2.26 USB XHC ohne USBInjectAll.kext oder USB-Portlimit Patch zum Laufen bringen

Hier der Versuch einer Anleitung um alle USB-3 Ports eines Rechners ans Laufen zu bringen ohne denn USBInjectAll.kext oder einem USB-Portlimit Patch.

Einmal eingerichtet ist diese Lösung update sicher. Voraussetzung zum Einrichten sind Grundkenntnisse im Auslesen aus dem Bios und Bearbeiten von DSDT.aml und SSDT's. Weiter wird das Programm MaciASL und IORegistryExplorer benötigt.

1. Vorbereitung:

Das Ganze funktioniert, wenn entweder in der DSDT.aml oder in einer SSDT Einträge zu finden sind wie hier zu sehen.

Code

```
1. Scope (\_SB.PCI0.XHC.RHUB.HS01)
2. {
3. Method (_UPC, 0, NotSerialized)
4. {
5. Return (GUPC (One))
6. }
7. Method (_PLD, 0, NotSerialized)
8. {
9. Return (GPLD (One, One))
10. }
11. }
12. }
```

Alles anzeigen

HS01 ist der erste USB-Port. Weiter gibt es noch HS02 – HSXX und USR1 – USRX und SS01 – SSXX.

Damit sind alle für diesen Rechner möglichen USB-Ports beschrieben.

OSX erkennt maximal die ersten 15 Ports. In dieser Anleitung geht es daher darum den Rechner auf maximal 15 Ports zu reduzieren was meistens möglich ist da nicht alle Ports in einem Rechner auch mit einem Anschluss belegt sind.

In diesem Beispiel gibt es eine SSDT mit dem Namen SSDT-4-xh_rvp08.aml welche genau diese Einträge für jeden USB-Port besitzt. Der Name der Datei kann auch etwas abweichend lauten wie etwa SSDT-4-xh_rvp10.aml, oder die Einträge befinden sich direkt in der DSDT.aml.

Hier im Beispiel bleiben wir bei der SSDT-4-xh_rvp08.aml.
Damit Änderungen in dieser Datei erhalten bleiben speichere ich die als Text Datei *.dsl mit dem MaciASL ab. Das hat den Vorteil das Notizen nicht gelöscht werden.

Für die spätere Verwendung der Datei im Ordner

Code

1. /EFI/CLOVER/ACPI/patched

Muss diese natürlich wieder als SSDT-4-xh_rvp08.aml abgespeichert werden damit das System die Datei versteht.

2. Hintergrundwissen:

Der SSDT Eintrag „Return (GUPC (One))" ruft eine Methode in der SSDT auf welche denn Anschlusstyp setzt. Möglichkeiten sind USB-2, USB-3, Interner Anschluss und Anschluss abgeschalten.

Im Bespiele finden wir eine Methode:

Code

```
1. Method (GUPC, 1, Serialized) // Method XUS3
2. {
3. Name (PCKG, Package (0x04)
4. {
5. 0xFF,
6. 0x03,
7. Zero,
8. Zero
9. })
10. PCKG [Zero] = Arg0
11. Return (PCKG)
12. }
```

Alles anzeigen

welche HS01 als einen USB-3 Anschluss ausweist. Im Grunde ist es mit diesem Hintergrundwissen einfach die Ports damit soweit zu reduzieren das nur noch die gewünschten und aktiven Ports übrig bleiben. Erreichen kann man das durch erzeugen einer eigener Methoden die den gewünschten Anschlusstyp definieren. Diesen ersetzen wir dann beim entsprechenden Anschluss durch den Namen dieser neunen Methode.

