AMD RADEON RX Grafikkarten ohne LILU & WhatEverGreen nutzen

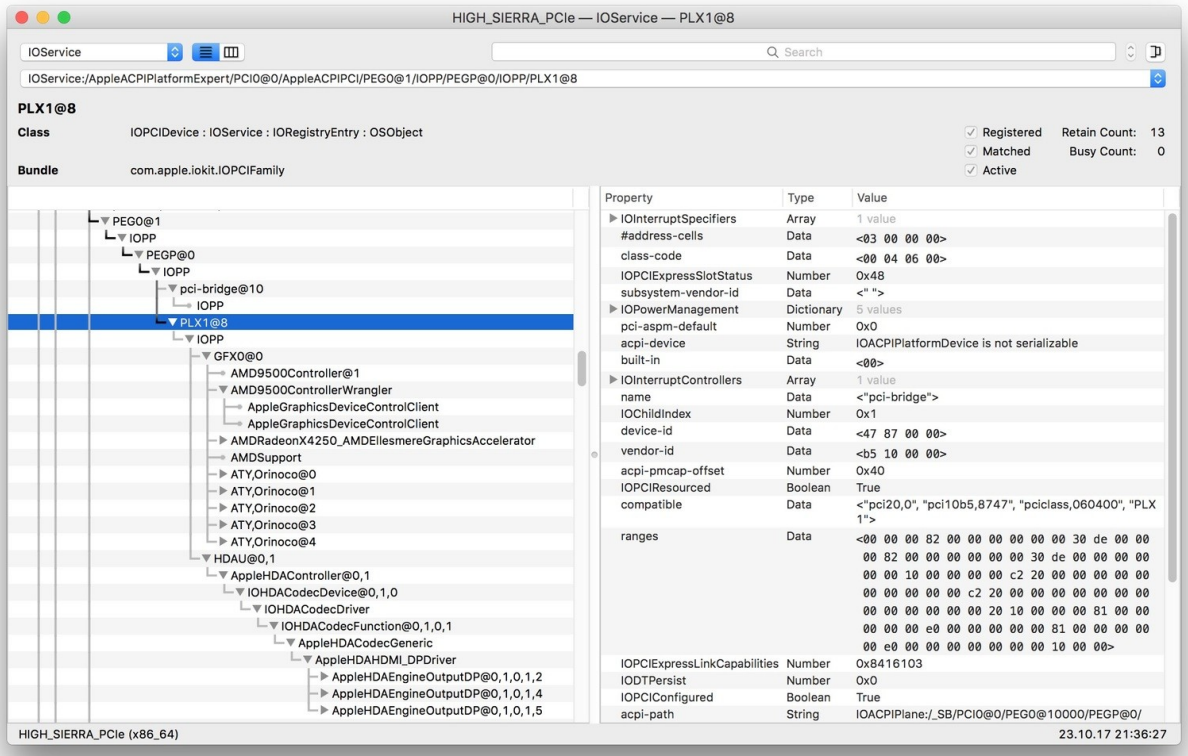
Beitrag von „Mork vom Ork“ vom 23. Oktober 2017, 23:30

Liebe AMD RADEON RX Nutzer,

die Coderin "Mieze" hat es **HIER** doch tatsächlich geschafft, daß wir ab sofort diverse Radeon Karten, die bislang nicht "OutOfTheBox" liefen, nun auch ohne die 2 Kexte "Lilu.kext" und "WhatEverGreen.kext" sowohl unter SIERRA (macOS 10.12.x), als auch unter HIGH SIERRA (macOS 10.13.x) *nativ* laufen lassen können.
Alles, was dazu benötigt wird, ist eine SSDT für eure Grafikkarte, den Patch von Mieze und ein paar kosmetische Einstellungen für das abschliessende Finetuning. Beides werde ich Euch so gut es eben geht hier näherbingen und erklären.

1.) Wir fangen mit der Erstellung der SSDT an. Dazu benötigen wir einen IORegisteryExlplorer Log unseres (noch mit LILU und WhatEverGreen) gestarteten Macs. Sollte dann in etwa so

aussehen:



Für die Ansicht habe ich mal bewusst die Unterpunkte eingefahren, damit die Übersicht gewahrt bleibt.
Hieraus können wir den folgenden Pfad des Steckplatzes unserer Grafikkarte entnehmen: in unserem Beispiel ist das "**_SB/PCI0/PEG0/PEGP/PLX1**". Das ist quasi der Startpfad in unserer kommenden SSDT.

2.) Als nächstes benötigen wir das Programm "MaciASL", mit dem wir unsere neue SSDT für die Grafikkarte erstellen. Programm öffnen, die sich daraufhin autom. öffnende DSDT schliessen und Command-N für ein neues Dokument.
In dieses kopieren wir als aller erstes die folgenden Zeilen:

Code

1. /*
2. * Intel ACPI Component Architecture
3. * AML/ASL+ Disassembler version 20161210-64(RM)
4. * Copyright (c) 2000 - 2016 Intel Corporation
5. *

```
6.  * Disassembling to non-symbolic legacy ASL operators
7.  *
8.  * Disassembly of iASLlpjYYo.aml, Mon Oct 23 21:46:36 2017
9.  *
10. * Original Table Header:
11. * Signature "SSDT"
12. * Length 0x0000014E (334)
13. * Revision 0x02
14. * Checksum 0x2A
15. * OEM ID "WEYW"
16. * OEM Table ID "SameHere"
17. * OEM Revision 0x00000000 (0)
18. * Compiler ID "INTL"
19. * Compiler Version 0x20161210 (538317328)
20. */
21. DefinitionBlock ("", "SSDT", 2, "WEYW", "SameHere", 0x00000000)
22. {
23.
24.
25. }
```

Alles anzeigen

Sicherlich ist Euch das "**WEYW**" und "**SameHere**" bereits aufgefallen. Taucht hier bereits jeweils 2x auf und kann von Euch durch eine beliebige Zeichenkombi ersetzt werden - wobei das "SameHere" bei mir immer auch der Dateiname der finalen SSDT ist, also in diesem Beispiel wäre das dann "SameHere.aml". Das kann aber auch jeder handhaben, wie er möchte.
Wichtig ist der Bereich zwischen den beiden Klammern "**{**" und "**}**", denn hier landet der eigentliche Code für die Grafikkarte und den Patch. In unserem oben genannten Beispiel sieht das ganze fertig eingerichtet dann so aus:

Spoiler anzeigen

Ich habe den Code, so gut es mir möglich war (bin selber **kein** Coder), versucht zu dokumentieren, damit man in etwa weiss, wofür welcher Part steht. Wenn MaciASL nun beim compelieren keine Fehler ausspuckt, speichern wir die fertige SSDT beispielsweise unter EFI/CLOVER/ACPI/patched als "AmdGfx.aml" ab und machen unseren ersten Test mit einem Neustart.
Hier sollte man sich ein wenig mit dem CLOVER Bootloader auskennen, dann kann man nämlich für den ersten Test die Lilu und WhatEverGreen kexte noch im Ordner

"EFI/CLOVER/kexts/Other" belassen und innerhalb von CLOVER deaktivieren, sodaß man, wenn etwas NICHT geht, ggf. nur die neue SSDT via CLOVER deaktiviert und der Rechner dann wieder hochfährt.

die hier erstellte SSDT findet ihr im Anhang als Download (sample_SSDT.dsl)

Ich habe mit dieser SSDT erfolgreich im BIOS die IGPU komplett deaktiviert und auch mein CSM steht von den Einstellungen her komplett auf UEFI (statt LEGACY) und ist ebenfalls komplett deaktiviert. Rechner startete erfolgreich, jedoch wurde einer meiner 3 Monitore nicht angesteuert - und hier kommen wir dann zum finalen Finetuning.

