HowTo: Thunderbolt HotPlug/HotSwap Finetuning für euren Hackintosh

Beitrag von „Mork vom Ork“ vom 26. August 2019, 22:01

Angefixt durch *diesen Beitrag* hier von Alex, habe ich mich die letzten 3 Tage mal nur um das Thema "Thunderbolt HotPlug" gekümmert.

Wie viele Andere (hier als auch in div. anderen Foren) auch, bin ich ebenfalls im Besitz einer Gigabyte GC Titan Ridge PCIe-Karte, welche in meinem ASRock Professional Gaming i7 in Steckplatz PCIe #5 steckt.

Bekanntlich verfügt die Karte über 2 TB3/USB-C Anschlüsse und angeschlossene TB3-Devices werden beim hochfahren des Rechners auch ordnungsgemäß erkannt. Zwar werden diese nicht, wie bei einem echten Mac,

im Bereich "Thunderbolt" der Systeminformation gezeigt, tun jedoch klaglos ihren Dienst. <--- wie gesagt, sofern sie **VOR** dem Start des Rechners angeschlossen wurden.

Nun hat Alex ja bekanntermassen eine Lösung gefunden, bei der das an- und abstecken von TB-Devices auch im laufenden Betrieb funktioniert - gleichzeitig hat er jedoch zu Protokoll gegeben, die gefundene Lösung

nicht öffentlich zu machen, sondern nur in seinen "Custom Build" zum Einsatz kommen zu lassen. Viele fanden das nicht korrekt und haben ihn dafür kritisiert (ob zu Recht steht hier **nicht** zur Debatte) - ich für meinen

Teil habe die Herausforderung für mich angenommen und mich auf die Suche nach "der" Lösung gemacht.

Hier also nun die bereits angekündigte Anleitung, mit der es möglich ist, vollständiges HotPlug/HotSwap für Thunderbolt 2/3 auf einem Hackintosh zu realisieren:

Was wird benötigt?

- Motherboard mit einem Thunderbolt-Header (um eine zusätzliche PCIe-Thunderbolt-Karte zu nutzen) oder Motherboard mit OnBoard Thunderbolt

- auf jeden Fall ein aktuelles BIOS des jeweils genutzten Boards

- SSDT für Thunderbolt (wird dieses Tutorial mitliefern)

- Kenntnisse im anpassen einer SSDT (nicht zwingend Voraussetzung, aber von Vorteil)

- mindestens ein Thunderbolt device und ein USB-C device (um HotPlug/HotSwap zu testen)

Ich schreibe dieses Tutorial auf Grund meiner Erfahrung mit folgenden ASRock Motherboards: ASRock Z270 Super Carrier (OnBoard TB) und ASRock Z370 Professional Gaming i7 (Thunderbolt-Header mit Gigabyte GC Titan Ridge PCIe-Karte)

Ich habe hier auch noch ein 10 Jahre altes ASRock Z170Extreme7+ liegen, weiss aber, das hier die Thunderbolt-Einstellungen im BIOS einfach zu mager sind, um zu garantieren, das der hier gezeigte Weg auch auf diesem Board funktionieren würde.

Warum spreche ich das an dieser Stelle an? Weil ich mir bereits diverse BIOS Dateien der verschiedensten Hersteller auf ihre Thunderbolt-Einstellungen angesehen habe und daher weiss, dass diese im Laufe der letzten Jahre wesentlich an Funktions-

umfang zugelegt und entsprechend "aufgepimpt" worden sind.

Schauen wir uns beispielsweise mal die Standardeinstellungen für Thunderbolt der von mir oben bereits erwähnten Boards ASRock Z270 und Z370 an:

ASRock Z270 Super Carrier: **ASRock Z370 Professional Gaming i7:**


Sieht reichlich dürftig aus, was die Einstellungen für Thunderbolt angeht. ABER: das sind nur die Einstellungen für Thunderbolt, die ASRock für den Enduser freigeschaltet hat. Im BIOS selber befinden sich noch diverse weitere Einstellungen, die es gilt freizuschalten.

ASUS und GIGABYTE bieten hier von Hause aus wesentlich mehr Einstellungen für den Enduser, die standardmässig freigeschaltet sind. Wie also schalten wir die anderen Funktionen für Thunderbolt frei, welche uns von Hause aus vorenthalten bleiben?

