

Anleitung: SIERRA/HIGH SIERRA auf einem ASROCK Z270 SUPER CARRIER Motherboard

Beitrag von „Mork vom Ork“ vom 20. August 2017, 17:50

Allen Interessierten zeige ich Euch hier mal meine neueste Errungenschaft in Sachen Hackintosh und erkläre Euch Schritt-für-Schritt, wie ich bei der Installation vorgegangen bin.

Fangen wir also mal bei der Hardware an. Folgende Sachen werden von mir in diesem Tutorial verbaut:

- 1x ASRock Z270 Super Carrier Motherboard
- 1x Corsair H115i Wasserkühlung
- 1x Intel i7 SKYLAKE CPU (4GHz)
- 4x Corsair 3200Mhz. 8GB RAM Module
- 1x Sapphire RX480 NITRO OC Grafikkarte
- 1x SAMSUNG 950 PRO NVMe M2. 256 GB SSD
- 1x SAMSUNG 960 EVO NVMe M2. 256 GB SSD
- 1x SAMSUNG MZVPV NVMe M.2 256 GB SSD
- 1x NZXT IU01 interner USB-Hub
- 1x 5.25" LG Sata BluRay Brenner
- 1x 5.25" Einschub-Hub mit 2x USB2 und 2x USB3 Anschlüssen

Ich werde in diesem Tutorial nicht jede einzelne Hardware vorstellen, sondern nur die

wichtigsten Komponenten, wie zum Beispiel das Motherboard: ASRock Z270 Super Carrier:



Warum gerade dieses Board?

Weil es

- a) über 4 PCIe x16 Slots verfügt, wovon 2 (bei entsprechender Belegung) auch wirklich mit x16 angesteuert werden
- b) über 3 interne M.2 Steckplätze verfügt, die jeweils mit vollen 4 Lanes angebunden sind

- c) über 2 Thunderbolt3 Ports am Backplane verfügt
- d) über 2 Intel 1GBit Ethernetports verfügt
- und e) sein BIOS vollständig AMI-BCP-kompatibel ist (dazu komme ich später noch genauer)

Angenehm auch das interne WIFI und Bluetooth, welches ich hier unter OS X aber nicht nutze und deren Installation auch nicht Bestandteil dieses Tutorials sind.

Warum als CPU nur einen Intel i7 6700K Skylake Prozessor?

Weil diese bereits von einem ASUS Z170 Maximus VIII Extreme Board von mir übernommen wurde und ich für mich keine Notwendigkeit (und auch nicht mehr das dazu nötige Kleingeld) sehe, auf diese Generation umzusteigen.

Wie sieht es mit der CORSAIR Wasserkühlung aus - ist diese über OS X steuerbar?

NEIN, sie wird von mir einzig und allein unter WINDOWS10 konfiguriert und gesteuert. Aber einmal sauber unter WIN10 konfiguriert, besitzt sie die selben Kühleigenschaften auch unter macOS.

Was hat es mit dem BIOS und AMI-BCP auf sich?

Dazu komme ich jetzt. Hat seinen Ursprung darin, das ich a) nicht nur Treiber und Software meiner im Einsatz befindlichen Systeme gerne auf dem neuesten Stand halte, sondern eben auch das jeweilige BIOS meiner Hardware.

Und genau hier lag bislang immer der Hase im Pfeffer: wann immer ich das BIOS meines PCs zurücksetzen musste (Werkseinstellung), musste ich anschliessend immer erst ins BIOS, um die für macOS optimalen Werte wieder einzustellen. Ich hatte bislang GIGABYTE, ASUS und ASRock-Boards - und keines der genannten BIOSe war bislang mit AMI-BCP kompatibel. Ich konnte das jeweilige BIOS zwar im AMI-BCP öffnen und auch modifizieren, doch wann immer ich die Modifizierung gespeichert habe, ist AMI-BCP abgestürzt, ohne die Modifikation gespeichert zu haben. Das BIOS des ASRock Super Carrier ist das bislang erste und einzige mir bekannte AMI V5 BIOS, welches sich vollständig mit AMI-BCP modifizieren lässt.

Wozu ist die Bearbeitung mit AMI-BCP sinnvoll?

Damit kann ich diverse voreingestellten BIOS Werte so einstellen, daß sie nach einem BIOS-Reset auf Werkseinstellungen gleich so gesetzt werden, wie ich sie für einen sauberen Betrieb unter WINDOWS und macOS benötige.

Desweiteren kann ich mit Hilfe dieses Programs Features "freischalten", welche für den

Enduser normalerweise nicht freigeschaltet sind. Ich zeige Euch das mal anhand der offiziellen Thunderbolt-BIOS-Einstellungen zu diesem Board.

Unter Nutzung des offiziellen BIOS von der Herstellerseite sehen die Einstellungen im BIOS für Thunderbolt wie folgt aus:



Die einzige Einstellmöglichkeit, die es hier gibt, ist der "Security Level":



Reichlich dürftig die Einstellmöglichkeiten, wenn man bedenkt, daß das BIOS sehr viel mehr

Einstellmöglichkeiten bietet, sofern man diese via AMI-BCP freischaltet:



und es geht noch weiter. Ich habe hier nicht alle Einstellungen freigeschaltet, sondern nur die

für mich wichtigsten:



Und so zieht es sich durch beinahe alle Einstellungen. Und nach diversen Tests habe ich dann die für mich passenden Einstellungen gefunden und als Standard-Werkseinstellungen festlegen können.

