Erledigt Problem die RX 480 Powercolour zu installieren

Beitrag von „Mork vom Ork“ vom 30. April 2017, 09:02

So, da ich nun alle für mich relevanten Daten von Dir (pawelpipowich) habe, fange ich mal an.

Als erstes müssen wir dafür sorgen, daß Dein Mainboard via HDMI mit deinem Monitor verbunden ist. Sollte Dein Monitor nur einen Displayport- oder DVI-Eingang haben, benötigst Du dafür ein HDMI-auf-Displayport- oder HDMI-auf-DVI-Kabel. Bekommst Du günstig und schnell via AMAZON.DE.

Als nächstes musst Du dein Gigabyte-BIOS so konfigurieren, daß Deine IGPU (Mainboardinterne Grafik) als primary gesetzt ist. Da dieses Mainboard (und das dazu passende letzte BIOS-Update) bereits von 2013 ist, schau bitte mal, ob Du im BIOS auch eine Funktion namens "CSM" hast, die Du ggf. ein- bzw. ausschalten kannst. Schalte diese Funktion bitte ein. Lt. meinen GOOGLE-Recherchen heisst die Funktion in deinem BIOS "CSM Support" und befindet sich unter den Einstellungen "BIOS Features".

Wie ich bereits in meinem letzten Beitrag erwähnt habe, kann die PowerColor RX480 bereits im PCIe slot #1 eingebaut sein. In erster Linie geht es darum, das Du mittels der IGPU ins BIOS kommst und somit auch unter CLOVER ein Bild auf dem Monitor hast.

Gehen wir mal davon aus, daß dies nun der Fall ist, so kommen wir zum nächsten Schritt: wir benötigen für die RX480 einen für SIERRA (macOS 10.12.4) passenden Framebuffer, mit dem wir die 3 Displayport sowie den HDMI- als auch den DVI-Anschluss ansprechen können. Ich habe mir das VBIOS Deiner PowerColor RX480 bei Techpowerup.com besorgt und hoffe mal, daß ich das richtige erwischt habe.

Für den Framebuffer-Patch benötigen wir 2 Scripts: **radeon_bios_decode.sh** und **redsock_bios_decoder.sh** sowie die ".rom"-Datei Deines VBIOS.
Beide Scripts findet Ihr als Anhang zu diesem Beitrag unten. Anwendung beider Scripts wäre wie folgt:

Unter OS X ein Terminalfenster aufmachen, das erste Script per drag-and-drop ins offene Terminalfenster ziehen, gefolgt von einem "**<**" und einem Leerzeichen und anschliessend das "xxx.rom"-File per drag-and-drop ins Terminalfenster ziehen. Dann sollte dort in etwa stehen (das erste ist jeweils der Pfadname, der bei Euch anders aussehen kann und derzeit meine Pfade zeigt, unter dem die Dateien bei mir liegen):

" /Volumes/Install\ macOS\ Sierra/APPS_and_FILES/Decoders/**radeon_bios_decode** < /Users/mvo/Desktop/**Powercolor.RX480.8192.160727.rom** "

Jetzt ENTER drücken und man erhält daraufhin folgende Ausgabe:

Code

1. ATOM BIOS Rom:
2. SubsystemVendorID: 0x148c SubsystemID: 0x2372
3. IOBaseAddress: 0x0000
4. Filename: I1727OAD.SLC
5.
6.
7. BIOS Bootup Message:
8. D00901 Polaris10 XT A1 GDDR5 256Mx32 8GB I1727OAD.SLC 2016
9.
10.
11. PCI ID: 1002:67df
12. Connector at index 0
13. Type [@offset 40846]: DisplayPort (10)
14. Encoder [@offset 40850]: INTERNAL_UNIPHY2 (0x21)
15. i2cid [@offset 40956]: 0x90, OSX senseid: 0x1
16. HotPlugID: 6
17.
18.
19. Connector at index 1
20. Type [@offset 40856]: DisplayPort (10)
21. Encoder [@offset 40860]: INTERNAL_UNIPHY2 (0x21)
22. i2cid [@offset 40983]: 0x92, OSX senseid: 0x3
23. HotPlugID: 4
24.
25.
26. Connector at index 2
27. Type [@offset 40866]: DisplayPort (10)
28. Encoder [@offset 40870]: INTERNAL_UNIPHY1 (0x20)

