

Kext as Kext can oder USB 3.0 ohne USBInjectAll

Beitrag von „Brumbaer“ vom 16. August 2017, 15:07

Worum geht's hier eigentlich ?

Das Folgende beschreibt eine Möglichkeit USB 3.0 Anschlüsse ohne USBInjectall zum Laufen zu bekommen.

Der Sinn dieses Artikels ist allerdings nicht einen Ersatz für USBInjectAll zu schaffen, sondern Hintergrundwissen über Kexte zu vermitteln.

Für eine vollständige Beschreibung der Funktionsweise, Anwendung und Einsatz von Kexten seien Interessierte an die Apple Dokumentation verwiesen.

Das beschriebene Verfahren lässt sich auf alle Arten von Treibern anwenden, ob Bluetooth, WiFi oder Grafik.

Das hier gegebene Beispiel wurde unter High Sierra getestet und sollte genauso unter Sierra funktionieren.

Kext, ein Kext ?

Kexte erlaube es einem Programme als wären sie Teil des Kerns ablaufen zu lassen.

Man kann drei generelle Arten von Kexten unterscheiden, die eine Routine enthalten die einfach gestartet wird, dann Services, die mit Treibern vergleichbar sind und Bibliotheken, die anderen Kexten Routinen zur Verfügung stellen.

Uns interessieren besonders die Kexte die Treiberfunktion haben.

Jeder hat sein Bündel zu tragen

Ein Kext ist ein Bundle - ein Bündel. Ein Bundle tritt nach aussen hin wie eine Datei auf, ist aber ein Ordner. Beim Doppelclick im Finder öffnet sich der Ordner nicht, sondern macht je nach Art des Bundles etwas anderes.

Apps sind Bundles und was die bei einem Doppelclick machen, weiß jeder.

Ein Doppelclick auf ein Kext führt zur Nachfrage, was man denn mit dem Ding machen will, wie bei einem unbekanntem Dateityp.

Allerdings kann man doch in ein Bundle hineinsehen. Im Finder macht man einen Rechtsklick auf das Bundle und über den Menüpunkt "Paketinhalt zeigen" kann man das Bundle wie einen Ordner öffnen und seinen Inhalt sehen und bearbeiten.

Es endet mit einem .kext

Kexte sind Bundles also Ordner die mit der Endung .kext enden.

Wie wir schon festgestellt haben, kann man ein Kext nicht mit einem Doppelclick starten. Stattdessen werden Kexte beim Systemstart geladen, wenn sie in einem der folgenden Ordner liegen:

- /System/Library/Extensions
- /Library/Extensions

Es genügt aber nicht, dass das Kext in einem dieser Ordner liegt, es muss auch

- als Eigentümer "root"
- als Gruppe "wheel"
- und die Rechte 755 haben.

Wenn man ein Kext mit einem Hilfsprogramm wie Kext Utility installiert werden die drei Dinge automatisch angepasst. Kopiert man das Kext händisch muss man selbst Sorge dafür tragen, dass sie angepasst werden.

Verwendet man Clover kann man Kexte beim Systemstart laden, indem man sie in den EFI/CLOVER/kexts/Other Ordner auf der EFI-Partition legt.

Clover sind Eigentümer, Gruppe und Rechte egal, deshalb genügt das Kopieren des Kextes in den Ordner.

Form ohne Inhalt ist nichts.

Wie gesagt ein Kext ist ein Bundle ist ein Ordner.

Der Inhalt eines Kextes ist in gewissem Rahmen vorgeschrieben.

Der eigentliche Inhalt kann direkt im Kext(dem Ordner) liegen. Häufig befindet sich aber im Kext ein Unterordner Contents, der dann den eigentlichen Inhalt enthält.

Ob man einen Contents Ordner zwischen schaltet oder nicht, bleibt jedem selbst überlassen.

Beide Varianten sind gleichwertig.

- Das Kext muss eine Datei Info.plist enthalten.
- Wenn das Kext ausführbaren Code enthält liegt er im Unter-Ordner MacOS.
- Wenn ein Kext Unter-Kexte hat, liegen diese im Unter-Ordner Plugins.
- Wenn ein Kext signiert ist, liegt die zugehörige Datei im UnterOrdner _CodeSignature. Über [SIP](#) kann man die Signatur Überprüfung abschalten. um nicht signierte oder Kexte mit falscher Signatur (weil man was am Kext geändert hat).

Legt man die Sachen in anderen Ordnern ab, findet macos sie einfach nicht.

Das Kext kann weitere Dateien und Ordner enthalten.

Ein schöner Vertreter seiner Art

Schauen wir uns doch mal ein Kext an.

IOUSBHostFamily verwendet einen Contents Ordner.

In dem gibt es die oben angeführten Dinge plus *version.plist*.

Die *version.plist* hat zusätzliche Versionsinformationen und ist für die Funktion nicht interessant.

Alleine anhand der Namen kann man schon ein paar Rückschlüsse ziehen

- **IO**: Ein- Ausgabe
- **USB**: betrifft USB 😊
- **Host**: keine Endgeräte, sondern die, die den Bus kontrollieren.
- **Familiy**: Gilt für eine ganze Reihe von Geräten.

Die allgemeine Funktionalität wird von IOUSBHostFamily zur Verfügung gestellt, aber es gibt Plugins, die die Details für bestimmte Controller oder Controller Arten abhandeln.

Auch bei den Plugins helfen die Namen weiter.

AppleUSB*HCIPCI sind

- **Apple** eigene Treiber
- für **USB** Controller
- der Typen **OHCI**, **UHCI**, **XHCI**, **EHCI**,
- die am **PCI** Bus angeschlossen sind.

Ein stolzer Spross der Familie

Wir interessieren uns für USB 3.0 d.h. XHCI Controller. Die Controller sind über PCI angeschlossen.

Schauen wir uns also das *AppleUSBXHCIPCI.kext* an.

Es zeigt sich ein inzwischen vertrautes Bild:

_CodeSignature enthält die Signatur, kümmert uns nicht.

MacOS enthält die Datei mit der Software, da reinzuschauen und zu analysieren was passiert, führt viel zu weit. Uns genügt es zu wissen, dass, da die Datei *AppleUSBXHCIPCI* in *MacOS* liegt, das Kext eigenen Programmcode enthält.

version.plist

ist wie gesagt nur Versionsinformation
Und das lässt nur noch die *Info.plist* übrig.

Informationen sind heutzutage Alles

Die *Info.plist* ist an jedem Kext das Interessanteste.

Eine *plist* ist eine Datei, die eine Liste von Eigenschaften enthält, eine Property List halt. *PLists* haben ein festes Format, so wie Word Files auch.

Eine *PList* ist ein XML Dokument und es gibt auch eine DTD dazu.

Man kann eine *PList* mit einem Text Editor oder einem XML Editor bearbeiten.

XCode enthält auch einen *PList Editor*, der das Bearbeiten etwas komfortabler als mit einem Texteditor macht.

XCode ist jedem zugänglich, also verwende ich im Folgenden dessen *PList Editor*, denn seine Darstellung ist übersichtlicher als die im Texteditor und er kümmert sich um die Datenstruktur, So werden Fehler bei Änderungen vermieden.

Die Darstellung ist vergleichbar mit der Listendarstellung im Finder. Statt Dateien haben wir Einträge der Datentypen Bool, Data, Date, Number oder String. Statt Ordner haben wir Container in Form von Dictionary oder Array.

Wie man Einträge ändert, löscht und hinzufügt, bitte ich, falls es sich einem nicht von selbst erschließt, im Netz zu recherchieren.

So sieht die *Info.plist* von *AppleUSBXHCIPCI.kext* aus

IOKitPersonalities ist mit Absicht nicht aufgeklappt, denn es ist ellenlang und ich möchte erst über die anderen Dinge sprechen.

Viele Einträge dienen nur der Information des Lesers, aber einige sind für die Funktion essentiell.

Wenn ein Kext geladen wird, startet es nicht automatisch. Es wird erstmal im Kernel mit einem Namen versehen abgelegt.

Diesen Namen findet man unter "*Bundle identifier*". Es kommt vor, dass man sich auf ein anderes Kext bezieht, dann braucht man diesen Namen.

Wir haben gesehen, dass dieses Kext Programmcode enthält (Datei im MacOS Ordner). Der Dateiname wird unter "*Executable file*" eingetragen. MacOS sucht im *MacOs* Ordner des Kextes nach der Datei,

"*OSBundleRequired*" gibt an wozu das Kext gebraucht wird. Das ist in den meisten Fällen "*Root*". Kexte für Netzwerkfunktionen haben oft "*Network-Root*" und Kexte, die auch im abgesicherten Modus geladen werden sollen "*Safe Boot*".

Das Kext kann Bibliotheken benötigen. Welche es benötigt und deren Version steht unter "*OSBundleLibraries*". Sollte das System keine kompatiblen Bibliotheken finden, startet das Kext nicht.

Starke Persönlichkeiten oder gespaltene Persönlichkeit ?

Und jetzt wird es spannend *IOKitPersonalities*.

Wenn es diesen Eintrag gibt, liegt ein Kext mit Service (Treiber) Funktion vor.

Gibt es einen solchen Eintrag, so enthält er einen oder mehrere Einträge für Services, die gestartet werden können.

Jeder Eintrag enthält Bedingungen, die erfüllt sein müssen damit der Service gestartet wird. Dabei sind die Services voneinander unabhängig, es sein denn man macht einen explizit von einem anderen abhängig.

Dieser Überprüfung findet zu bestimmten Zeitpunkten bzw. bei bestimmten Ereignissen statt. D.h. auch wenn ein Service nicht gleich gestartet wird, so kann er noch später, sobald alle Bedingungen erfüllt sind, gestartet werden.

Schauen wir uns solch einen Eintrag an:

Von diesen Einträgen sind zwei Startbedingungen.

Die erste ist "*IOProviderClass*". Dort wird der Name eines Services angegeben. Wird solch ein Service gestartet, dann ist es das Startsignal, dass die Überprüfung losgeht.

In diesem Fall heißt der Service *IOPCIDevice*. Für jedes PCI Gerät wird solch ein Service gestartet. Damit ist schon mal klar, dass sich der Service um den es hier geht um ein PCI Gerät kümmert.

Welches Schweinderl hätten sie denn gerne ?

Aber welches ?

"*IOPCIClassMatch*" gibt an, welcher Geräteklasse, das Gerät angehören muss, damit der Service in Frage kommt. In diesem Falle 0x0c033000. In der PCIe Spezifikation sind die Klassen und ihre Werte beschrieben.

- 0C Serielles Gerät
- 03 USB
- 30 XHCI

In diesem Fall gibt es keine weiteren Bedingungen.

Der Service ist also zuständig für alle PCI Geräte mit der Klasse XHCI Controller.

Der Programmcode des Service steht im Bundle "*CFBundleIdentifier*" in diesem Falle "*com.apple.driver.usb.AppleUSBXHCIPCI*".

Ein kurzer Blick nach oben zeigt und, dass das das Kext selbst ist.

Der Programmcode kann mehr als einen Service enthalten. "*IOClass*" gibt an welcher verwendet werden soll. In diesem Fall "*AppleUSBXHCIPCI*".

"*IOPCIPauseCompatible*" und "*IOPCITunnelCompatible*" haben mit dem Teilen von PCI Kanälen bzw. dem Anschluss über Thunderbolt zu tun und interessieren uns hier nicht.

Eins geht noch

Ein zweites Beispiel

Die Überprüfung beginnt natürlich ebenfalls mit dem Finden eines PCI Gerätes "*IOProviderClass*".

Doch diesmal haben wir kein *IOPCIClassMatch*. Statt dessen haben wir ein *IOPCIPrimaryMatch*. PCI Geräte haben eine Primary und eine Secondary Id.

Sie bestehen jeweils aus einer Produkt- und einer Hersteller-Id. Beides sind 16bit Hex Werte, die Produkt Id kommt zuerst.

Also der Service kommt in Frage, wenn ein Gerät mit der Produkt-Id 0x9c31 und der Hersteller-Id 0x8086 (steht für Intel) gefunden wird.

Der ProgrammCode befindet sich in diesem Kext und heißt *AppleUSBXHCILPT*.

Man kann davon ausgehen dass auch dieses Gerät der PCI Klasse XHCI Controller angehört.

Es kann nur einen geben

Woher weiß das System nun welchen Treiber es nehmen soll, diesen oder den den wir uns vorher angeschaut haben ?

"*IOProbeScore*" hilft an der Stelle. Es wird der Treiber mit dem höheren *IOProbeScore* genommen.

Wird also ein Device mit der Primary-Id 0x9c318086 gefunden so hat der Service *AppleUSBXHCILPT* ein *IOProbeScore* von 1000, *AppleUSBXHCI* einen *IOProbeScore* von 0 (kein Eintrag bedeutet 0).

Also wird in dem Fall *AppleUSBXHCILPT* verwendet. Bei XHCI Geräten mit anderen Primary Ids kommt der Eintrag nicht in Frage, aber *AppleUSBXHCI* ist für diese immer noch ein Kandidat.

Soviel hierzu

Und schon haben wir die erste Gruppe von IOKitPersonalities analysiert

All diese Einträge stellen Treiber für verschiedene XHCI Controller zur Verfügung.

Es gibt Treiber für bestimmte Controller und einen generischen Treiber für die Controller, für die es keinen besonderen Eintrag gibt.

Alle Treiber basieren auf dem *AppleUSBXHCI* Treiber, sie sind Subklassen von *AppleUSBXHCI* dem generischen Treiber.

Wenn wir nun ein Board mit einem macos nicht bekannten XHCI Controller verwenden würden, würde macos den generischen Treiber verwenden, solange der Controller zur PCI-XHCI-Klasse gehört - wovon auszugehen ist.

Das ist ne tolle Sache, wenn der generische Treiber mit dem Controller funktioniert.

Nun kann es aber sein, dass es nicht der Fall ist. In diesem Fall könnte man einen neuen Eintrag für diesen Controller anlegen, als IOPrimaryMatch dessen Id eintragen und dann als IOClass alle von den anderen verwendeten Treiber durchprobieren bis man einen findet der funktioniert.

Später dazu mehr.

