

SSDT für USB-Ausgänge unter OpenCore ohne Kext erstellen

Beitrag von „Der_Trottel“ vom 23. Oktober 2020, 22:21

Ich versuche Anhand dieser kurze Anleitung zu zeigen, wie ihr einfach unter OpenCore eine SSDT für die USB-Ausgänge erstellt.

Was wir benötigen:

- Hackintool
- MaciASL
- 2 USB-Stick (1x nur USB2.0 und 1x USB3.0)

Erste Schritt:

Zuerst wollen wir die USB-Ausgänge indentifizieren, welsche Nummer gehört zum welchen Ausgang, daher aktivieren wir den quirk "XhciPortLimit" Unter Kernel -> Quirks

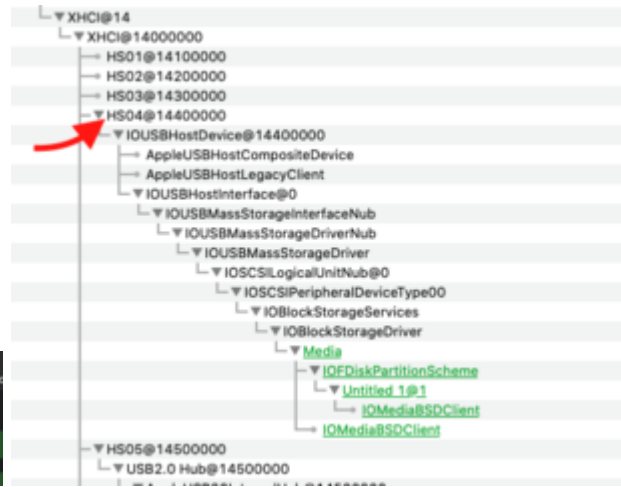
dann starten wir den PC neu.

Hier öffnen wir Hackintool und weckseln zu USB Fenster, hier werden wir alle USB-Ausgänge ansehen, wenn nicht einfach mal mit dem Tool aktualisieren. Wer mag kann auch an diese Stelle iORegistryExplorer verwenden.

Dann stecken wir in jeden Ausgang 1x USB2.0 und USB3.0 und notieren auf einem Zettel die Nummern.

Hier an diesem Beispiel sehen wir HS04 mit USB2.0-Stick

Type	Name	Location ID	Port	Connector	Dir. Speed	Device	
XHCI	HS01	0x14100000	0x01	Internal	Unknown		
XHCI	HS02	0x14200000	0x02	Internal	Unknown		
XHCI	HS03	0x14300000	0x03	Internal	Unknown		
XHCI	HS04	0x14400000	0x04	Internal	480 Mbps	IOUSBHostDevice	
XHCI	HS05	0x14500000	0x05	Internal	480 Mbps	USB2.0 Hub	
XHCI	HS06	HS06	0x14800000	0x06	Internal	Unknown	
XHCI	HS07	HS07	0x17000000	0x07	Internal	480 Mbps	Keyboard Hub



Und hier den selben USB-Ausgang mit USB3.0 Stick

Type	Name	Location ID	Port	Connector	Dir. Speed	Device
XHCI	HS01	0x14100000	0x01	Internal	Unknown	
XHCI	HS02	0x14200000	0x02	Internal	12 Mbps	Apple Custom Header Interface
XHCI	HS03	0x14300000	0x03	Internal	Unknown	
XHCI	HS04	0x14400000	0x04	Internal	480 Mbps	Ultra USB 3.0
XHCI	HS05	0x14500000	0x05	Internal	Unknown	
XHCI	HS06	0x14800000	0x06	Internal	Unknown	
XHCI	HS07	0x14900000	0x07	Internal	Unknown	



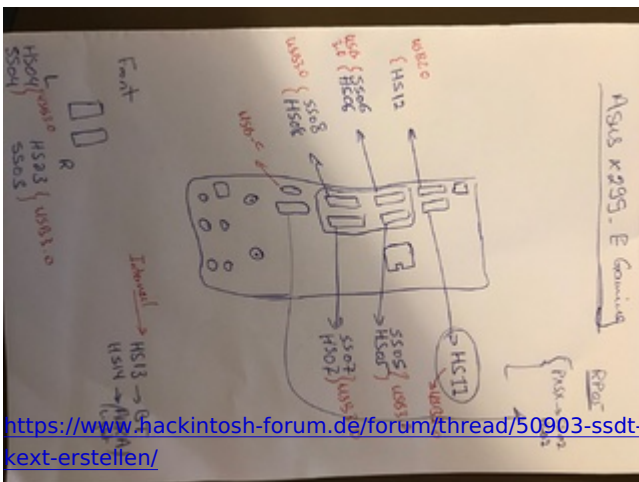
Das heißt diesen Ausgang hat die Nummer 4 und gleichzeitig besitzt USB2.0 & USB3.0, daher müssen wir diesen Ausgang nachher als USB3.0 deklarieren. Wenn dieser Ausgang nur USB2.0 hat dann müssen wir ihn als USB2.0 deklarieren