29. i2cid [@offset 41010]: 0x91, OSX senseid: 0x2
30. HotPlugID: 1
31.
32.
33. Connector at index 3
34. Type [@offset 40876]: HDMI-A (11)
35. Encoder [@offset 40880]: INTERNAL_UNIPHY1 (0x20)
36. i2cid [@offset 41037]: 0x93, OSX senseid: 0x4
37. HotPlugID: 5
38.
39.
40. Connector at index 4
41. Type [@offset 40886]: DVI-D (3)
42. Encoder [@offset 40890]: INTERNAL_UNIPHY (0x1e)
43. i2cid [@offset 41064]: 0x95, OSX senseid: 0x6
44. HotPlugID: 3

Alles anzeigen

Auf die selbe Art und weise holen wir uns mit dem zweiten Script nun noch die fehlenden Werte:
script plus "< " plus VBIOS.ROM-File ins Terminalfenster ziehen und ENTER drücken. Ergebnis sieht dann wie folgt aus:

Code

1. D00901 Polaris10 XT A1 GDDR5 256Mx32 8GB I1727OAD.SLC 2016
2. Subsystem Vendor ID: 148c
3. Subsystem ID: 2372
4. Object Header Structure Size: 340
5. Connector Object Table Offset: 48
6. Router Object Table Offset: 0
7. Encoder Object Table Offset: fb
8. Display Path Table Offset: 12
9.
10.
11. Connector Object Id [19] which is [DISPLAY_PORT]
12. encoder obj id [0x21] which is [INTERNAL_UNIPHY2 (osx txmit 0x12 [duallink 0x2] enc 0x4)] linkb: false
13.
14.
15. Connector Object Id [19] which is [DISPLAY_PORT]
16. encoder obj id [0x21] which is [INTERNAL_UNIPHY2 (osx txmit 0x22 [duallink 0x2] enc 0x5)] linkb: true

17.
18.
19. Connector Object Id [19] which is [DISPLAY_PORT]
20. encoder obj id [0x20] which is [INTERNAL_UNIPHY1 (osx txmit 0x11 [duallink 0x1] enc 0x2)] linkb: false
21.
22.
23. Connector Object Id [12] which is [HDMI_TYPE_A]
24. encoder obj id [0x20] which is [INTERNAL_UNIPHY1 (osx txmit 0x21] [duallink 0x1] enc 0x3)] linkb: true
25.
26.
27. Connector Object Id [4] which is [DVI_D]
28. encoder obj id [0x1e] which is [INTERNAL_UNIPHY (osx txmit 0x10 [duallink 0x0] enc 0x0)] linkb: false

Alles anzeigen

Jetzt haben wir alle relevanten Infos, die wir zum Erstellen eines passenden Frambuffer-Patches brauchen.
Holen wir uns also die originalen Framebufferdaten aus SIERRA, um diese für uns passend zu patchen. Hierzu benötigen wir das PHP-file "ATI_FrameBuffers_Sierra_Edition.php" und ebenfalls das Terminal unter OS X. Dort tippen wir den Befehl "php " ein, ziehen wieder die Datei "ATI_FrameBuffers_Sierra_Edition.php" direkt dahinter ins Terminalfenster und drücken Enter (sollte macOS Euch jetzt auffordern XCODE zu laden, tut dies bitte, denn wir werden es später erneut benötigen).
Ergebnis sieht wie folgt aus (da für uns nur der AMD9150Controller wichtig ist, hier nur dessen Einträge):