So habe ich zum Beispiel folgende Einstellungen von standard "disabled" für meine Zwecke auf "enabled" setzen können - und wann immer ich nun mein BIOS mittels Reset-Taster auf Werkseinstellungen

zurücksetzen muss, weiss ich, das folgende Werte standardmässig gesetzt sind:

- VT-d = **disabled**
- Intel Virtualization = **disabled**
- Memory Profile = **XMP 2.0**
- USB EHCI/XHCI Hand-off = **enabled**
- Serial Port = **disabled**
- Onboard HD-Audio = **disabled** (da ich nur den HDMI-Port der RX480 für AUDIO an meinen DENON-AV Receiver nutze)

- WIFI = **disabled** (nutze WIFI bei mir in der Wohnung nur für mein iPhone, Computer sind alle via Ethernet am Netzwerk)
 - Bluetooth = **disabled**
 - CSM = **disabled**
 - primärer Grafikadapter = **PCI-Express**
- etc, etc.

Wann immer ich also in den Eingeweiden des BIOS rumfricke und dabei etwas so einstelle, dass meine Hardware danach zicken macht, drücke ich den BIOS-Reset Taster, boote einmal durch und schon passt

wieder alles, ohne das ich erst wieder die Grundeinstellungen setzen muss. Hat seine Vorteile, wenn erstmal alles gesetzt ist, wie gewünscht - hat mich aber auch ungefähr eine Woche Zeit und diverse Neu-

starts gekostet, herauszufinden, welche Einstellung wie am optimalsten zu setzen ist.

Wurden sonst noch Änderungen am original BIOS von mir vorgenommen?

Ja. Denn wie ich bereits anfangs erwähnte, bin ich ein "Immer auf dem neuesten Stand" Fanatiker. Vielleicht kennt der Eine oder Andere ja den UEFI-BIOS-UPDATER, kurz UBU genannt. Hiermit lassen sich

ebenfalls nützliche BIOS-Änderungen bzw. Aktualisierungen vornehmen. So kann man über diesen beispielsweise die im BIOS befindlichen UEFI-Treiber für die iGPU, RAID und Ethernet updaten und auch den

Micro Code auf den neuesten Stand bringen.

Hier mal wieder zuerst das Original BIOS von der Herstellerseite:

Code

1. ASRock Super Carrier BIOS 2.20 original
- 2.
- 3.
4. 1 - Intel RST(e) OROM and EFI SATA Driver
5. OROM IRST RAID for SATA - 15.5.0.2875
6. EFI IRST RAID for SATA - 15.5.0.2875
7. EFI AMI NVME Driver present
8. 2 - Intel OROM VBIOS and EFI GOP Driver
9. OROM VBIOS SKL-KBL - 1046
10. EFI GOP Driver SKL-KBL - 9.0.1053

11. 3 - LAN OROM PXE and EFI UNDI - Intel, RTK, BCM, QCA
12. OROM Intel Boot Agent GE - 1.5.50
13. OROM Intel Boot Agent CL - 0.1.10
14. EFI Intel PRO/1000 UNDI - 6.1.14
15. EFI Intel Gigabit UNDI - 0.0.13
16. 7 - Update Intel CPU MicroCode
17. Kabylake CUID 0906E9 - 48
18. Skylake CUID 0506E3 - A6
19. 9 - ME Analyzer
20. FW ME of scanned BIOS File 11.6.0.1126 CON_H_D0
21. i - Versions, HomePages, Donate
22. 0 - Exit

Alles anzeigen

Hier sieht man die diversen UEFI ROM-Treiber für RAID, iGPU GOP Treiber, Intel LAN OROM und EFI Treiber sowie der angesprochene Micro Code (sowohl für SKY- als auch KABYLAKE)

Und hier nun die von mir aktualisierte Fassung des selben BIOS:

Code

1. ASRock Super Carrier BIOS 2.20 modified
- 2.
- 3.
4. 1 - Intel RST(e) OROM and EFI SataDriver
5. OROM IRST RAID for SATA - 15.5.0.2875
6. EFI IRST RAID for SATA - 15.7.0.3054
7. EFI AMI NVME Driver present
8. 2 - Intel OROM VBIOS and EFI GOP Driver
9. OROM VBIOS SKL-KBL - 1051
10. EFI GOP Driver SKL-KBL - 9.0.1066
11. 3 - LAN OROM PXE and EFI UNDI - Intel, RTK, BCM, QCA
12. OROM Intel Boot Agent GE - 1.5.62
13. OROM Intel Boot Agent CL - 0.1.12
14. EFI Intel PRO/1000 UNDI - 7.4.36
15. EFI Intel Gigabit UNDI - 0.0.17
16. 7 - Update Intel CPU MicroCode
17. Kabylake CUID 0906E9 - 5E
18. Skylake CUID 0506E3 - BA

19. 9 - ME Analyzer
20. FW ME of scanned BIOS File 11.6.0.1126 CON_H_D0
21. i - Versions, HomePages, Donate
22. 0 - Exit

Alles anzeigen

Dies sind die aktuellsten OROMs und EFI-Treiber, die die aktuelle Version 1.69.5 des UBU-Tools anbietet. Zusätzlich wurde die Intel MEI-Firmware von mir noch auf die aktuelle Version 11.7.4.3314 upgedatet.

Macht das einen Unterschied für die Nutzung des Super Carrier unter macOS?

Nicht wirklich, sowohl WINDOW10 als auch macOS laufen ebenso rund mit dem aktuellen BIOS der Herstellerseite. Nur wie bereits mehrfach erwähnt: ich hab's halt gerne aktuell.