USB3.0 = 0x03

USB2.0 = 0x00

Internal = 0xFF

USB-C = 0x0A



X299-E Gaming Mainboard

Da in macOS System die Ausgänge auf 15 begrenzt sind, dann dürfen wir nur 15 Ausgänge auswählen.

zum Beispiel habe ich mich für 3(2x), 4(2x), 5(2x), 6(2x), 7(2x), 8(2x), 11, 13, 14 entschieden, heißt es insgesamt 15.

Zweite Schritt:

Öffnen wir MaciASL und unter File -> New from ACPI sieht man alle DSDT und SSDT des Systems, hier suchen wir die SSDT für unsere USB-Ausgänge (meisten heißt es "A M I" unter X299 Mainboards und "xh_rvp" unter Zxxx Mainboards)

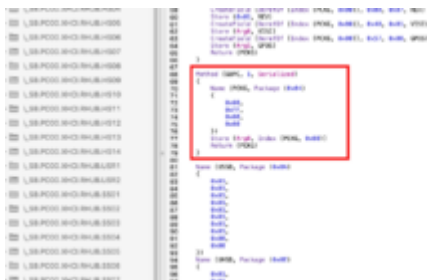
So sieht für mein Asus X299 aus



Bevor wir mit irgendwas anfangen, schreiben wir die Länge der SSDT auf, weil nachher diese SSDT in config.plist deaktivieren wollen und stattdessen unsere SSDT laden lassen.

Am Anfang steht Methode GUPC, wo jeder Ausgang es zurück gibt, daher steht für alle Ausgänge.

Hier wollen wir diese Methode für jeden Ausgang allein schreiben, deshalb löschen wir sie.



danach löschen wir die Ausgänge, die wir nicht brauchen, bei mir sieht es so aus



Unter jedem Ausgang steht die Methode "Method (_UPC, 0, NotSerialized)", die die Methode GUPC zurück gibt, was wir vorher gesehen haben, deshalb ändern wir diese Methode zu

Code

1. Method (_UPC, 0, NotSerialized) // _UPC: USB Port Capabilities
2. {
3. Name (UPCP, Package (0x04)
4. {
5. One,
6. 0x03,
7. Zero,
8. Zero
9. })
10. Return (UPCP)
11. }

Alles anzeigen

Und zwar für jeden Ausgang, hier ist dabei zu beachten, an erste Stelle steht One und an der 2. Stelle die Codierung für deinen Ausgang, ob USB3.0(0x03) oder USB2.0(0x00) oder Internal (0xFF) ist

Hier ist einer Ausschnitt aus mein SSDT

```

1009 Scope (\_SB.PCI0.H01.0000.0000)
1010 {
1011     Method (_PRP, 0, NonSerialized) // _PRP: USB Port Capabilities
1012     {
1013         Name (_PRP, Package (1x04))
1014         {
1015             0x,
1016             0x,
1017             0x,
1018             0x
1019         }
1020         Return (_PRP)
1021     }
1022 }
1023 Scope (\_SB.PCI0.H01.0000.0000)
1024 {
1025     Method (_PLD, 0, NonSerialized) // _PLD: Physical Location of Device
1026     {
1027         Return (_PRP.ChassisID (Index 0x00, 0x001, 0x001))
1028     }
1029 }
1030 Scope (\_SB.PCI0.H01.0000.0000)
1031 {
1032     Method (_PRP, 0, NonSerialized) // _PRP: USB Port Capabilities
1033     {
1034         Name (_PRP, Package (1x04))
1035         {
1036             0x,
1037             0x,
1038             0x,
1039             0x
1040         }
1041         Return (_PRP)
1042     }
1043 }
1044 Scope (\_SB.PCI0.H01.0000.0000)
1045 {
1046     Method (_PLD, 0, NonSerialized) // _PLD: Physical Location of Device
1047     {
1048         Return (_PRP.ChassisID (Index 0x00, 0x001, 0x001))
1049     }
1050 }
1051 Scope (\_SB.PCI0.H01.0000.0000)
1052 {
1053     Method (_PRP, 0, NonSerialized) // _PRP: USB Port Capabilities
1054     {
1055         Name (_PRP, Package (1x04))
1056         {
1057             0x,
1058             0x,
1059             0x,
1060             0x
1061         }
1062         Return (_PRP)
1063     }
1064 }

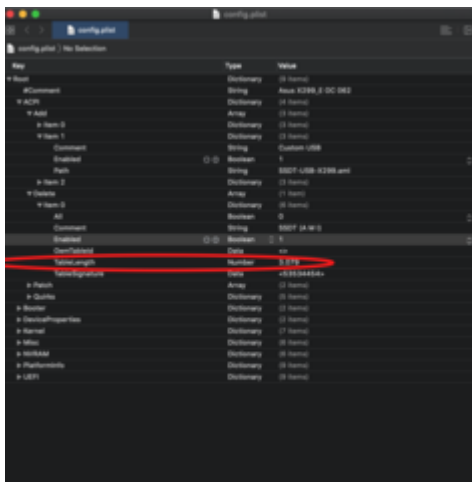
```

