CPUFriend Guide, HWP & Speedstep: X86PlatformPlugin vs ACPI_SMC_PlatformPlugin

Beitrag von „kuckkuck“ vom 26. April 2020, 00:49

CPU Friend Guide

X86PlatformPlugin Customization und Anpassung – Perfektes Speedstepping

Vor und Nachteile des X86PlatformPlugin vs ACPI_SMC_PlatformPlugin in der Diskussion: plugin-type=1 vs plugin-type=0

In diesem ausführlichen Guide geht es um absolutes Feintuning – um das Ziel perfektes Apple-konformes CPU-Speedstepping und Frequenz-/Powermanagement zu erreichen. Ich will ein paar Anpassungsmöglichkeiten und Herangehensweisen für Speedstepping erklären und durchgehen, mitunter wird es aber nicht besonders unkompliziert und nutzerfreundlich.

Das am häufigsten genannte Indiz für korrektes PowerManagement ist meistens das TaktFrequenz Verhalten der CPU. Dies kann unter macOS beispielsweise mit dem Intel Power Gadget ausgelesen werden, hierbei gilt jedoch zu beachten, dass der angezeigte Graph häufig und schnell die Realität ein wenig verzerrt. Deswegen ist es wichtig beim Testen von Speedstep im Intel Power Gadget das Log zu aktivieren und die dabei erzeugte Tabelle im Nachhinein auszuwerten. Hier werden die "echten" Frequenzen angezeigt.

Vereinfacht gesagt stellt macOS zwei Treiber zur Verfügung, die genutzt werden können um allgemein das Powermanagement zu kontrollieren: ACPI_SMC_PlatformPlugin und X86PlatformPlugin. Beide Kexts liegen unter /System/Library/Extensions/IOPlatformPluginFamily.kext/Contents/PlugIns.

Das ACPI_SMC_PlatformPlugin stammt zwar aus Zeiten vor Ivy Bridge Prozessoren, hat aber die schöne Eigenschaft, dass es die benötigten Daten über die CPU größtenteils aus dem ACPI ließt, was in den meisten Fällen dazu führt, dass viele CPU Frequenzen verfügbar sind und die CPU "schön" taktet. Das ACPI_SMC_PlatformPlugin ist für aktuelle Prozessoren und Hardware jedoch nicht wirklich konzipiert, sondern nur ein Fallback. Das X86PlatformPlugin ist das korrekte Plugin für aktuelle CPUs und Chipsets. Das Plugin sowie die bekannte Technologie XCPM (Xnu Power Management) ist tief in den Kernel Integriert und hat Einfluss auf viele

Verschiedene Bereiche:

> Zitat von kuckkuck
>
> Xcpm ist tief in den Kernel Integriert und hat weitreichende Auswirkungen und Dependenzen. Das X86PlatformPlugin sorgt (von seiner Funktion her) für korrektes Speedstepping, aber hat Einfluss auf viel mehr als nur das, wie zB System Capabilities, Sleep, Wake, DVFS, HWP, (AGPM), (Temp-/FanManagement), Shutdown-, Reboot-, Reset- und PowerFailure Monitoring, ME Events, Energiehaushalt/PowerProfiles, Boot Geschwindigkeit, (Systemeinstellungen), etc.

GUIDE:

Das X86PlatformPlugin lädt nur, wenn macOS für die installierte CPU das Property `plugin-type=1` bereitgestellt wird. (`plugin-type=0` sorgt dafür, dass das ACPI_SMC_PlatformPlugin lädt)

Um `plugin-type=1` zu injecten können wir eine SSDT benutzen, ein gutes Template hierfür ist diese SSDT-PLUG. (Für IvyBridge CPUs hier entlang)

Im Gegensatz zum ACPI_SMC_PlatformPlugin holt sich das X86PlatformPlugin seine Informationen nicht aus dem ACPI, sondern aus SMBios-abhängigen Profilen. In /System/Library/Extensions/IOPlatformPluginFamily.kext/Contents/PlugIns/X86PlatformPlugin.kext/Contents/R sind für sämtliche Board-IDs Profile mit Properties und Daten hinterlegt. Zu diesen Daten gehören ebenfalls die `FrequencyVectors` (siehe Plists), welche das Speedstepping der CPU beeinflussen, und mit genau diesen Frequency-Vectors wollen wir uns im Laufe des Guides beschäftigen.