Ihr findet das von mir wie oben beschrieben angepasste aktuelle BIOS für das ASRock Super Carrier auch als Anhang in diesem Beitrag. Ich nutze exakt dieses BIOS just in dem Moment, in dem ich diesen

Beitrag verfasse. Dennoch gilt an dieser Stelle: **Installieren auf eigene Gefahr**

von mir gepatchtes BIOS = Z270SC_AMIBCP_UBU_patched_2.20.zip
original BIOS der ASRock Webite = Z270 SuperCarrier(2.20)ROM.zip

Kommen wir nun zu den nötigen Vorbereitungen für die aktive Installation von macOS SIERRA/HIGH SIERRA:

ich selber nutze beide Versionen von macOS auf jeweils unterschiedlichen M.2 SSDs: SIERRA auf der SAMSUNG MZVPV, HIGH SIERRA auf der SAMSUNG 960EVO und WINDOWS10 auf der SAMSUNG 950PRO.

Zunächst solltet ihr Euch den aktuellen SIERRA 10.12.6 Installer aus dem Appstore ziehen - oder aber die Developer-/Public-Beta Installer von HIGH SIERRA. Dann benötigt Ihr einen USB-Stick, welcher minimum 8GB Platz hat.

Wer bereits einen funktionfähigen Hackintosh oder einen original Mac (welcher Art auch immer) hat, ist hier klar im Vorteil, da er die Installer auf diesen Geräten runterladen und sich einen aktuellen

Installer Bootstick mittels folgendem Terminalbefehl erstellen kann. Der USB Stick sollte im nachfolgenden Terminalbefehl einfach in "USB" umbenannt werden. Vorausgesetzt, die Installer wurden von Euch aus dem Appstore geladen und liegen somit unter "Programme", gibt man die folgenden Terminalbefehle zum erstellen des Bootsticks ein:

SIERRA:

Code

1. `sudo /Applications/Install\ macOS\ Sierra.app/Contents/Resources/createinstallmedia --volume /Volumes/USB --applicationpath /Applications/Install\ macOS\ Sierra.app --nointeraction`

HIGH SIERRA:

Code

1. `sudo /Applications/Install\ macOS\ High\ Sierra\ Beta.app/Contents/Resources/createinstallmedia --volume /Volumes/USB --applicationpath /Applications/Install\ macOS\ High\ Sierra\ Beta.app --nointeraction`

Warten, bis der Bootstick erstellt wurde. Kopiert anschliessend den jeweiligen EFI Ordner (EFI_SIERRA.zip oder EFI_HIGH_SIERRA.zip) auf die EFI partition des Sticks, um von diesen booten zu können.

In diesen EFI Ordnern liegen bereits alle nötigen SSDTs, KEXTE, SMBIOS.plist und CONFIG.plist fertig konfiguriert, so wie ich sie nutze. Auf die angepassten SSDTs, sowie die installierten Kexte gehe ich im weiteren Verlauf dieses Tutorials noch ein.

Solltet ihr das von mir gepatchte BIOS nutzen, muesst ihr KEINE BIOS-Einstellungen anpassen (zum flashen des BIOS (egal ob gepatched oder original) einfach das BIOS ungezippt auf einen FAT32 formatierten USB Stick kopieren, in einen beliebigen USB-Port stecken und beim booten F6 drücken), solltet ihr das original BIOS des Herstellers nutzen, sollten die Einstellungen im BIOS wie folgt vorgenommen werden:

BIOS SETUP EASY:



The screenshot displays the ASRock EZ Mode BIOS interface. At the top, it shows the motherboard model 'Z270 SuperCarrier P2.20' and system information: 'Intel(R) Core(TM) i7-6700K CPU @ 4.00GHz', 'Prozessorgeschwindigkeit: 4000 MHz', and 'Speicher gesamt: 32 GB'. The top right corner features a digital clock '13:08', the date 'So 08/20/2017', and sensor readings: 'CPU Temperature 40.0 °C', 'M/B Temperature 34.0 °C', and 'CPU Voltage 1.280 V'. The main area is divided into several sections: 'DRAM-Informationen' lists four Corsair 8GB DDR4 modules; 'CPU EZ OC' shows a speed dial set to 'OFF'; 'Lüfterstatus' (Fan Status) shows 'CPU-Lüfter 1' at 'N/A' and 'Water Pump' at '974'; 'Startpriorität' (Boot Priority) lists 'UEFI OS (Samsung SSD 960 EVO 250GB)', 'Windows Boot Manager (Samsung SSD 950 PRO)', and 'UEFI OS (SAMSUNG MZVPV256HDGL-00000)'; and 'Speicherkonfiguration' (Storage Configuration) shows SATA and M.2 drives, including 'HL-DT-S', 'Samsung SSD 950', and 'SAMSUNG MZVPV25'. The bottom right includes a 'Tools' section with icons for a monitor, globe, and keyboard. The interface is in German, with 'Deutsch' selected in the top right.

ASRock EZ MODE

Z270 SuperCarrier P2.20

Intel(R) Core(TM) i7-6700K CPU @ 4.00GHz
Prozessorgeschwindigkeit: 4000 MHz
Speicher gesamt: 32 GB

13:08
So 08/20/2017

CPU Temperature 40.0 °C
M/B Temperature 34.0 °C
CPU Voltage 1.280 V

DRAM-Informationen

DDR4_A1: Corsair 8GB (3200)
DDR4_A2: Corsair 8GB (3200)
DDR4_B1: Corsair 8GB (3200)
DDR4_B2: Corsair 8GB (3200)

Profile 1: DDR4-3200 16-18-18-36 1.35V

XMP-Profil Profil 1

Speicherkonfiguration

SATA3_0/M.2_1(SATA) : HL-DT-S
SATA3_1 : N/A
SATA3_2 : N/A
SATA3_3/M.2_3(SATA) : N/A
SATA3_4/M.2_2(SATA) : N/A
SATA3_5 : N/A
SATA3_A1 : N/A
SATA3_A2 : N/A