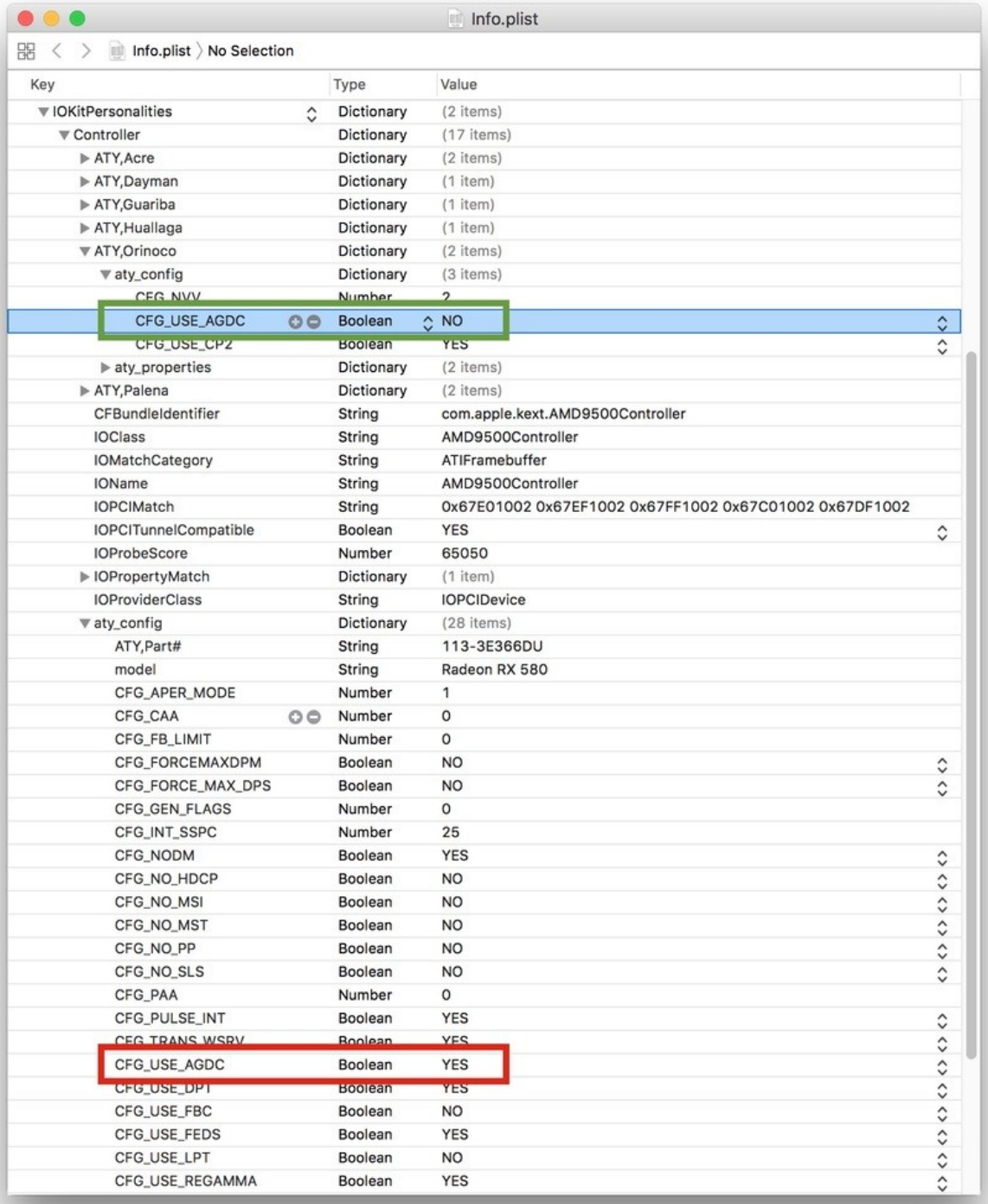
3.) FINETUNING

"Warum geht bei mir der eine Monitor nicht?" war gleichmal meine erste Frage. Unterstützt der patch etwa kein Multi-Monitor-Setup? DOCH, tut er - der Verursacher ist der altbekannte AGDC-Fix. Hier muss dafür gesorgt werden, daß der gewählte Framebuffer nicht auf die AGDC zugreift. Wie machen wir das? Ganz einfach:
Dazu sehen wir uns mal die "info.plist" unserer durch die Grafikkarte gewählte "AMD9xxxController.kext" an. In dieser findet man die durch diesen Controller nutzbaren Framebuffer plus div. Config und Properties Einstellungen. Und unter "aty_config" findet man unter anderem die folgenden Angaben:

<key>CFG_USE_AGDC</key>
<true/>

Setzen wir hier den Wert "<true/>" auf "<false/>", so wird AGDC von diesem Controller nicht mehr aufgerufen. Besser noch: sollte dieser Eintrag unter eurem genutzten Framebuffer stehen, ändert Ihr ihn dort von TRUE auf FALSE und lasst ihn unter den aty_config Einstellungen so stehen, wie er ist. Dann gilt die Einstellung von Euch nur für den von Euch eingesetzten Framebuffer. Ist dieser Eintrag für Euren Framebuffer nicht vorhanden, einfach an

der passenden Stelle einfügen. Sollte dann in etwa so aussehen:



GRÜN = disabled nur für den gewählten Framebuffer - **ROT** = enabled für den 9500Controller

Um die Datei (info.plist) zu ändern und die geänderte Version zu nutzen, OHNE die original AMD9xxxController.kext zu modfizieren, bin ich wie folgt vorgegangen:
! ich empfehle an dieser Stelle mal ganz bewusst die Installation der XTools, da hier der Umgang mit info.plist Dateien um ein vielfaches einfacher ist, als diese schonmal recht umfangreichen Dateien jedesmal mit einen Texteditor zu bearbeiten !

info.plist aus der AM9xxxController.kext auf den Desktop KOPIEREN (nicht verschieben). Kopie via Texteditor öffnen und nach folgenden Zeilen suchen und diese in die Zwischenablage kopieren: (*hier dient der Teilbereich aus der* ***AMD9500Controller.kext*** *als Beispiel*)

Spoiler anzeigen

info.plist eurer FakeSMC.kext öffnen und den in der Zwischenablage befindlichen Part nach folgender Codezeile einfügen:

<key>IOKitPersonalities</key>
<dict>

die FakeSMC.kext info.plist speichern - FERTIG. nun koennen wir in dem gerade eingefügten Bereich nach Lust und Laune "wildern" und Werte ändern, welche sich dann auf die Einstellungen für Eure AMD Grafikkarte auswirken. ! Doch **VORSICHT**: nicht jeder Wert mag es, beliebig geändert zu werden !
Und was ich heute rausgefunden habe, ist auch geil: man kann die "aty_config" sogar um solche Einträge aufstocken:

<key>ATY,Part#</key>
<string>113-3E366DU</string>
<key>model</key>
<string>Radeon RX 580</string>

So kann man dadurch beispielweise den Namen der Grafikkarte anpassen, oder (wie hier) die Part# der Grafikkarte ändern). Durch diese beiden Einträge ist mir die folgende Einstellung geglückt: statt normaler METAL Unterstützung unterstützt die Karte nun auch "Metal: Unterstützt, Funktionsset macOS GPUFamily1 v3" !!!
Ich kann hier also beliebig probieren und patchen, solange ich annähernd weiss, was ich da

tue. Und merke: es ist immer wichtig, einen Ersatz CLOVER-Bootstick parat zu haben, von dem aus man den Rechner wieder erfolgreich booten kann, um misslungene Einstellungen zurückzusetzen.