Da die Profile also SMBios abhängig sind, müssen wir jetzt erstmal ein SMBios finden, dass am besten zu unserer CPU passt (wir werden unser per Clover/OC/Oz injectetes SMBios nicht ändern! Bitte weiterlesen). Es empfiehlt sich MacTracker zu nutzen. Ihr solltet unbedingt einen Mac finden, der die gleiche CPU Generation besitzt (zB Kaby Lake). Die genauen CPU-Modell Bezeichnungen können von Mac zu eigener Hardware variieren, also am besten etwas möglichst Nahes wählen. Bei zB einem i5-8265U bietet sich das MacBookPro15,4 SMBios mit i5-8257U an, denn beide Prozessoren haben laut Intel Website die gleiche maximale Taktrate von 3,9 GHz (Wichtig! Wenn nicht möglich, dann möglichst nah) und gehören der gleichen Prozessor Generation an. Wir merken uns "MacBookPro15,4" und/oder die passende Board-ID von zB hier.

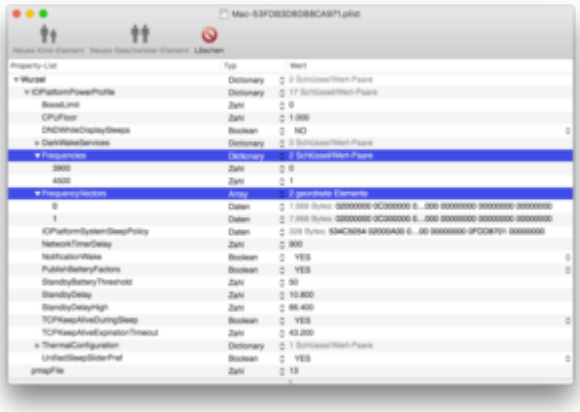
Pike R. Alpha hat Teile der Frequency-Vectors entschlüsselt und ein Skript namens freqVectorsEdit

geschrieben. Dieses Script wollen wir jetzt erstmal für den Anfang benutzen, da es uns hilfreiche Informationen gibt und erste sinnvolle Anpassungen vornimmt, auf die ich jetzt nicht explizit eingehen will.

Also, Script herunterladen, ggf ausführbar machen (chmod, chown) und auf ein geöffnetes Terminal-Fenster ziehen. Dahinter setzen wir noch die Debug Anweisung, sodass es so aussieht: `./freqVectorsEdit.sh -d 1` und führen den Befehl aus.

Das Script analysiert jetzt alle "Mac-... .plists" aus dem X86PlatformPlugin und stellt uns verschiedene Informationen bereit, Mac-Modelle mit passender maximal Frequenz werden automatisch blau markiert. Warum ist das wichtig?

In den Profilen aus dem X86PlatformPlugin sind häufig mehrere FrequencyVectors hinterlegt. macOS wählt die entsprechende Spalte je nach Maximalfrequenz der CPU aus. Beispiel /System/Library/Extensions/IOPlatformPluginFamily.kext/Contents/PlugIns/X86PlatformPlugin.kext/Contents/R
53FDB3D8DB8CA971.plist



Hat unsere CPU einen MaxClock von 3900mHz wird Eintrag 0 unter FrequencyVectors benutzt, bei 4500mHz der andere.

Mit den gegebenen Daten haben wir jetzt also das korrekte CPU SMBios ausgewählt. Jetzt machen wir folgendes:

1. SIP deaktivieren (per config.plist Eintrag oder Terminal) -> Neustarten
2. Aus /System/Library/Extensions/IOPlatformPluginFamily.kext/Contents/PlugIns/X86PlatformPlugin.kext/Cont folgende Plists finden und als Backup auf dem Schreibtisch sichern: 1. die Plist unserer aktuell gesetzten SMBios Board ID; 2. die Plist unserer für die CPU gewollten Board ID (unter Catalina+ mit KextUpdater -> Werkzeuge die Systempartition als read/write mounten)
3. Danach freqVectorsEdit im Terminal ausführen (siehe oben) und die Plist die wir für die CPU wollen per Nummer auswählen (entsprechend unseren Recherchen)

