HACKINTOSH mit OpenCore und GIGABYTE Z390 DESIGNARE

ANLEITUNG zu JimSalabim's Thread im Hackintosh-Forum

https://www.hackintosh-forum.de/forum/thread/46701-gigabyte-z390-designare-opencore-efi-ordner-und-anleitung/

Ein paar Hinweise vorweg:

Beachtet bitte grundsätzlich, dass OpenCore nicht Clover ist und wir beispielsweise sämtliche Kexts, SSDTs, Drivers und Tools, die wir in die Ordner legen, grundsätzlich auch in der config.plist eintragen (und dort ggf. noch aktivieren) müssen.
Das ist in meinen Ordnern alles schon erledigt, aber es ist wichtig fürs Verständnis bzw. bei eventuellen Änderungen eurerseits immer zu beachten.

Wer von Clover auf OpenCore wechselt und schon die Clover-EFI-Ordner aus meinem anderen Thread verwendet hat, kann im Grunde einfach mit **Schritt 5** („EFI-Ordner auf SSD kopieren und weiter anpassen") anfangen.

Bitte beachtet auch die Hinweise im ersten Post des Threads!

Vorbereitung am echten Mac:

1. Schritt: macOS-Installationsstick erstellen:

USB-Stick (oder ext. Platte) mit mehr als (!) 8 GB Speicher mit dem Festplattendienstprogramm formatieren. Diesen Schritt bitte nicht überspringen, da hierbei die anschließend benötigte EFI-Partition automatisch im Hintergrund erzeugt wird.

Im Festplattendienstprogramm links oben auf „Darstellung" klicken und „Alle Geräte einblenden" wählen. Zum Formatieren dann anschließend das ganze Gerät auswählen (nicht nur die bereits vorhandene Partition, die unter dem Gerät angezeigt wird).

Name: **INSTALLER**
Format: macOS Extended (Journaled)
Schema: GUID-Partitionstabelle

Dann:
Catalina (https://apps.apple.com/de/app/macos-catalina/id1466841314?mt=12) laden.
Oder:
Big Sur (https://apps.apple.com/de/app/macos-big-sur/id1526878132?mt=12) laden.
Wenn das Installationsprogramm nach dem Download startet, dieses einfach wieder beenden.

Terminal öffnen und (für **Catalina**) folgendes in EINER Zeile eingeben:
```
sudo /Applications/Install\ macOS\ Catalina.app/Contents/
Resources/createinstallmedia --volume /Volumes/INSTALLER
```

für **Big Sur**:
```
sudo /Applications/Install\ macOS\ Big\ Sur.app/Contents/
Resources/createinstallmedia --volume /Volumes/INSTALLER
```

ACHTUNG: Beim Kopieren der Befehle aus der PDF-Datei kann es passieren, dass dort, wo hier formatierungsbedingt Zeilenumbrüche zu sehen sind, Leerzeichen eingefügt werden. Diese müssen entfernt werden. Es dürfen sich also keine Leerzeichen oder Umbrüche hinter „Resources/“ oder „Contents/“ befinden.

Nach Eingabe des Befehls Computerpasswort eingeben und mit Enter bestätigen, danach auf Aufforderung mit Y und Enter das Löschen des Volumes bestätigen und warten bis der Stick erstellt ist.

2. Schritt: EFI-Konfiguration auf den Stick kopieren:

Welcher EFI-Ordner für welche Konfiguration?

- Wenn man ohne externe GPU, sondern *nur mit iGPU* arbeitet:
  - **OC-0.6.X_Z390-Designare_nur-iGPU_iMac19,1**

- Wenn man ne Radeon-GPU hat:
  - Kommt drauf an, was einem wichtiger ist: DRM-Inhalte problemlos abspielen zu können oder Sidecar.*
    - DRM: **OC-0.6.X_Z390-Designare_Radeon-GPU_iMacPro1,1**
    - Sidecar: **OC-0.6.X_Z390-Designare_Radeon-GPU_iMac19,1**

Mit dem (bereits eingetragenen) Bootargument `shikigva=80` kann DRM unter **Catalina** nun auch mit dem iMac19,1 SMBIOS vollständig funktionieren (hängt von der Hardware ab). Sollte es beim Abspielen von DRM-Inhalten zu Freezes kommen, bitte stattdessen `shikigva=16` benutzen (Einschränkungen möglich).
Mit dem iMacPro1,1-SMBIOS funktioniert DRM von Haus aus vollständig, hier werden keine der beiden genannten shikigva-Bootargumente benötigt.
Unter Big Sur funktioniert DRM mit dem iMac19,1-SMBIOS selbst mit shikigva-Bootargument NICHT. Mit iMacPro1,1 funktioniert DRM auch hier wie gewohnt.

Kleine Info: Ich verwende selbst das iMac19,1-SMBIOS, da mir Sidecar wichtig ist. Beim Arbeiten mit der GPU merke ich keinen spürbaren Leistungsunterschied gegenüber dem iMacPro1,1-SMBIOS (bei dem die GPU eigentlich noch besser ausgenutzt wird, da hier die iGPU überhaupt nicht mitarbeitet – selbst wenn sie im Bios aktiviert ist). DRM funktioniert bei mir mit shikigva=80 perfekt. Ich benutze eine Asus ROG Strix Vega 64 OC.

Weiter gehts!