SATA3_A3 : N/A
SATA3_A4 : N/A
M.2_1(PCI-E) : Samsung SSD 950
M.2_2(PCI-E) : Samsung SSD 960
M.2_3(PCI-E) : SAMSUNG MZVPV25
SATAE_0 : N/A
SATAE_1 : N/A

RAID-Modus Deaktiviert

CPU EZ OC

Lüfterstatus

CPU-Lüfter 1 N/A
Water Pump: 974
Gehäuselüfter 1 N/A
Gehäuselüfter 2 N/A
Gehäuselüfter 3 N/A

Startpriorität

UEFI OS (Samsung SSD 960 EVO 250GB)
Windows Boot Manager (Samsung SSD 950 PRO)
UEFI OS (SAMSUNG MZVPV256HDGL-00000)

Tools

CPU-Lüfter 1-Einstellung
Standard

BIOS SETUP ADVANCED:



BIOS SETUP OC TWEAKER:



BIOS SETUP ERWEITERT:



BIOS SETUP CHIPSET-KONFIGURATION:





BIOS SETUP SPEICHER-KONFIGURATION:



* der ASMedia SATA3 Control ist von mir auf DISABLED gesetzt, weil ich die ASMedia SATA Ports nicht nutze. Lasst ihn auf ENABLED, wenn ihr die ASMedia SATA Ports nutzt.

BIOS SETUP THUNDERBOLT:

siehe Screenshots weiter oben in der Beschreibung

BIOS SETUP SUPER IO-KONFIGURATION:



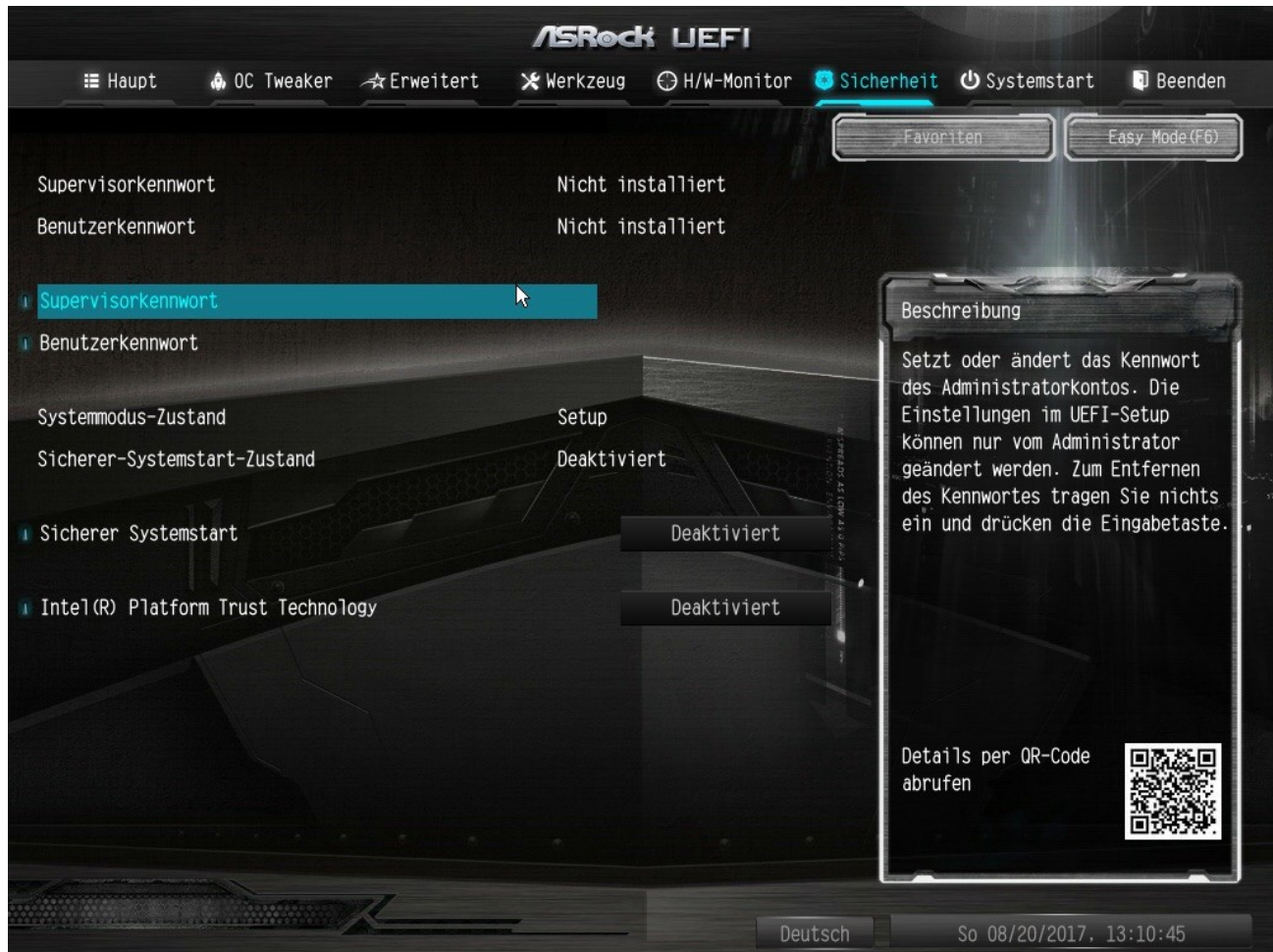
BIOS SETUP USB-KONFIGURATION:



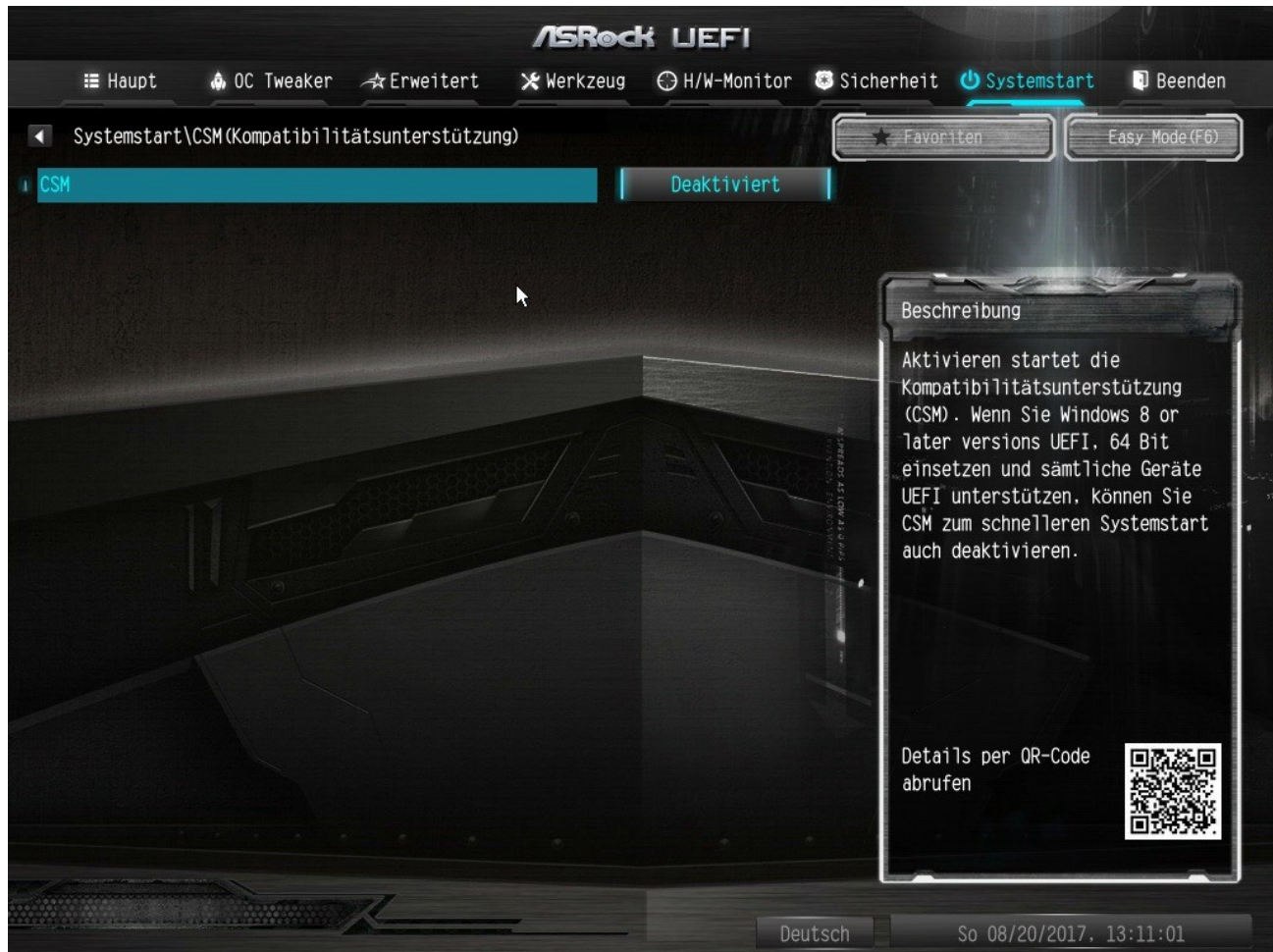
BIOS SETUP TRUSTED COMPUTING:



BIOS SETUP SICHERHEIT:



BIOS SETUP SYSTEMSTART - CSM:





Mit diesen [BIOS Einstellungen](#) habe ich die besten Erfahrungen gemacht, sowohl unter macOS, als auch unter WINDOWS.

Aber wie oben bereits erwähnt, sind folgende Komponenten bei mir disabled, weil ich sie weder unter macOS noch unter WINDOWS nutze: Bluetooth, Onboard-AUDIO und WLAN.

Was beinhaltet der von mir genutzte EFI CLOVER Folder?

In diesem Ordner habe derzeit alle für einen erfolgreichen Bootvorgang unter macOS benötigten Dateien zusammengefasst, damit dieses Board optimal läuft. Dies beinhaltet die nachfolgend aufgelisteten Dateien:

- EFI/BOOT = BOOTX64.efi - der aktuelle (rev. 4178) EFI Loader
- EFI/CLOVER = CLOVERX64.efi - in der aktuellen Rev. 4178
- EFI/CLOVER = config.plist - die CONFIGURATION plist mit den nötigen Einstellungen für ACPI, Boot, CPU, Devices, GUI, Graphics, KernelAndKextPatches, RTVariables und Systemparameters