4. freqVectorsEdit patcht jetzt die entsprechende Plist im X86PlatformPlugin und überschreibt die Alte (welche wir gebackupt haben)
5. Danach unsere Plist aus /System/Library/Extensions/IOPlatformPluginFamily.kext/Contents/PlugIns/X86PlatformPlugin.kext/Cont kopieren und sichern. Wir wollen genau das Produkt von freqVectorsEdit, wir wollen aber nicht, dass das X86PlatformPlugin modifiziert wurde
6. Also kopieren wir unsere gebackupte Plist von Schritt 1 wieder zurück an ihren alten Ort, sodass dort alles wie eh und jeh ist
7. Mit KextUpdater unter Werkzeuge Kextcache neu aufbauen und Rechte reparieren

Wir haben jetzt also eine X86PlatformPlugin Plist – von freqVectorsEdit gepatcht – in einem eigenen Ordner vorliegen. Weiter gehts.

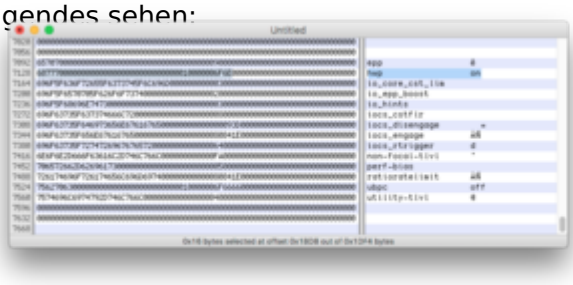
Wir öffnen diese Plist mit einem Plist Editor und beachten 2 Dinge:

1. Unter Frequencies steht ein Eintrag mit genau der Maximalfrequenz unserer installierten CPU. Wenn nicht, passen wir den nähesten Eintrag auf unseren MaxClock an und merken uns den Index.
2. Wer gerne 2 (statt einem) Slider in Systemeinstellungen -> Energie Sparen haben will, setzt ganz unten `UnifiedSleepSliderPref` auf `false`

Jetzt suchen wir unter FrequencyVectors den Eintrag raus, der unserem Index/Takt bei Frequencies entspricht. Der i5-8265U MaxClock ist 3900 und das steht auf Index 0 (siehe Bild oben), also wähle ich die Daten bei FrequencyVectors Index 0 aus.

Wir kopieren die FrequencyVectors-Daten in einen Hex Editor.

1. Wenn wir eine CPU größer, gleich Skylake haben, unterstützt sie wahrscheinlich HWP (Hardware P-States) (am besten nochmal für die CPU googeln!) und deswegen ist es wichtig, dass unsere FrequencyVectors dies auch tun. Am Ende der Hex Daten können wir evtl folgendes sehen:



(Mac-53FDB3D8DB8CA971) "hwp on"

und das zeigt uns, dass HWP supported ist. Ist der Eintrag nicht vorhanden und wir benutzen eine HWP CPU, müssen wir uns eine neue Plist suchen! In Hex lautet die Folge: `687770` [...] `010000006F6E`, also einfach danach suchen.

2. Alle FrequencyVectors beginnen mit `02000000`, danach folgt beispielsweise `0C000000`. Diese 4 byte (8 Zahlen) sind der LFM (Low Frequency Mode), welchen wir anpassen müssen. Also googeln wir den LFM oder TDP-down für unsere CPU. In unserer Plist entspricht `0C 00 00 00` = `0C` (hex) was in dezimal 12, also 1200MHz ist. Der i5-8265U hat einen 800MHz LFM, also tragen wir 8 (dec) -> `08000000` (hex) anstatt `0C000000` ein.

Manche CPUs vertragen sogar noch weniger, beispielsweise hat sich bei meinem i5-8265U herausgestellt, dass selbst 500MHz möglich sind. Also evtl ausprobieren.

