

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 identifizieren, welche 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 wechseln 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 | 0x14700000 | 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 | 0x14700000 | 0x07 | Internal | Unknown | |



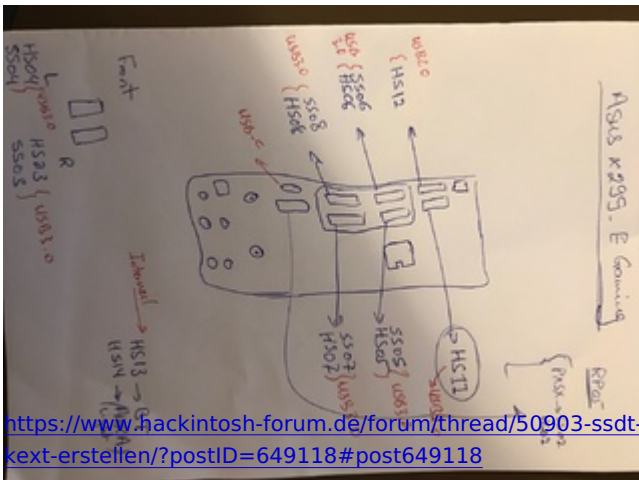
Das heißt diesen Ausgang hat die Nummer 4 und gleichzeitig besitzt USB2.0 & USB3.0, daher müssen wir diesen Ausgang nachher als USB3.0 deklarieren. Wenn dieser Ausgang nur USB2.0 hat dann müssen wir ihn als USB2.0 deklarieren

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)

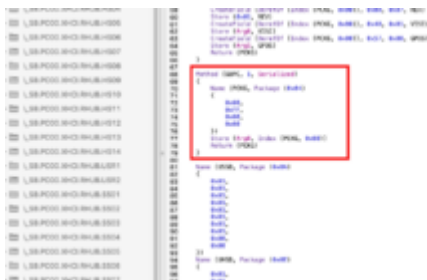
So sieht für mein Asus X299 aus



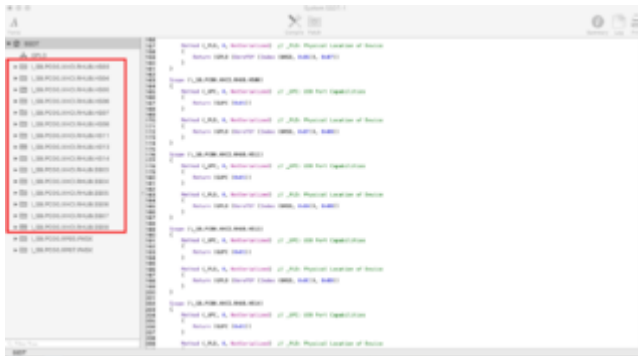
Bevor wir mit irgendwas anfangen, schreiben wir die Länge der SSDT auf, weil nachher diese SSDT in config.plist deaktivieren wollen und stattdessen unsere SSDT laden lassen.

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

```

System 0000-0
Complete Patch

1000 Scope (\_SB.PCI0.HXCI_0000)
1001 {
1002     Method _SOPC, 0, NotSerialized // _SOPC: USB Port Capabilities
1003     {
1004         Name _SOPC, Package (0x00)
1005         {
1006             0x00,
1007             0x00,
1008             0x00
1009         }
1010         Return (_SOPC)
1011     }
1012 }
1013
1014 Scope (\_SB.PCI0.HXCI_0001)
1015 {
1016     Method _SOPC, 0, NotSerialized // _SOPC: USB Port Capabilities
1017     {
1018         Name _SOPC, Package (0x00)
1019         {
1020             0x00,
1021             0x00,
1022             0x00
1023         }
1024         Return (_SOPC)
1025     }
1026 }
1027
1028 Method _PUB, 0, NotSerialized // _PUB: Physical Location of Device
1029 {
1030     Return (_PUB)
1031 }
1032 }
1033
1034 Scope (\_SB.PCI0.HXCI_0002)
1035 {
1036     Method _SOPC, 0, NotSerialized // _SOPC: USB Port Capabilities
1037     {
1038         Name _SOPC, Package (0x00)
1039         {
1040             0x00,
1041             0x00,
1042             0x00
1043         }
1044         Return (_SOPC)
1045     }
1046 }
1047
1048 Method _PUB, 0, NotSerialized // _PUB: Physical Location of Device
1049 {
1050     Return (_PUB)
1051 }
1052 }
1053
1054 Scope (\_SB.PCI0.HXCI_0003)
1055 {
1056     Method _SOPC, 0, NotSerialized // _SOPC: USB Port Capabilities
1057     {
1058         Name _SOPC, Package (0x00)
1059         {
1060             0x00,
1061             0x00,
1062             0x00
1063         }
1064         Return (_SOPC)
1065     }
1066 }
1067
1068 Method _PUB, 0, NotSerialized // _PUB: Physical Location of Device
1069 {
1070     Return (_PUB)
1071 }
1072 }

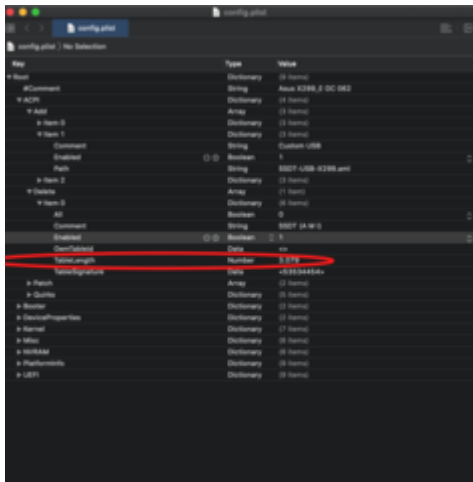
```

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.