Code

1. ----------AMD9510Controller.kext----------
2. Exmoor (6) @ 0x107080
3. LVDS, LVDS, DP, DP, DP, DP
4. 020000000001000000010151000000002205020400000000
5. 020000000001000000010261010000001204010300000000
6. 000400000403000000010343000000001102030100000000
7. 000400000001000000010431000000002103050500000000
8. 000400000403000000010523000000001000040200000000
9. 000400000001000000010611000000002001050500000000
10.
11.
12. Berbice (5) @ 0x107110
13. LVDS, DP, DP, DP, DP

14. 020000000001000039050108000000002001010100000000
15. 000400000001000000010243000000001000020200000000
16. 000400000403000000010313000000002103030300000000
17. 000400000403000000010453000000001102040400000000
18. 000400000403000000010533000000001204050500000000
19.
20.
21. Baladi (6) @ 0x107300
22. DP, DP, DP, DP, DP, DP
23. 000400000403000000010300000000001204030300000000
24. 000400000403000000010100000000001102010100000000
25. 000400000403000000010200000000002103020200000000
26. 000400000403000000010400000000002205040400000000
27. 000400000403000000010500000000001000050500000000
28. 000400000403000000010600000000002001060600000000

Alles anzeigen

Die von uns genutzte PowerColor RX480 hat insgesamt 5 Anschlüsse, also ist der für uns am besten passende Framebuffer "BERBICE", da dieser ebenfalls über 5 Anschlüsse verfügt. Wir können aber erkennen, das BERBICE für einen internen Anschluss (LVDS) sowie 4 externe Displayport-Anschlüsse ausgelegt ist.
Nun heisst es diesen zu patchen. Dadurch wird dann aus:

Code

1. LVDS, DP, DP, DP, DP
2. 020000000001000039050108000000002001010100000000
3. 000400000001000000010243000000001000020200000000
4. 000400000403000000010313000000002103030300000000
5. 000400000403000000010453000000001102040400000000
6. 000400000403000000010533000000001204050500000000

dieser Code mit Anschlüssen für 3x Displayport, 1x HDMI und 1x DVI-D:

Code

1. DP, DP, DP, HDMI, DVI-D
2. 00040000040300000001024300000000120406010000000 0
3. 00040000040300000001031300000000220504030000000 0
4. 00040000040300000001045300000000110201020000000 0
5. 00080000040200000001053300000000210305040000000 0
6. 04000000140200000001030000000000100003060000000 0

Daraus ergibt sich dann für die spätere Verwendung unter CLOVER folgender Patchcode:

original FB:
02000000000100003905010800000000200101010000000000040000000100000001024300000000100002

patched FB: 00040000040300000001024300000000**12040601**
00000000000400000403000000010313000000000**22050403**
00000000000400000403000000010453000000000**11020102**
00000000000800000402000000010533000000000**21030504**
00000000040000001402000000010300000000000**10000306**00000000

Soweit, so gut. Jetzt müßen wir noch die entsprechenden AMD-Kexte patchen, damit unsere RX480 Karte überhaupt von den Apple Treibern erkannt wird. Dazu legen wir uns Kopien der nachfolgend genannten Kext-Dateien an:

AMD9510Controller.kext und **AMDRadeonX4100.kext**

Bei beiden müssen wir die interne "Info.plist"-Datei anpassen. Dazu einen Rechtsklick auf die jeweilige Kext-Datei und "Paketinhalt zeigen" auswählen, den Ordner "Contents" öffnen und nun die "Info.plist" per doppelklick öffnen. Wer sich, wie ich gesagte hatte, XCode heruntergeladen und installiert hat, sollte daraufhin folgendes Bild bekommen (am Beispiel der **AMDRadeonX4100.kext**

Für uns relevant ist hier der Eintrag unter "IOPCIMatch", denn hier müssen wir nun die Device-ID unserer RX480 nachtragen: **0x67DF1002**. Einfach diesen Wert mit einem Leerzeichen davor hinter dem Wert "0x67EF1002" eintragen. Info.plist speichern - fertig.