EFI-Partition des USB-Sticks mounten:
Auf welche Weise man die EFI-Partition mountet, ist egal. Beispielsweise übers Terminal mit
`diskutil list EFI`
den Identifier für die EFI-Partition des USB-Sticks ausfindig machen (z. B. *disk3s1*), dann mit folgendem Befehl mounten:
`sudo diskutil mount disk3s1`
(bitte *disk3s1* mit dem bei euch zutreffenen Identifier ersetzen!)

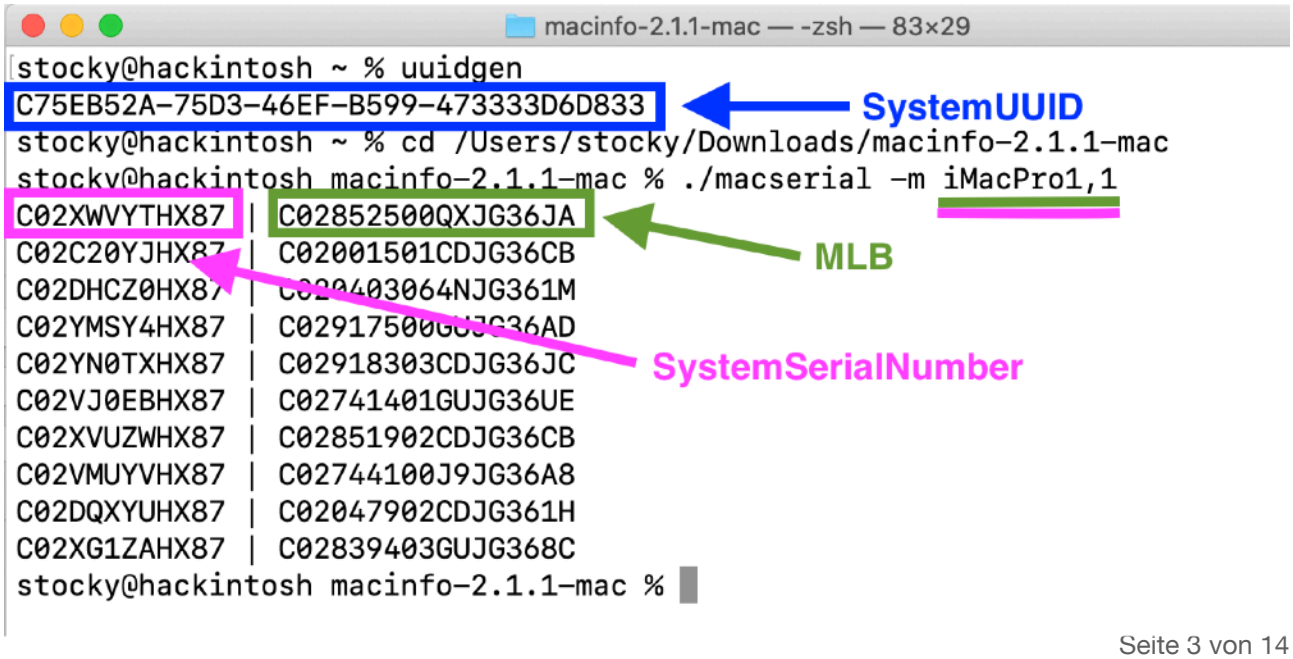
Nun bitte meinen Ordner „EFI“ sowie die Ordner „Docs“ und „Utilities“ und bei Bedarf auch den Ordner „BIOS“ auf die soeben gemountete **EFI-Partition** des Sticks kopieren (**nicht** auf die „Install macOS …“-Partition). Sollte da schon ein EFI-Ordner drauf sein, diesen einfach ersetzen.

Benötigte Nummern generieren:

Wir öffnen das Terminal und generieren zunächst eine SystemUUID, die wir uns notieren. Der Befehl hierfür lautet:
`uuidgen`

Anschließend nutzen wir ***macserial*** aus dem Utilities-Ordner auf der EFI-Partition des Sticks. Wir navigieren wir mit dem Terminal in den Ordner, indem wir im Terminal folgendes eingeben:
`cd /Volumes/EFI/Utilities/macserial`

Danach geben wir `./macserial -m` ein, gefolgt vom Modellnamen, der dem EFI-Ordner entspricht, den wir gewählt haben. Also beispielsweise:
`./macserial -m iMacPro1,1`
oder eben:
`./macserial -m iMac19,1`
Es werden uns einige Seriennummern und MLBs generiert. Davon wählen wir ein zusammengehöriges Paar aus und notieren uns die Nummern.
Die Ausgabe sollte etwa so aussehen:



Die Datei **config.plist** im Ordner EFI/OC auf der EFI-Partition des Sticks mit einem geeigenten Plist-Editor (z. B. *PLIST Editor*, *Plist Edit Pro* oder *Xcode*) öffnen. Verwendet hier bitte **NICHT** den *OpenCore Configurator* (und den *Clover Configurator* natürlich erst recht nicht)!
Ich verwende die App *PLIST Editor*, die im AppStore sehr günstig zu haben ist und im Gegensatz zu z. B. *Xcode* sehr praktische Funktionen wie beispielsweise die Möglichkeit des rekursiven Aus- und Einklappens aller Unter-Items eines Eintrags bietet. Das macht die Bearbeitung sehr übersichtlich.