Einen hab ich noch

In den IOKitPersonalities dieses Kextes gibt es noch eine zweite Art von Einträgen:

Anhand der Namen kann man darauf schliessen, dass sie dazu dienen Anpassungen in Abhängigkeit vom vorliegenden Rechner durchzuführen.

In diesem Fall dienen sie dazu dem Treiber mitzuteilen, welche Ports des Controllers bei diesem Computer verwendet werden.

Wann geht's los ?

"IOProviderClass" zeigt uns "AppleUSBXHCIPCI". D.h. wird ein Service dieses Namens oder eine Subklasse davon geladen, dann startet die Überprüfung.

Wir müssen nicht mal raten, wann dies der Fall ist. Wir wissen, dass *AppleUSBXHCIPCI* oder eine seiner Subklassen gestartet wird, wenn ein Treiber für einen XHCI Controller gefunden wird.

Und sonst ?

"*IONameMatch*" ist der Name des PCI Gerätes. Damit der Test erfolgreich ist muss der Name "*XHCI*" sein.

Das genügt noch nicht das "*model*" muss vom Typ "*iMac17,1*" sein.

Also dieser Service wird gestartet, wenn ein XHCI Treiber geladen wurde, das PCI Gerät *XHCI* heißt und der Rechner ein *iMac17,1* ist.

Der Service der gestartet wird befindet sich nicht in diesem Kext, denn der "*CFBundleIdentifier*" ist "*com.apple.driver.AppleUSBMergeNub*". Wo das Kext tatsächlich steht ist egal, denn alle Kexte wurden ja schon geladen und mit ihrem Bundle-identifier gespeichert.

Der Service heißt laut "*IOClass*" Eintrag "*AppleUSBMergeNub*".

Merge bedeutet zusammenführen. Da werden wohl Eigenschaften zusammengeführt. D.h. dem USB Treiber (XHCI Controller sind USB Controller) werden Parameter übergeben.

Der Name "*IOProviderMergeProperties*" ist ein deutlicher Hinweis, dass es sich hier um die Parameter handelt.

Wie man sieht, sieht man bei Data nichts. Das liegt daran, dass der PList Editor versucht die Daten als Text anzuzeigen, es aber gar kein Text ist.

Ein Rechtsklick auf das Fenster und Show Raw Keys/Values angewählt ändert das

Das kenn ich doch

Wer sich mit USB und DSDT/SSDT beschäftigt hat, dem kommt das sehr vertraut vor.

port-count ist die höchste verwendete Port Id.

Die Ports werden meist einfach durchnummeriert die HS Ports beginnen bei 1, die SS Ports bei 17. 17 macht Sinn, wenn man Hexadezimal denkt, denn dann ist es 0x11.

Wenn 0x1a, siehe *port-count*, die höchste verwendete Port Id ist, müssten wir einen SSP10 Eintrag haben. Haben wir aber nicht. Vermutlich hat der Programmierer einfach den höchsten bei diesem Controller möglichen Wert genommen - so ein Faultier.

HS02 ist der Name unter dem das Port verwaltet wird. Typischerweise HS für High Speed und SS oder SSP für Super Speed Port.

Unter "*USB Connector*" wird die Anschlussart angegeben. Damit ist der Stecker bzw. die Buchse gemeint mit dem/der der Port verbunden ist.

- 0 ist die normale USB2.0 Typ A Buchse - die breite Fläche in Schwarz. Nur HS.
- 3 ist die normale USB3.0 Typ A Buchse - die breite Fläche in Blau. SS und HS.
- 9 ist USB-C mit HS und SS.
- 255 ist ein unbekannter oder eigener Stecker. MoBo Header fallen in diese Rubrik.

HS02 liegt also an einer USB3.0 Buchse und die Port Id ist 0x02. Das ist der zweite SS Port am Controller - deshalb auch HS02.

Das Kext für EHCI is genauso organisiert. Dort gibt es keinen Eintrag für iMac17,1, denn der iMac17,1 hat nur einen XHCI Controller und keinen EHCI Controller.

Wird ein PCI Gerät erkannt, checkt das Kext ob es sich um einen XHCI Controller handelt und wenn dem so ist, lädt es einen Treiber.

Wenn es danach einen passenden Eintrag für Rechner und Gerätenamen findet, wählt es die dazu passenden Ports aus und teilt sie dem Treiber mit, findet es keinen Eintrag, verwendet der Treiber die ersten 15 Ports des Controllers.

Das Problem bei einem Hack ist, dass die Ports meist nicht die sind, die man gerne hätte. Sollte man mit einem Board ein SMBIOS verwenden, das eine Controllerart die das Board hat nicht verwendet, werden die ersten 15 Ports davon eingetragen, was u.U. nicht hilfreich ist.

Und was hilft mir das ?

Das sehen wir morgen im nächsten Beitrag.

Beitrag von „KayKun“ vom 16. August 2017, 15:34

Respekt für diesen Beitrag! einfach nur top!!!

Beitrag von „Dr.Stein“ vom 16. August 2017, 15:40

Das ist viel zu viel Wissen auf ein mal für mein Hirn.

Klasse Job! 😊

Beitrag von „derHackfan“ vom 16. August 2017, 15:50

Erst mal Danke für diesen Beitrag. 👍

Ich persönlich habe es zwar (alles) gelesen, bin aber ziemlich schnell ausgestiegen, entweder liegt es an der Tageszeit oder an meinen geistigen Fähigkeiten.

Beitrag von „al6042“ vom 16. August 2017, 16:04

Es ist immer wieder eine Freude deine Beiträge zu lesen...

Zum einen super informativ und fundiert, zum anderen ein Intellektueller Leckerbissen, bei dem man mit Lesen gar nicht aufhören mag... 😊

vielen Dank für den tollen Beitrag... 👍

Beitrag von „McRudolfo“ vom 16. August 2017, 16:44

Vielen Dank für diesen - mal wieder großartigen - Post. Mir geht es ein bisschen wie dem Hackfan, aber ich hoffe, nach dem dritten oder vierten Lesen kommt etwas Licht ins Dunkel...



Beitrag von „Brumbaer“ vom 17. August 2017, 14:39

Ran ans Eingemachte.

Dann wenden wir das Gelernte mal an.

Ich habe hier ein Asus Strix Z270i Gaming Board.
Es wird als iMac18,3 betrieben, wegen Kaby Lake und so.

Ziel ist: Keine Änderungen an der DSDT, kein USB InjectAll.

USB 2.0 Ports funktionieren, deshalb kann man mit einem USB 2.0 Stick (oder einem USB 3.0 Stick an einem USB2.0 Anschluss oder hinter einem USB 2.0 Hub) macos installieren.

Wenn ich nur wüsst was drinnen ist.

Wenn man etwas über Geräte und Services wissen will, hilft *IORegistryExplorer* weiter.

Wenn man *IORegistryExplorer* startet, sieht man die *IOService (Plane)*. Das ist eine Auflistung der installierten Services.

Da Services häufig an Geräte gebunden sind, sieht es auf den ersten Blick wie eine Auflistung der Geräte aus.

Man kann sich auch die *IOACPIPlane* anzeigen lassen. Die zeigt uns dann die Geräte an, wie sie in der DSDT/SSDT verzeichnet sind.

XHCI Controller werden in der DSDT für gewöhnlich *XHC* nix oder irgendwas benannt. Gott sei Dank, sonst ging die Sucherei jedes Mal aufs Neue los.

Die Services verwenden ebenfalls den Namen aus der DSDT/SSDT und somit heißen die auch irgendwas mit XHC.

Auf der rechten Seite sehen wir, dass die DSDT ein Gerät namens *XHC* hat, das hat einen *RHUB* und dann 26 Ports.

Auf der linken Seite sehen wir keinen *RHUB* und nur 15 Ports.

Von der Problematik mit den 15 Ports hat jeder schon gehört. Es werden, wenn wie nicht im ersten Artikel beschrieben, die Ports explizit ausgewählt werden, die ersten 15 mit Services versehen.

Ein Blick auf die rechte Seite zeigt uns das sind HS01-HS14 und SS01 und tatsächlich genau diese Ports tauchen auf der linken Seite auf.

Der *RHUB* taucht auf der linken Seite nicht auf. USB Controller haben interne HUBs an denen die Ports angeschlossen sind.

Das ist sinngemäß ein eigenes Gerät bzw. eine eigene Funktionalität, aber Treiber-/Service-seitig ist dessen Behandlung Teil des XHC Treibers. Deshalb sieht man ihn auf der ACPI aber nicht auf der Service Seite.

Auf der rechten Seite sieht man hinter jedem Ports die Portnummer.

Auf der rechten Seite sieht man die erwarteten Ports plus *USR1* und *USR2*.

Was sind *USR1* und *USR2*.

Der Controller nummeriert die Ports durch.

Port 0x00 ist der Hub !!!!!!!!!!!

Port 0x01 bis Port 0x0E sind HS01 bis HS14

Port 0x11 bis Port 0x1A sind SS01 bis SS10.

Dem Ordnungsfanatiker fällt auf dass da eine Lücke ist, 0x0F und 0x10 sind nicht belegt.

0x0F ist nicht belegt, weil es nur 14 HS Ports gibt und 0x10 ist nicht belegt, da es der HUB für die SS Ports wäre.

Damit die Portnummern aber nicht frei sind und man beim "Durchzählen" nicht auf "Nichts" trifft, werden sie mit zwei Dummy-Ports belegt.

Was soll mir das sagen ?

Was wir anhand dieser Daten sagen können ist:

Der Gerätenamen des XHCI Controllers ist *XHC*.

Die *Portnummern* sind 0x01 bis 0x0E für HS Ports und 0x11 bis 0x1A für SS Ports.

Es wird ein Treiber für den Controller geladen, sonst würden die Ports nicht in der Service Plane auftauchen.

Es gibt keinen Eintrag in *AppleUSBXHCIPCI* für diese Mactyp/Gerätenamen Version, sonst wären nicht die ersten 15 Ports in der Service Liste, sondern die die der Original Mac hat.

HS02,5,7,9 und 11 haben Dreiecke, das heißt die Ports sind in Benutzung. Diese Ports werden auf jeden Fall von meinem Computer benutzt.

"Entfaltet" man das Port, sieht man welcher Service/Gerät an dem Port aktiv ist. Dann weiss man auch schon welche Buchse am Computer mit diesem Port verbunden ist.

Wir können sehen, dass Port HS11 mit dem internen BT Adapter verbunden ist.

Sonst noch was ?

Auf der linken Seite sehen wir zwei Einträge für XHC, also zwei Services. Warum ?

Bei unseren gestrigen Betrachtungen haben wir gesehen, dass der XHCI Treiber in Abhängigkeit von einem PCI Device Treiber gestartet wird.

Da der PCI Device Treiber zuerst "da war" steht es oberhalb des XHCI Treibers. Beide heißen XHC, weil das Gerät laut DSDT/SSDT XHC heißt.

Auf der rechten Seite sehen wir, dass der Treiber *AppleUSBXHCIPCI* ist und dass er aufgrund von *IOPCIClassMatch* geladen wurde.

Erinnern wir uns, das ist der Notfall-Treiber, wenn es keinen Controller-eigenen Treiber gibt.

Auf der linken Seite finden wir den *class-code*, *vendor-id* und *device-id* (Produkt-Id). Das sind die Rohdaten aus den Controllerregistern, deshalb sind die Werte etwas verdreht.

Unter *name* finden wir eine lesbarere Form. 0x8086 steht für Intel und 0xa2af ist die *device-id* des XHCI Controllers des Z270 Chipsatzes.

Geradlinig ist anders

Voll interessant - na ja vielleicht nicht - aber wir werden es später brauchen.

AHDS ?

Treiber wird geladen, Ports sind da, nur nicht genug, meine Aufmerksamkeitsspanne ist am Ende, l. mich.

Na ja wenn das so ist, den USB-Port-Anzahlveränderungs-Patch rein und weiter bei "Was will man mehr?".

Nö

Tja, die Aufgabe ohne Patch ist bei obiger Lösung nicht erfüllt. Außerdem ist mir der Patch suspekt. Also die Ports auf 15 begrenzen.

Port sucht Buchse

Dazu müssen wir herausfinden welche Ports auf Buchsen oder Header geführt sind.

Ein paar Ports haben wir schon gefunden, nämlich die die in der Service Plane mit einem Dreieck markiert waren.

Jetzt stecken wir nacheinander in alle anderen Ports ein USB 2.0 Gerät oder Hub und schauen bei welchem Port dann ein Dreieck erscheint.

Die Anschlüsse sind

- HS01 intern USB 3.0
- HS02 intern USB 3.0
- HS03 hinten außen rechts USB 3.0
- HS04 hinten außen rechts USB 3.0
- HS05 hinten außen links USB 3.0
- HS07 hinten Mitte USB 2.0 only
- HS08 hinten Mitte USB 2.0 only
- HS09 hinten Mitte USB 2.0 only
- HS10 hinten Mitte USB 2.0 only
- HS11 intern BCM Bluetooth Module

Das sind schon mal 10 HS Ports. Davon sind 5 Ports an USB 3.0 Buchsen geführt. Diese haben jedes zusätzlich noch ein SS Port. Macht zusammen 15. Wenn das kein Wink des Schicksals ist.

Die Frage ist wie bekommen wir raus welche SS Ports an den Buchsen/Headern liegen. Na ja wir tragen die ersten 5 ein und schauen welche Verbindung haben. Werfen die anderen raus probieren wieder mit dem nächsten Satz Ports.

Oder aber wir schauen in die Dokumentation und graben in der Erfahrungskiste.

Die Erfahrung sagt uns, das Asus gerne SS und HS Ports mit der selben Nummer an die selbe Buchse legt.

D.h. das wären dann SS01 bis SS05, weil HS01 bis HS05 an den USB 3.0 Buchsen/Header anliegen.

Asus benennt in seinen Handbüchern die Stecker/Buchsen nach den Port Nummern. USB3_12 Ist ein USB3.0 Header an dem SS01 und SS02 anliegen.