Hierzu benötigen wir das Tool "**AMIBCP**". Leider ist dieses Tool kommerziell und auch auf Anfrage nicht bei AMI zu bekommen, da es nur für Entwickler gedacht ist. Somit kann ich es auch nicht mit diesem Tutorial zur Verfügung stellen. Wer aber in der Lage ist GOOGLE

richtig zu benutzen, wird sicherlich fündig. Ich nutze für dieses Tutorial die Version 5.02.0023. Hiermit öffnen wir unser BIOS-File, wie wir es von der Hersteller Supportseite heruntergeladen haben:

ASRock Z270 Super Carrier BIOS 2.40 default	**ASRock Z370 Professional Gaming i7 BIOS 4.00 default**

WOW, was für eine Möglichkeit, sich hier an den Einstellungen auszuprobieren! Und wie schalten wir diese nun frei, damit wir Sie ggf. alle ändern können? Das wird im nächsten Schritt erklärt.

Wir stellen einfach in der Spalte "Access/Use" jeden "Default" Wert auf "USER" - wie im nachfolgenden Bild:

und speichern das ganze dann über das kleine Diskettensymbol oben links. <--- **Anmerkung von mir:** ich habe es mit div. BIOS Dateien der versch. Hersteller getestet: ASRock BIOS Dateien waren die einzigen, die sich danach auch wieder speichern ließen. BIOS Dateien von Gigabyte, Asus und MSI brachten das Programm "AMIBCP" zum Absturz, ohne das die Änderungen in der BIOS Datei gespeichert wurden.

Wenn die Änderungen erfolgreich gespeichert werden und das so "gepimpte" BIOS erfolgreich geflashed werden konnte, sollten Eure Thunderbolt-Einstellungen beim nächsten Boot wie folgt aussehen:

ASRock Z270 Super Carrier Thunderbolt settings modified:	**ASRock Z370 Professional Gaming Thunderbolt settings modified:**
 	 

Ihr seht übrigens in beiden BIOS Screenshot (Z270 und Z370) die bereits korrekten Einstellungen, die wir benötigen, damit **HotPlug/HotSwap** korrekt funktioniert. **WICHTIG** an

dieser Stelle: die Funktion "**GPIO3 Force Pwr**" muss auf "**ENABLED**" stehen! Sollte Euer Board, so wie das hier gezeigte Z270 Super Carrier, auch eine "**Thunderbolt (TM) Force Power**" Funktion haben, so ist diese ebenfalls auf "**ENABLED**" zu setzen.

Bei Boards mit Thunderbolt OnBoard kann es auch nötig sein, die zweite "**AIC Location**" Funktion von ggf. standardmäßig "**NB PCIE D01F0**" auf "**NB PCIE D01F2**" zu setzen. <--- war bei dem ASRock Z270 Super Carrier der Fall - und ich habe mir einen Wolf gesucht, warum **HotPlug/HotSwap** nicht funktioniert ! Das ASRock Z370 Professional Gaming i7 stand standardmäßig auf "**NB PCIE D01F2**", was mir beim direkten Vergleich der Einstellungen aber jedes mal entgangen ist.

Das war der **BIOS**-Teil, welcher ggf. abgearbeitet werden muss, um **HotPlug/HotSwap** zu gewährleisten. Jetzt folgt der leichte Teil: anlegen einer passenden **SSDT** für Thunderbolt. Ihr findet im Anhang zu diesem Tutorial eine von mir vorgefertigte SSDT, die Ihr mit an Sicherheit grenzender Wahrscheinlichkeit für Euer Board anpassen müsst (aber keine Sorge, die Anpassung hält sich wirklich in Grenzen, versprochen).

Beispiel SSDT für Thunderbolt: hackintosh-forum.de/attachment/111617/

Zunächst einmal müssen wir wissen, wo unsere Thunderbolt Hardware eingebunden wird. Dazu sehen wir uns mal die Einträge im IORegistryExplorer an (beide Beispiele erfolgten nach einem Boot **ohne** spezieller SSDT):

ASRock Z270 SuperCarrier (Thunderbolt OnBoard):

ASRock Z370 Professional Gaming i7 (Thunderbolt via GC Titan Ridge PCIe):