Die soeben generierte SystemSerialNumber, MLB und SystemUUID nun bitte unter „PlatformInfo“ -> „Generic“ in die entsprechenden Felder der config.plist eintragen. Das Feld für „ROM“ passen wir später noch an.

PlatformInfo	Dictionary	7 items
Automatic	Boolean	☑
CustomMemory	Boolean	☐
Generic	Dictionary	9 items
AdviseWindows	Boolean	☐
SystemMemoryStatus	String	Auto
MLB	String	HIER_DEINE_MLB_EINTRAGEN
ProcessorType	Number	0
ROM	Data	<>
SpoofVendor	Boolean	☑
SystemProductName	String	iMac19,1
SystemSerialNumber	String	HIER_DEINE_SERIENNUMMER_EINTRAGEN
SystemUUID	String	HIER_DEINE_SMUUID_EINTRAGEN
UpdateDataHub	Boolean	☑
UpdateNVRAM	Boolean	☑
UpdateSMBIOS	Boolean	☑
UpdateSMBIOSMode	String	Custom

Hinweis:
Da viele eine Radeon RX 5700 (XT) Grafikkarte benutzen, habe ich unter „NVRAM“ --> „Add“ --> „7C436110-AB2A-4BBB-A880-FE41995C9F82“ --> „boot-args“ folgendes Bootargument eingetragen:
`agdpmod=pikera`
Wer **keine** RX 5700 (XT) oder RX 5600 (XT) verwendet (also keine GPU vom Typ Navi 10), kann dieses Bootargument entfernen.

Je nach Bildschirm, den man verwendet, lässt man den Eintrag für „UiScale“ unter „NVRAM“ --> „Add“ --> „4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14“ auf `<02>` oder stellt ihn auf `<01>`.
`<01>` ist für „normale“ Monitore, `<02>` für HiDPI-Monitore. Diese Einstellung hat auch Einfluss auf die Größe des Apfels beim Systemstart.

Nun wechseln wir zum Hackintosh und gehen zunächst ins BIOS.

3. Schritt: BIOS einstellen

Version F9i:
Den Ordner „BIOS“ kann man einfach auf einer FAT-formatierten Partition ablegen (zum Beispiel auf der EFI-Partition des USB-Sticks direkt neben dem EFI-Ordner).
Dann macht man über die Q-Flash-Utility im BIOS das BIOS-Update auf Version F9i. Dort den entsprechenden Drive, den Ordner „BIOS“ und darin den Ordner „mb_bios_z390-designare_f9i“ auswählen, und los gehts.

Nun Settings einstellen:

BIOS-SETTINGS:
„Load Optimized Defaults“ wählen, dann folgende Änderungen vornehmen
(oder stattdessen unter „Save & Exit“ meine Datei aus dem Ordner „BIOS“ als Profil laden, dann ist fast alles folgende bereits erledigt. Trotzdem bitte nochmal durchchecken.)

F2 drücken (Advanced Mode)

TWEAKER:
Extreme Memory Profile(X.M.P.): Profile 1
Advanced CPU Settings:
VT-d: Enabled oder Disabled (je nachdem, ob man das in einem anderen Betriebssystem braucht)
SETTINGS:
Platform Power:
Platform Power Management: Enabled
PEG ASPM: Enabled
PCH ASPM: Disabled
DMI ASPM: Enabled
ErP: Enabled (kein Strom auf den USB-Ports bei ausgeschaltetem Rechner) oder Disabled (das Gegenteil eben ;-)), ist Geschmackssache
IO Ports:
Initial Display Output: PCie 1 Slot
(es sei denn, die Grafikkarte steckt in einem anderen Slot).
Wenn man **keine** Grafikkarte hat und nur die iGPU verwendet, stattdessen *IGFX* auswählen!
Internal Graphics: Enabled (bei iMac19,1 SMBIOS) oder ggf. Disabled (bei iMacPro1,1) (wenn man Final Cut Pro X und/oder Compressor benutzt, sollte man hier auch beim iMacPro1,1-SMBIOS Enabled wählen, da sonst die GPU in diesen Programmen nicht alle Aufgaben übernimmt, die sie soll (obwohl die iGPU unter iMacPro1,1 dennoch nicht arbeitet). Die Gründe sind unklar, aber es ist so. Es scheint sich um einen Bug in diesen Programmen zu handeln.
Falls Probleme auftreten oder man Final Cut oder Compressor nicht benutzt, auf Disabled stellen (gilt wie gesagt **nur** für den Ordner mit iMacPro1,1 SMBIOS)

Thunderbolt Configuration:
Discrete Thunderbolt(™) Support: Enabled
TBT Vt-d base security: Disabled
Thunderbolt Boot Support: Disabled
Wake From Thunderbolt(™) Devices: Enabled
Security Level: No Security
Discrete Thunderbolt(™) Configuration:
Thunderbolt Usb Support: Enabled
GPIO3 Force Pwr: Enabled
USB Configuration:
Legacy USB Support: Enabled
XHCI Hand-off: Enabled
USB Mass Storage Driver Support: Enabled
Port 60/64 Emulation: Enabled oder Disabled
SATA And RST Configuration:
SATA Mode Selection: AHCI
Smart Fan 5:
Die Lüftersteuerung (Smart Fan) bitte nach euren Bedürfnissen einrichten. Bei mir sind alle Lüfter auf „Manual“ mit eigenen Lüfterkurven eingestellt, die leiser sind als die Silent-Einstellung. Außer beim CPU-Lüfter und der Pumpe hab ich bei allen Lüftern „VRM MOS“ als Sensor ausgewählt.
BOOT:
CFG Lock: Disabled
Windows 8/10 Featues: Windows 8/10 WHQL
CSM Support: Disabled. Auf Enabled stellen, wenn das Boot-Menü nicht in der richtigen Auflösung bzw. verzerrt oder gar nicht angezeigt wird.
Preferred Operating Mode: Advanced Mode

