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