Nun wünsche ich viel Spass mit einer LILU.kext und WhatEverGreen.kext freien AMD RADEON Grafikkarte.

UPDATE #1: ich habe hier auch mal einen AMD9xxxControllerPatcher.kext angehängt. Ist ein DummyKext, sodass man nicht mehr die FakeSMC.kext patchen muss (wie oben im Text beschrieben), sondern einfach diesen DummyKext den eigenen Bedürfnissen anpassen (folgt in einer separaten Anleitung).
dieser DummyKext ist derzeit auf mein Setup abgestimmt und soll Euch nur als Richtlinie dienen. Wie gesagt: Anleitung zum anpassen dieses DummyKextes folgt in einer späteren Anleitung - bin zu müde das jetzt noch zusammenzufriemeln.

UPDATE #2: ich habe die SSDT nochmal angepasst:ich habe den Part mit

Code

```
1. Method (DTGP, 5, NotSerialized)
2. {
3. .
4. .
5. .
6. Return (Zero)
7. }
```

rausgeschmissen und dafür in Zeile 24 folgen Code eingefügt: **External (DTGP, MethodObj)**
Diese Zeile bewirkt das die DTGP Methode aus der systemeigenen DSDT, oder der von Euch gepatchten DSDT aufgerufen wird. Das macht den Code für die hier beschriebene SSDT übersichtlicher.

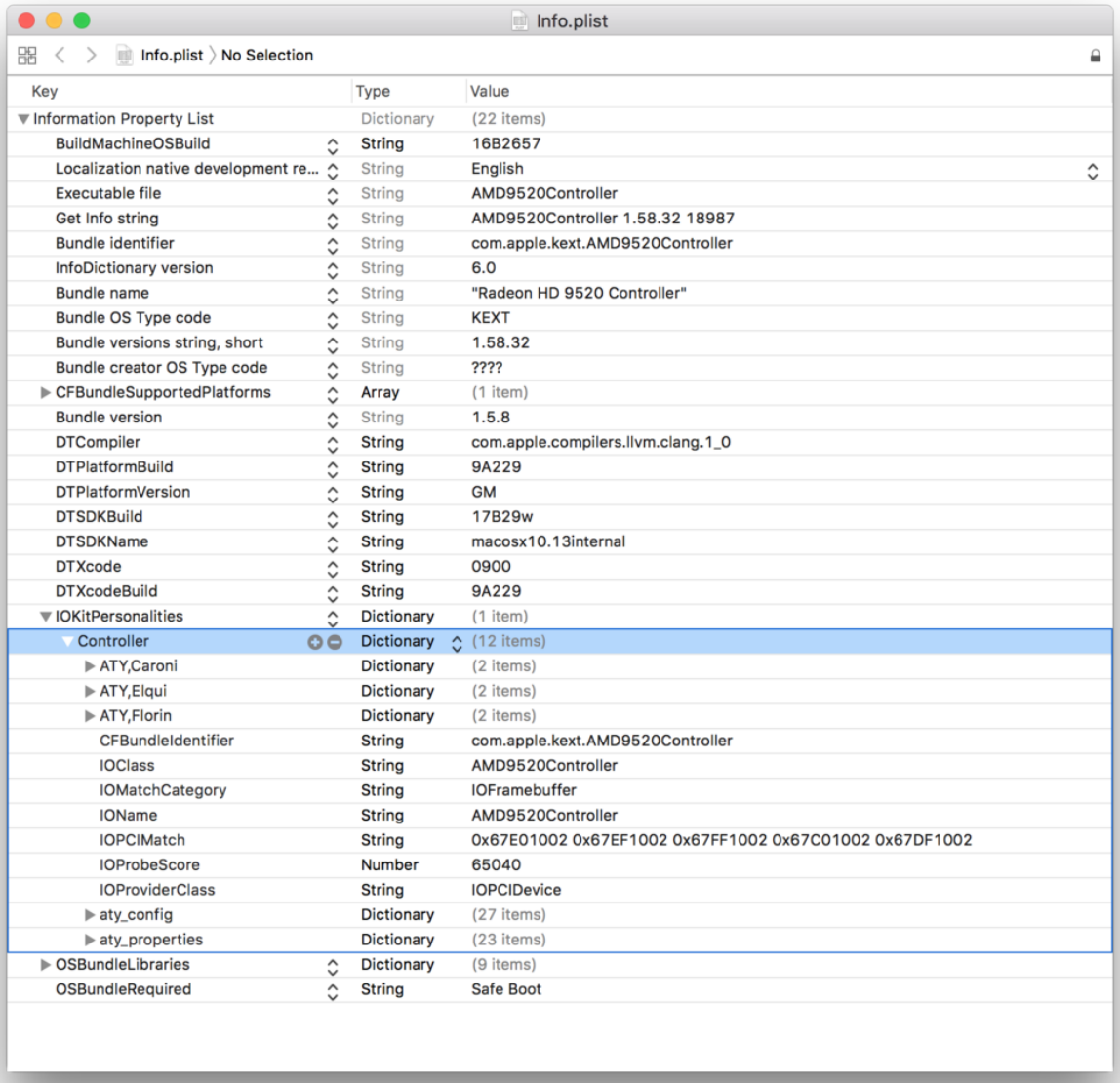
= = = = = = = = = = = = = = =

Anleitung zum Thema "Warum einen Dummy.kext einsetzen":

Aus der bisherigen Diskussion zu diesem Beitrag von mir ist relativ schnell zu erkennen, daß viele von Euch sich fragen, wozu der Dummy.kext gut sein soll.
Ich möchte es Euch an dieser Stelle gerne näher bringen - habe jedoch vorher einen guten Rat an jeden von Euch: installiert Euch unbedingt die **Apple XCode** (auch wenn diese im Download 5GB+ gross sind) - **ES LOHNT SICH**, da die Bearbeitung der im folgenden beschrieben Routinen damit um ein vielfaches einfacher umzusetzen sind.