- EFI/CLOVER = smbios.plist - die Einstellungen zu definition der zu nutzenden SMBIOS-Einstellungen (separiert, da ich mir hier verschiedene Konstellationen zum testen angelegt habe wie beispielsweise iMac17,1 - iMac18,1 - iMac18,3 - iMacPro - MacBookPro11,3 etc. - einfach die smbios.plist tauschen und neustarten.
- EFI/CLOVER/ACPI/origin = die original ungepatchten DSDT-SSDTs des vom Board genutzten BIOS zum patchen für den ACPI/patched Folder
- EFI/CLOVER/ACPI/patched = CpuPmPike.aml, IgpuGfx1.aml, UsbFix.aml und Xosi.aml <--gehe ich später noch genauer drauf ein
- EFI/CLOVER/drivers64UEFI = apfs.efi, FSInject-64.efi, HFSPlus-64.efi, OsxAptioFixDrv-64.efi, Test2-2016.efi (ähnlich dem Treiber OsxAptioFix2Drv-64.efi - habe ich bislang für die Gigabyte AlpineRidge Thunderbolt3 PCIe Karte benötigt) - noch nicht probiert, ob dieses Board auch ohne funktioniert. Ggf. besser ich an dieser Stelle nach
- EFI/CLOVER/kexts/Other = AppleGB.kext (v5.3.5.3) , FakeSMC.kext (v6.25-333), IntelMausiEthernet.kext (v2.3.0), Lilu.kext (v1.1.6), WhateverGreen.kext (v1.0.3) und XHCI-200-series-Injector.kext (v0.5.0)

Warum befindet sich unter EFI/CLOVER/ACPI/patched keine DSDT.aml?

Weil ich mittels der config.plist die original DSDT schon soweit patchen kann, daß keine spezielle DSDT.aml in diesem Ordner benötigt wird. Durch die Einstellungen in der config.plist wird die originale DSDT bereits soweit angepasst, sodaß ich nicht mit jedem neuen BIOS die DSDT wieder neu patchen muss.

Und was machen die vier SSDTs, welche im Ordner EFI/CLOVER/ACPI/patched liegen?

Diese Dateien passen die Nutzung der AMD RX480/iGPU, das PowerManagement und die USB-Ports optimal an die Systemeinbindung an.

Mehr brauchte ich für meine Zwecke nicht. Die "CpuPmPike.aml" basiert auf dem Script von Pike R. Alpha (siehe [hier](#)), die "IgpuGfx1.aml" passt die Einbindung von iGPU und der in Slot1 steckenden RX480 inklusiv

HDMI-Audio an und die "UsbFix.aml" steuert die der Hardware zur Verfügung stehenden USB-Ports an, so daß in der config.plist kein "Enhance USB-Port Limit"-patch benötigt wird und man darüber relativ einfach die zu nutzenden USB-Ports steuern kann.

Btw: Dank LILU/WhateverGreen benötige ich in der config.plist für SIERRA nur noch die NVMe-Patche von Pike R. Alpha und in der config.plist von HIGH SIERRA habe ich gar keinen Kernel-/Kextpatch mehr drin!

Wie steuert man über die "UsbFix.aml" denn bitte die nötigen USB-Ports, ohne einen "Enhance USB-Port Limit"-Patch?

Relativ einfach und äußerst genial:

schauen wir uns zunächst den Code der SSDT an:

Spoiler anzeigen

In diesem Code werden ALLE auf dem Board befindlichen USB-Ports gesteuert - sowohl die an der Backplate befindlichen, als auch die OnBoard Ports. Insgesamt haben wir beim Super Carrier 14 davon:

auf der Rückseite: 2x USB2.0, 2x USB3.1 (Typ A), 2x USB 3.0 (Typ A)

auf dem Motherboard: 4x USB2.0 und 4x USB3.0

Dank einer einzigen Zeile des Codes innerhalb der SSDT kann ich nun genau steuern, welche der USB2/USB3-Ports unter macOS nutzbar sein sollen:

Code

1. Store (0xFFE8, \HUBC)
2. Store (0x10, \UMAP)

Wie genau funktioniert das? Mittels der macOS eigenen "Taschenrechner" App! Wir benötigen dazu nur die "Programmieren"-Funktion der App:



Aber fangen wir von vorne an. Zunächst benötigen wir dann doch noch den "Enhance USB-Port limited"-Patch, als auch die/den "USBInjectAll.kext".

Letzteres findet ihr ebenfalls als Anhang in diesem Post (siehe RehabMan-USBInjectAll-2017-

0724.zip). Der/die/das kext wie immer unter EFI/CLOVER/kexts/Other plazieren.

Anschließend in der config.plist in der Sektion "KernelAndKextPatches" folgenden Code einfügen:

SIERRA:

Code

1. <dict>
2. <key>Comment</key>
3. <string>change 15 port limit to 26 in XHCI kext (100-series) 10.12</string>
4. <key>Disabled</key>
5. <false/>
6. <key>Find</key>
7. <data>g710///EA==</data>
8. <key>Name</key>
9. <string>AppleUSBXHCIPCI</string>
10. <key>Replace</key>
11. <data>g710///Gw==</data>
12. </dict>

Alles anzeigen

HIGH SIERRA:

Code

1. <dict>
2. <key>Comment</key>
3. <string>Change 15 port limit to 24 in XHCI kext 10.13 PB1</string>
4. <key>Disabled</key>
5. <false/>
6. <key>Find</key>
7. <data>
8. g32MEA==
9. </data>
10. <key>Name</key>
11. <string>AppleUSBXHCIPCI</string>
12. <key>Replace</key>
13. <data>
14. g32MGw==
15. </data>