4. Schritt: macOS installieren:

Netzwerkkabel für die Internetverbindung vorläufig in den **unteren** Ethernet-Anschluss des Mainboards stecken.
Rechner von der EFI-Partition des USB-Sticks booten (beim Starten mehrfach F12 drücken, dann den Stick wählen). Anschließend im OpenCore-Menü die Installations-Partition starten („Install macOS Catalina“). Danach geht man im Installationsprogramm zunächst über die Dienstprogramme ins Festplattendienstprogramm und formatiert die zukünftige System-SSD als APFS. Anschließend installiert man macOS auf eben dieser SSD und bootet anschließend (hoffentlich erfolgreich) das System.

5. Schritt EFI-Ordner auf SSD kopieren und weiter anpassen:

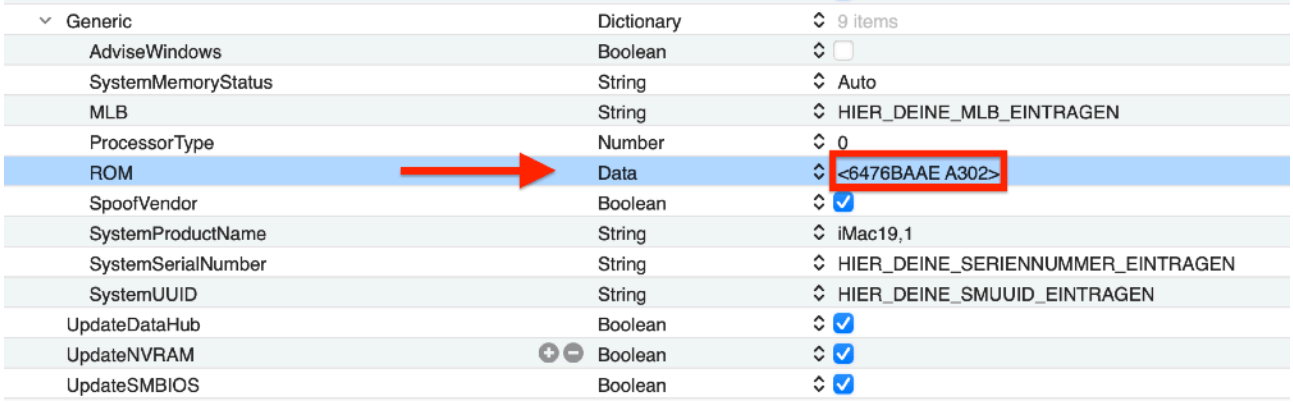
EFI-Partition des USB-Sticks auf dem Hackintosh mounten.
Nun den EFI-Ordner auf den Rechner kopieren (beispielsweise auf den Schreibtisch oder in den Downloads-Ordner) und die EFI-Partition wieder auswerfen (`diskutil unmount EFI`).
Jetzt den USB-Stick vom Rechner entfernen und die EFI-Partition der System-SSD mounten. Terminal:
`diskutil list EFI`
Identifier des EFI-Ordners der SSD notieren (z. B. *disk0s1*)
`sudo diskutil mount disk0s1`
(Identifier wieder entsprechend ersetzen)

Nun den EFI-Ordner (falls dort schon einer vorhanden ist) auf der EFI-Partition der System-SSD löschen und stattdessen den EFI-Ordner drauf kopieren, den wir gerade vom Stick auf den Rechner kopiert haben.

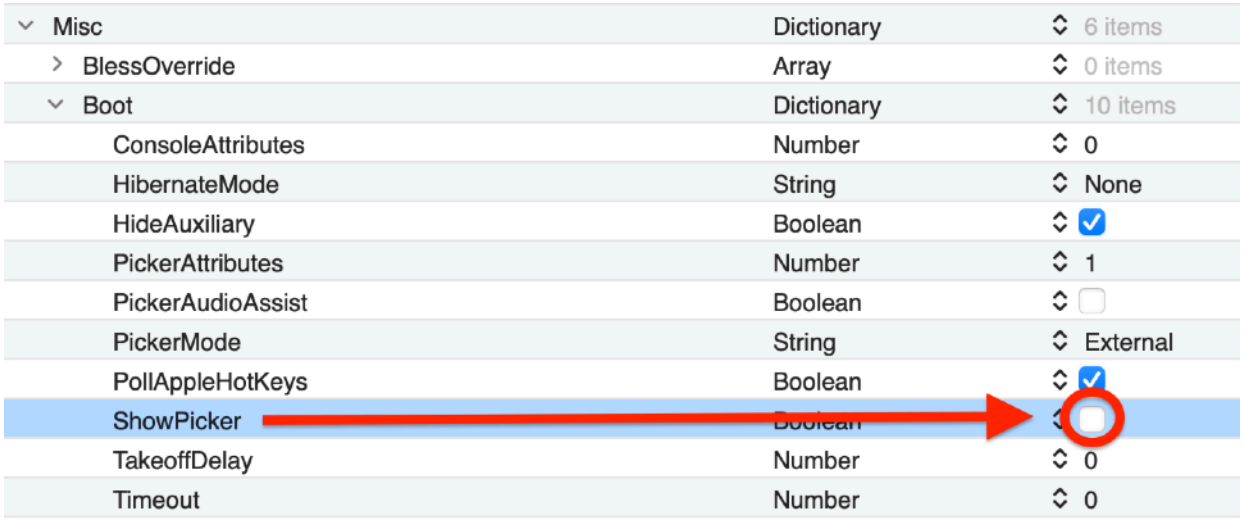
Wir vergewissern uns, dass unser Netzwerkkabel nach wie vor im **unteren** Netzwerkanschluss steckt.
Jetzt öffnen wir das Terminal und geben ein:
`ifconfig en0 ether`