Wir sehen beim Z270 (links) den Eintrag "**AppleThunderboltHAL**" unter "**RP01@1C**", beim Z370 (rechts) den Eintrag "**AppleThunderboltHAL**" unter "**PEG2@1,2**". Ebenso sehen wir aber beim Z270 auch, dass hier noch "**PXSX**" und beim Z370 noch "**PEGP**" ins Spiel kommen. Also sind die anzupassenden Werte:

beim Z270: "**RP01**" und "**PXSX**" (wer aufmerksam hinschaut, wird feststellen, daß unsere Beispiel-SSDT offensichtlich vom Z270 stammt)

beim Z370: "**PEG2**" und "**PEGP**"

Wenn wir unsere Beispiel-SSDT öffnen, sehen wir darin den folgenden Code (der Übersichtlichkeit halber verpackt in einen SPOILER):

Spoiler anzeigen

Um die **SSDT** nun für die jeweils eigenen Bedürfnisse anzupassen, müssen von Euch folgende Zeilen geändert werden (ich nenne an dieser Stelle die entsprchenden Zeilennummern):

Zeile 23: External (_SB_.PCI0.**RP01**, DeviceObj) // (from opcode)

Zeile 24: External (_SB_.PCI0.**RP01**.**PXSX**, DeviceObj) // (from opcode)

Zeile 27: Scope (_SB.PCI0.**RP01**)

Zeile 29: Scope (**PXSX**)

Diese Zeilen bestimmen, auf welchem PCI0 Platz Eure Thunderbolt-Hardware sitzt. Die im Beispiel gezeigte Lösung kommt von meinem ASRock Z270 Super Carrier, welches Thunderbolt OnBoard mitbringt und daher unter **RP01/PXSX** zu finden ist.

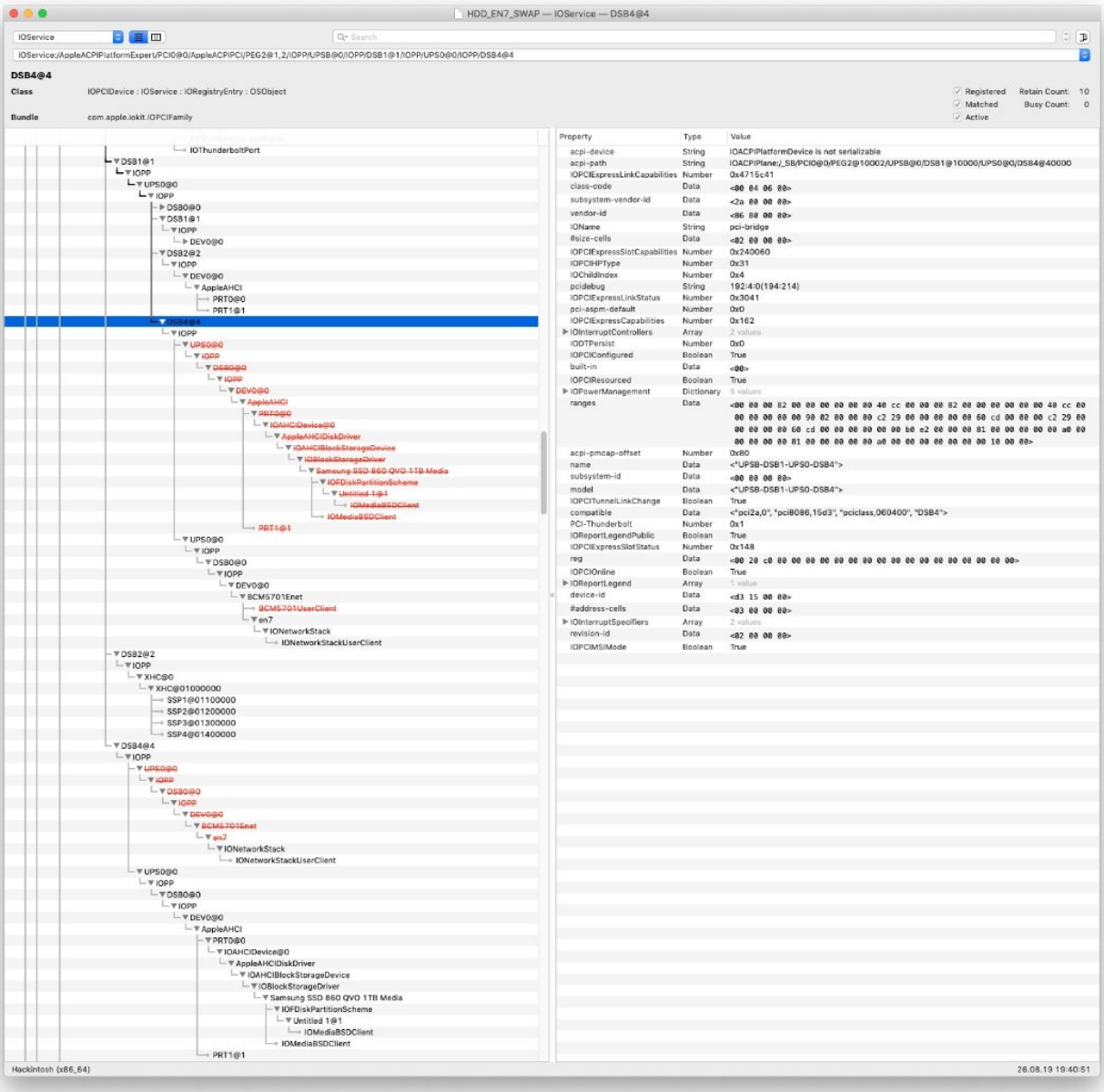
Mein ASRock Z370 Professional Gaming i7 nutzt einen GC Titan Ridge PCIe Karte und sitzt bei mir daher standardmässig auf _SB_.PCI0.PEG2.PEGP, dementsprechend würde die Änderung wie folgt aussehen (wieder im Spoiler):