16. </dict>

Alles anzeigen

Rechner neu starten und ihr solltet in etwa folgendes im IORegistryExplorer sehen:

```
graph TD
    SBUS@1F,4 --> XHC@14
    XHC@14 --> XHC@14000000
    XHC@14000000 --> HS01@14100000
    XHC@14000000 --> HS02@14200000
    XHC@14000000 --> HS03@14300000
    XHC@14000000 --> HS04@14400000
    XHC@14000000 --> HS05@14500000
    XHC@14000000 --> HS06@14600000
    XHC@14000000 --> HS07@14700000
    XHC@14000000 --> HS08@14800000
    XHC@14000000 --> HS09@14900000
    XHC@14000000 --> HS10@14a00000
    XHC@14000000 --> HS11@14b00000
    XHC@14000000 --> HS12@14c00000
    XHC@14000000 --> HS13@14d00000
    XHC@14000000 --> HS14@14e00000
    XHC@14000000 --> SS01@15100000
    XHC@14000000 --> SS02@15200000
    XHC@14000000 --> SS03@15300000
    XHC@14000000 --> SS04@15400000
    XHC@14000000 --> SS05@15500000
    XHC@14000000 --> SS06@15600000
    XHC@14000000 --> SS07@15700000
    XHC@14000000 --> SS08@15800000
    XHC@14000000 --> SS09@15900000
    XHC@14000000 --> SS10@15a00000
    XHC@14000000 --> USR1@14f00000
    XHC@14000000 --> USR2@15000000
```

SBUS@1F,4

- ▼ XHC@14
 - ▼ XHC@14000000
 - ▶ HS01@14100000
 - ▶ HS02@14200000
 - ▶ HS03@14300000
 - HS04@14400000
 - ▶ HS05@14500000
 - HS06@14600000
 - HS07@14700000
 - HS08@14800000
 - ▶ HS09@14900000
 - HS10@14a00000
 - HS11@14b00000
 - HS12@14c00000
 - HS13@14d00000
 - HS14@14e00000
 - SS01@15100000
 - SS02@15200000
 - SS03@15300000
 - SS04@15400000
 - ▶ SS05@15500000
 - SS06@15600000
 - SS07@15700000
 - SS08@15800000
 - SS09@15900000
 - SS10@15a00000
 - USR1@14f00000
 - USR2@15000000

Jetzt in jeden der zur Verfügung stehenden Ports einen USB-Stick einstecken und schauen, unter welchem Anschluss sich dieser meldet. Bei Euren USB3-Ports müsst Ihr sowohl einen USB3- als auch einen USB2-Stick einstecken. Schreibt euch dann sowohl die genutzten SSxx als auch HSxx Ports auf. Nun benötigen wir, wie Eingangs bereits erwähnt den Taschenrechner. Stellt diesen im Menü "Ansicht" auf "Programmer" um. Um nun den Wert für die genutzten USB2 Ports zu ermitteln, tippt zunächst "FF FF" in den Rechner. Ihr seht in der zweiten Zeile die Werte 0 bis 15 alle auf "1" gesetzt:



Jetzt tippt ihr auf die "1"en, für die ihr einen jeweiligen HS-Port ermittelt habt (sieht man oben im Screen durch die davor stehenden Dreiecke).

Bei mir ergibt sich dadurch folgendes Bild: für die aktiven USB2 Ports gilt der Wert 0xFEE8



Die durch das auf die "1"en klicken erzeugten "0"en zeigen die genutzten HSxx-Ports. Hier sollte man meinen 1 = aktiv, 0 = inaktiv. Aber bei den USB2 Ports ist es eben umgekehrt: 0 ist ein aktiver, 1 ein inaktiver Port.

Um nun den Wert für die SS-Ports zu ermitteln, setzt Ihr den Rechner durch klicken auf "C" wieder zurück auf NULL. Jetzt klickt ihr die "Nullen" an, bei denen Ihr einen passenden USB3 Port ermittelt habt. Dieser wechselt dann auf "1". Für USB3 gilt also: 0 ist ein inaktiver, 1 ein aktiver Port.

Aus dem Beispiel oben im Bild wäre das also die EINS auf Position 5 (SS05). Dadurch ergibt

sich unserem Beispiel für die aktiven USB3 Ports also der Wert 0x10:



Der Erklärung halber: die hinteren USB3 Ports wurden in diesem Beispiel alle deaktiviert, ich nutze mein Case in einem 19"-Rack, komme also an die hinteren Ports nicht so gut ran und habe daher nur die 4 USB3-Frontports aktiviert. Und diese 4 Ports liegen alle an SS05 an.

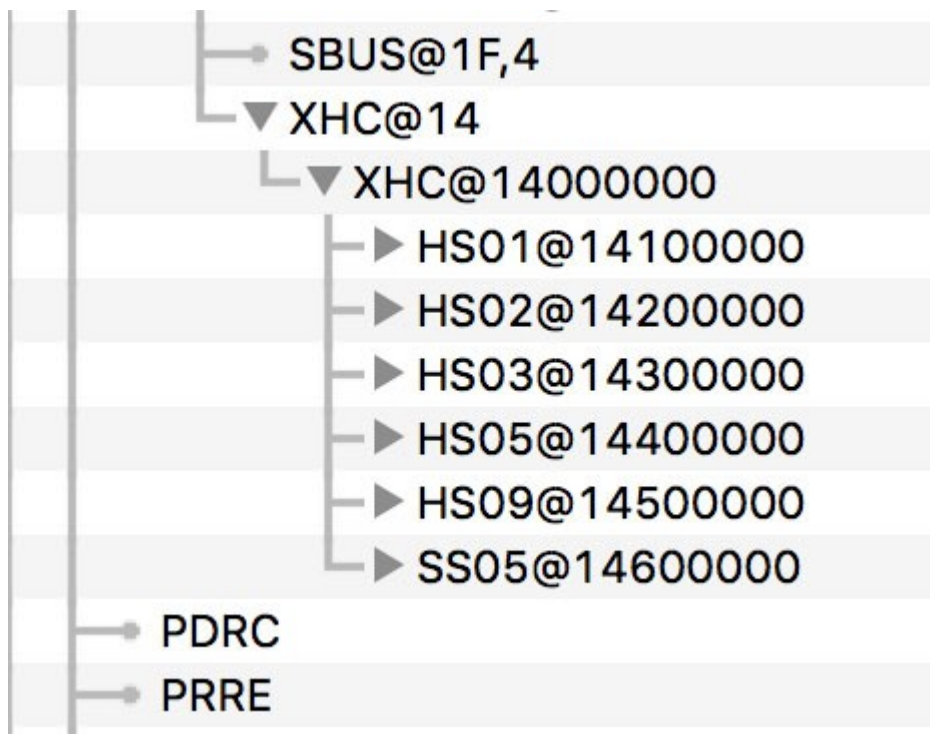
Tragt die von Euch ermittelten Werte nun in die beiden Zeilen ein (dieses hier sind die aus unserem Beispiel ermittelten Werte):