Wir notieren uns die MAC-Adresse, die direkt hinter „ether“ steht.
Ein anderer Weg, die MAC-Adresse herauszufinden, ist über den Apfel in der Menüleiste auf „Über diesen Mac“ zu klicken, dann auf „Systembericht“ und dort direkt auf „Netzwerk“ zu klicken. Dort wählen wir den Dienst mit dem BSD-Gerätenamen „en0“ (in der Regel sollte der Dienst einfach „Ethernet“ heißen). Dann findet man weiter unten auch den Eintrag für die MAC-Adresse.

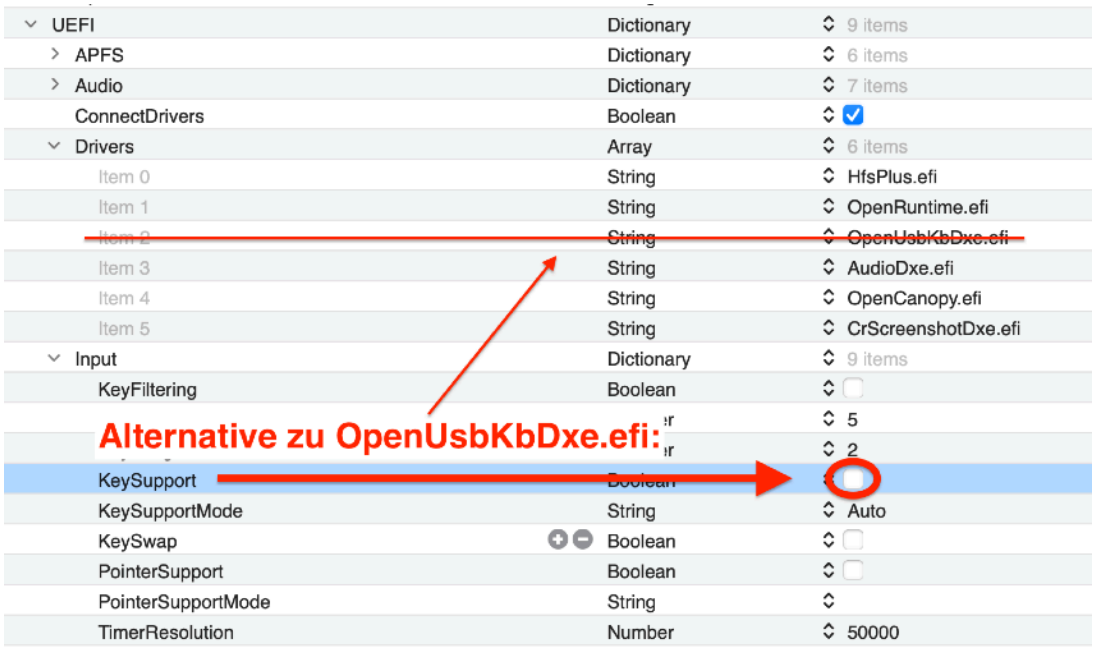
Nun die **config.plist** des neuen EFI-Ordners auf der EFI-Partition der System-SSD mit einem Plist-Editor öffnen (ich wiederhole: bitte **nicht** mit dem OpenCore Configurator oder gar dem Clover Configurator):
Unter „PlatformInfo“ -> „Generic“ tragen wir nun noch bei „ROM“ die MAC-Adresse **zwischen** den <>-Zeichen ein, lassen dabei alle Doppelpunkte weg und fügen ein Leerzeichen vor den letzten vier Stellen ein. Das Ergebnis sollte nach fertiger Eingabe dann zum Beispiel so aussehen:



Wer von Clover auf OpenCore wechselt und daher die Schritte 1–4 übersprungen hat, trägt hier selbstverständlich auch noch seine MLB (entspricht der Board Serial Number), seine Seriennummer und seine SmUUID ein.
Wer will, dass das OpenCore-Menü nicht auftaucht und stattdessen direkt von der Hackintosh-SSD gebootet wird, entfernt (je nach verwendetem Plist Editor) unter „Misc“ --> „Boot“ entweder den Haken bei „ShowPicker“ oder setzt den Eintrag von „YES“ auf „NO“ bzw. von „true“ auf „false“. Will man trotzdem ins OpenCore-Menü, hält man, während das Bios-Logo angezeigt wird, einfach die Alt-Taste gedrückt, bis das Menü erscheint. Verwenden wir eine Bluetooth-Tastatur, kann es sein, dass wir die Alt-Taste mehrfach drücken müssen (beispielsweise immer eine Sekunde oder etwas länger drücken, dann wieder loslassen, dann wieder drücken etc.)