Hier die zur Umsetzung nötigen neuen Methoden:

Methode XUSB2 = USB2

Code

```
1. Method (XUS2, 1, Serialized) // Method XUS2
2. {
3. Name (PCKG, Package (0x04)
4. {
5. 0xFF,
6. Zero,
7. Zero,
8. Zero
9. })
10. PCKG [Zero] = Arg0
11. Return (PCKG)
12. }
```

Alles anzeigen

Method XUS3

Code

```
1. Method (XUS3, 1, Serialized) // Method XUS3
2. {
3. Name (PCKG, Package (0x04)
4. {
5. 0xFF,
6. 0x03,
7. Zero,
8. Zero
9. })
10. PCKG [Zero] = Arg0
11. Return (PCKG)
12. }
```

Alles anzeigen

Methode XUSI = Internal

Code

```
1. Method (XUSI, 1, Serialized) // Method USB Intern
2. {
3. Name (PCKG, Package (0x04)
4. {
5. 0xFF,
6. 0xFF,
7. Zero,
8. Zero
9. })
10. PCKG [One] = Arg0
11. Return (PCKG)
12. }
```

Alles anzeigen

Methode XUSC = USB-C

Code

```
1. Method (XUSC, 1, Serialized) // Method USB-C
2. {
3. Name (PCKG, Package (0x04)
4. {
5. 0xFF,
6. 0x0A,
7. Zero,
8. Zero
9. })
10. PCKG [One] = Arg0
11. Return (PCKG)
12. }
```

Alles anzeigen

Method USB OFF

Code

```
1. Method (XOFF, 1, Serialized) // Method USB OFF
2. {
3. Name (PCKG, Package (0x04)
4. {
5. Zero,
6. Zero,
7. Zero,
8. Zero
9. })
10. PCKG [One] = Arg0
11. Return (PCKG)
12. }
13. }
```

Alles anzeigen

Als Beispiel wird jetzt, wenn wir „Return (GUPC (One))“ bei HS01 ersetzen durch XUSC aus dem USB-3 ein USB-C Anschluss. Das Ganze kann natürlich nicht wirklich einen USB-C erstellen, aber zumindest erhält OSX jetzt die Information HS01 du bist jetzt ein USB-C Anschluss.

Besonders wichtig ist hier die Methode XOFF. Diese definiert den USB-Port als deaktiviert und der Port wird nicht weiter als USB-Port unter OSX angezeigt.

Das ist auch der Trick um auf das Portlimit von maximal 15 Ports zu kommen.

3. Umsetzung:

Wichtig ist für das Gelingen mit dieser Anleitung das USBInjectAll.kext nicht verwendet wird. USBInjectAll.kext aktiviert immer alle Ports, egal ob diese aktiv sind oder nicht.

Jetzt kommt der IORegistryExplorer ins Spiel. Mit diesem Programm kann man herausbekommen welche Ports aktiv sind und mit was die aktiven Ports belegt sind. Weiter zeigt das Tool beim Einstecken eines USB-Sticks an auf welchen Port dieser gerade eingesteckt wurde.

Wir suchen dazu im IORegistryExplorer nach USB. Dann sehen wir alle aktiven Ports. Wir der Stick eingesteckt leuchtet der Port grün auf. Beim Abziehen des Sticks rot.

Wir haben somit ein Tool mit welchen wir erkennen können hinter welchen USB-Ports ein aktiver Anschluss hängt und hinter welchem Port nicht.

Genau auf diese Informationen setzten wir dann die entsprechenden „Return (GUPC (One))“ Einträge der dazugehörigen Ports.

Nach einem Neustart sehen wir jetzt weitere Ports welche vorher nicht sichtbar waren. Jetzt wiederholt sich das Spiel mit dem einstecken des USB-Sticks solange bis wir alle unsere Ports mit den neunen Methoden belegt haben.

Am Schluss sollten dann maximal 15 aktive Ports übrig bleiben und wir haben das Ziel erreicht das alle Ports ab jetzt auch nach Updates funktionieren.

Im Anhang findet ihr eine SSDT-4-xh_rvp08.dsl in welchem dieses Wissen das als Beispiel umgesetzt wurde.

Viel Glück!

Anregungen und Diskussionen zu dieser Anleitung bitte in diesem Thread hier führen.