Ein kurzer Blick ins Handbuch bestätigt unsere Vermutung bezüglich SS01 bis SS05

Die komplette Port-/Buchsenliste sieht also so aus:

- SS01, HS01 intern USB 3.0
- SS02, HS02 intern USB 3.0
- SS03, HS03 hinten außen rechts USB 3.0
- SS04, HS04 hinten außen rechts USB 3.0
- SS05, HS05 hinten außen links USB 3.0
- HS07 hinten mitte USB 2.0 only
- HS08 hinten mitte USB 2.0 only
- HS09 hinten mitte USB 2.0 only
- HS10 hinten mitte USB 2.0 only
- HS11 intern BCM Bluetooth Module

Ein Blick zurück ohne Zorn

In der *AppleUSBXHCIPCI* wurden dem Treiber die Ports mit Hilfe von *AppleUSBMergeNub* und einer Parameterliste zugewiesen.

Es wird also ein Service aus einem Kext heraus gestartet. Der Service muss nicht im selben Kext stehen, denn *AppleUSBMergeNub* ist ja nicht Teil des *AppleUSBXHCIPCI*.

Also können wir ein Kext erzeugen, das *AppleUSBMergeNub* startet und dem Treiber unsere Portliste übergibt. Wir müssen nichts patchen oder verändern, wir fügen einfach nur ein neues Kext hinzu.

Es ist ein Kext

Wir legen einen Ordner an, der unser Kext sein wird.

Der Ordner bekommt den schönen Namen BBStrixUSB.kext oder was auch immer ihr sonst bevorzugt. Hauptsache der Suffix ist kext.

Das Icon im Finder wird mit dem Umbenennen zum Kext Icon wechseln.

Ein Rechtsklick auf das Icon und "**Paketinhalt zeigen**" öffnet den Ordner.

Wir werden keinen Contents Ordner verwenden und so legen wir auch keinen an.

Nun erzeugen wir die *Info.plist* Datei.

XCode hat unter **New -> File** eine Vorlage für Property Lists. Diese erstellt eine leere PList.

Es ist wichtig die Datei *Info.plist* zu nennen sonst funktioniert das Kext nicht. Außerdem schaltet der Name *Info.plist* im PList Editor PopUps für die Feldnamen frei. Wählt man den Feldnamen, wird auch der Datentyp gleich passend gewählt - praktisch.

Man kann nun die benötigten Felder hinzufügen. Man kann natürlich auch eine vorhandene *Info.plist* kopieren und rauswerfen, was man nicht braucht und editieren was nötig ist.

Wir fangen mit den allgemeinen Kext Feldern an.

Weniger geht nicht, oder ?

Das sieht im PList Editor so aus

Ich will nicht ausschließen, dass Clover auf ein paar dieser Felder verzichten kann, wer will kann es ja mal ausprobieren.

Der *Bundle Identifier* muss auf jeden Fall vorhanden sein. Diesen Namen darf sonst kein Kext tragen. Es ist üblich mit der eigenen umgedrehten Webdomain anzufangen und mit dem Projektnamen zu enden. Keine Leerschritte.

Bundle name, und die zwei *Bundle versions*, kann man ebenfalls frei vergeben. *Bundle version* muss allerdings das Format Zahl.Zahl.Zahl haben.

Den Rest übernimmt man wie er da steht.

Jetzt wird's persönlich

Wie wir wissen machen die *IOKitPersonalities* die Arbeit.

Also legen wir sie an. Wir nennen unsere Personality *Strix 270i Gaming*. Apple hätte sie wohl iMac18,3-XHC genannt.

Die Überprüfung soll starten, wenn ein XHCI Treiber geladen wird (*IOProviderClass*)

Der Rechner vom Typ iMac18,3 (*model*) ist.

Und unser Gerät *XHC* heisst (*IONameMatch*). In *AppleUSBXHCIPCI* steht an dieser Stelle *XHCI*, aber wir haben im IORegistryExplorer gesehen, dass unser Device *XHC* heisst. Statt das Device in der DSDT oder Clover umzubenennen, ändern wir hier den *IONameMatch*, da wir schon mal hier sind und dann auch nur an einer Stelle Änderungen vornehmen.

Unser Test hat einen *IOProbeScore* von 5000, um auch in Zukunft wichtiger als andere Einträge mit den selben Bedingungen zu sein.

Der Treiber der geladen werden soll ist, wie wir im Original gesehen haben, *AppleUSBMergeNub* im Kext mit dem *Bundle Identifier* *com.apple.driver.AppleUSBMergeNub*.

Jetzt müssen wir nur noch die Ports eintragen. Die gehören unter *IOProviderMergeProperties*.

Der erste Eintrag bei den *IOProviderMergeProperties* ist der *page-count*, der die höchste Port-Id enthält.

Danach kommt die Liste der Ports.

Die Daten des *port-count* Eintrages tragen wir später ein, wenn wir die höchste Portnummer kennen.

Um Platz zu sparen nur der relevante Ausschnitt

Wie ein Eintrag für ein Port aussieht haben wir ja gestern beim Betrachten von *AppleUSBHCIPCI* gesehen.

Für unsere erstes Port HS01 sieht das dann wie folgt aus

Unser erstes Port ist auch das erste Controller Port und ein HS Port deshalb heißt es HS01.

Es ist auf einen Header geführt, deshalb ist der *USB Connector Typ* 255

Die Port Id ist 0x01

Der am selben Stecker liegende SS Port ist der erste SS Port des Controllers und heißt deshalb SSP1 - weil es im iMac17,1 Eintrag auch so hieß. Wir könnten es auch SS01 oder XA4R nennen. Das erhöht den Wiedererkennungswert für Eingeweihte, verwirrt aber alle anderen, deshalb verwenden wir einen "Standardnamen".

Der USB Connector ist der selbe und die Port Id 0x11.

Das macht man für alle Ports.

Das Port mit der höchsten Id ist SSP5

Das liegt an einer USB 3.0 Typ A Buchse, deshalb ist der USB Connector 3. Die Port Id ist 0x15 das ist deshalb erwähnenswert, weil wir die höchste Port Id im port-count Eintrag brauchen.

Die ganze Datei sieht dann so aus, beachte den bei port-count eingetragenen Wert.

Feddich, wie en Reddich.

Yeah, lasst es Konfetti regnen.

In den EFI/CLOVER/kexts/Other Ordner der EFI Partition kopiert, neugestartet und IORegistryEditor zeigt uns

Was will man mehr ?

Z.B. das es funktioniert.

Wir stellen nämlich schnell fest, dass die USB 3.0 Einträge nicht funktionieren.

Wie wir aus dem IORegistryEditor wissen, wurde der "Notfall Treiber" für XHCI Geräte geladen. der scheint dann wohl nicht zu passen, zumindest nicht für USB 3.0.

Die Qual der Wahl

AppleUSBXHCIPCI hat eine Reihe von auf verschiedene Controller zugeschnittene Treiber. Welchen nehmen ?

Ausprobieren oder einen "educated guess" - gibt's da einen deutschen Ausdruck für ? - wagen.

Wie wir aus dem IORegistryEditor wissen, hat user USB Controller die Primary-Id 0xa2af8086.

Bei Durchsicht der Treiber Einträge in *AppleUSBXHCIPCI* finden wir einen für 0xa12f8086 - das Vorgängermodell.

Also probieren wir es doch mit dem.

Um den Treiber starten zu können brauchen wir wieder ein Kext. Na ja wir haben schon eins, also verwenden wir das.

Die Struktur für einen Treibereintrag haben wir gestern gesehen.

Also legen wir eine neue Personality an und nennen sie *AppleUSBXHCISPTB Z200* oder irgendwie anders.

Unser Treiber soll geladen werden, wenn ein PCI Device entdeckt wird und es die Id 0xa2af8086 hat.

Den *IOProbeScore* setzen wir auf 5000, damit macht unser Eintrag einen auf wichtig.

Wir schreiben ja keinen eigenen XHCI Treiber, sondern nehmen einen aus dem *AppleUSBXHCIPCI* Kext. Das ist dem Kernel als *com.apple.driver.usb.AppleUSBXHCIPCI* (*CFBundleIdentifier*) bekannt.

Der Treiber den 0xa12f8086 (Vorgänger) verwendet ist (*IOClass*) *AppleUSBXHCISPT*. Also nehmen wir den auch.

Die letzten beiden Einträge übernehmen wir, da wir davon ausgehen, dass unser Controller alles kann, was der alte konnte.

Und war's das ?

In den Other Ordner, Neustart und
Alles geht, wie es soll.

Das war's

P.S.

Im Anhang findet ihr das Kext.

Beitrag von „Doctor Plagiat“ vom 17. August 2017, 15:33

[@Brumbaer](#)

Vielen Dank dafür, dass du dein hart erarbeitetes Wissen hier mit uns teilst. Einfach grandios

deine Beiträge. 👍 

Beitrag von „andreas_55“ vom 17. August 2017, 21:45

Na das hat jetzt mal Spaß gemacht. 👍

Gerade den ersten Teil gelesen. Vielen Dank für die tolle Erläuterung. Das ist ja hier der Goldstandard.

Freue mich schon auf Morgen Abend: 2. Teil.

Ein kleine Anmerkung: Das zweite Bild der Info.plist sollte ein anderes sein, da Du von "IOPCIClassMatch" redest, diese dort aber nicht auftaucht, sondern nur die "IOPCIPrimaryMatch". Es ist also bereits das dritte Bild drin.

Beitrag von „Brumbaer“ vom 17. August 2017, 22:06

[andreas_55](#)

Vielen Dank fürs aufmerksame Lesen. Habe das Bild getauscht .

P.S.

Gestern war morgen heute. Der morgige Beitrag von gestern aus gesehen ist ein paar Posts weiter oben.

Beitrag von „andreas_55“ vom 17. August 2017, 22:16

Ich weiss.

Freue mich ja auch auf mein Morgen. Dein Morgen war ja schon heute. 😊

Beitrag von „StevePeter“ vom 17. August 2017, 22:19

Vielen Dank für die tolle ausführliche Erläuterung. Das ist mal ein richtig guter Batzen Wissen.

Man braucht aber schon ein paar Anläufe bis man durch ist.
Bringt aber wichtiges wissen mit.

Thanks 🙏
StevePeter

Beitrag von „Thogg Niatiz“ vom 17. August 2017, 23:57

Auch von mir vielen Dank für die Mühen und die ausführlichen Erläuterungen. Das ist ein wichtiges Thema und sicher für viele hier sehr aufschlussreich.

Eine Anmerkung, da mich das beim ersten Lesen selbst verunsichert hatte: Der port-count Eintrag entspricht, wie beschrieben, der höchsten verzeichneten Portnummer. Damit lässt sich aber *nicht* das nebenbei angesprochene Port Limit von 15 Ports umgehen, da dieses in den Treibern im AppleUSBXHCIPCI Binary fest definiert ist, also *nicht* auf port-count reagiert. Mit dieser Dummy Kext können beliebig viele Ports definiert werden, auch mehr als 15, wenn vorhanden, jedoch lädt der Treiber nur die ersten maximal 15 davon. Wem das nicht ausreicht, der kann:

- weiterhin auf den Port Limit Raiser Patch (Clover "KextsToPatch" / Perl / etc - von Entwicklern immer wieder als riskant eingestuft) setzen, um mehr als 15 der definierten Ports ansprechen zu können, oder
- nicht explizit benötigte Ports (freie interne Header / freie Mainboard Buchsen / bei USB 3.0 Typ A Ports wahlweise auch entweder den 2.0 (HS) oder 3.0 (SS) Teil, wenn man nur den jeweils anderen braucht) *nicht* in die Info.plist eintragen

Ich persönlich konnte einen halben USB 2.0 Header und den 2.0 Part zweier rückseitiger Buchsen opfern. Sicher - das lässt sich mit USBInjectAll auch leicht machen. Diese Dummy Kext Methode funktioniert aber seit El Capitan einwandfrei, während USBInjectAll schon bei einigen Versionen des Betriebssystems angepasst werden musste, was für manche Early

Adopters sicher nicht unproblematisch war. Also der Aufwand lohnt sich meiner Meinung nach in jeden Fall, wenn nicht nativ alle USB Ports erkannt werden, weil man damit auch in den kommenden Jahren Ruhe haben dürfte (vorausgesetzt Apple findet kein zweites technisches Fauxpas, das uns einen weiteren komplett überarbeiteten USB Stack beschert) 😊

Beitrag von „andreas_55“ vom 18. August 2017, 22:36

Wow!! Also das war ja nochmal besser! Super lehrreich!

Vor ´ner Weile habe ich bei RehabMan gelesen, "... man könnte einen eigenen Kext bauen, muss nur darauf achten, das der IOProbeScore hoch genug ist ..." und ich dachte nur "Wovon redet der Kerl eigentlich?"

Jetzt weiss ich nicht nur was er meinte sondern freue mich darauf, bald meinen ersten Kext zu bauen!

Daran hätte ich vorgestern im Traum nicht gedacht.

Ihr seid hier schon ein toller Haufen.



Beitrag von „derHackfan“ vom 18. August 2017, 22:47

[@Brumbaer](#) Das muss ich dann wohl 33 mal lesen um es teilweise zu verstehen, aber trotzdem vielen Dank für deinen Einsatz. 👍

Beitrag von „Brumbaer“ vom 18. August 2017, 23:09

Dann habe ich wohl nicht die richtigen Worte gefunden.

Beitrag von „derHackfan“ vom 18. August 2017, 23:23

Aber sicher doch, es liegt entweder an meinen begrenzten Möglichkeiten oder der Uhrzeit/Jahreszeit, auf jeden Fall weiss ich nicht was ich wo wie machen soll. 😄

Mal Spaß beiseite, ich habe einfach neben dem Job keinen Kopf um mich in solche Brocken einzulesen und zu verstehen und damit zu arbeiten, zumal dass ja ein ganz anderes Berufsfeld ist.

Kurz und knapp, die Festplatte ist voll und auslagern in die Cloud geht nicht, meine Frau will nix mehr vom Forum und Hackintosh hören. 🙄

Btw: Wie machen die anderen das, lesen, verstehen, Aha Effekt und anschließend umsetzen und freuen? 😞

Beitrag von „Brumbaer“ vom 19. August 2017, 00:20

Vielleicht äußert sich ja noch jemand.

Aber ich sehe schon die Gefahr, dass bei unterschiedlichem Erfahrungshorizont, Dinge nicht unbedingt so aufgenommen werden, wie der Autor es erwartet.

Vielleicht liegt es aber auch einfach daran, dass im Text ein Fehler ist.