Code

1. Store (0xFEE8, \HUBC)
2. Store (0x10, \UMAP)

speichert die SSDT, löscht der/die/das Kext "USBInjectAll.kext" aus dem Ordner "EFI/CLOVER/kexts/Other" und den Patch aus der config.plist (oder setzt den Wert für "Disabled" auf "/true") und startet den Rechner neu.

Ihr solltet dann in etwa dieses Bild erhalten:



Wann immer ein Port dazukommt oder wegfällt, müsst Ihr nur mittels des Taschenrechners die Werte neu ermitteln und in die beiden gezeigten Zeilen einfügen. Port-limit Patch adé.

Was muss bei der Nutzung der AMD Radeon Grafikkarte beachtet werden?

Dank des LILU und WhateverGreen Kextes eigentlich gar nichts mehr. Meine RX480 steckt im ersten PCIe x16 Port und ist von mir mittels SSDT auch sauber ins System eingebunden. Die SSDT bindet dabei

nicht nur die Grafikkarte sauber ein, sondern auch den dazugehörigen Ton via HDMI. Hier muss keinerlei "ATI = inject" mehr gesetzt werden, noch ist ein Framebufferpatch mehr von Nöten. Solltet Ihr Eure

Karte in einen anderen Slot des Super Carrier Boards stecken, müsst ihr nur die SSDT an den dann zu ermittelnden Steckplatz anpassen. Bei Fragen, wie man das macht, könnt Ihr mich gerne ansprechen.

An die Leute, die statt einer AMD lieber eine NVIDIA GFX-Karte nutzen: sorry Jungs, hier kann ich bei Fragen nicht helfen, da mir durch fehlende Hardware hier das passende KnowHow fehlt.

Das war es eigentlich auch schon, wenn Ihr dieses Tutorial Punkt für Punkt abgearbeitet habt, sollte Euer ASRock Super Carrier Hackintosh jetzt schonmal richtig gut laufen.

PS: das Onboard WLAN werdet ihr unter OS X nicht zum laufen bekommen, weil es sich bei der benutzten Hardware um eine Intel WLAN-Karte auf M.2 Basis handelt, für die es laut meinen Recherchen keinerlei

Treiber oder Patches oder FakePCI-Lösungen gibt. Der Bluetoothadapter tut jedoch einwandfrei seinen Dienst, solange man den dafür genutzten USB-Port (HS09) nicht deaktiviert.

PPS: anbei auch noch ein Firmwareupdate für den Thunderbolt3 Controller, den mir ASRock auf Anfrage zur Verfügung gestellt hat. Damit wird die NVM-Firmware von v12 auf v20 gehoben (zur Unterstützung von eGFX)

DANKSAGUNG: mein Dank geht an alle, die sich seit Jahren mit dem Thema HACKINTOSH befassen und soviele nützliche Treiber, Patches, Ideen oder Anleitungen verfasst haben. Besonderer Dank geht an

- SLICE - für das Projekt CLOVER
- RehabMan - für die vielen nützlichen Tutorials rund um DSDT, SSDT, USB sowie die vielen Treiber und Patches, die von Ihm stammen
- Pike R. Alpha - ebenfalls für die vielen Patches und Ideen aus seinem Blog
- Mizi - für den genialen Ethernettreiber
- vandroiy2012 - für die LILU und WhatEverGreen Lösungen rund um die Grafikkarten
- Sherlocks - für seine außergewöhnlichen Leistungen an der Optimierung der letzten CLOVER Versionen (hinsichtlich HIGH SIERRA)

- Mickey1979 - für sein geniales "Build Clover"-Script, mit dem es ein Kinderspiel ist, CLOVER auf dem aktuellen Stand zu halten

und für die Macher und User dieses genialen, deutschen Forums zum Thema Hackintosh - live long and prosper.

Cheers,

Mork vom Ork

oops, eine Sache habe ich noch vergessen zu erwähnen: auch wenn das ASRock Super Carrier richtig geile BIOS-Einstellungen für THUNDERBOLT liefert, so ist es mir bislang dennoch nicht gelungen, daß sich angeschlossene Thunderbolt-Hardware auch im SystemProfiler als solche zu erkennen gibt. Angeschlossene Komponenten werden zwar erkannt und, sofern der passende Treiber vorhanden ist, auch unterstützt, aber im SystemProfiler bleibt der Baum unter dem Punkt Thunderbolt noch immer leer, bzw. dort steht nur der Satz: "Thunderbolt: no Hardware was found". Meine CalDigit ThunderStation2 wird jedoch korrekt erkannt und mir steht der darin befindliche Ethernet- USB3- und eSATA-Port zur Nutzung zur Verfügung.