OpenCore Sammelthread (N-D-K Fork)

Beitrag von „apfelnico“ vom 16. Februar 2020, 20:35

> Zitat von JimSalabim
>
> Was macht denn eigentlich die SSDT-DTGP (oder DTPG?)?

Grundsätzlich ist eine SSDT eine individuelle Table, die man der ACPI hinzufügen kann. Man kann bestehende bearbeiten, oder eigene hinzufügen. Letzteres ist oft einfacher und überschaubarer. In der von dir genannten SSDT ist nur eine Methode drin, nämlich die "DTGP". Diese findest du auch in Apples originalen DSDT.

Code

```
1.  Method (DTGP, 5, NotSerialized)
2.  {
3.  If ((Arg0 == ToUUID ("a0b5b7c6-1318-441c-b0c9-fe695eaf949b")))
4.  {
5.  If ((Arg1 == One))
6.  {
7.  If ((Arg2 == Zero))
8.  {
9.  Arg4 = Buffer (One)
10. {
11. 0x03 // .
12. }
13. Return (One)
14. }
15.
16. If ((Arg2 == One))
17. {
18. Return (One)
19. }
20. }
21. }
22.
23. Arg4 = Buffer (One)
24. {
25. 0x00 // .
```

```
26. }
27. Return (Zero)
28. }
```

Alles anzeigen

Gebraucht wird diese Methode wiederum von anderen Methoden, den "_DSM-Methods". Wann immer man also per _DSM-Methode zusätzliche Properties neuen Geräten "Device" oder vorhandenen Geräten via "Scope" Properties via "_DSM-Methode" beibringen möchte, referenziert diese wiederum auf die Methode "DTPG". Diese kann in einer extra SSDT stehen und wird via Link im Header von anderen SSDTs aufgerufen, oder die Methode steht in der SSDT drin, wo sie eigentlich gebraucht wird.

Beispiel einer kompletten SSDT:

Code

```
1. DefinitionBlock ("", "SSDT", 2, "NICO", "TATA", 0x00000000)
2. {
3. External (_SB_.PCI0, DeviceObj)
4. External (DTGP, MethodObj) // 5 Arguments
5.
6. Scope (\_SB.PCI0)
7. {
8. Device (TATA)
9. {
10. Name (_ADR, 0x00140002) // _ADR: Address
11. Method (_DSM, 4, NotSerialized) // _DSM: Device-Specific Method
12. {
13. Local0 = Package (0x0A)
14. {
15. "AAPL,slot-name",
16. Buffer (0x09)
17. {
18. "Built In"
19. },
20.
21. "built-in",
22. Buffer (One)
23. {
```

```
0x00 // .
},

"name",
Buffer (0x12)
{
"TATAs Tata-Device"
},

"model",
Buffer (0x18)
{
"Tata-Device Version 2.0"
},

"location",
Buffer (0x02)
{
"1"
}
}
DTGP (Arg0, Arg1, Arg2, Arg3, RefOf (Local0))
Return (Local0)
}
}
}
}
```

Alles anzeigen

Wie du siehst, wird oben im Header zum einen auf das Vorhandensein von "PCI0" hingewiesen, es wird ein "Scope" zur näheren Beschreibung des vorhandenen Devices "PCI0" eröffnet, in diesem wird ein neues Device "TATA" erstellt welches an einer bestimmten Adresse zu finden ist. Über die "_DSM-Methode" werden dem Device TATA weiter Properties mitgegeben, die Methode hat ganz unten die Methode DTGP eingebunden. Damit das funktioniert, ist oben im Header ein Link nach extern für dieses "MethodObj" festgelegt. Damit ist diese SSDT zunächst "sauber" definiert. Damit das Ganze auch läuft, muss in einer anderen Table eben auch die Methode "DTGP" enthalten sein.

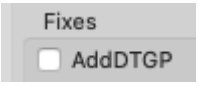
Das es bei dir auch ohne läuft, kann verschiedenste Gründe haben.

• du benutzt keine weiteren SSDTs mit _DSM-Methoden

• du benutzt weitere SSDT mit _DSM-Methoden aber ohne Einbindung von "DTGP"

• du benutzt weitere SSDT mit _DSM-Methoden aber mit der zusätzlichen DTGP-Methode innerhalb dieser SSDT(s)

• du weist Clover per config.plist an, eine "DTGP-Methode zu implementieren



EDIT:

In deinem EFI-Ordner benutzt du in den meisten SSDTs _DSM-Methoden ohne Einbindung von "DTGP", in der Thunderbolt-SSDT ebenso die meisten, nur für die Devices "NHI" und "XHC5" benutzt du eine Mischform, die so nicht sonderlich schlau ist. Jedenfalls ist da zwar DTGP eingebunden und im Header auch korrekt als extern definiert. Sieht trotzdem "unprofessionell" aus und spielt so auch keine Rolle, daher funktioniert eben auch erstaunlicherweise ...

In deiner config.plist ist mir noch aufgefallen, da ist bei USB der "EH00" (Rename von EHCI") angewählt. Ist natürlich Unfug, diesen alten reinen USB2-Controller hat das Board überhaupt nicht und ist somit auch nicht in der ACPI vorhanden.