[@andreas_55](#) hat schon ein paar Dreckfuhler gefunden, die ein Stirnrunzeln erzeugen können, und mit netterweise mitgeteilt. Vielen, vielen Dank dafür.

Denn egal wie oft ich es lese irgendwann ist Schicht mit dem Finden von Fehlern in eigenen Texten. Das geht dann erst wieder mit Vierzehntagen Abstand.

Ggf. Fragen, vielleicht kann ich es mit anderen Worten erklären.

Beitrag von „andreas_55“ vom 19. August 2017, 07:00

[@derHackfan](#)

Das Du weniger Zeit hast liegt selbstverständlich auch daran, dass Du unermüdlich im Forum Anderen hilfst.

The difference between amateurs and professionals:

"Amateurs think knowledge is power. Professionals pass on wisdom and advice."

"Amateurs show up inconsistently. Professionals show up every day."

Du bist ein Profi. 👍

[@Brumbaer](#)

Unter der Überschrift Sonst noch was ? ist das Bild "IOService" einmal XHC@14 und XHC@14000000:

Dort findet sich links class-code <30 03 0c 00> und rechts IOPCIClassMatch 0x0c033000

Ich bin über was gestolpert:

Das ist nicht die Umwandlung von Little in Big-Endian, denn das ist 30 03 0c 00 -> 00 0c 03 30. Also hätte ich eigentlich IOPCIClassMatch 0x000c0330 erwartet.

Oder ist das in dem Fall nicht der Zusammenhang?

Beitrag von „Doctor Plagiat“ vom 19. August 2017, 09:58

[Zitat von Brumbaer](#)

Vielleicht äußert sich ja noch jemand.

Ich habe die beiden Beiträge ganz in Ruhe durchgelesen und glaube soweit alles verstanden zu haben. Es ist schon beachtlich was da so ineinander greift oder auch nicht, wenn bestimmte Bedingungen nicht gegeben sind.

Zum Schluss war ich dann die "Faulheit in Person" und habe deinen hochgeladenen Kext einfach an mein System angepasst. Ich bin gerade fertig geworden und werde es gleich testen. Melde mich gleich wieder.

EDIT: Meine Faulheit wurde postwendend bestraft. Es wurden 15 HS-Ports und ein SS-Port erkannt. Geladen lt. Systembericht wurde der Treiber für den Host-Controller AppleUSBXHCISPT mit der PCI-Geräte-ID 0xa12f.

Ich habe in meinem Kext auch Bundle Identifier, Bundle name und in IOKitPersonalities das Dictionary angepasst. Vielleicht liegt da irgendwo der Fehler.

Beitrag von „Brumbaer“ vom 19. August 2017, 14:06

[@andreas 55](#)

Jein.

Deine Umwandlung ist korrekt. Die PCI Register sind als 32 bit Worte organisiert.

Das Wort, das die Klasse enthält, wird bei diesem Chip als 0x000C0330 gelesen. Allerdings bezeichnen nur die unter 24 Bit also 0C0330 die Klasse. Die obersten 8 Bit, die 00, stammt aus von einem anderen Feld, einer Revision ID und werden ausgeundet, haben also nichts mit der 00 am Endes des IPClassMatches zu tun.

Der IOClassCode ist auch ein 32 Bit Wert. Organisatorisch ist die 0C die höchste Hierarchiestufe, da geht nix drüber. Sie gehört also im Verständnis des "westlichen" Raums ganz nach links. Da wir aber nur 24 Bit haben, bleiben am rechten Rand 8 Bit übrig und die füllt man wie sooft wenn man nicht weiß was man tun soll oder was da kommen wird mit Nullen. Nebenbei gibt es einem noch die Möglichkeit ggf. eine weitere Hierarchiestufe einzubauen.

$\text{IOClassMatch} = (\text{Register} \& 0x00FFFFFF) \ll 8.$

[@Doctor Plagiat](#)

Du hast ein Z170 Board und dessen Controller ist der mit der Id 0xa12f8086. Den verwendet Apple auch und deshalb ist er in der AppleUSBXHCIPCI mit einem eigenen Treiber nämlich AppleUSBXHCISPT vertreten. das ist also ok und sollte funktionieren.

Wenn die 14+1 Ports angezeigt werden greift die erste (in meinem Kext) Personality nicht.

Lade doch mal deine Info.plist hoch.

Zusätzlich brauchen wir noch, den SMBIOS Namen, den dein Rechner verwendet und wie das XHCI Gerät heisst. Das ist der Name des Devices über den Ports in IORegistryExplorer. Bei

meinem Beispiel XHC.

Beitrag von „Doctor Plagiat“ vom 19. August 2017, 14:30

Ich habe den Kext jetzt noch mehr verschlimmbessert, es will aber nicht funktionieren. SMBIOS ist iMac 17,1. Das hatte ich aber schon geändert. Der Name des Devices ist XHCI. Vielleicht habe ich auf bei port-Data was falsch gemacht. Bei den Ports HS13 und 14 habe ich dort 0d und 0e eingetragen, da 13 und 14 für SS03 und SS04 vergeben sind.

[Info.plist](#)

Beitrag von „Brumbaer“ vom 19. August 2017, 14:55

Bitte verwende nicht de.brumbaer, ich möchte keine Fragen beantworten müssen, über Dinge die ich nicht geschrieben habe 😊

Das ist nur ein Name. Es muss keine Webseite unter dem Namen existieren. Du kannst genau so gut doctor.plagiat.USBGAZ170M verwenden.

Du hast in der IOKitPersonality GA-Z170M-D3H unter *IONameMatch* den Wert *XHC* eingetragen. Dein Gerät heißt aber Hier (und in der Info.plist) passendes eintragen. Dann sollte es funktionieren.

Die zweite Personality für den Treiber, kannst du löschen, da der Treiber für diesen Controller in *AppleUSBXHCIPCI* enthalten ist.

Beitrag von „Doctor Plagiat“ vom 19. August 2017, 16:49

Erstmal vielen Dank an dieser Stelle.

Sorry, den String "Bundle identifier" habe ich geändert. Da musst du dir also keine Sorgen machen über irgendwelche Anfragen. Fragen kommen also nur noch von mir. 😊

Den Wert XHC in IONameMatch habe ich nicht eingetragen, der stand so drin. Weil das Devices in der DSDT bei mir auch XHC heißt, dachte ich das ist so ok.

Wie du siehst guckt da ein Laie, der sich das erste Mal damit auseinander setzt , wie ein Schwein in 's Uhrwerk.

Was muss da jetzt rein ? XHCI ?

EDIT: Ich habe auch die info.plist mit einem geänderten Bundle identifier ausgetauscht.

Beitrag von „Brumbaer“ vom 19. August 2017, 17:24

Du hast geschrieben, dass der Name in IORegistryEditor XHCI ist.

Wenn er in der DSDT XHC ist, dann wird er irgendwo umgepatched. Entweder in einer SSDT oder in Clover oder sonstwo.

Finde den Patch und nimm ihn raus. Es darf auch kein anderes USB Kext mehr aktiv sein, USBInjectAll oder X99-USB irgendwas.

Der Name in IORegistryEditor ist der unter dem der Service gestartet wurde und wenn nix gepatched wird ist es der, der in der DSDT steht.

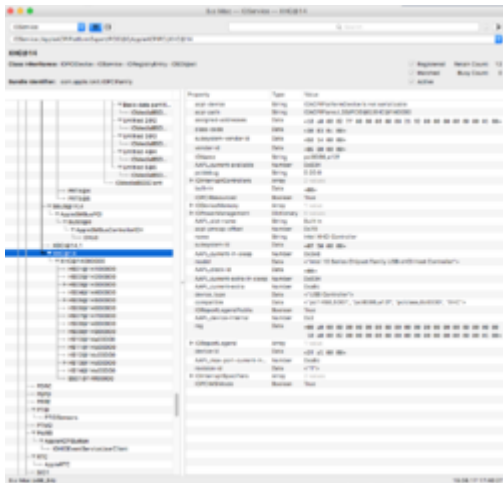
Alternativ verwende den den du im IORE siehst. Ist nicht "perfekt", aber sollte funktionieren.

Beitrag von „Doctor Plagiat“ vom 19. August 2017, 21:33

Ups, das mit XHCI war ein Schreibfehler, hatte ich noch gar nicht gemerkt. Falls ich das nicht hinkriege, habe ich meine USB-SSDT.aml, die funktioniert perfekt. Ich wollte nur deine sehr gute Anleitung testen und stelle mich wahrscheinlich einfach nur doof an.

Hier noch ein IOReg Screen oder doch gleich das ganze ioreg-File. Aber wenn du da keine Lust

drauf hast, ist das auch ok.



[iMac17,1_docplag.ioreg.zip](#)

EDIT: Ich bin nochmal ganz genau der Anleitung gefolgt. Ich kann keine Fehler feststellen, aber möchte auch keine ausschließen. Es ist nicht ein einziger USB-Kext im Other-Ordner und keine USB-KextToPatch-Einträge vorhanden. Es funktioniert leider nicht.

Falls noch andere User diese Methode ausprobiert haben, immer her mit dem Ergebnis, ob positiv oder negativ.

Beitrag von „Brumbaer“ vom 19. August 2017, 22:09

Ok,
ich dachte vielleicht liegt es an Sierra, also habe ich von einem Sierra Stick mit iMac17,1 gebootet - ging.

Im Anhang das Kext, das ich verwendet habe.

Leg es doch bitte in den Other Ordner deiner EFI Partition und teste.

Beitrag von „cobanramo“ vom 20. August 2017, 10:30

Zunächst mal vielen dank an [@Brumbaer](#),s tolle Anleitung, klasse Arbeit.

Bei mir funktioniert auch nicht so wie ich es gerne hätte, hab auch alle USB relevantes raus geschmissen, den erstellten kext rein.

Kann mal einer mein info.plist kontrollieren ob ich eventuell ein Fehler drinnen hab?

Grus Coban

Edit: ~~Noch was bitte, wie zum Geier bringe ich den PortLimit Patch zum laufen Ohne den USBInjectAll.kext?~~

Edit 2:

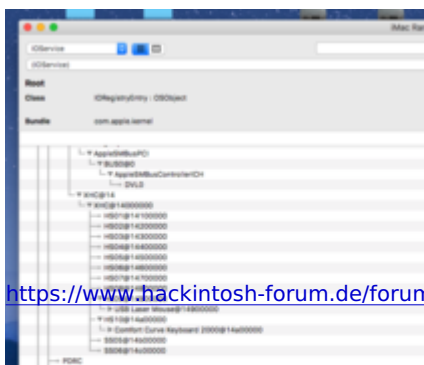
Also ich weiss zwar immer noch nicht wo ich den Fehler gemacht habe, jetzt nahm ich den Kext von vorpost vom Brumbaer und erweiterte ihn um Port HS06, passte das ganze zu meiner Board USBConnector Nummern an.

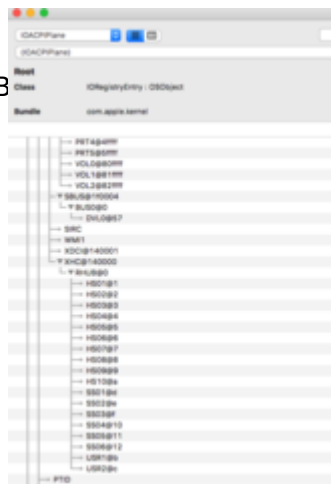
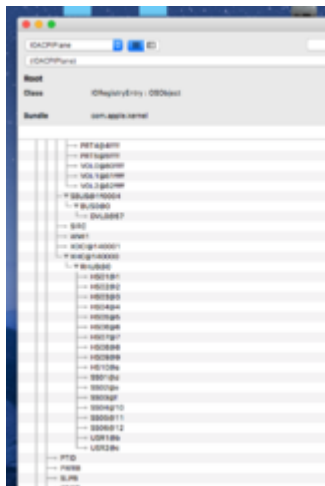
Siehe da, es funzt! 👍

Die Sache um den Port HS11 (Bluetooth Anbindung) verstehe ich nicht so ganz, den haben wir doch nirgends definiert taucht aber auf.

[Info.plist A](#) (diesen hab ich selber nach Anleitung gebastelt, funktioniert nicht)

[Info.plist B](#) (Funktioniert)





Beitrag von „Dr.Stein“ vom 20. August 2017, 12:04

[@Brumbaer](#)

Ich hab ja auch ein Asus Board und die letzte Kext läuft mit High Sierra. Sierra geht auch.

Beitrag von „Doctor Plagiat“ vom 20. August 2017, 17:08

[Zitat von Brumbaer](#)

Im Anhang das Kext, das ich verwendet habe.
Leg es doch bitte in den Other Ordner deiner EFI Partition und teste.

Leider auch nicht. Ich habe den/die/das kext so genommen wie er war und bekomme nach Neustart wieder bloß 14 + 1.

Dann habe ich die ports nochmal an mein Board angepasst und festgestellt dass der port-count auf 26 steht, aber auch das hat nicht geholfen.

Ich hatte bis jetzt immer nur USBInjectAll im Einsatz und den habe ich gelöscht. USB-KextToPatches sind auch nicht mehr anwesend.

Ich danke dir ganz doll für die Mühe und Zeit, die du extra für mich aufgebracht hast. Verfolge die Sache einfach nicht mehr, vielleicht liegt ja der Fehler ganz woanders.

Ich hatte in letzter Zeit gelegentlich USB-Probleme und irgendwann funktionierte die exclude-Liste nicht mehr und da bin ich auch nicht dahinter gekommen warum das so ist. Ich will das jetzt nicht bis ins Detail schildern. Auf der Suche nach dem "warum" bin ich dann auf GitHub auf ein USB-SSDT.command - script gestoßen, welches mir eine USB-SSDT erstellt hat. Die funktionierte auf Anhieb, ebenfalls ohne USBInjectAll.