danach speichern wir die SSDT als ACPI Machine Language Binary und wir geben sie irgendeine Name zum Beispiel SSDT-USB-X299.aml

Dritte Schritt:

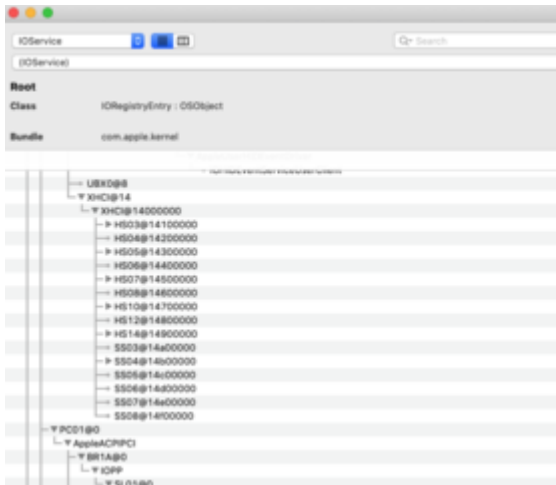
wählen wir die EFI Ordner und öffnen die config.plist, hier tragen wir unsere SSDT unter ACPI ein und SSDT selber kommt in den ACPI Ordner

Unter ACPI -> Delete deaktivieren wir die originale SSDT



danach deaktivieren wir wieder den quirk "XhciPortLimit" und starten den PC neu.

Am Ende Sieht so aus unter iOREg.



Beitrag von „Schorse“ vom 23. Oktober 2020, 22:41

Der_Trottel Fein fein, prima Anleitung. Danke

Beitrag von „Gichin“ vom 24. Mai 2021, 15:42

Danke für die Anleitung! Durch das Update auf 11.3.1 mit teilweise komplettem USB-Versagen hat das nochmal Aktualität bekommen. Der gestern Abend frisch installierte iHac (OC 0.6.9, Big Sur 11.3 auf i7-6700, GA Z170X-Gaming7, RX580) hat bis heute Morgen anstandslos funktioniert, bis wir das "Sicherheitsupdate" eingespielt haben und der Rechner sich beim Sleep-Modus verschluckt hat und neu gebootet hat...

Ich habe aber mal eine Frage: Du schreibst "aktivieren wir den quirk "XhciPortLimit" Unter Kernel -> Quirks". Du meinst in der config.plist?

Und: Muss man vorher USBInjectAll.kext entfernen?

Über eine kurze Rückantwort würde ich mich sehr freuen!

Beitrag von „Rentier Rudi“ vom 24. Mai 2021, 16:19

Bei mir gingen nur meine vier USB2.0-Ports vorne am Gehäuse. Ich habe dann den *quirk "XhciPortLimit" disabled*. Dann gingen alle Ports.

Die Frage ist, kann ich das so lassen? Der USBInjectAll.kext ist weiterhin aktiv.

Komischerweise wird keiner der USB-Ports im Hackintool als aktiv angezeigt, wenn ich Sticks reinstecke.

Beitrag von „HackBook Pro“ vom 24. Mai 2021, 16:24

Du kannst auch einen USB Injector Kext nehmen, den kannst du ganz einfach mit [USBMap](#) erstellen.

Edit: USBInjectAll brauchst du damit nicht mehr.