Bei Problemen mit der Tastatur im OpenCore-Menü kann man folgenes probieren: „KeySupport” unter UEFI --> Input aktivieren und gleichzeitig den Eintrag „OpenUsbKbDxe.efi“ unter UEFI --> Drivers löschen.
Beim einen funktioniert OpenUsbKbDxe.efi besser (bei mir zum Beispiel, sonst funktionieren bei mir die Mac-typischen Hotkeys für NVRAM-Reset, Recovery Mode etc. beim Boot nicht), beim anderen funktioniert KeySupport besser.



Falls wir beobachtet haben, dass das Apfel-Logo, das während des macOS-Startvorgangs zweimal hintereinander erscheint, beim ersten Mal eine andere Größe hat als beim zweiten Mal (also so wie es bei Clover in der Regel zu beobachten ist), können wir ausprobieren, ob sich das beheben lässt, wenn wir den Eintrag für „UiScale“ unter „NVRAM“ --> „Add“ --> „4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14“ statt `<02>` auf `<01>` stellen oder umgekehrt. Richtig ist es, wenn der Apfel immer dieselbe Größe hat. Bei meinem 4K-Monitor ist `<02>` (also wie in meiner config.plist schon voreingestellt) die korrekte Einstellung.

Datei speichern, Rechner neu starten.

6. Schritt: Boot-Reihenfolge im BIOS anpassen:

Die Hacktintosh-SSD, auf die wir vorhin den EFI-Ordner kopiert haben (z. B. „UEFI OS (Samsung SSD 960 EVO 1TB“) sollte im BIOS unter „Boot“ bei den „Boot Option Priorities“ an erster Stelle stehen.

7. Schritt: USB-Stick entfernen, booten, FERTIG (zumindest mit dem Hauptteil für macOS).

— Zwischenstopp: WICHTIGE HINWEISE:

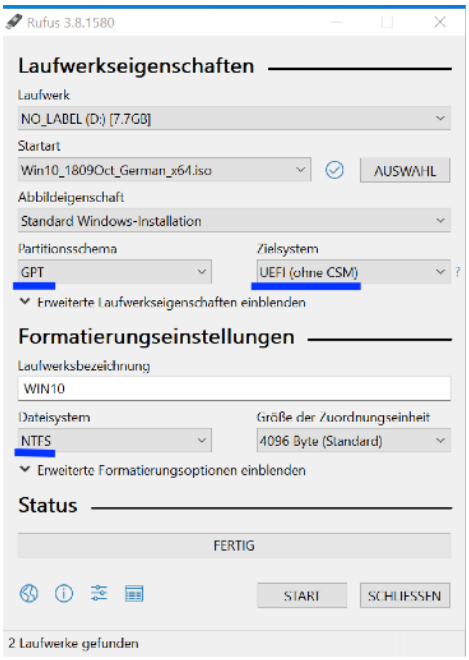
Standard-Boot-Volume: Wir wählen in den macOS-Systemeinstellungen unter „Startvolume“ einmalig unsere macOS-SSD aus (ein Neustart ist nicht erforderlich, auswählen reicht), damit sie das Standard-Boot-Volume für OpenCore wird, aus. Diese Einstellung bleibt erhalten, bis wir einen NVRAM-Reset machen. Alternativ kann man macOS auch als Standard-Boot-Volume festlegen, indem man es im OpenCore-Menü einmalig mit gedrückter Ctrl-Taste startet. Auch hier bleibt die Einstellung bis zum nächsten NVRAM-Reset oder der nächsten Änderung erhalten.

Thunderbolt:
Der Thunderbolt-Controller taucht in den macOS-Systeminformationen unter „PCI“ auf, nicht unter „Thunderbolt“.
Im Gegensatz zum echten Mac sind die Thunderbolt-Ports bei Hackintoshes grundsätzlich nicht netzwerkfähig und auch bei RAIDs kann es ggf. Probleme geben, aber da bin ich nicht genau im Bilde. Die meisten Funktionen und die meisten Mac-kompatiblen Thunderbolt-2- und 3-Geräte funktionieren jedoch einwandfrei.

Updates:
Für Hinweise und Anleitungen zu Updates etc. bitte immer in meinen Thread schauen. Ich aktualisiere den ersten Post dort regelmäßig und lade nach OpenCore-Updates etc. aktualisierte EFI-Ordner hoch.

8. Schritt: Dual Boot mit WINDOWS 10 und Menü-Anpassung

Windows-Installation



Um Windows 10 im UEFI-Modus zu installieren, ist es empfehlenswert, sich einen Install-Stick beispielsweise mit dem Tool „Rufus“ zu erstellen:
https://rufus.ie
Hierfür wählen wir als Partitionsschema „GPT“ und als Zielsystem „UEFI (ohne CSM)“.
ACHTUNG: Während der Windows-Installation sind alle SSDs und Platten außer der Windows-SSD **unbedingt** vom Mainboard zu trennen! Windows schreibt uns sonst mit hoher Wahrscheinlichkeit entweder auf unserer EFI-Partition rum und wählt nicht das richtige Volume für seine eigene EFI-Partition – und das wollen wir unbedingt vermeiden – oder lässt sich gar nicht erst installieren.
Also alles abstöpseln, was nicht Windows ist und warten, bis die Installation erledigt ist. Anschließend kann man alles wieder anschließen.