Beitrag von „cobanramo“ vom 20. August 2017, 19:26

[@Doctor Plagiat](#)

[Zitat von Thogg Niatiz](#)

Eine Anmerkung, da mich das beim ersten Lesen selbst verunsichert hatte: Der port-count Eintrag entspricht, wie beschrieben, der höchsten verzeichneten Portnummer. Damit lässt sich aber nicht das nebenbei angesprochene Port Limit von 15 Ports umgehen, da dieses in den Treibern im AppleUSBXHCIPCI Binary fest definiert ist, also nicht auf port-count reagiert. Mit dieser Dummy Kext können beliebig viele Ports definiert werden, auch mehr als 15, wenn vorhanden, jedoch lädt der Treiber nur die ersten maximal 15 davon. Wem das nicht ausreicht, der kann:

weiterhin auf den Port Limit Raiser Patch (Clover "KextsToPatch" / Perl / etc - von Entwicklern immer wieder als riskant eingestuft) setzen, um mehr als 15 der definierten Ports ansprechen zu können, oder nicht explizit benötigte Ports (freie interne Header / freie Mainboard Buchsen / bei USB 3.0 Typ A Ports wahlweise auch entweder den 2.0 (HS) oder 3.0 (SS) Teil, wenn man nur den jeweils anderen braucht) nicht in die Info.plist eintragen

Mag ja sein das ich das missverstanden habe, diese Anleitung bring den USB 3 ans laufen ohne USBInjectAll.kext und hebelt den Portlimit nicht oder?

Gruss

Beitrag von „Doctor Plagiat“ vom 20. August 2017, 19:36

Fast richtig. Die info.plist bringt USB2 und USB3 ohne USBInjectAll zum Laufen. Und richtig, dass Portlimit wird nicht ausgehebelt.

Beitrag von „Brumbaer“ vom 21. August 2017, 12:38

[@cobanramo](#)

Deine Info.plist hat als Model iMac18,3 eingetragen, aber du hast wohl einen iMac17,1 - was man daran sieht, dass die andere Version mit dem model iMac17,1 läuft.

Beitrag von „cobanramo“ vom 21. August 2017, 12:47

Heilige bimbam, ich könnt schwören das ich das geändert hatte, ich hatte mehrere info.plist

auf dem Schreibtisch, muss wohl was durcheinander gebracht haben.
Danke dir Brumbaer, ich teste mal weiter.

Gruss Cobanramo

Beitrag von „Brumbaer“ vom 21. August 2017, 12:50

Bzgl. des Port Limit Patches.

Der Port Limit Patch ist unabhängig von dieser Sache.

Unter HS genügt der Port Limit Cache um alle Ports zur Verfügung zu stellen - man brauch dann gar kein Kext mehr - es sei denn der Controller wird von MacOS nicht unterstützt.

Beitrag von „kgp-illacpro“ vom 22. August 2017, 19:11

Hi Brumbaer,

Glückwunsch zu Kext as Kext can!! Ich hab mal versucht Deinen fertigen Kext einfach mal auf mein ASUS Prime X299 Deluxe mit SMBIOS iMAC17,1 los zu lassen und ich muss sagen 1A USB 3.0!!!

Ich versuche nun das ganze nachzuvollziehen und meinen eigenen board-spezifischen Kext zu bauen. Leider sind meine IORegistry Einträge anders als beim strix... und da komme ich mit meinem Verständnis ins Wanken (siehe Bilder im Anhang) ... Des Weiteren erscheint kein Dreieck wenn ich einen USB 2.0 Stick in die verschiedenen 3.0 Buchsen stecke nachdem ich ohne USBkext gebootet habe.. , was mache ich falsch?

Kannst Du mir etwas unter die Arme greifen? Das wäre wirklich sehr nett ! Dein USBKext ist für SMBIOS iMac17,1 und damit für alle X299/Skylake-X Boards wirklich fundamental!

Vielen Dank im Voraus!

KGP

Beitrag von „Brumbaer“ vom 22. August 2017, 22:30

Du kopierst das Kext und ersetzt in der ersten IOKitPersonality den SMBIOS Name und den Geräte-Namen (findest du im IORegistryEditor).

Dann änderst du die PortNamen, Portnummern und die Anschlüsse. Wenn du die Anschlüsse nicht kennst nimm 255.

Damit die Dreiecke erscheinen muss der Treiber zum Controller passen. Welchen USB Controller hat dein Board ?

Ich vermute mal, dass er von macos nicht unterstützt wird, also brauchst du einen Treiber dafür.

dafür ist der zweite IOKitPersonality Eintrag da.

Wie man die Controller Id und den passenden Treiber findet und wo man sie einträgt, steht in meinem zweiten Post.

Beitrag von „kgp-imacpro“ vom 24. August 2017, 00:59

Vielen Dank für deine rasche Antwort... ich weiss nicht welcher USB controller genau in dem Board verbaut ist... Ich schau mal wie ich an Hand Deiner Posts irgendwie weiterkomme...

nach mals vielen Danke für Deine Ratschläge,

KGP

Beitrag von „apfelnico“ vom 24. August 2017, 01:04

[Zitat von kgp](#)

ich weiss nicht welcher USB controller genau in dem Board verbaut ist...

"DPCIManager" ist dein Freund, der sagt dir das inkl. aller wichtigen Adressen.

Beitrag von „Brumbaer“ vom 24. August 2017, 12:04

Wie in der zweiten Post gezeigt, kann man die Controller ID im IORegistryEditor sehen.

Beitrag von „kgp-imacpro“ vom 24. August 2017, 21:18

[@Brumbaer'](#),

da hast Du dir aber ein einfaches mobo ausgesucht 😊

Fragen:

1.) USB3.1 Typ A Buchse: welche Nummer? Wie USB3.1 Typ C Buchse? Also (9) ??

2.) Alle USB3.1 Typ A und Typ-C Buchsen liegen bei mir nicht auf XHCI sondern auf

gesonderten Controllern auf denen auch alle Festplatten liegen...

3.) zur Belegung der SS Einträgen hab ich im Handbuch nichts gefunden...

4.) Das Ganze funktioniert nur unter 10.12.6 - Unter 10.13 sehe ich gar keine Rückmeldung in der IOREG... Ich hoffe wenn man einen kext dazu überhaupt erstellen kann, dass dieser dann auch für 10.13 funzt und dort nicht wieder die Einträge anders sind ...

Attached meine Portbelegung und Bilder von der IOREG...

Was meinst? Kannst mir bitte noch ein paar Tipps geben? Ich hänge unten auch nochmal meine IOREG an, erstellt mit IOReg Explorer 2.1

Vielen Dank im Voraus,

KGP

Beitrag von „Brumbaer“ vom 25. August 2017, 13:02

Die Intel XHCI Controller können kein Gen2. Fast alle verwenden dafür Asmedia Controller und die funktionieren für gewöhnlich. Die Asmedia Controller haben auch nur 4 Ports und die werden auch nicht gegen das Port Limit gerechnet.

Ich wiederhole meine Frage: Welcher XHCI Controller (mit mehr als 4 Ports) wird von deinem Board verwendet ?

Beitrag von „kqp-illacpro“ vom 25. August 2017, 13:26

Ich hab meinen letzten Post nochmals aktualisiert... Hab jetzt auch die USB 2.0 Portbelegung herausgefunden... Das einzige was jetzt noch fehlt sind die USB 3.1 Gen2 Typ-C 1x intern und 1x extern....

Ich weiss nicht welchen XHCI Controller das ASUS Prime X299 Deluxe verwendet.. Die USB3.0 und USB3.1 Ports sind jedenfalls nicht nativ eingebunden und laufen ausschliesslich mit USB2.0 Geschwindigkeit ohne kext...

Wie kann ich herausfinden welcher XHCI Controller genau im ASUS Prime X299 Deluxe verbaut ist?

Beitrag von „apfelnico“ vom 25. August 2017, 13:31

RP1/5/7 sollten hier keine Rolle spielen, sind die ASM3142 (USB3.1), werden auch nativ unterstützt, haben eh nur 1 bis zwei Schnittstellen, also max 4 Ports (USB2/USB3). Die Grenze von 15 Ports betrifft NICHT die maximale Anzahl an Ports des Rechners, sondern je eigenen Controller.

Konzentriere dich also auf das Device "XHCI". HS6 und (das dazugehörige) SS6 scheinen ganz interessant zu sein, dort hängen jeweils ein USB2 und USB3 Hub (ASM107x) dran mit aufgedröselten vier Schnittstellen. Das ist also jeweils nur ein Port, der Hub mit weiteren Schnittstellen dahinter interessieren nicht bei der Zählung. Sollten die also aktiv sein, so sind das schon mal eine Menge der vorhandenen tatsächlichen Schnittstellen am Board, die USB3.1 kannste ja eh wegdenken.

Sollte nicht so schwer sein, rauszufinden, welche der in der DSDT (pauschal) angelegten Ports von XHCI tatsächlich vorhanden sind – und darüber hinaus – welche du auch tatsächlich aktiv verwendest (Stichwort interne USB3). Da lässt sich einiges deaktivieren, somit sollte das Port-Limit auch für dieses Device keine Rolle spielen.

Edit: USB3.0 ist Standard Intel PCH.

Deine Probleme mit USB rühren vielleicht daher, dass du irgendwelche Kexte dazu via Clover lädst? Raus damit, läuft alles nativ.

Beitrag von „kqp-illacpro“ vom 25. August 2017, 14:16

[@apfelnico](#),

die genaue Portbelegung findest du in meinem Post oben angehängt.. Hier also nochmals als Attachment. Du hast doch dasselbe Mobo, oder?

USB 3.0 funktioniert bei mir in keiner Weise nativ! USB 3.0 Geräte werden nicht am USB 3.0 Port erkannt. Letztere funktionieren nur wenn man ein USB 2.0 Gerät in die USB 3.0 Ports steckt.

USB 3.0 funktionierte erst mit [@Brumbaer](#) s kext den ich als Versuch zu erst mal ohne jegliches editieren in /EFI/CLOVER/kexts/10.12 (10.12.6) und /EFI/CLOVER/kexts/Other/ (10.13) losgelassen hatte...

Also einfach in [@Brumbaer](#) s kext die XHCI die Einträge an Hand der aktiven Portliste (hier nochmals im Anhang) editieren und dann müsste zumindest USB 3.0 richtig funktionieren?

Un bezüglich USB 3.1 Gen 2 Type-A und Type-C kann man gar nichts machen? [@Brumbaer](#) ?

Ich wüsste nicht welche kexts ich laden würde die Probleme mit USB 3.0 verursachen würden. [Hier der Link zu meiner aktuellen EFI](#). Bitte wirf mal einen Blick drauf...

Dann kannst Du mir vielleicht auch gleich die notwendigen Audio Einträge setzen damit Mirones Tool läuft.... Wärs Du bitte so nett?

Vielen Dank,

KGP

P.S. Ich krieg bei Mirone für den Realteck ALC S1200A die folgende Dump (im Anhang)

Ist aber eigentlich anderer thread, sorry [@Brumbaer](#) ! Antwort dazu also bitte im dazugehörigen Thread... Danke !

Beitrag von „apfelnico“ vom 25. August 2017, 14:32

Sorry für OT.

Bitte in einen anderen Fred, Herr Doktor. 😊

Zweimal FakeSMC?

config.plist:

Die DropTables kannst alle rauswerfen, sind eh nicht vorhanden in deiner ACPI.

SmartUPS?

CPU QPI kann raus, hast eh keinen solchen Xeon drin (und kein Mehrprozessorboard)

USB alle Haken weg

FakeCPUID "0x0506E4"? ~~Du hast einen SkylakeX -> 0x050654~~-edit: ah, Broadwell-E

Möglicherweise hängt es ach mit deinem gewählten SMBIOS zusammen.

Bin ja nicht der Meinung, dass der iMac17 deinem Build nahe kommt. Nur weil der Prozi etwas ähnlich klingt? Ganz anderer Sockel, Technologie, Speicheranbindung (nur zweikanalig).

Denke du solltest erst mal die Grundkonfiguration in Schuss bringen. USB läuft jedenfalls nativ.

Ansonsten bin ich allerdings auf HighSierra unterwegs, kann schon sein, dass es hier eh anders aussieht.

Beitrag von „Brumbaer“ vom 25. August 2017, 14:54

Ich versuche zu helfen, aber ich werde dir kein Kext bauen.

In der zweiten Post steht wie man die Controller (device) id bestimmt.

Wenn du die bestimmt hast. Wirst du feststellen (falls du den Text aufmerksam gelesen hast), dass Treiber und Controller nicht zusammen passen.

Wie man das ändert steht da auch.

Dann hast du schon mal USB 3.0 (USB 3.1 Gen1) Unterstützung.

Dann kannst du über das Kext die Ports limitieren oder den Port Limit Patch benutzen.

Wie oben geschrieben wird USB 3.1 bzw. USB3.1 Gen 2 nicht über den Chipsatz USB Controller gehandhabt, sondern über einen anderen Controller. Diese Ports haben mit den Sachen in diesem Thread nur indirekt zu tun. Und diese Ports sollten funktionieren.

Das selbe Kext funktioniert für Sierra und High Sierra.

Ich nehme an du weißt, dass der Port Limit Patch für 10.3 ist ein anderer als der für 10.2 ist. Wenn ich mich richtig erinnere hat sich der Patch bei irgendeiner Beta noch einmal geändert.

[@apfelnico](#)

USB 3.0 funktioniert beim X299 Chipsatzes nicht ohne Anpassung.

Beitrag von „kgp-illacpro“ vom 25. August 2017, 14:59

[@apfelnico](#) ,

Zwei mal FakeSMC? Wo?

FakeCPU "0x0506E4" ist Skylake-X nicht Broadwell-E ! Ist aber reine Kosmetik für den Apple System Report... Läuft auch ohne...

SMBIOS iMac17.1 ist absolutes Muss für ssdtPRGen, ssdt.aml und natives XCPM !!!

Und da SMBIOS iMac17.1, kein USB 3.0 🙄

Daher muss ich unbedingt [@Brumbaer](#) s kext zum Laufen bringen...

Such mal im Internet nach "Skylake-X/X299 - The Ultimate Customac Pro - Live the Future now on macOS 10.12 Sierra" und "Skylake-X/X299 - The Ultimate Customac Pro - Live the Future now on macOS 10.13 High Sierra", zwei Skylake-X/X299 Guides die ich grade entwickle... Dort findest Du bereits viele Antworten auf Deine Fragen 👍

Beitrag von „apfelnico“ vom 25. August 2017, 15:17

Zitat

Zweimal FakeSMC?

