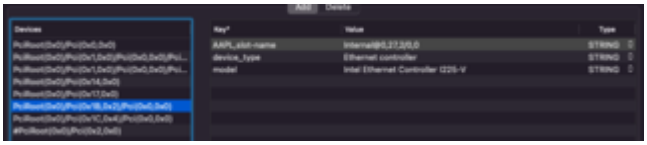
# Z490 Vision G - OpenCore 0.7.5 zeigt nur Windows Partition beim Booten

### Beitrag von „apfelnico“ vom 10. November 2021, 08:46

> Zitat von greecedrummer
>
> 

greecedrummer 5T33Z0

All diese Beschreibungen bewirken nichts im Sinne der Lauffähigkeit. "AAPL,slot-name" für ein "internes" Device zu nehmen ist an sich unsinnig. Der einzige Zweck dessen ist, damit dieses Gerät im Systembericht unter PCI gelistet wird. Und in dem Fall dann nicht wie normal in einem bestimmten PCI-Slot, sondern eben in einem "Phantasie-Gebilde". "device-type" könnte hilfreich sein, ist aber in der Regel schon erkannt, auch über die Geräteklasse. "model" ist dann abschliessend nur "wichtig" durch den ersten Eintrag, damit hier auch ein Device namentlich steht und nicht dessen Adresse.

Wirklich helfen könnten die Angaben zu "device-, vendor-, sub-vendor- und sub-device-id" sowie "compatible". Hier könnte man von den eigentlichen Daten abweichende Beschreibungen festlegen (spoofen), so dass in den Treibern hinterlegte Adressen angesprochen werden und somit eine Kext andockt. In diesem Fall würde ich aber NICHT diese OpenCore-Funktionalität bemühen, sondern dieses über eine SSDT bereitstellen. Denn diese werden frühzeitig beim Systemstart eingebunden, gegenüber OpenCores inject. Der Eintrag "name" könnte gegenüber "model" noch hilfreich sein (ebenfalls frühzeitig über SSDT) und hat mitunter nicht nur einen "kosmetischen" Auftrag. Denn je nach gewähltem SMBIOS "kann" für eine bestimmte PCI-Adresse ein bestimmtes Gerät von macOS erwartet werden, was dann von der tatsächlichen Bestückung abweicht. Das sieht man wunderbar in der IORegistry. Ein Beispiel: man hat nichts weiter deklariert, in einem PCIe-Slot steckt beispielsweise eine Soundkarte. Diese mag nicht laufen, obwohl die Treiber dazu installiert sind. Im IOReg sieht man einen Eintrag "name <ethernet>". Warum? Weil eben auf dieser Adresse bei diesem Apple-Gerät ein Ethernet-Gerät vorhanden ist. Schon wird eine völlig unpassende Geräteklasse bemüht, dessen Treiber laufen natürlich nicht und docken nicht an, die vorgesehenen Treiber werden auch nicht geladen, weil das Gerät nicht gefunden wird. Hier kann über eine SSDT frühzeitig nachgeholfen werden. Ob das jetzt im konkreten Fall auch so ist, keine Ahnung, habe ich nicht, kann ich nicht testen. Aber dieses Szenario macht deutlich, dass selbst eine einwandfreie Hardware (oder mit korrekten Adressen geflashte) unter bestimmten Umständen

eben nicht von allein laufen mag, weil eben von macOS aufgrund einer bestimmten PCIe-Position (Device Path) auf ein anderes Gerät geschlossen wird. Das ist auch nicht unbedingt ein "Fehler" von macOS gegenüber anderen Systemen wie Linux oder Windows – macOS ist nunmal grundsätzlich für eine Handvoll bekannter Macs bestimmt, und nicht für uns Hackentosher. In diesem Kontext ist ein "läuft auf jeden Fall ohne Probleme" nie hundertprozentig vorhersagbar.

Vorteil OpenCore Device Properties:

unkompliziert, in der Regel funktionierend für kosmetische Eingriffe

Nachteil OpenCore Device Properties:

erst zu einem späten Zeitpunkt realisiert, viele Properties durch ACPI und macOS in der Folge festgelegt, eigene Properties können ausschliesslich im Format "DATA" injiziert werden und öfter werden andere Formate verlangt, eigene Properties können nicht bereits gleiche vorhandene mit anderen Werten "überschreiben".

SSDT:

Properties können in den Formaten "DATA", "Number" und "String" übergeben werden, und da die ACPI (und SSDT als Teil dessen) zuerst noch vor dem System geladen wird, werden diese Properties auch "festgeklopft". Nachteil ist das Zusammenspiel von SSDT und anderen Tabellen der ACPI, vornehmlich anderer SSDT und DSDT im Auge zu behalten, die Skriptsprache zu verstehen, kann im Detail recht aufwändig werden.