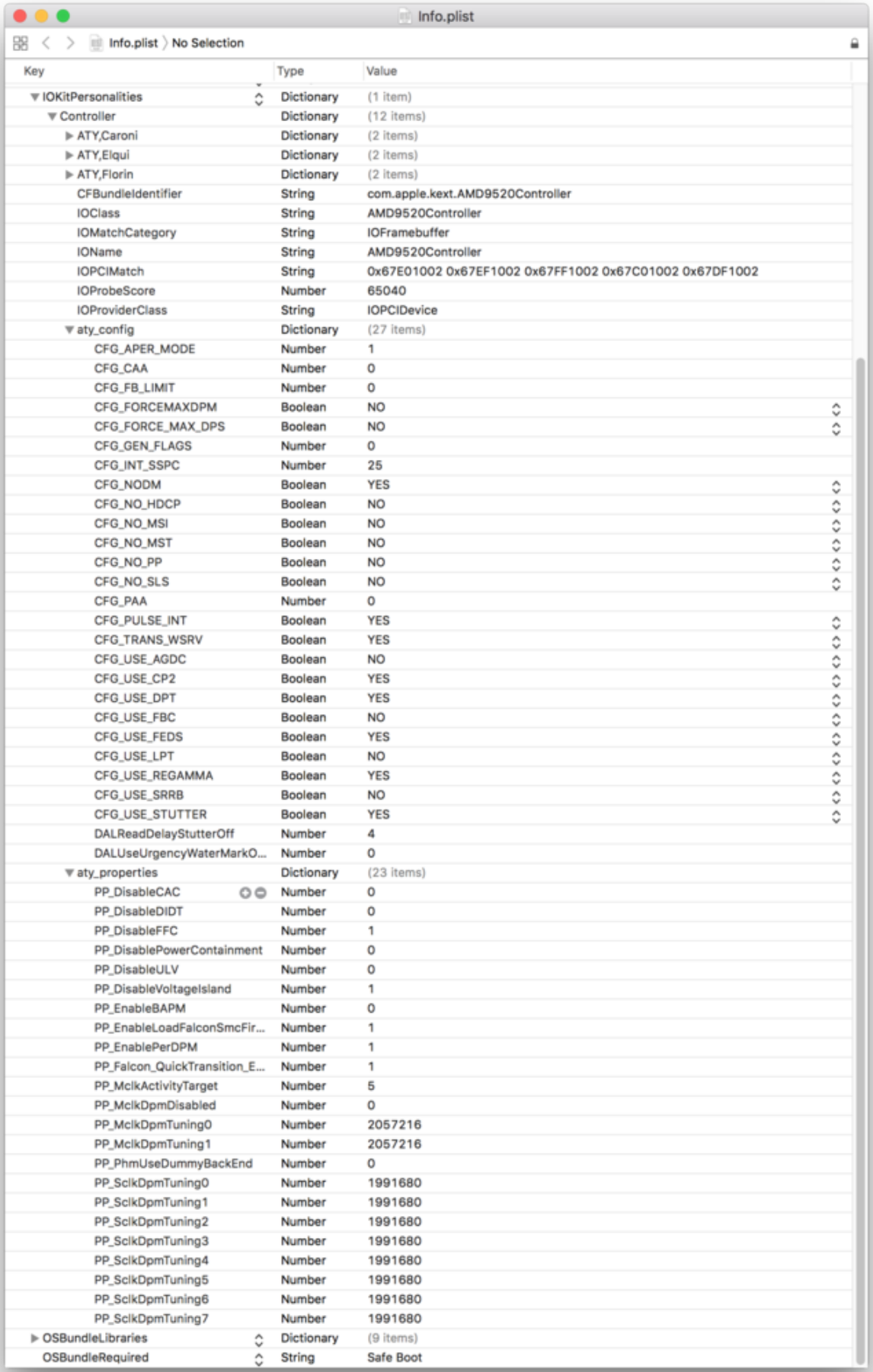
Der Grundgedanke von mir für einen Dummy.kext ist der folgende: die AMD9xxxController Kexte enthalten jeweils eine "config.plist", die dazu dient, diverse Werte, die der Controller nutzt, vorzudefinieren. Bedeutet im Umkehrschluss: man kann die Werte ändern, bzw. nach Bedarf anpassen. Damit man eine solche Anpassung nicht in der originalen Apple Kext macht, welche dann ein Update auf Version macOS 10.x.x nicht überleben würde, weil Apple auch an den AMD9xxxController Kexten etwas modifiziert hat, basteln wir uns einfach einen Dummy.kext für den von unserer AMD GFX-Karte genutzten Controller. Ihr findet im Anhang an diesen Post einen solchen Dummy.kext, den Ihr (hoffentlich 😉) nach dieser Anleitung an Eure Bedürfnisse anpassen könnt.

Schauen wir uns also mal die "info.plist" der original Apple AMD9520Controller.kext an:



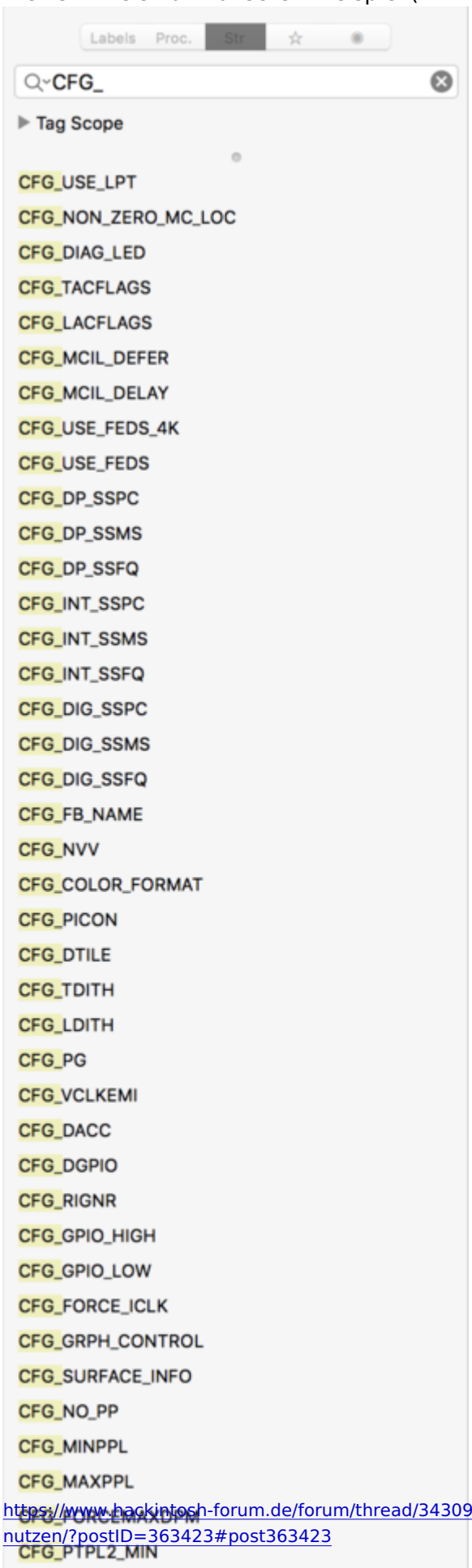
Der in diesem Screenshot umrandete Teil ist der von mir bereits angesprochene Teil, in dem die verschiedensten Settings für diesen Controller festgelegt sind. Was einem zuerst ins Auge fällt ist, daß dieser Controller die Framebuffer CARONI, ELQUI und FLORIN unterstützt. Desweiteren wird dieser Controller nur von AMD Karten genutzt, deren Device-ID 67E0, 67EF, 67FF, 67C0 oder 67DF lauten. Damit Ihr also den korrekten Controller patcht, ist es wichtig, das ihr wisst, welchen Controller eure Grafikkarte beim booten wählt. Das findet Ihr raus, indem ihr Euren Hackintosh bootet und dann mittels IORegistryExplorer den Bereich Eurer genutzten Grafikkarte aufruft (siehe erster Screenshot am Anfang dieses Beitrags). Die Einstellungen von "CFBundleIdentifier" bis "IOProviderClass" bleiben bitte **unberührt** und werden **keinesfalls** geändert - andernfalls wird die finale Fassung Eures Dummy-Kexts nicht funktionieren.