AppleEmulator.kext (ebenfalls FakeSMC)

Zitat

SMBIOS iMac17.1 ist absolutes Muss für ssdtPRGen, ssdt.aml und natives XCPM !

Ach was. Geht genau so mit MacPro6.1.

Die ssdtPRGen muss nicht sein, "plug"1" bekommt man auch leichter an die CPU.

[Natives Powermanagement - Alternative für ssdtPRGen - Hilft ggf. bei Sleep/Wake-Problemen](#)

Zitat

Such mal im Internet nach...

Hab ich gelesen, bin auch bei Tony. Steht nicht viel drin, einiges falsch, zweiter Ethernet geht selbstverständlich auch, ist auch kein 10Gbit.

Zitat

Ist aber reine Kosmetik für den Apple System Report... Läuft auch ohne...

Nicht unbedingt. Clover wertet das aus und bringt "unter der Haube" schon hilfreiche Patche mit. Schau dir dazu aktuelle Clover-Entwicklung an.

Lass uns aber dazu einen eigenen Thread erstellen zum Thema X299. Bin mit meiner DSDT fast durch.

Beitrag von „kgp-imaopro“ vom 25. August 2017, 15:39

1.) O.K.. ich schmeiss den AppleEmulator dann raus...

2.) Ich bleib bei ssdtPRGen und SMBIOS iMac17,1! Über den richtigen Skylake-X/X299 SMBIOS Ansatz läßt sich streiten .. Wenn Du meinst dass Du am richtigen Weg bist dann viel Glück dabei 👍

3.) [@Brumbaer](#) 's kext ist genial... endlich los von USBInjectAll un all dem Müll.. Du kannst aber auch hier gerne mit Deinem MacPro6,1 Ansatz und mit USBInjectAll weiter kurbeln... Aber bitte erzähle mir nicht dass USB 3.0 bei den X299 Boards nativ läuft! 😊

4.) Meine Threads bei Tony sind noch in Entwicklung.. Kannst ja gerne helfen wenn noch was fehlt oder berichtigt werden muss 👍 Das mit dem Ethernet z.B. werde ich asap korrigieren... Sonst noch was, was falsch wäre? Wie gesagt bzgl. Audio werde ich dann auch auf Mirone setzten... Mehr dazu im entsprechenden Thread...

5.) Wir können gerne zum Thema FakeCPUID und Clover einen eigenen Thread zum Thema X299 hier erstellen wenn Du magst.. Bin da gerne dabei...

[@Brumbaer](#) ich hab noch ein paar Fragen an Dich, ich hab meinen eigenen kext fast fertig:

1.) Ich hab gesehen, dass in Deinem Kext HS06 und HS12 fehlen... Ich gehe daher davon aus dass ich die HS Einträge weglassen kann die nicht belegt sind.. Also bei wären das HS01, HS02, HS13, HS14. Ich musste aber im Gegensatz zu Deiner Kext auf Grund meiner Port-Belegung HS06 und HS12 hinzufügen... Ist das so richtig?

2.) Nun sagst Du dass ASUS normaler Weise 5 von 6 SSPs Einträgen mit den ersten 5 HS Einträgen verknüpft. Bei mir sind HS01 und HS02 aber offensichtlich gar nicht belegt (zumindest hat das von Dir vorgeschlagene Experiment das ergeben). Wenn ich HS01 und HS02 rauswerfen würde, da nicht belegt, würden aber SSP1 und SSP2 ins Leere laufen... Der ganze Spass beginnt bei mir offensichtlich erst ab HS03. Wie kann ich also wissen welchen HS ports die 5 oder sogar vielleicht 6 SSPs zugeordnet sind? Trotzdem SSP1-SSP5 zu HS01-HS05?

3.) port counts

Wenn ich also dann davon ausgehe dass SS1 bis SS5, HS01 bis HS05 zugeordnet sind (obwohl HS01 und HS02 nicht belegt sind), kann ich auch nicht HS01 und HS02 rauswerfen, da mit SSP1 und SSP2 verknüpft. Die entsprechende Werte für alle 4 Einträge würde ich in diesem Fall auf 255 setzten, da zumindest HS01 und HS02 gar nicht belegt.

4.) Von den HS Einträgen hab ich dann 12 (an Stelle von 10 ohne HS01 und HS02 Einträge) + zumindest 5 SSPs Einträgen, was dann einen port count von 17 an Stelle von 15 ergibt.. Übrigens Die Zahl die Du unter port-count angibst <1a000000> ist doch 26 und nicht 15, oder hab ich da einen Denkfehler?

5.) Da Dein Kext schon ohne Änderungen und Anpassungen bei mir zuvor mehr oder weniger funktioniert hat, gehe ich davon aus dass der Treiber-Eintrag IOPCIPrimary Match 0xa2af8086 auch mit meinem Controller funktioniert. Wo finde ich in der IOREG die PrimaryID meines Controllers? Ich sehe nur class code "30 03 0c 00", device id "af a2 00 00" und vendor "id 86

80 00 00". Dann gibt es noch einen "compatible" Eintrag "pci1043,873c", "pci8086,a2af", "pciclass, 0c0330", "XHCI" und einen "name" Eintrag "pci8086,a2af"... Sollte ich Dank Deiner Hilfe den Eintrag dann doch finden, und sollte dieser dann von deinem Wert Abweichen, muss ich nur noch in der AppleUSBHCIPCI nachsehen um den passenden Treiber zu finden, richtig?

Ich weiss, viele Fragen. Aber ich mach was ich kann...

Auf alle Fälle funktioniert interessanter Weise Dein iMac17,1 kext auf meinem System durchaus erfolgreich. Mein Kext hingegen funktioniert im Moment gar nicht mit den 17 Einträgen oder weitere große Änderungen...

Vielen Dank für Deine Hilfe!   

Gruß,

KGP

Beitrag von „Brumbaer“ vom 25. August 2017, 21:51

1) Ja

2) Wenn USB 3.0 läuft steckst du einen USB 3.0 Gerät an die Buchse und schaust an welchem SS Port das Gerät erscheint. Dann machst du es ab und steckst ein USB 2.0 Gerät an und schaust an welchem HS Port das Gerät erscheint.

3) Siehe doch mal zu dass du den richtigen Treiber installierst. Dann Stecke ein USB 2.0 Gerät nacheinander in die Buchsen die du zu verwenden gedenkst. Schau bei welchem HS Port das Gerät erscheint. Dann hast du eine Liste aller verwendeten HS Ports. Dann machst du das gleiche mit einem USB 3.0 Gerät und schreibst dir die SS Ports auf.

4) Das Feld port-count enthält keinen port-count. Das Feld ist verwirrend benannt. Port-count enthält die höchste Port-Id. Die entspricht häufig der Anzahl Ports, aber nicht immer.

Der Gedanke hinter port-count ist wohl, dass man gerne ein Array mit einem Eintrag für jeden Port hätte. Damit man schnell auf den Eintrag für ein Port zugreifen kann verwendet man die Port-Id als Index für das Array. Das Array, muss dann nicht so viele Einträge haben, wie man Ports benutzt, sondern wie hoch die Port-Id sein kann. Port-Count ist also nicht die Anzahl der

Ports, sondern die Anzahl der Einträge in dem Feld für Ports, und die entspricht der höchsten Port Id. Da der verwendete Controller als höchste Port-Id 0x1A hat, ist man mit der 0x1A auf der sicheren Seite, aber "korrekter" wäre in deinem Beispiel natürlich 0x15 (falls SSP5) das höchste verwendete Port ist.

5) X299 und Z270 verwenden den gleichen XHCI Controller, deshalb funktioniert die Treiber Anpassung für mein System auch für dein System. Wie in meinem zweiten Post steht findet man die Id in lesbarer Form im "name" Feld des ersten XHCI Eintrages in IORegistryExplorer. da steht pci8086,a2af. Das entspricht dem IOPrimaryMatch 0xa2af8086.

Was an deinem Kext schiefliegt, kann ich nicht sagen ohne es zu sehen.

Beitrag von „cobanramo“ vom 25. August 2017, 22:10

[@kgp](#) bei dir greift der Portlimit eben, max 15 Port's / Controller !

Portlimit Patch ins Clover oder 2 USB Port's opfern.:~)

Gruß Cobanramo

Beitrag von „Noir0SX“ vom 25. August 2017, 22:57

[@kgp](#) mal so am Rande noch. Ich fände es auch mehr als Fair die Web-Seite im Profil raus zunehmen. Ist ja auch nicht wirklich Deine Kontaktmöglichkeit 😊

Beitrag von „kgp-illacpro“ vom 25. August 2017, 23:05

[@Brumbaer](#)

genau da liegt das Problem.. Im Notfalltreiber Modus (also ohne Deinen Kext) tut sich mit USB 3.0 in USB 3.0 absolut gar nichts in der IOREG. Deswegen sehe ich auch nicht welche SS Ports aktiviert werden... USB 2.0 in USB 3.0 funzt und USB 2.0 in USB 2.0 natürlich auch.. daher kann ich auch die HS ports zuordnen!

Sobald ich Dein Kext verwende, klappt dann auch USB3.0 in USB 3.0! Aber dann sind die Ports ja schon wie in Deinem Kext definiert zugeordnet, oder? Das hilft mir also auch nicht weiter....

Interessant ist übrigens auch dass auf meinem System der Gerätename des XHCI Controllers im Notfallmodus XHCI ist, du aber in Deinem Kext als IONameMatch XHC angibst. Wenn ich aber mit Deinem Kext boote steht trotzdem wieder als Gerätename des XHCI Controllers XHCI da. Ich hätte erwartet dass Dein Kext gar nicht funktioniert auf Grund des falschen Gerätenamens im Kext.

Also ich weiss nicht wie ich die SS-Port Zuordnung ohne funktionierendes USB 3.0 in USB 3.0 im Notfall Modus raus kriegen soll oder kann....

[@cobanramo](#), natürlich verwende ich den Port-Limit Patch! Das ist nicht das Problem!

[@BlackOSX](#) , Deinem Wunsch wurde Genüge geleistet

Beitrag von „Brumbaer“ vom 25. August 2017, 23:20

Welche Personality ist für den Treiber zuständig ?

Die Personality wird aktiv wenn ?

Das sollte eine Frage beantworten.

Du kannst die Personality in dein Kext kopieren und dann hat dein Kext Treiber Unterstützung.

Da USB 2.0 funktioniert solltest du schon wissen welche SS Ports belegt sind.

Du trägst die in dein Kext ein und alle SS Ports egal ob du denkst dass sie belegt sind oder nicht. Dann testest du die USB 3.0 Buchsen mit einem USB 3.0 Gerät.

Falls du den Port Limit Switch benutzt, wirfst du dann noch alle Ports aus deinem Kext die nicht benutzt werden.

Falls du den Port Limit Switch nicht benutzt, wirfst du dann alle Ports aus deinem Kext die nicht benutzt werden **und** die im IORegistryEditor sichtbar waren.

Dann testest du erneut. Jetzt sollten einige oder alle noch nicht in IORE sichtbaren SS Ports sichtbar werden. Du testest wieder die Buchsen (nur die für die du noch kein USB 3.0 Gerät gefunden hast) ob bei einem USB 3.0 Port auftaucht. Dann wirfst du Ports die mit keiner Buchse verbunden sind aus dem Kext, startest neu, die anderen Ports rutschen nach, testest usw.

Beitrag von „kgp-illacpro“ vom 25. August 2017, 23:36

YES..... 👍 Genial [@Brumbaer](#) thats the way! Großartige Idee !

Update:

@Brumbear es funzzzzzzttttt 👍👍👍

[Full Success on the ASUS Prime X299 Deluxe with SMBIOS iMac17,1]

Alle USB 2.0 und 3.0 Ports funktionieren jetzt wie sie sollen !!!!! 👍



Hoch lebe [@Brumbaer](#) !!!!

Die angehängte Datei sollten man runterlesen um die Resultate wirklich zu sehen...

Beeindruckend!

Ich hab jetzt meine kext as kext can 👍

Beitrag von „ernie-t“ vom 26. August 2017, 13:44

Funktioniert das auch für Haswell Z97X Chipsatz?

Beitrag von „Brumbaer“ vom 26. August 2017, 14:32

Ja die selbe Vorgehensweise.

Beitrag von „derHackfan“ vom 26. August 2017, 20:08

Zitat von Brumbaer

Nun erzeugen wir die Info.plist Datei.

XCode hat unter New -> File eine Vorlage für Property Lists. Diese erstellt eine leere PList.

Geht das auch ohne Xcode?

Immerhin sind das (nebenbei) mal eben 4.54 GB die man da runterlesen muss.

Beitrag von „Brumbaer“ vom 26. August 2017, 20:19

Eine existierende kopieren und dann bearbeiten mit:
Texteditor - weniger übersichtlich und mit Fehlerpotential,
XML Editor - wenn man einen hat, kann man's probieren,
und es gibt spezielle PList Editoren von "unabhängigen Entwicklern" - kann mangels Erfahrung keine Empfehlung aussprechen

Beitrag von „derHackfan“ vom 26. August 2017, 20:24

Also nein.
Gott seid Dank, ich dachte schon ich muss deine Anleitung durcharbeiten.

Beitrag von „Brumbaer“ vom 26. August 2017, 21:27

Na da hamm wa ma wieda Glück gehabt 😊

Beitrag von „Altemirabelle“ vom 26. August 2017, 21:43

Schon geniale Anleitung, edukativ sehr wertvoll.
Werde das durcharbeiten. Macht echt Spass eigene Kext zu erzeugen.