3. Wir kommen zum EPP, dem Energy Performance Preference (im Screenshot oben auch sichtbar). Dies ist ein Wert für HWP CPUs, der die Aggressivität des Taktverhaltens steuert und wie eine Präferenz festgelegt werden kann. Der EPP liegt zwischen 0x00 (0) und 0xFF (255) und entspricht grob folgenden Richtlinien:

Richtlinie	**EPP**	**EPB (siehe 4.)**
performance	0x00 - 0x40	0 - 4
balance-performance	0x40 - 0x80	4 - 8
normal, default	0x60	6
balance-power	0x80 - 0xC0	8 - 12
power	0xC0 - 0xFF	12 - 15

Wir müssen uns für einen Wert zwischen 0x00 (0) und 0xFF (255) entscheiden, je niedriger desto schneller taktet die CPU nach oben. Wenn wir ihn zu niedrig setzen kann es sein, dass die CPU später praktisch nie im Idle die LFM Frequenz erreichen wird. Wenn wir den Wert zu hoch setzen kann es sein, dass unsere CPU praktisch garnicht mehr in den Turbo schaltet. Hier ist also evtl später etwas probieren gefragt. Für Desktops eignet sich meist ein Wert zwischen performance und balance-performance, zB 0x40. Für Laptops empfiehlt sich meist ein Wert im balance-power bis power Bereich, zB 0xC0.

Sobald wir uns entschieden haben suchen wir im Hex Editor die Folge `65707000` (epp) und passen 16 bytes später den Wert an, also zB von 65707000000000000000000000000000000000**60**00 zu 65707000000000000000000000000000000000**C0**00 (normal -> stromspar)

4. EPB ist eine Art Fallback zu EPP, wenn EPP nicht unterstützt ist. Also sollten wir EPB (IA32_ENERGY_PERF_BIAS) auch noch passend zum gewählten EPP setzen (siehe Tabelle). Der Wert findet sich 16 bytes hinter `perf-bias`, eine mögliche Anpassung wäre zB 706572662D626961730000000000000000000000**05**00 zu 706572662D626961730000000000000000000000**12**00.

Unsere angepassten FrequencyVectors können wir jetzt aus dem Hex Editor wieder in das entsprechende Feld des PlistEditors kopieren und unsere Custom X86Platform Plist speichern.

Jetzt laden wir als letztes CPU-Friend herunter und ziehen das Tool ResourceConverter.sh ins Terminal. Der Befehl lautet `./ResourceConverter.sh -a "Mac-... .plist"` wobei wir anstatt "Mac-... .plist" unsere Custom X86Platform Plist ins Terminal ziehen. Es wird eine ssdt im Ausgangsordner erstellt.

Wir öffnen MaciASL, lassen uns die DSDT anzeigen und suchen nach CPU0 oder PR00 um die ACPI ID unserer CPU herauszufinden. Sobald wir etwas finden wie

Code

1. Scope (_PR)
2. {
3. Processor (CPU0, 0x01, 0x00001810, 0x06) {}

wissen wir, dass unsere CPU zB bei _PR_.CPU0 liegt.

Als letztes öffnen wir die von ResourceConverter erstellt ssdt_data.dsl mit MaciASL und stellen sicher, dass als External und Scope jeweils die gerade gefundene ACPI ID eingetragen ist. Danach Save As -> File Format: ACPI Machine... aml, speichern. Die SSDT wird normal über die EFI eingebunden (nicht parallel mit SSDT-PLUG nutzen!) und CPUFriend normal als Kext eingebunden.

Neustart -> testen -> hoffentlich freue. Voraussetzung ist, dass das X86PlatformPlugin jetzt geladen wird. Dies lässt sich prüfen mit `kextstat | grep -R X86` im Terminal.

Und für die ganz Harten gibts dann auch noch VoltageShift, aber das wird jetzt wirklich zu viel.

Älterer Lesestoff: SMBIOS iMac17,1 / Skylake i76700K und Powermanagement - wie funktioniert es richtig?

Guide wurde eingefügt. Ehemaliger Beginn des Threads ohne Guide:

> Zitat von karacho
>
> *Du kannst den PluginType in der SSDT-PLUG.aml auf 0 setzen*

Darf ich fragen wieso?