In den „Boot Option Priorities“ im BIOS setzen wir nun wieder die macOS-SSD an die erste Stelle und löschen den Eintrag für den Windows Boot Manager raus.

OPTIONAL: Custom-Eintrag für Windows erstellen und OpenCore-Menü anpassen

Wer mögliche weitere EFI-Einträge im Boot-Menü nicht sehen möchte (was sinnvoll ist, da sie sich von OpenCore aus oft gar nicht starten lassen), kann gemäß der Beschreibung in der *Configuration.pdf* (im Ordner Docs) den Wert für ScanPolicy (in der config.plist unter „Misc“-->„Security“) entsprechend anpassen. Den einzutragenden Wert kann man sich selbst ausrechnen. Ich möchte beispielsweise keinen Device Lock drin haben (damit mir auch USB-Sticks, externe Festplatten etc. angezeigt werden, die sich über OpenCore booten lassen) und die Anzeige der Dateisysteme APFS, HFS, NTFS und EXT erlauben, aber eben keine ESP-Dateisysteme (also eben EFI-Partitionen an sich). Dafür gebe ich als ScanPolicy ein:

```
6913
```

Das setzt sich beispielsweise zusammen aus:
OC_SCAN_FILE_SYSTEM_LOCK (0x1)
OC_SCAN_ALLOW_FS_APFS (0x100)
OC_SCAN_ALLOW_FS_HFS (0x200)
OC_SCAN_ALLOW_FS_NTFS (0x800)
OC_SCAN_ALLOW_FS_EXT (0x1000)
Addiert man die Hex-Werte zusammen, ergibt das *0x1B01*. Umgerechnet in eine Dezimalzahl ist das eben der gesuchte Wert *6913.*

Da wir OC_SCAN_ALLOW_FS_ESP (also EFI-File-Systems) nicht für die Anzeige aktiviert haben, erscheint nun leider auch Windows nicht mehr im Bootmenü (der Eintrag für NTFS bringt uns hier also eigentlich gar nichts, da die boot.efi von Windows sich auf der

Windows-EFI-Partition befindet, nicht auf der NTFS-Partition). Mit ausgeschalteter ScanPolicy erscheinen natürlich alle Einträge (womöglich aber eben auch unerwünschte). Aber natürlich gibt es einen **Workaround**: Wir legen einfach einen Custom-Eintrag für Windows in der config.plist an. Ein Vorteil ist hier außerdem, dass der Windows-Eintrag damit nicht als erster Eintrag im Menü erscheinen kann, da die Custom-Einträge erst hinter den automatisch erkannten erscheinen.

Im OpenCore-Menü drücken wir die Leertaste, navigieren zum Eintrag für die **UEFI Shell** und starten diese durch Betätigung der Return- oder Enter-Taste.

Wir drücken kurz eine beliebige Taste oder warten ein paar Sekunden, bis wir zur Eingabe gelangen.

Wir müssen jetzt eine andere Tastaturbelegung beachten:
Doppelpunkt: **Umschalt** (also Shift) + **Ö**
Größer-Zeichen (**>**): **Umschalt** + **Punkt**
Backslash (\): **#** (Raute-Zeichen)
y: **z**

Wir suchen zunächst unsere OpenCore-EFI-Partition.
Wir geben ein:
```
ls fs1:EFI
```
und lassen uns so den Inhalt des EFI-Ordners auf „fs1“ anzeigen (sofern dort einer vorhanden ist).
Wir machen weiter mit
```
ls fs2:EFI
ls fs3:EFI
```
usw., bis uns der Inhalt unseres OpenCore-EFI-Ordners angezeigt wird, also die Ordner „BOOT“ und „OC“.
Fehlermeldungen bei den anderen Volumes, dass der Pfad oder Ordner nicht existiert oder ähnliches, ignorieren wir einfach. Auf diese Weise identifizieren wir die richtige Partition aber am schnellsten.

Nun wechseln wir auf genau dieses Volume (in meinem Fall ist es „fs8“), indem wir eingeben:
```
fs8:
```
(Ihr gebt natürlich stattdessen die Nummer ein, bei der bei euch der Inhalt der OpenCore-EFI-Partition angezeigt wurde).
Jetzt lassen wir uns zunächst die Map mit allen Pfaden als Textdatei auf unsere OpenCore-EFI-Partition schreiben:
```
map > map.txt
```

Anschließend suchen wir unsere **Windows-EFI-Partition**:
```
ls fs1:EFI
ls fs2:EFI
```
usw., so lange, bis uns die Ordner „Microsoft“ und „Boot“ angezeigt werden. Auch hier gilt: Fehlermeldungen bei einigen anderen Volumes, dass der Pfad oder Ordner nicht exisitiert oder ähnliches, ignorieren wir einfach.

Wir notieren uns das Volume, bei der die genannten beiden Ordner gelistet werden. Bei mir ist das zum Beispiel fs5. Wir schreiben uns auf, dass es sich dabei um die Windows-EFI-Partition handelt.