Spoiler anzeigen

Sprich: wenn Ihr erstmal rausgefunden habt, wo Eure Thunderbolt Hardware sitzt und entsprechend die sechs genannten Zeilen angepasst habt, dann seid Ihr auch schon fertig. Wenn Ihr die Datei anders benennen wollt ist das auch kein Problem. Ich habe sie nur so benannt, weil ich mir das Namensschema an einem original iMac19,1 abgeschaut habe.

Nun müsst Ihr nur noch die fertige SSDT nach "***EFI/CLOVER/ACPI/patched***" kopieren, Rechner neu starten und wenn alles richtig gemacht wurde, Eure Thunderbolt Devices an-, ab- oder umstecken - und alles im laufenden Betrieb!

Im IORegistryExplorer sollte das HotPlug/HotSwap sich dann ähnlich diesem Screenshot bemerkbar machen:



Hier wurde beispielsweise eine TB-SSD und ein TB-to-Ethernetadapter beim hochfahren eingesteckt und während des laufenden Betriebs wurden beide Geräte jeweils auf den anderen Port geHotSwitched. Man sieht es an den **roten** Einträgen, die zeigen, das an jenem Port zuvor mal ein anderes Gerät vorhanden war.

That's **HotPlug/HotSwitch** for Hackintosh. Kein Hexenwerk, wenn man weiss, wie es geht. Aber eins kann ich Euch sagen: der Weg zu einer funktionierenden Lösung war steinig und geprägt von diversen Neustarts, lautstarken Flüchen und einem sehr lauten **YES**, als es dann doch funktioniert hat.

BtW: wer sich fragt, warum die SSDT nur aus den Einträgen UPSB und DSB0 besteht - und nicht aus weit mehr UPS0 und DSBx Einträgen, dem sei gesagt, dass ich versucht habe, weitere Einträge hinzuzufügen (wie es in einer original APPLE TB-SSDT der Fall ist), dann jedoch massiv Probleme beim HotPlug/HotSwap

innerhalb einer längeren TB-Device-Kette bekommen habe. So hatte ich versuchsweise mal alle meine TB-Geräte "in Reihe" gesteckt, und die dafür vorgesehenen UPSB und DSBx Einträge gesetzt, als dann jedesmal nach einem HotPlug an irgendeinem Punkt der Reihe mein Rechner jedesmal prompt neu gestartet

ist. Nutze ich jedoch die SSDT aus diesem Tutorial, kann ich jederzeit an jedem Punkt der Kette ein TB-Device trennen, ohne das mein Rechner neu startet. Natürlich geht so der Weg verloren, über einen _DSM-Eintrag angeschlossene Geräte mit einem "name" und "model" zu versehen, mit welchem es sich dann

innerhalb der Systeminformationen unter dem Punkt "PCI" registrieren würde, aber hier ging mir Funktionalität **vor** Design.