Schauen wir uns mal die aufgeklappten Einstellungen von "aty_config" und "aty_properties" an:

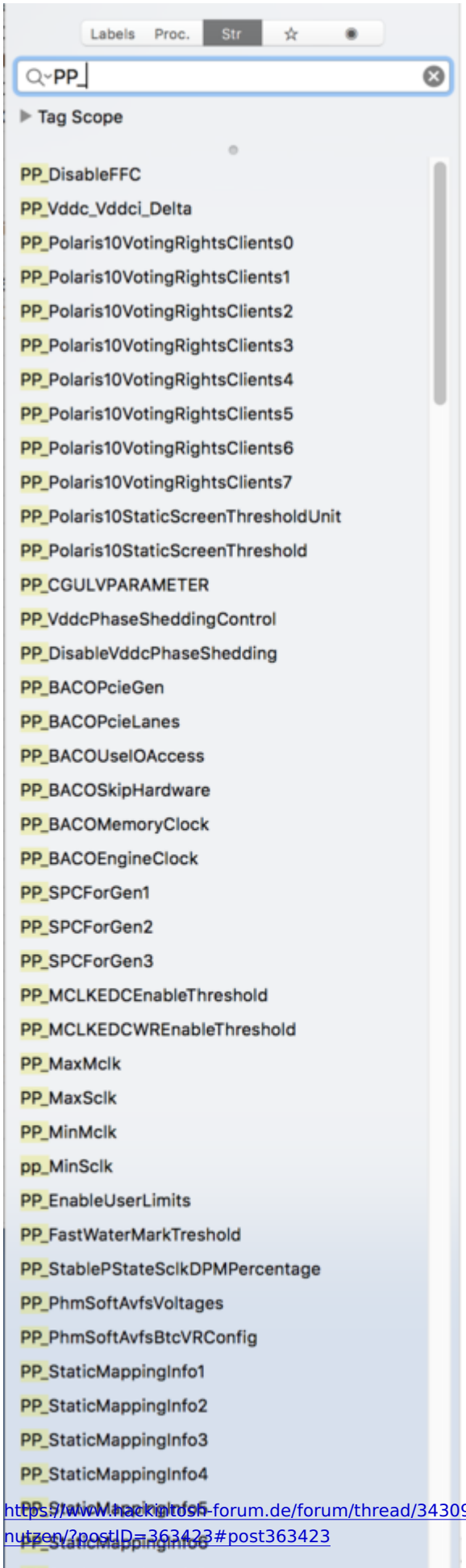


WOW, viele Einstellmöglichkeiten! Doch das ist nur ein Bruchteil dessen, was der Controller wirklich zu bieten hat. Fragt mich bitte nicht, was jeder einzelne Wert zu bedeuten hat, denn ICH WEISS ES NICHT. Einige davon teilweise selbsterklärend (dazu komme ich noch), andere deuten weder vom Namen noch sonst irgendwie auf deren Funktion! Und für die experimentierfreudigen unter Euch zeigt sich hier auch schon der erste Vorteil dieses Dummy Kextes: haben wir einen Wert geändert und der Rechner bootet nicht mehr sauber oder womöglich gar nicht mehr, können wir diesen Dummy Kext einfach im CLOVER Bootmenü deaktivieren, sodaß der Mac wieder auf den original Apple AMDController zugreift und somit wieder die ursprünglichen Werte geladen werden. Zweiter Vorteil: ich kann die "CFG_"- und "PP_"-Einstellungen um viele weitere Werte ergänzen. Jetzt fragt Ihr Euch sicherlich: um welche denn zum Beispiel? Jeder Controller beinhaltet eine Liste mit ALLEN von ihm unterstützten

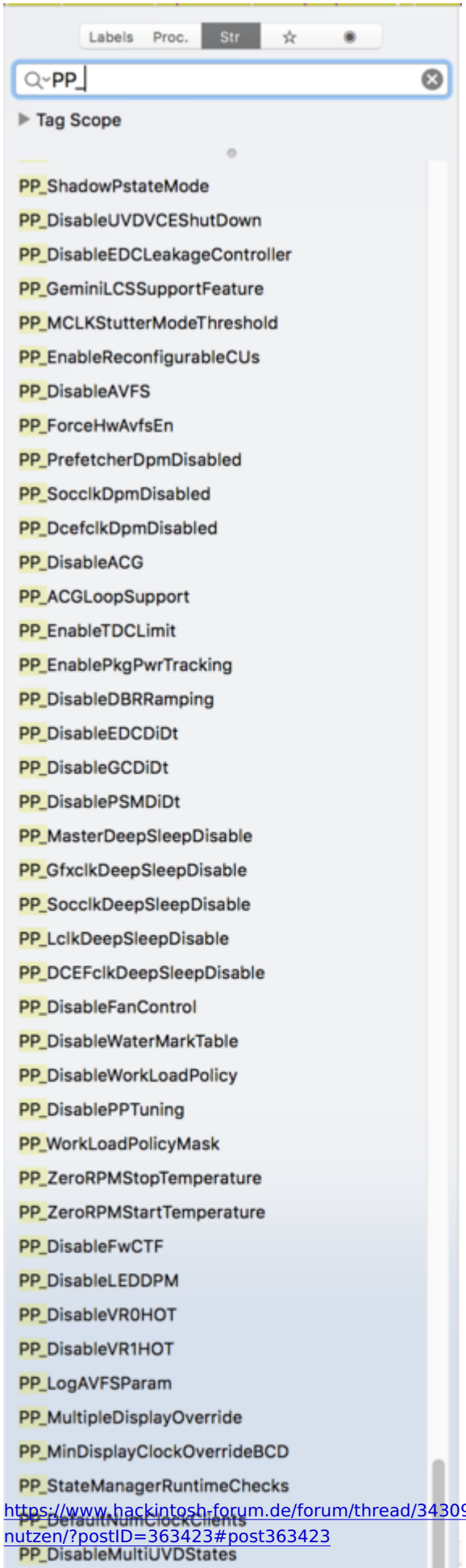
Werten. Die sind in unserem Beispiel (AMD9520Controller.kext) dann so aus:



nur mal die "**CFG_**" Werte. Noch mehr Werte findet Ihr für die "**PP_**"-Einstellungen:

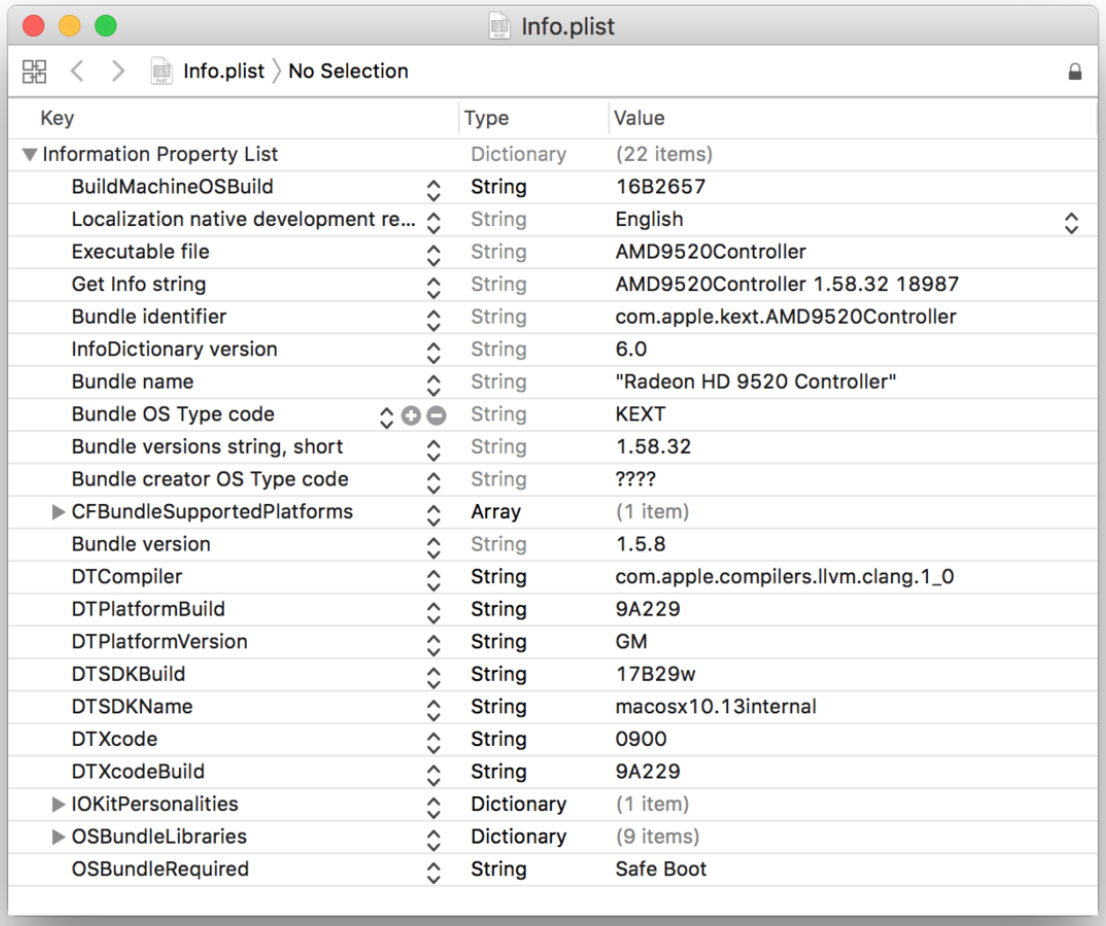

Labels
Proc.
Str
PP_
Tag Scope
PP_DisableFFC
PP_Vddc_Vddci_Delta
PP_Polaris10VotingRightsClients0
PP_Polaris10VotingRightsClients1
PP_Polaris10VotingRightsClients2
PP_Polaris10VotingRightsClients3
PP_Polaris10VotingRightsClients4
PP_Polaris10VotingRightsClients5
PP_Polaris10VotingRightsClients6
PP_Polaris10VotingRightsClients7
PP_Polaris10StaticScreenThresholdUnit
PP_Polaris10StaticScreenThreshold
PP_CGULVPARAMETER
PP_VddcPhaseSheddingControl
PP_DisableVddcPhaseShedding
PP_BACOPcieGen
PP_BACOPcieLanes
PP_BACOUseIOAccess
PP_BACOSkipHardware
PP_BACOMemoryClock
PP_BACOEngineClock
PP_SPCForGen1
PP_SPCForGen2
PP_SPCForGen3
PP_MCLKEDCEnableThreshold
PP_MCLKEDCWREnableThreshold
PP_MaxMclk
PP_MaxSclk
PP_MinMclk
pp_MinSclk
PP_EnableUserLimits
PP_FastWaterMarkTreshold
PP_StablePStateSclkDPMPercentage
PP_PhmSoftAvfsVoltages
PP_PhmSoftAvfsBtcVRConfig
PP_StaticMappingInfo1
PP_StaticMappingInfo2
PP_StaticMappingInfo3
PP_StaticMappingInfo4

bis hin zu

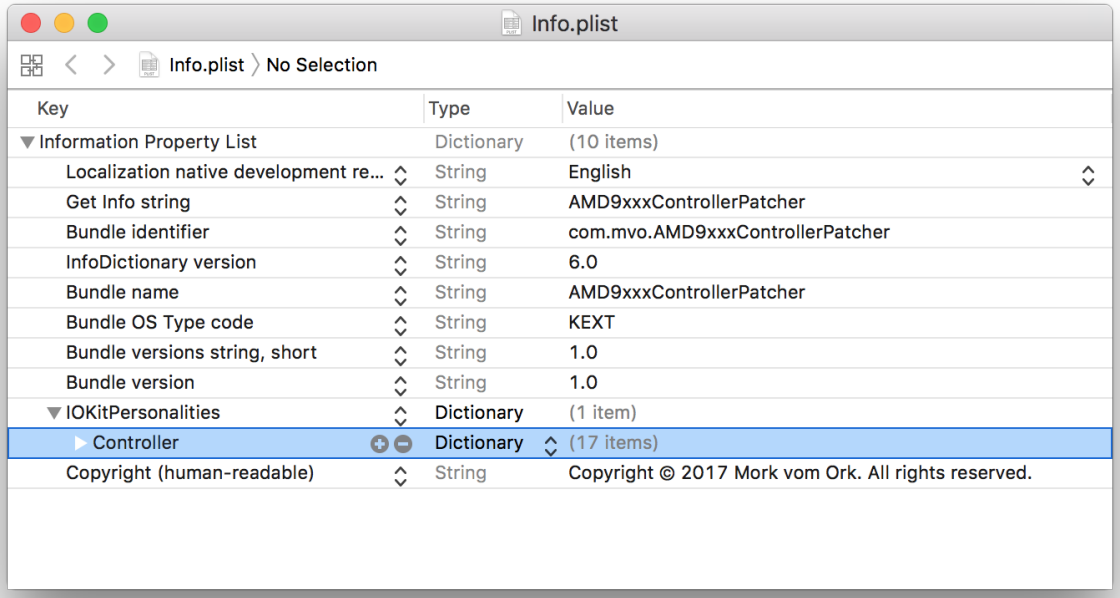

Labels
Proc.
Str
PP_
Tag Scope
PP_ShadowPstateMode
PP_DisableUVDVCEShutDown
PP_DisableEDCLeakageController
PP_GeminiLCSSupportFeature
PP_MCLKStutterModeThreshold
PP_EnableReconfigurableCUs
PP_DisableAVFS
PP_ForceHwAvfsEn
PP_PrefetcherDpmDisabled
PP_SocclkDpmDisabled
PP_DcefclkDpmDisabled
PP_DisableACG
PP_ACGLoopSupport
PP_EnableTDCLimit
PP_EnablePkgPwrTracking
PP_DisableDBRRamping
PP_DisableEDCDiDt
PP_DisableGCDiDt
PP_DisablePSMDiDt
PP_MasterDeepSleepDisable
PP_GfxclkDeepSleepDisable
PP_SocclkDeepSleepDisable
PP_LclkDeepSleepDisable
PP_DCEFclkDeepSleepDisable
PP_DisableFanControl
PP_DisableWaterMarkTable
PP_DisableWorkLoadPolicy
PP_DisablePPTuning
PP_WorkLoadPolicyMask
PP_ZeroRPMStopTemperature
PP_ZeroRPMStartTemperature
PP_DisableFwCTF
PP_DisableLEDDPM
PP_DisableVR0HOT
PP_DisableVR1HOT
PP_LogAVFSParam
PP_MultipleDisplayOverride
PP_MinDisplayClockOverrideBCD
PP_StateManagerRuntimeChecks
PP_DefaultNumClockClients
PP_DisableMultiUVDStates

Ihr seht: hier kann man sich (wenn man denn will) richtig *austoben*.

Nun, wie erstellt Ihr einen Dummy.Kext, der auf Euren genutzten Controller zugeschnitten ist? Ladet Euch zunächst den hier angehängten Dummy Kext "AMD9xxxControllerPatcher.kext.zip" runter und entpackt diesen beispielsweise auf dem Desktop. Öffnet diesen per Rechts-Klick - "Packetinhalt zeigen". Im darin befindlichen Contents-Ordner findet ihr die zu modfizierende "info.plist". Das selbe macht Ihr mit einer **Kopie** der von Euch ermittelten AMD9xxxController.kext aus "System/Library/Extensions". Seit ihr meinem Rat gefolgt und habt Euch XCODE installiert, sollte sich euch folgendes Bild zeigen:



Screenshot der original Apple AMD9520Controller.kext "info.plist"