Dann geben wir ein:
`exit`
und verlassen damit die Shell.
Dann starten wir macOS und mounten die EFI-Partition. Dort befindet sich nun unsere Datei „**map.txt**“, die wir vorhin erzeugt haben.
Wir öffnen sie mit TextEdit und öffnen daneben auch die **config.plist** mit dem Plist Editor unserer Wahl.

Unter „Misc“ --> „Entries“ machen wir nun folgenden Eintrag:

Wir kopieren aus der map.txt den PciRoot-Pfad des File Systems, das wir als Windows-EFI identifiziert haben. Und fügen ihn bei „Path“ ein. Dahinter kommt direkt:
`/\EFI\Microsoft\Boot\bootmgfw.efi`
Der vollständige Windows-Pfad könnte also zum Beispiel lauten:
`PciRoot(0x0)/Pci(0x17,0x0)/Sata(0x2,0xFFFF,0x0)/HD(2,GPT,9A5BA866-EBC5-47F5-9BDD-9A75E30139F2,0xFA000,0x32000)/\EFI\Microsoft\Boot\bootmgfw.efi`

So sähe der vollständige Eintrag in einem Standard-Texteditor aus:

```
<key>Entries</key>
<array>
    <dict>
        <key>Arguments</key>
        <string></string>
        <key>Auxiliary</key>
        <false/>
        <key>Comment</key>
        <string>Not signed for security reasons</string>
        <key>Enabled</key>
        <true/>
        <key>Name</key>
        <string>Windows</string>
        <key>Path</key>
        <string>DEIN-PciROOT-PFAD/\EFI\Microsoft\Boot\bootmgfw.efi</string>
    </dict>
</array>
```

Letzte Schritte:
Ich empfehle nun noch, wie oben schon erwähnt, den Haken bei „Misc“ --> „Boot“ --> „ShowPicker“ zu entfernen (bzw. den Eintrag auf „NO“/„false“ zu setzen).
Anschließend legen wir in den macOS-Systemeinstellungen unter „Startvolume“ noch die macOS-Platte fest. Dann können wir den Rechner neu starten.

Wenn das Gigabyte-Logo erscheint, drücken wir die Alt-Taste und halten sie gedrückt (bzw. drücken mehrfach), bis das OpenCore-Menü erscheint.

Weitere nützliche Hinweise:
Ist die macOS-Platte mal nicht als Startvolume festgelegt (beispielsweise nach einem NVRAM-Reset!) oder haben wir ein anderes Startvolume festgelegt, können wir nach Erscheinen des Gigabyte-Boot-Logos auch die X-Taste gedrückt halten (bzw. mehrfach drücken), bis macOS von selbst bootet.

Einen **NVRAM-Reset** können wir mit der Tastenkombination **Cmd**+**Alt**+**P**+**R** machen
Auch hier einfach die Tasten gedrückt halten, wenn das Bios-Logo erscheint, und so lange gedrückt lassen, bis der Rechner neu startet und das Bios-Logo erneut erscheint.
(Achtung: funktioniert mit Bluetooth-Tastatur ggf. nicht – hier ist bei Bedarf stattdessen der NVRAM-Reset-Eintrag im OpenCore-Menü zu wählen. Dieser und weitere Tools erscheinen nach Drücken der Leertaste im Menü).
Achtung: Nach einem NVRAM-Reset (der wirklich nur selten notwendig ist), sollten wir ins BIOS gehen (mehrfaches Drücken der Entf.- oder Backspace-Taste) und dort die Boot Option Priorities erneut checken. Diese können nach dem NVRAM-Reset evtl. durcheinander geraten. Der Windows-Boot-Manager-Eintrag kann dabei auch wieder komplett entfernt werden, wenn wir vorhaben, Windows sowieso über OpenCore zu booten.

Checkliste für EFI-Ordner-Updates:

Sichert euren alten EFI-Ordner auf dem Schreibtisch oder einer Backup-EFI-Partition und ersetzt die Ordner „Docs“, „EFI“ und „Utilities“ vollständig. Es ist nicht empfehlenswert, die alte config.plist weiterzuverwenden, weil mit den OpenCore-Updates auch immer wieder neue Funktionen hinzukommen und alte entfernt werden, die sich in der config.plist entsprechend widerspiegeln. Man kann natürlich auch die neue config.plist mit der alten Eintrag für Eintrag vergleichen und die alte entsprechend anpassen und neue Einträge ergänzen sowie alte entfernen, aber es ist einfacher und überschaubarer, die neue zu benutzen und einfach folgende Schritte vorzunehmen:

- Notiert euch eure MLB, ROM, SystemSerialNumber und SystemUUID aus der alten config.plist („Platform Info“ --> „Generic“).
  Tragt diese in der neuen config.plist wieder ein.
- Kopiert euch eure Pfade und ggf. weitere Anpassungen für die Custom Entries unter „Misc“ --> „Entries“ und tragt diese in der neuen config.plist ein.
- Überprüft, ob der „UIScale“-Eintrag („NVRAM“ --> „Add“ --> „4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14“) mit eurem übereinstimmt und passt ihn ggf. an.
- Überprüft, ob eure Bootargumente unter „NVRAM“ --> „Add“ --> „7C436110-AB2A-4BBB-A880-FE41995C9F82“ --> „boot-args“ mit denen in der neuen config.plist übereinstimmen und passt sie ggf. an.

Viel Erfolg und liebe Grüße
JimSalabim