Information

suchst oder mal nur nach _BIF oder Battery Information ob dort was kommt. In dieser Methode sollten auch die Worte stehen die verändert werden müssen.
Sollte dort nichts kommen kannst Du vielleicht danach noch schauen

Code

1. Method (_BST, 0, NotSerialized) // _BST: Battery Status

oder dann mal nur nach _BST: Battery Status bzw. Battery Status suchen.

Vielleicht findest Du über die Worte dann die Methode die für deine DSDT korrekt ist. Du kannst mir ja sonst deine DSDT auch mal hochladen dann kann ich mal nachschauen.

Beitrag von „Noir0SX“ vom 15. Juli 2017, 18:45

Ich vermute mal `Field (WNBD, ByteAcc, Lock, Preserve)` ist der Schlüssel dazu und `BCM1` , `BCM2` die Werte die angepasst werden müssen.

Ausserhalb vom Thema Batterie:
Wenn Du einmal schaust oben nach (iASL Warning: .....), gibt es da noch andere Methoden die man in die refs.txt schreiben kann ?

Beitrag von „hitman20“ vom 15. Juli 2017, 20:12

In der Methode

Code

1. Field (WNBD, ByteAcc, Lock, Preserve)

findest Du deine Werte die für die Batterie zuständig sind. In dem Fall musst Du nur BCM1 und BCM2 anpassen dieser hat aber ein Wert von 40. Diese gehen dann aber mit meiner Methode nicht mehr. Dort muss ich mal schauen wie das mit denen geht. Meine DSDT's hatten immer nur bis 32. Bei meinem Dell XPS musste ich z. B. gar nichts anpassen. Die Zeile

Code

1. Concatenate (BCM1, BCM2, Local0)

die Du Patchen musst, macht eine Verkettung aus den zwei Werten BCM1 und BCM2.

Welche Warning meinst Du oben mit iASL Warning:? In dem Thread habe ich nichts gefunden, das Du was mit iASL Warning geschrieben hast oder habe ich es vielleicht nur übersehen? Man kann noch andere Methoden mit aufnehmen in die refs.txt aber das ist dann wieder DSDT abhängig ob dort vom iASL Compiler etwas nicht korrekt interpretiert werden kann. In der refs sind eigentlich schon die wichtigsten Methoden enthalten, die man benötigt.

Edit: @BlackOSX Ich habe mal die DSDT bearbeitet, bin mir aber nicht sicher ob das wirklich funktioniert. Kannst Du das bitte mal bei Dir testen?

Beitrag von „Noir0SX“ vom 15. Juli 2017, 21:50

Danke, das Prolem an der Sach die ging auch vorher. Ich wollte halt an Hand Deiner Anleitung das mal praktisch nach vollziehen und merkte das es halt anders ist. Werde sie trotzdem mal booten.

Beitrag von „hitman20“ vom 15. Juli 2017, 22:16

Kannst die DSDT ja mal testen auch wenn es vorhin schon ging. Die Anleitung kann man halt nicht 1:1 her nehmen weil nicht überall die Methoden den gleichen Namen haben. Wie ich ja oben geschrieben habe, kann die Anleitung halt nur bis zu den Werten 32 hergenommen werden, weil ich mit Werten über 32 noch nie zu tun hatte. Vielleicht kann die ja jemand erweitern der damit schon zu tun hatte.

Beitrag von „kuckkuck“ vom 16. Juli 2017, 07:20

> Zitat von BlackOSX
>
> Nach was kann man suchen wenn z.B. EC(EmbeddedControl) gar nicht vorkommt.

Der Embedded Controller heißt in der DSDT meist EC, EC0 oder H_EC. Bei laptops mit hardwaretechnisch wirklich vorhandenem Embedded Controller sollte dann H_EC in EC umbenannt werden, oder auch EC0->EC.

Beitrag von „derHackfan“ vom 16. Juli 2017, 18:59

Wer will sich denn mal an meiner Batterieanzeige und dem Sleep versuchen, mit Sierra und High Sierra ist da Schlaflosigkeit und Ungenauigkeit eingekehrt, ich finde weder das eine noch das andere.

Beitrag von „hitman20“ vom 16. Juli 2017, 20:14

In der SSDT-1 habe ich mal ein BAT0 Device gefunden und diese Operation Region dazu

Code

1. OperationRegion (WNBD, SystemMemory, 0xFF700100, 0x0100)
2. Field (WNBD, ByteAcc, Lock, Preserve)

In der DSDT habe ich nichts gefunden auch nichts mit EmbeddedControl. Die Batterieanzeige hat noch nie funktioniert bei Dir oder?

Was funktioniert bei Dir mit dem Sleep nicht, weil den WAK Patch hast Du aujedenfall in deiner DSDT.

Beitrag von „derHackfan“ vom 16. Juli 2017, 20:17

Oh doch, mit ML, Mav, Yos und El Capitan funzt die Batterieanzeige und Sleep, aber nicht mehr unter Sierra und HS Beta.

Beitrag von „hitman20“ vom 16. Juli 2017, 20:32

Dann wird es denke ich nicht an der DSDT, bzw. SSDT's liegen weil die Funktionsweise sich eigentlich nicht geändert hat. Hast Du mal eine aktuelle ACPIBatteryManager.kext getestet ob es vielleicht daran liegt und hast Du mit IOReg mal geschaut ob dort unter BAT0 etwas auftaucht? Ich hab meine ACPIBatteryManager.kext unter /Library/Extensions liegen aber mit Clover müsste die eigentlich auch funktionieren. Unter High Sierra lief die Batterie anzeige bei mir auch.

Wacht der Rechner gleich wieder auf wenn Du ihn in den Sleep schickst oder wacht dieser nicht mehr korrekt auf wenn er einmal im Sleep war?

Beitrag von „derHackfan“ vom 16. Juli 2017, 20:50

> Zitat von hitman20
>
> Dann wird es denke ich nicht an der DSDT, bzw. SSDT's liegen weil die Funktionsweise sich eigentlich nicht geändert hat.

Sehe ich auch so.

> Zitat von hitman20
>
> Wacht der Rechner gleich wieder auf wenn Du ihn in den Sleep schickst ...

Jup so ist es.

Ehrlich gesagt habe ich bei der Möhre keine Lust auf Fehlersuche, ich habe auch keine Lust auf Experimente mit Kexte und oder IOReg, bist jetzt brauchte ich bei den 4 x USB 3.0 nicht mal den USB InjectAll.kext.
Da lebe ich lieber mit diesen beiden Problemen, der Akku hält keine 45 Minuten und die 5Kg von dem Notebook machen es nicht unbedingt transportabel, außerdem brauche ich es unter Windows für CAD und Office.

Vielen Dank für deine Bemühungen.