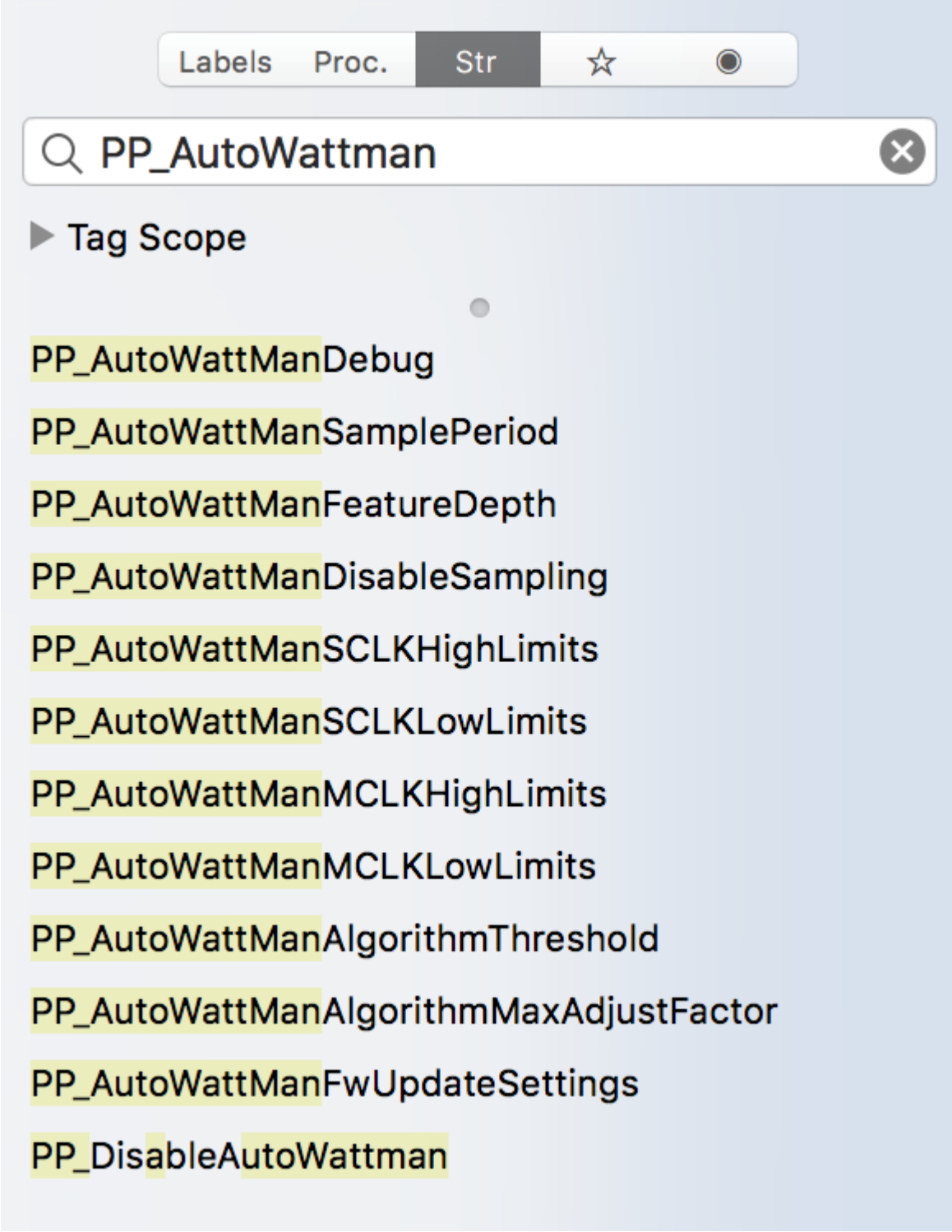
Screenshot der Dummy Kext "info.plist"

Klickt das Dreieck vor "IOKitPersonalities" des von Euch ermittelten Controllers an und anschliessend auf "Controller", dann Command-C zum kopieren in die Zwischenablage und schliesst die "info.plist" des Apple Controller.kext. Öffnet nun den Dummy.kext auf die selbe Weise (Rechtsklick blabla) und markiert wieder den Eintrag "Controller", löscht diesen durch drücken der Backspace-Taste, markiert dann "IOKitPersonalities" und drückt Command-V, um den Inhalt der Zwischenablge ("Controller" der original Datei) unter dem markierten "IOKitPersonalities" einzufügen. Dann sollte die Dummy.kext nun so aussehen, wie im Screenshot direkt über diesen Zeilen. Jetzt noch Command-S zum sichern der "info.plist" und umbenennen der ".kext" Datei wie es euch gefällt, beispielsweise in "AMD9520ControllerPatcher.kext" (um bei unserem Beispiel zu bleiben).

Der Grundstock ist gelegt, der nutzbare Dummy.kext ist fertig und kann unter "EFI/CLOVER/kexts/Others/" zum Einsatz kommen. Jetzt kann man anfangen, mit den einzelnen Werten zu "spielen" und diese ggf. zu ändern. Neustarten, schauen ob der Rechner noch bootet und gucken, was sich geändert hat. Aber ich warne an dieser Stelle noch einmal: das sollten wirklich nur erfahrene User tun, denn ich weiss aus eigener Erfahrung, das nicht jede der CFG_ oder PP_ - Einstellungen beliebige Werte verträgt.

Aber, und das ist das geniale: es gibt Settings, die von Hause aus auf "enabled" = 1 oder "disabled" = 0 stehen und bei denen man einfach mal durchprobieren kann, was passiert, wenn man diese "enabled" oder "disabled". Ich erkläre Euch das an Hand eines Beispiels:

Es gibt folgenden Wert für die "aty_properties":