Das selbe machen wir im Falle der **AMD9510Controller.kext**:

Auch hier tragen wir wieder unsere Device-ID nach: " **0x67DF1002**" direkt hinter "0x67EF1002". Speichern, Fertig.
Jetzt beide gepatchten Kexte mittels Kext Wizard wieder in den Ordner /System/Library/Extensions laden, den KernelCache refreshen und weiter geht es.

Finaler Schritt: anpassen der CLOVER "**config.plist**" Datei.
Hierzu öffnen wir unsere CLOVER config.plist und suchen uns folgenden Bereich raus:

Code

```
<key>Graphics</key>
<dict>
<key>DualLink</key>
<integer>1</integer>
<key>FBName</key>
<string>Berbice</string>
<key>ig-platform-id</key>
<string>0x19120000</string>
<key>Inject</key>
<dict>
<key>ATI</key>
<true/>
<key>Intel</key>
<false/>
</dict>
<key>InjectEDID</key>
<false/>
</dict>
```

Alles anzeigen

Kann bei Euch ein bißchen anders aussehen, da der hier zu sehende Teil aus meiner config.plist stammt (welche bereits für eine RX480/RX580 genutzt wird). Wichtig sind hierbei die Einträge **<key>FBName</key>** und **<key>ATI</key>**. Bei FBNamen tragt Ihr darunter den Namen des von Euch gepatchten Framebuffers ein (in unserem Fall eben BERBICE, und für ATI setzt Ihr darunter den Wert auf <true/>, da CLOVER ja das Injecten für AMD Grafikkarten übernehmen soll. Das ist alles, was in dieser Sektion eingestellt werden muss.

Nächster Part für die CLOVER config.plist: der Bereich "**<key>KernelAndKextPatches</key>** " - bei mir sieht dieser so aus:

Code

```
1. <key>KernelAndKextPatches</key>
2. <dict>
3. <key>ATIConnectorsController</key>
4. <string>9510</string>
5. <key>ATIConnectorsData</key>
6. <string>02000000000100003905010800000000200101010000000000040000001000000010243000
7. <key>ATIConnectorsPatch</key>
8. <string>000400000403000000010100000000001204060100000000000400000403000000010200000
```

Hier sind nun folgende 4 Einträge durch die von uns früher ermittelten Werte zu erstzen:

Code

```
1. <key>ATIConnectorsData</key>
2. <string></string>
3. <key>ATIConnectorsPatch</key>
4. <string></string>
```

Hier setzen wir zwischen die beiden "string"-Felder unseren ermittelten Wert des original BERBICE-Framebuffers, in unserem Fall also:

Code

```
1. <key>ATIConnectorsData</key>
2. <string>02000000000100003905010800000000200101010000000000040000001000000010243000
```

Das selbe machen wir für den gepatchten Framebuffereintrag, so daß dort dann folgendes steht:

Code

```
1. <key>ATIConnectorsPatch</key>
2. <string>000400000403000000010243000000001204060100000000000400000403000000010313000
```

FERTIG! Eigentlich gar nicht so schwer, wenn man weiss, wie es geht 
Wenn wir nun unseren Hackintosh durchbooten, sollte SIERRA die RX480 vollständig erkennen und unterstützen.
ABER: jedoch nur solange wir unsere IGPU als primary GFX im BIOS gesetzt haben! Als alleinige und primary Grafikkarte läuft die RX480 **nicht**, bzw wenn nur ohne METAL-Unterstützung, also ohne Beschleunigung. **Und bislang ist für dieses Problem noch keine Lösung in Sicht.**

Solange man aber eine Helperkarte hat, welche man als primary GFX deklarieren kann, kann man die RX-Karten in ihrer vollen Pracht auch unter macOS nutzen.
Anbei die von mir in diesem Tutorial erwähnten Scripte zum auslesen der VBIOS-Daten und der Framebuffer unter SIERRA.

Ist leider ein bißchen länger als gedacht geworden, aber ich dachte mir: einfach nur die fertigen Patche hier reinklatschen kann ja jeder. Vielleicht interessiert es ja den Ein oder Anderen, wie ich auf die benötigten Patche komme - daher dieser Beitrag.