Und am Ende werde ich:

Im Terminal folgenden Befehl eingeben und enter drücken

```
say -v pipe organ "hi hih hah hah hah hi hi hi hih hah hah hah hi hi hi hih hah hah hah hi hi hi hih hah hah hah hi hi hi hih hah hah hah hi hi hi hih hah hah hah hi hi "
```



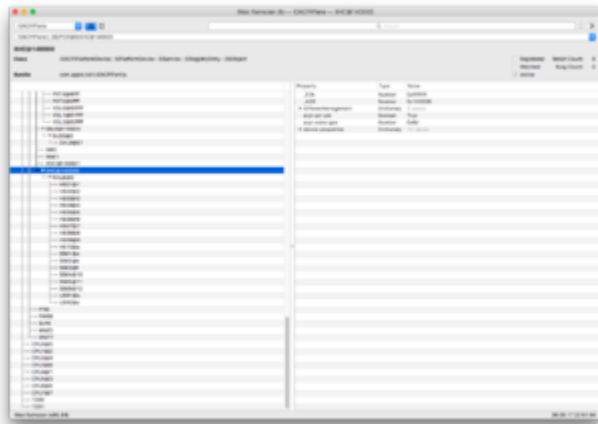
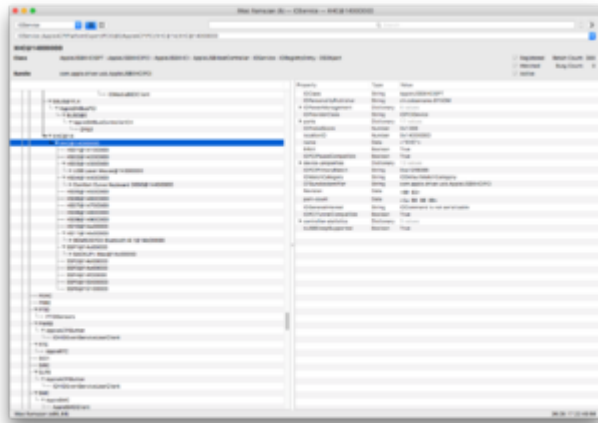
Beitrag von „cobanramo“ vom 26. August 2017, 22:04

[@Brumbaer](#)

Du Brumbaer, hab seit ein paar tagen nach deiner Anleitung erstellten Plist im Einsatz.

Im IoReg hab ich alles so wie es sollte, müsste ich eigentlich immer noch die nicht vorhandenen Ports im clover excluden? oder einfach sein lassen?

Ich hab zusätzlich den PortLimit Patch drinne.



Beitrag von „Brumbaer“ vom 26. August 2017, 22:32

Die exclude Liste in Clover gehört zu welchem Kext ?
Benutzt du es noch ?

Beitrag von „cobanramo“ vom 27. August 2017, 02:41

Die liste hab ich neulich selber berechnet, mit PortlimitPatch und USBInjectAll.kext zeigt es bei mir 14 HS & 10 SS Ports an.
Nein Ich habs noch nicht eingetragen.

uia_exclude=HS12;HS13;HS14;SSP7;SSP8;SSP9;SSP10;USR1;USR2

Gruss

Beitrag von „Doctor Plagiat“ vom 27. August 2017, 08:45

[Zitat von Brumbaer](#)

Die exclude Liste in Clover gehört zu welchem Kext ?
Benutzt du es noch ?

[@cobanramo](#) Brumbaer wollte dich mit dieser Frage, die du ihm nicht beantwortet hast, auf das eventuelle Problem stoßen. Mit seiner Kext-Methode musst du alle anderen USB-Kexte entfernen.

Beitrag von „cobanramo“ vom 27. August 2017, 11:59

[@Doctor Plagiat](#) ach Mann so ist das gemeint, danke 🙄

[@Brumbaer](#) hab bitte bisschen geduld mit mir, sprachlich bin leider nicht der Hirsch 😄
Exclude Liste gehört zu USBInjectAll.kext, da wir den nicht einsetzen obsolet, richtig?
wenn man die richtigen fragen stellt & versteht klapst ja, danke dir. 😊

Gruss

Beitrag von „Brumbaer“ vom 27. August 2017, 12:58

[@cobanramo](#)

Stimmt - kein USBInjectAll - keine Excludeliste in Clover.

Beitrag von „Noir0SX“ vom 27. August 2017, 18:02

Fragt jetzt nicht wie oft ich das nun rauf und runter gelesen habe.....
....aber es läuft

DANKE [@Brumbaer](#) für Deine ausführliche Anleitung.

Beitrag von „derHackfan“ vom 27. August 2017, 18:14

Habe das ganze jetzt auch mehrfach gelesen, dabei um einiges mehr verstanden und versucht einen Kext zu basteln, also doch sehr gut geschrieben. 😁

Hat nur nicht funktioniert.

Es fängt damit an dass ich im IOReg zum Teil ganz andere Einträge finde, dass ich ohne Xcode die plist bearbeiten muss, dass mein Kext nicht geladen wird.

Beitrag von „Brumbaer“ vom 27. August 2017, 18:38

[@derHackfan](#)

Was spricht gegen XCode ? Kein Internet ? Ansonsten, kann das Rechner laden, wenn man schläft.

Es geht auch mit TextEdit, einige wenige Sachen sogar besser, aber es erfordert Disziplin und Aufmerksamkeit - lieber XCode 😊

Wie du aus eigener Erfahrung weißt ist es schwer zu helfen, wenn man keine Infos hat.

Wenn du denn soweit bist, dass du Hilfe annehmen würdest, könntest du ja mal dein Kext und ein IORegistryEditor-File deines Systems posten.

Beitrag von „apfelnico“ vom 29. August 2017, 01:28

[Zitat von Brumbaer](#)

USB 3.0 funktioniert beim X299 Chipsatzes nicht ohne Anpassung.

Richtig, danke. Hatte ich nur "halb" hingeschaut. Controller da, Treiber geladen. So ist's natürlich viel besser.

Beitrag von „cobanramo“ vom 31. August 2017, 16:08

Hallo [@Brumbaer](#), hoffe kannst mir da weiterhelfen.

In einem anderen tread versuchen wir gerade Wake/Sleep problem zu lösen und da muss anscheinend die USB geschichte gelöst sein.

Da ich nach deiner Anleitung vorgegangen bin frag ich mal hier was bei mir so schief läuft 😊

zunächst mal der plist den ich einsatz hab;

[Info.plist](#)

Hier physische Zustand aller Ports;



Ich hoffe mal das ich das ganze nicht wieder völlig verkehrt verstanden hab.

Hab alle Board USB 2 Connectoren 0x00, alle USB 3 Connectoren 0x03 gesetzt.

Alle Front/Header Connectoren sind auf 0xff, WLAN/Bluetooth ist auch auf 0xff
dass ganze funktioniert auch bis auf's inter/extern Icon geschichte.

Wenn ich alles korrekt nach anleitung vorgehe werden USB HD's an den HeaderPort's (0xff) als interne erkannt.

An den anderen Port's ist alles ok.

Wie wichtig ist das oder eben kann ich die HeaderPort's auch als usb2 oder usb3 setzen?


getestet hab ich das und es tut auch so funktionieren, eben weiss nicht so recht ob mein sleep/wake daher nicht funktioniert.

Könntest du bitte mal mein plist angucken ob ich da was falsch mache oder eben ne andere lösung brauche.

Danke. Gruss Cobanramo



Beitrag von „SirusX“ vom 31. August 2017, 16:23

Ja xCode kann man immer mal gebrauchen, man muss es ja nur einmal laden nicht immer wieder. 

Beitrag von „Brumbaer“ vom 31. August 2017, 17:14

[@cobanramo](#)

Du kannst statt 0xFF auch 0x03 bzw 0x00 nehmen.

Die Kennung dient u.a. zur Icon Auswahl.

Ich nehme an, dass sie auch für die max. Stromaufnahme des/der angeschlossenen Geräte verwendet wird.

Da der MoBo Header letztendlich für den Anschluss einer USB 3.0 bzw. 2.0 Buchse vorgesehen ist, sehe ich kein Problem den Anschluss entsprechend zu benennen.

Das Kext enthält keinen Programmcode und ruft Apple Kexte mit den selben Parametern auf, wie die Original Apple Kexte.

Ich bezweifle, dass das Kext (selbst) Probleme bei Sleep/Wake verursacht. Auf der anderen Seite korrigiert es auch keine Unstimmigkeiten, die BIOS und macos eventuell haben.

Sleep/Wake verursacht bei vielen Systemen Probleme.

Da ich ein bekennender Nicht-Ruhemodus-er bin, habe ich keine Erfahrungen mit deren Ursachen und Behebung.

Tut mir leid.

Beitrag von „cobanramo“ vom 31. August 2017, 17:20

Danke Brumbear, werde umstellen damit hat sich das Icon problem erledigt.

Glaub das mit sleep kann man vergessen.

Danke fürs unterstützung und deinem tollen anleitung.

Gruss Cobanramo

Beitrag von „DSM2“ vom 3. September 2017, 04:27

Also irgendwie komme ich damit gar nicht voran... wird sicherlich mein Fehler sein und doch komme ich nicht drauf.

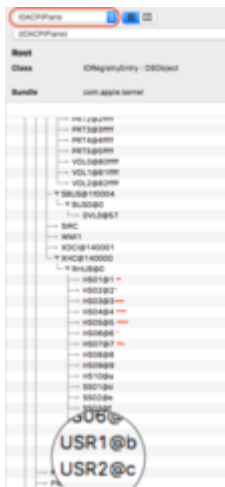
Hab das Kext erstellt und alle HS Ports aufgeschrieben, Kext geladen und wollte dann nach den SS Ports suchen,
doch leider wird mir nur ein einziger SSP Port angezeigt und das ist der SSP1 unter Sierra 10.12.6,
alle anderen USB 3.0 Ports funktionieren gar nicht erst...
Hab probiert das SMBios zu ändern weil ich dachte vielleicht hat es damit zu tun, verschiedene USB Limit Patches versucht etc und doch leider keine Besserung.
Ok dachte ich mir, probiere ich es doch mal unter High Sierra und siehe da es funktioniert wie es soll unter 10.13., also habe ich alles unter 10.13 vollständig erstellt aber es will einfach nicht unter 10.12.6 laufen!

Habe selbstverständlich alle HS/SS Ports aus dem Kext entfernt die nicht genutzt werden.

Hänge Bilder und Kext an, vielleicht erkennt es ja jemand auf anhieb.
Danke schonmal im Voraus!

Beitrag von „cobanramo“ vom 3. September 2017, 07:40

Guck mal unter "IOACPIPlane" die durchnummerierung obs passt.



Beitrag von „Brumbaer“ vom 3. September 2017, 10:15

[@DSM2](#)

IORegistry zeigt XHCI als DeviceName.

In deinem Kext verwendest du aber XHC als IONameMatch - muss auch XHCI sein.

Beitrag von „DSM2“ vom 20. September 2017, 16:36

Danke dir für das Tutorial und natürlich auch für die Fehlersuche [@Brumbaer](#)
Das war natürlich goldrichtig. Werkelt nun alles wie es soll!

Mein funktionierendes File für alle anderen gibt es natürlich ebenfalls hier zum Download.

EDIT:

Hab das alte File rausgenommen weil da noch ein kleiner Fehler drin war und gegen ein aktuelles ausgetauscht.

Achja in meinem File ist nur der USB 3.0 Controller Port neben den RAM Bänken des Asus X99-A II definiert,

sprich wenn ihr eure Gehäuse USB 3.0 Anschlüsse am unteren Host angeschlossen habt wird USB 3.0 nicht funktionieren!

Beitrag von „apfelnico“ vom 20. September 2017, 19:04

Mein Board hat EHCI, EHCI2 und XHCI (X99) und zusätzlich einen ASM1042A, der jetzt keine weitere Rolle spielt. Damit XHCI problemlos läuft, benötigt dieser "AppleUSBXHCILPTH". Egal ob nun deine individuelle Kext oder über Rehabmans UsbInjectAll - was letztendlich ebenfalls bewirkt, dass "AppleUSBXHCILPTH" geladen wird, verstehe ich eine Sache daran nicht:

Solange XHCI mit dem "Wald und Wiesen Treiber AppleUSBXHCIPCI" betrieben wird (was ja nur eingeschränkte Funktionalität bietet), werden die EHCI korrekt behandelt. Das heißt, reine USB2 vom Mainboard an entsprechende Buchsen am Gehäuse und daran angeschlossene Geräte werden auch korrekt im Systembericht (wie auch IORegistryExplorer) angezeigt. Auch das auf dem Board integrierte Bluetooth-Modul hängt intern am EHCI und wird dort angezeigt. Sobald aber AppleUSBXHCILPTH sich an XHCI andockt, werden sämtliche Geräte von EHCI unter XHCI verbucht.

Es funktioniert alles, ich hab auch keinen Stress mit dem Portlimit, mir war nur schon vor deiner sehr einfachen und individuellen Kext nicht klar, was da eigentlich diesbezüglich intern läuft. Alle Versuche auch EHCI vernünftig zu deklarieren ändern nichts an diesem Verhalten. Nicht das ich hier unbedingt etwas ändern muss, ich verstehe es nur einfach nicht.

Beitrag von „kaneske“ vom 20. September 2017, 21:28

Hat jemand das schon auf ein X99er Gigabyte Board implementiert?

Beitrag von „Brumbaer“ vom 21. September 2017, 09:20

[@apfelnico](#)

Der X99 Chipsatz hat 2 EHCI und einen XHCI Controller.

Allerdings teilen sich beide die selben Ports. Ein Port kann nur von einem Controller verwendet werden.

Wird der XHCI Controller korrekt initialisiert bekommt er die Ports, sonst nehmen sie sich die EHCI Controller (einer 8, einer 6 Ports).

Der AppleUSBXHCIPCI passt nicht zum X99 Controller, deshalb funktioniert USB 3.0 auch nicht mit ihm.

Die Ports werden zwar korrekt (im Sinne von IORegistryExplorer) eingetragen, funktionieren aber nicht.

AppleUSBXHCILPTH ist der richtige Treiber für den XHCI Controller. D.h. der XHCI Controller wird korrekt initialisiert, alle Ports gehen an den XHCI Controller und die EHCI Controller haben das nachsehen.

Wenn man sich im AppleUSBXHCIPCI.kext den Eintrag für AppleUSBXHCILPTH anschaut sieht man, dass dieser für das Device mit der ID 8c31 angelegt wurde.

Das ist der XHCI Controller des Vorgängers des X99 Chipsatzes (C220 aus 2013, C610 aus 2014).

Beitrag von „apfelnico“ vom 22. September 2017, 01:23

Prima, danke. Jetzt ist es klar.