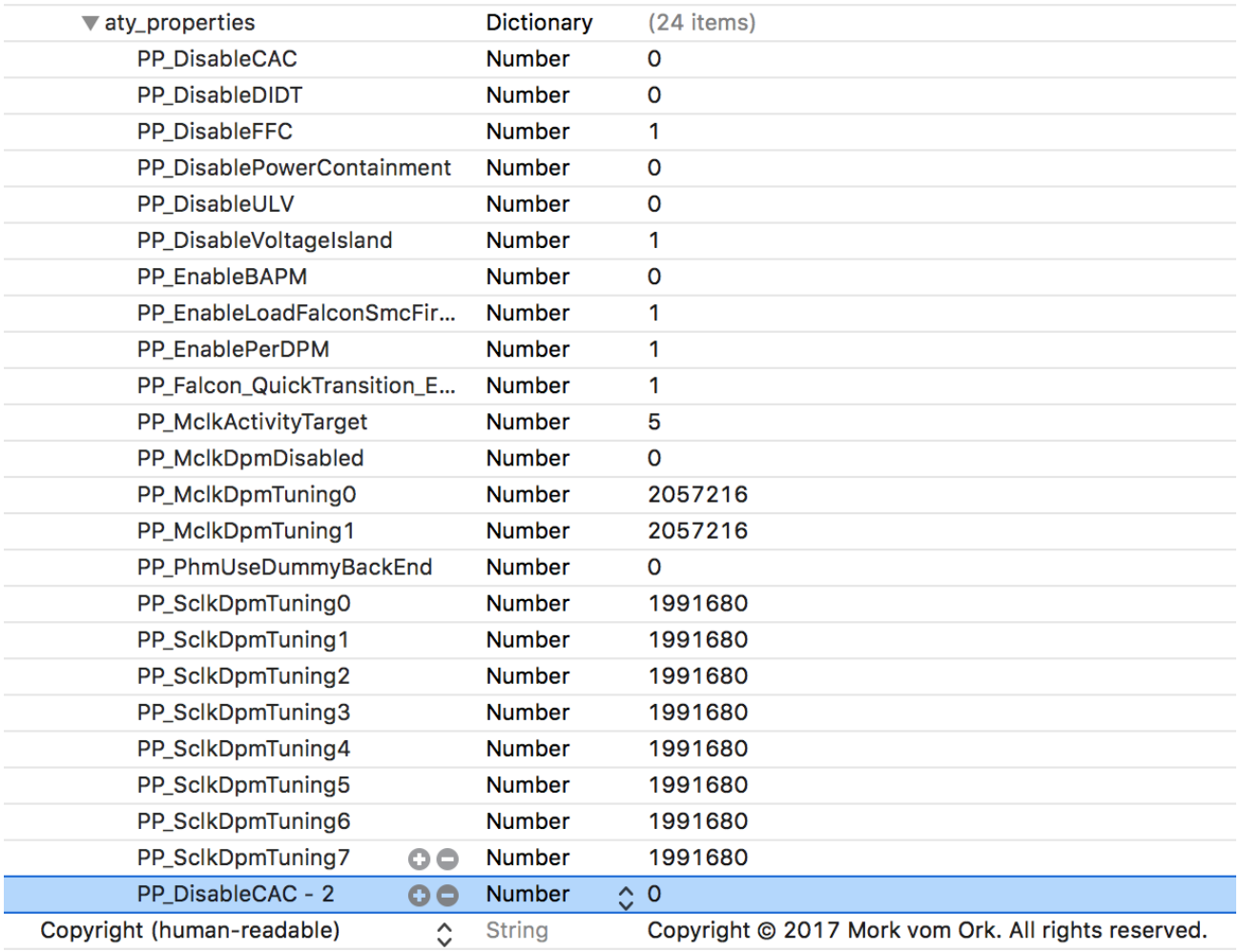


Wer sich mit den AMD-Treibern der RX-Karten unter WINDOWS ein wenig auskennt, der weiss, daß WATTMAN für die GPU-temperaturabhängige Lüftersteuerung zuständig ist. Unter Apple steht die Einstellung "PP_DisableAutoWattman" auf "true" = "1", was bedeutet, sie ist

standardmäßig deaktiviert (disabled = true).
Nun können wir hergehen, und bei unserem Dummy.kext diesen Wert hinzufügen und in"PP_DisbledAutoWattman = 0" ändern und sie somit aktivieren (Disabled = false). Klingt zunächst erstmal kompliziert, weil warum wird diese Funktion mit dem Wert "0" gesetzt, wenn doch "0" = AUS und "1" gleich EIN bedeutet? Weil die
Funktion "PP_Disable..." heisst. Es gibt auch Funktionen wie "PP_Enable..." und da zählt dann tatsächlich: "0" = AUS, "1" gleich EIN. Also merke:

"PP_Disable..." dann "0" = die Funktion wird genutzt, "1" = die Funktion wird NICHT genutzt
"PP_Enable..." dann "1" = die Funktion wird genutzt, "0" = die Funktion wird NICHT genutzt

Wie bekommen wir diese Funktion nun in unsere Dummy.kext "info.plist" ?
Öffnet Eure Dummy.kext "info.plist", klickt auf das Dreieck vor "aty_properties", klickt auf irgendeine vorhandenen "PP_..." Eintrag und kopiert diesen via Command-C in die Zwischenablage. Markiert nun den letzten "PP_"-Eintrag und drückt Command-V zum Einfügen der Zwischenablage:



aty_properties	Dictionary	(24 items)
PP_DisableCAC	Number	0
PP_DisableDIDT	Number	0
PP_DisableFFC	Number	1
PP_DisablePowerContainment	Number	0
PP_DisableULV	Number	0
PP_DisableVoltageIsland	Number	1
PP_EnableBAPM	Number	0
PP_EnableLoadFalconSmcFir...	Number	1
PP_EnablePerDPM	Number	1
PP_Falcon_QuickTransition_E...	Number	1
PP_MclkActivityTarget	Number	5
PP_MclkDpmDisabled	Number	0
PP_MclkDpmTuning0	Number	2057216
PP_MclkDpmTuning1	Number	2057216
PP_PhmUseDummyBackEnd	Number	0
PP_SclkDpmTuning0	Number	1991680
PP_SclkDpmTuning1	Number	1991680
PP_SclkDpmTuning2	Number	1991680
PP_SclkDpmTuning3	Number	1991680
PP_SclkDpmTuning4	Number	1991680
PP_SclkDpmTuning5	Number	1991680
PP_SclkDpmTuning6	Number	1991680
PP_SclkDpmTuning7	Number	1991680
PP_DisableCAC - 2	Number	0
Copyright (human-readable)	String	Copyright © 2017 Mork vom Ork. All rights reserved.

Jetzt doppeklick auf den soeben eingefügten Eintrag (vorne auf den Namen, hier eben "PP_DisableCAC - 2") und ändert diesen in "PP_DisableAutoWattman", die "0" bleibt stehen, damit WATTMAN genutzt wird und eben NICHT gedisabled wird. FERTIG.

▼ aty_properties	Dictionary	(24 items)
PP_DisableCAC	Number	0
PP_DisableDIDT	Number	0
PP_DisableFFC	Number	1
PP_DisablePowerContainment	Number	0
PP_DisableULV	Number	0
PP_DisableVoltageIsland	Number	1
PP_EnableBAPM	Number	0
PP_EnableLoadFalconSmcFir...	Number	1
PP_EnablePerDPM	Number	1
PP_Falcon_QuickTransition_E...	Number	1
PP_MclkActivityTarget	Number	5
PP_MclkDpmDisabled	Number	0
PP_MclkDpmTuning0	Number	2057216
PP_MclkDpmTuning1	Number	2057216
PP_PhmUseDummyBackEnd	Number	0
PP_SclkDpmTuning0	Number	1991680
PP_SclkDpmTuning1	Number	1991680
PP_SclkDpmTuning2	Number	1991680
PP_SclkDpmTuning3	Number	1991680
PP_SclkDpmTuning4	Number	1991680
PP_SclkDpmTuning5	Number	1991680
PP_SclkDpmTuning6	Number	1991680
PP_SclkDpmTuning7	Number	1991680
PP_DisableAutoWattman	Number	0
Copyright (human-readable)	String	Copyright © 2017 Mork vom Ork. All rights reserved.

Speichern - rebooten und über eine temperaturgeregelte Lüftersteuerung der Lüfter Eurer Grafikkarte freuen.
! ACHTUNG ! - das hier genannte Beispiel gilt NUR für AMD Karten der RX-Serie. Karten der HD-Serie unterstützen kein WATTMANN.

Das war es auch schon mit der Erklärung, warum eine Dummy.kext von Vorteil sein kann. Letzlich entscheidest DU, ob DU eine nutzt oder nicht. Viel Spass für den Fall daß...

EDIT:
Der User "modzilla" hat mich auf folgenden Umstand aufmerksam gemacht, welchem einem das Nutzen einer Dummy.kext erspart. Die hier soeben beschrieben Werte kann man auch direkt in der erstellten SSDT einsetzen. Das Ganze muss dann ungefähr so aussehen:

> Zitat von modzilla
>
> Hey
> Mork vom Ork
> ich habe ne Methode gefunden, um die ControllerPatcher Kext zu umgehen, man kann nämlich alles in der DSDT respektive SSDT unterbringen, die angefügt DSDT wurde von revunix getestet und für funktional befunden
>
> Man muss halt das einfach in der Injection hinzufügen:
>
> Code
>
> ```
> 1. ...
> 2. "CFG,CFG_USE_AGDC",
> 3. Buffer (One)
> 4. {
> 5. 0x00
> 6. },
> 7.
> 8.
> 9. "PP,PP_DisableAutoWattman",
> 10. Buffer (One)
> 11. {
> 12. 0x00
> 13. },
> 14. ...
> ```
>
> Alles anzeigen
> Alles anzeigen

Einfach in dem Bereich hinzufügen, an dem in Eurer SSDT die Werte für den Framebuffer, den Slot oder den Kartennamen steht.