Erledigt Wechsel von Clover auf OpenCore

Beitrag von „apfelnico“ vom 18. Dezember 2019, 19:46

> Zitat von bluebyte
>
> Bei der SSDT wusste ich nicht wie das mit OSY-Weiche funktioniert.

Gibt viele Wege. Hier ein ganz eleganter:

1. Zuerst eine "OSDW"-Methode erstellen und oben in der SSDT platzieren (oder ausserhalb in einer anderen, dann verlinken):

Code

```
1. Scope (\)
2. {
3. Method (OSDW, 0, NotSerialized)
4. {
5. If (CondRefOf (\_OSI, Local0))
6. {
7. If (_OSI ("Darwin"))
8. {
9. Return (One)
10. }
11. }
12.
13. Return (Zero)
14. }
15. }
```

Alles anzeigen

Was passiert hier? OSDW wird abgearbeitet, wenn "Darwin" erkannt, dann bekommt "OSDW" den Wert "1", ansonsten "0".

Wieso "OSDW"? Ist in der Apple ACPI zu finden und wird dort "zweiteilig" ermittelt:

Code

```
1.  Method (OSDW, 0, NotSerialized)
2.  {
3.  If ((OSYS == 0x2710))
4.  {
5.  Return (One)
6.  }
7.  Else
8.  {
9.  Return (Zero)
10. }
11. }
12.
13. Method (PINI, 0, NotSerialized)
14. {
15. OSYS = 0x07DC
16. If (CondRefOf (_OSI, Local0))
17. {
18. If (_OSI ("Darwin"))
19. {
20. OSYS = 0x2710
21. }
22. ...
```

Alles anzeigen

... das brauchen wir nicht, ist zu umständlich. Eine generelle Abfrage der Systeme ist schon in der DSDT enthalten.

Warum benutze ich "OSDW" und schreibe nicht "(_OSI ("Darwin"))" direkt rein? Weil ich's elegant haben will. Etliche modifizierte Apple SSDT haben auch abfragen nach "OSDW" drin und diese werden dann auch beantwortet.

Jetzt also die Abfrage, ob ein bestimmtes Device bei einem bestimmten System genutzt werden soll oder nicht:

Code

```
1. Scope (TATA)
2. {
3. Method (_STA, 0, NotSerialized) // _STA: Status
4. {
5. If (OSDW ())
6. {
7. Return (0x0F)
8. }
9. Else
10. {
11. Return (Zero)
12. }
13. }
14.
15. Method (_DSM, 4, NotSerialized) // _DSM: Device-Specific Method
16. {
17. ...
```

Alles anzeigen

Hier finden wir die Methode _STA (Status) und legen fest, dass das vorhandene Device "TATA", welches wir mit "Scope" weitergehend beschreiben, je nach "OSDW"-Wert ein "0x0F" (aktiv) oder "Zero" (deaktiviert) bekommt. Im Anschluss dann eine _DSM-Methode, um in das Device weitere Eigenschaften zu injecten.

Natürlich kann ich mir "OSDW" auch als Methode sparen, und schreibe gleich:

Code

```
1. Scope (TATA)
2. {
3. Method (_STA, 0, NotSerialized) // _STA: Status
4. {
5. If (_OSI ("Darwin"))
6. {
7. Return (0x0F)
8. }
9. Else
10. {
```

11. Return (Zero)
12. }
13. }
14.
15. Method (_DSM, 4, NotSerialized) // _DSM: Device-Specific Method
16. {
17. ...

Alles anzeigen