Gesendet von iPhone mit Tapatalk Pro

Beitrag von „kqp-illacpro“ vom 22. September 2017, 09:04

[@Brumbaer](#)

eine Frage. Das ASUS X99-A II hat 2 interne USB 3.0 Stecker (für z.B. 4 externe Front Panel USB3.0 Anschlüsse) sowie 4 externe Back Panel USB3.0 Anschlüsse. Solange man nur einen der zwei internen USB 3.0 Stecker (SSP1,SSP2 oder SSP3,SSP4 und respektive HS Ports) sowie die 4 externen USB3.0 Anschlüsse (SSP5,SSP6 und respektive HS Ports) einbindet, funktioniert alles wunderbar. Bindet man aber beide internen USB3.0 Stecker (SSP1, SSP2, SSP3, SSP4 und respektive HS-Ports) und alle 4 externen Back Panel USB3.0 Anschlüsse (SSP5,SSP6 und respektive HS-Ports) ein, ist plötzlich Schluss mit lustig. In der IOREG scheinen dann nur noch SSP1 und SSP2 auf und alle USB3.0 ports, die mit SSP3,SSP4, SSP5 und SSP6 verbunden sind funktionieren gar nicht mehr oder nur noch als USB2.0. Um alle USB3.0 Anschlüsse des Asus X99-A II einzubinden, braucht man alle 6 SS-Ports und respektive HS-Ports und das scheint eben nicht zu funktionieren.

Bei, ASUS Prime X299 Deluxe sind die USB3.0 Anschlüsse besser implementiert. HS01, HS02 sowie SSP1 und SSP2 werden von dem Board gar nicht verwendet. Man benötigt also nur 4 SS-Ports (SSP3,SSP4,SSP5 und SSP6) und respektive HS-Ports und alle verfügbaren USB3.0 Anschlüsse sind damit wunderbar eingebunden.

Gibt es irgendeine Erklärung warum man nicht mehr als 4 SS-Ports und respektive HS-Ports einbinden kann? Die erste Vermutung war das USB Port Limit. Mein USB Port Limit KextToPatch Eintrag ist aber richtig implementiert und wird auch korrekt vom System verwendet. Auch die Variation der Port limits (zw. z.B. 24 und 30) scheint nichts daran zu ändern dass beim

Einbinden von mehr als 4 SS-Ports und respektiver HS-Ports, eben Schluss mit lustig ist.

Um alle USB2.0 und USB3.0 Anschlüsse einzubinden bräuchte man (HS01 bis HS10 und HS12 bis HS14 sowie alle 6 SSP-Ports), also insgesamt 19 Ports. Was nicht funktioniert ist z.B. (HS01,HS02, HS05 bis HS10 und HS12 bis HS14, sowie SSP1, SSP2, SSP5 und SSP6), also genau 15 Ports!

Den Port Count habe ich im Kext übrigens auf 1a000000 gesetzt.

Als USB Port Limit KextToPatch Eintrag verwende ich unter 10.13:

AppleUSBXHCIPCI Find:837d8c10 Replace:837d8c19 (wobei ich 19 als XX verstehe und entsprechend variere, ohne jeden Erfolg

Als USB Port Limit KextToPatch Eintrag verwende ich unter 10.12.6:

AppleUSBXHCIPCI Find:83bd74ff ffff10 Replace:83bd74ff ffff1b

Ich bin gespannt was Du dazu sagen kannst.... Weisst Du irgendeine Lösung für das 6 SS-Port Problem?

Beitrag von „apfelnico“ vom 22. September 2017, 09:26

[@kgp](#) - da ist nix dran. Es funktionieren sowohl beim X99 alle SSP nebst ihren zugehörigen HS vom XHCI, wie auch beim X299. Selbstverständlich sind beim X299 SSP1/HS01 und SSP2/HS02 vorhanden. Bei mir hängen daran zum Beispiel die beiden Front-USB3 des Gehäuses. Ob die nun bei deinem Board verwendet/verkabelt sind, oder du diese per DSDT, SSDT oder Kext umbenannt hast, weißt nur du. Auch ist es möglich, diese im BIOS hardware-seitig abzustellen.

Beitrag von „kjp-illacpro“ vom 22. September 2017, 10:06

[@apfelnico](#)

ich hab beim ASUS Prime X299 Deluxe gar nichts umbenannt. Weder mit DSDT, noch SSDT noch kext... HS01,HS02,SSP1 und SSP2 werden nicht verwendet. Meine USB3.0 Front Ports hängen an HS03, HS04 sowie SSP3 und SSP4... eine Diskussion zum ASUS Prime X299 Deluxe ist hinfällig da mein XHC USB Kext für dieses Port eh perfekt funktioniert.

Bzgl. meiner eigentlichen Frage hätte ich lieber wenn möglich eine Antwort von [@Brumbaer](#).

Den ASUS X99-A II XHC USB Kext entwickelt eigentlich eh der [@DSM2](#) und nicht ich... Ich hatte ihm dabei nur ab und zu etwas geholfen. Trotzdem würde ich gerne wissen ob es bzgl. der HS und SS Port-Zahl wirklich ein Problem gibt, und wenn ja wo dabei genau das Problem liegt und ob es dafür eine Lösung gibt.

Danke

KGP

Beitrag von „apfelnico“ vom 22. September 2017, 10:26

Sorry, du hattest ja das X299 ins Spiel gebracht. War nur ein Hinweis. Dann hatte ich das wohl missverstanden: "HS01, HS02 sowie SSP1 und SSP2 werden von dem Board gar nicht verwendet." Sollte gemeint sein, werden von Dir nicht verwendet.

Zum X99, schau dir noch mal die Port-Nummerierung an.

Beitrag von „kjp-illacpro“ vom 22. September 2017, 10:47

Die entsprechenden Ports werden von mir nicht verwendet da sie auch mein X299 Mobo ohne jeden USB kext in der OS X USB-Notkonfiguration nicht verwendet. Auf alle Fälle funktioniert mein X299 kext ohne wenn und aber...

Die Lösung zum ASUS X99-A II kext überlasse ich [@DSM2](#). Ich wollte von [@Brumbaer](#) nur wissen, ob es generell ein Problem bei der verwendeten HS und SS Port-Anzahl geben kann. Und wenn ja, warum in diesem Fall der USB Port Limit Patch möglicherweise nicht greift. Eigentlich war dies die Vermutung von [@DSM2](#) und ich wollte mich da bei [@Brumbaer](#) diesbezüglich nur rückversichern.

Ich hatte eigentlich nicht vor mit Dir eine ausgedehnte Diskussion zum Thema zu starten 😊

Beitrag von „apfelnico“ vom 22. September 2017, 11:23

Dann antworte doch nicht immer, bin ja somit auch im Zugzwang. 😊

He, alles gut. Wollte nur damit ausdrücken, beim Asus Prime X299 Deluxe sind alle Ports unter macOS vorhanden. Auch in der Basiskonfiguration mit AppleUSBXHCIPCI. XHCI funktioniert natürlich dann korrekt mit AppleUSBXHCILPTH.

Ebenfalls beim X99. Bin aber sehr gespannt auf Brumbaer, schätze seine Kompetenz sehr.

Edit: Was mich nur verwirrte und nun Klarheit besteht, war obige Frage von mir zum Thema X99. Diesbezüglich ist X99 "komplizierter" als X299, da neben dem XHCI noch zwei EHCI existieren und je nach Konfiguration (auch schon im BIOS) die Controller verschieden auf die Ports zugreifen, teilweise auch im Bootprozess noch wechseln. In diesem Zusammenhang ist mir nun auch klar, warum es deutlich mehr HS## als SSP# am XHCI gibt. Fakt ist aber (ich habe auch so ein Board), du kannst alle sechs SSP# nebst deren korrespondierenden HS## nutzen.

Grüße (ich halte jetzt meinen Mund, muss endlich arbeiten)

Beitrag von „kgp-imaopro“ vom 22. September 2017, 11:36

Der Diskussionsreigen zwischen uns wird so wohl sicher kein Ende nehmen 😞

Die IOPCIPrimaryMatch des ASUS Prime X299 Deluxe XHC onboard controllers ist "0xa2af8086". Der entsprechende AppleUSBXHCIPCI-Treiber is daher also nicht wie beim X99-A II ("0x8d318086") der "AppleUSBXHCILPTH" sondern der "AppleUSBXHCISPT" und den hab ich in meinem Kext auch eingebunden.

Nun aber was Versöhnliches zum Abschluss 😞

Soviel ich weiss verwendest Du noch immer SMBIOS macpro6,1. Ich verwende aber von Anfang an SMBIOS iMac17,1.... Vielleicht liegt hier die Divergenz in unseren Aussagen? 😊

Grüße

KGP

Beitrag von „DSM2“ vom 22. September 2017, 11:49

[@kgp](#) : Ja, das ist der Unterschied -> SMBios

Jedenfalls funktioniert der Kext von mir nicht, sobald ich iMac als Definition einstell.

Beitrag von „apfelnico“ vom 22. September 2017, 11:51

Sitze gerade am real MacPro im Schnitt, war aus dem Kopf. Du hast sicher recht mit dem konkreten Treiber, werde ich wohl auch so haben. Ändert aber nichts an der Kernaussage, dass alle Ports vorhanden sind, von Anfang an. Macht aber nichts, weil du alles mit Sicherheit

korrekt ansteuert und die verbleibenden Ports nicht nutzt. Ich wollte nur (auch für andere hier Mitlesende), dass nicht der Eindruck entsteht, das Board könne es nicht, die Ports würden nicht zur Verfügung stehen.

Edit: BIOS-Definition spielt eine Rolle, in dem Zusammenhang muss natürlich auch (zurück zum Thema) die hier besprochene Kext angepasst werden. Sonst greift ja eine der Voraussetzungen nicht zum erfolgreichen laden und andocken der nachfolgenden Kext an den Controller.

Edit2: Hat auch die korrekte Definition des LPC/LPC0/LPCB damit zu tun?

Nachdem ich hier auch eine Änderung, speziell zu "compatible" vorgenommen hatte, tauchte beim X99 auch der zweite EHCI auf, der vorher völlig ausgeblendet war.

Beitrag von „kgp-illacpro“ vom 22. September 2017, 12:19

[@DSM2](#)

Du sollst ja beim Asus X99-A II auch gar nicht SMBIOS iMac17,1 verwenden. Hier ist SMBIOS MacPro6,1 absolutes MUSS!!!

Ich verwende beim ASUS Prime X299 Deluxe das SMBIOS iMac17,1 weil letzterer auf Skylake basiert und daher die SMBIOS iMAC17,1 Definition auch mit ssdtPRGen.sh von Pike Alpha kompatibel ist und korrekt funktioniert. Auf Grund von SMBIOS iMac17,1 ist aber grad [@Brumbaer](#)'s Kext-Ansatz fundamental und überlebenswichtig, da es keine Implementierung von iMac17,1 in dem IOUSBHostFamily.kext oder AppleUSBXHCIPCI.kext gibt!

[@apfelnico](#) schwört auf XCPM ohne ssdtPRGen.sh und ssdt.aml von Pike Alpha, daher verwendet er wie bei den X99 Boards und bei den Broadwell-E/EP (Haswell-E/EP) Prozessoren auch bei Skylake-X/X299 weiterhin SMBIOS MacPro6,1. Aber ich will da nicht die nächste Diskussion lostreten 😊 Das wäre auch der falsche Thread dazu...

Gruß,

KGP

Beitrag von „DSM2“ vom 22. September 2017, 12:25

Das ich das SMBios nicht benutzen muss war mir von Anfang an klar und braucht mir gar nicht erläutert werden! Es ging bei meiner Aussage lediglich darum das die Adressierung AppleUSBXHCILPTH vs. AppleUSBXHCISPT SMBios spezifisch ist!!! Deshalb meine Anmerkung das der Kext nicht funktioniert wenn ich 17,1 anwähle weil eben für smbios des MacPro 6,1 geschrieben!

Beitrag von „apfelnico“ vom 22. September 2017, 12:33

Was ja nun tatsächlich ein ganz anderes Thema ist. Ich schwöre auch nicht drauf, es hatte konkrete Gründe. Prozessor ist das eine, Speicheranbindung und restliche Architektur eine andere. Speziell bei mir war es aber der ausschlaggebende extrem wichtige Punkt, eine bestehende Systemdefinition inkl. Seriennummer und darauf aufsetzender AppleID zu übernehmen. Daran wollte ich nicht rütteln. Das man dann XCPM auch einem System mit MBIOS MacPro6.1 beigegeben bekommt und die Frequenzvectors vom zum Beispiel iMac17 nimmt, ist doch möglich. Was ist daran nun lustig und was hat das hier zu suchen?

Beitrag von „kgp-imacpro“ vom 22. September 2017, 12:39

[@apfelnico](#), ich hab nur gesagt dass beim ASUS Prime X299 Deluxe unter SMBIOS iMac17,1 die Ports HS01, HS02, SSP1 und SSP2 nicht verwendet werden, auch nicht in der OSX USB Notfall-Treiberkonfiguration. Ich kann das leider nicht ändern... ist einfach so...

Lustig ist bei der ganzen Diskussion gar nichts... ich hatte den Smily nur angehängt um nicht wie gesagt noch eine weitere Diskussion loszutreten.

Gruß,

KGP

Beitrag von „apfelnico“ vom 22. September 2017, 12:52

Ah, das ist jetzt ein neuer Zusammenhang (SMBIOS), den ich vorher nicht hatte. Da ich immer noch am Schnittplatz hocke und weiter machen muss, kann ich das nicht weiter prüfen. Dann entschuldige, verstehe meinen Beitrag als allgemeines Veto (bezogen auf das Board könne es nicht), relativiere es auf, unter bestimmter SMBIOS mag es nicht. Wobei ich auch hier die Erfahrung gemacht habe, dass (und das zeigt ja auch dieses Kext eindrucksvoll) man zuzüglich zum SMBIOS durch weitere Beschreibungen die verwendete Hardware macOS begreiflicher macht. Egal ob nun per DSDT, zusätzlicher SSDT oder quasi live durch Clovers ACPI-Patches erledigt wird, oder durch "Injectoren".

Beitrag von „kgp-illacpro“ vom 22. September 2017, 13:24

[@Brumbaer](#)

kannst Du bitte sobald Du Zeit findest nach dem ganzen Hin und Her bitte meine einfache ursprüngliche Frage beantworten?

Vielen Dank im Voraus,

KGP