Asus Notebook - Fan-Control ACPI - Schreiben in EC-RAM?

Beitrag von „Holz_Michel“ vom 19. März 2023, 18:45

Hallo zusammen,

ich möchte das Thema Fan-Control bei Asus Laptops (in meinem Fall speziell der UX310UAK) noch einmal aufrollen. Der auch bei niedrigen Temperaturen ständig mitlaufende Lüfter nervt schon sehr.

Aktueller Stand

- Unter Linux habe ich es z. B. mit NBFC schon hinbekommen, eine sinnvolle Lüftercharakteristik zu definieren
- Der hier beschriebene SSDT-Patch funktioniert bei meinem Modell nicht, da die verwendeten Methoden (z. B. QMOD) nicht mehr vorliegen bzw. Asus allgemein die Lüftersteuerung geändert hat.
- Zufällig bin ich auf diese Posts gestoßen, welche für mich sehr vielversprechend klingen: https://github.com/daringer/as...44#issuecomment-307589414

Ich suche eine Möglichkeit um

- zunächst einen EC RAM Dump zu machen (das kann ich zur Not aber natürlich auch von einem Linux aus tun), um daraus die Register zu suchen, welche die Kennline bzw. Temperaturen definieren
- dIe im o. g. Github Verlauf dargestellten acpi_calls in mein System einzubinden. Ich stelle mir vor, dass dies eigentlich recht einfach gehen müsste, jedoch bin ich planlos bezüglich der Herangehensweise.

  _SB.PCI0.LPCB.EC0.WRAM 0x537 0x2d

  _SB.PCI0.LPCB.EC0.WRAM 0x538 0x30

  _SB.PCI0.LPCB.EC0.WRAM 0x539 0x33

  _SB.PCI0.LPCB.EC0.WRAM 0x53a 0x37

  _SB.PCI0.LPCB.EC0.WRAM 0x53b 0x3a

In der DSDT wird WRAM durchaus verwendet, z. B:

Code

```
1. If ((IIA0 == 0x00110014))
2. {
3. Local0 = ^^PCI0.LPCB.EC0.RRAM (0x0520)
4. If ((Local0 & 0x02))
5. {
6. Local0 = ^^PCI0.LPCB.EC0.RRAM (0x0522)
7. If ((IIA1 == Zero))
8. {
9. Local1 = (Local0 & 0xFFFFFFFFFFFFFFBF)
10. }
11. ElseIf ((IIA1 == One))
12. {
13. Local1 = (Local0 | 0x40)
14. }
15.
16. ^^PCI0.LPCB.EC0.WRAM (0x0522, Local1)
17. Return (One)
18. }
19.
20. Return (Zero)
21. }
```

Alles anzeigen

Hat jemand von euch eine Idee, wie ich einen Aufruf dieser WRAM Methode mit entsprechenden Parametern implementieren kann? Ich hätte jetzt aus dem Bauch heraus gesagt, dass das in eine SSDT muss, jedoch bräuchte ich dafür Hilfe.

Vielen Dank im Voraus!

Beitrag von „Holz_Michel“ vom 22. März 2023, 21:27

Hallo zusammen,
hier ein kleines Update:

Unter Linux kann ich einfach diese Befehle als Script ausführen und habe dann mit Hilfe des acpi_call Pakets eine perfekte Lüftersteuerung.

Wenn jemand eine Idee hat, wie ich diese funktionierenden ACPI Calls in MacOS, OC oder in eine SSDT einbauen kann, wäre ich sehr glücklich.

Shell-Script: set_kennlinie.sh

1. #!/bin/bash
2.
3. echo '_SB.PCI0.LPCB.EC0.WRAM 0x537 0x3C' | sudo tee /proc/acpi/call && sudo cat /proc/acpi/call
4. echo '_SB.PCI0.LPCB.EC0.WRAM 0x538 0x3E' | sudo tee /proc/acpi/call && sudo cat /proc/acpi/call
5. echo '_SB.PCI0.LPCB.EC0.WRAM 0x539 0x40' | sudo tee /proc/acpi/call && sudo cat /proc/acpi/call
6. echo '_SB.PCI0.LPCB.EC0.WRAM 0x540 0x44' | sudo tee /proc/acpi/call && sudo cat /proc/acpi/call
7. echo '_SB.PCI0.LPCB.EC0.WRAM 0x541 0x48' | sudo tee /proc/acpi/call && sudo cat /proc/acpi/call
8. echo '_SB.PCI0.LPCB.EC0.WRAM 0x542 0x4C' | sudo tee /proc/acpi/call && sudo cat /proc/acpi/call
9. echo '_SB.PCI0.LPCB.EC0.WRAM 0x543 0x50' | sudo tee /proc/acpi/call && sudo cat /proc/acpi/call
10. echo '_SB.PCI0.LPCB.EC0.WRAM 0x544 0x54' | sudo tee /proc/acpi/call && sudo cat /proc/acpi/call

Ich kann mir sehr gut vorstellen, dass diese Vorgehensweise für eine Vielzahl von Asus Notebooks funktionieren wird.

Wenn noch weitere Infos benötigt werden, kann ich diese natürlich sehr gern nachliefern.
Vielen Dank!

Beitrag von „griven“ vom 23. März 2023, 10:31

Naja Du braucht unter macOS etwas das die Calls triggert damit da was passiert es muss also irgendetwas her das vereinfacht gesprochen sagt wenn der Temperatursensor den Wert X liefert denn setze den Wert Y für die Drehzahl des Lüfters. Von sich aus passiert das nicht da macOS ja nunmal einen SMC erwartet und das darüber regeln würde (AppleSMC.kext). Hier

muss man ansetzen. Es gab eine ähnliche Problematik damals bei den alten Thinkpads (T60/T61) schau Dir mal das hier an: Temperaturabhängige Lüftersteuerung für Lenovo T6X - T4XX (möglicherweise auch andere Notebooks) ich könnte mir vorstellen das sich eine ähnliche Lösung auch mit dem Asus realisieren lässt.

Beitrag von „Holz_Michel“ vom 25. März 2023, 20:41

Guten Abend griven ,

vielen Dank für die Antwort. Ich muss glaube ich dazu sagen, dass die Lüftersteuerung durchaus temperaturabhängig funktioniert. Und das mit exakt gleichen Kennlinien in Windows, Linux und MacOS.

Daher fällt es mir sehr schwer zu verstehen, dass der EC hier außer Gefecht ist.

In der DSDT gibt es keine explizite Funktion zum direkten Setzen einer Drehzahl. Da die "hardwareseitige" Steuerung grundsätzlich funktioniert und ich nur einen „Offset" der Kennline brauche, würde ich gerne einmal probieren, ob es mit macOS nicht vielleicht auch ausreicht, wenn ich diese 5 Werte im EC RAM setze.

Ein einmaliges Setzen z. B. beim oder nach dem Systemstart würde völlig ausreichen.

Ich würde das wirklich gern experimentell ausprobieren, mach dann natürlich auch eine Doku davon. Jedoch fehlt mir das letzte Puzzlestück: Wo und wie kann ich diese Calls als „Init“ in eine SSDT oder sonst wohin einbinden, sodass sie beim Start gesetzt werden?

Vielen Dank noch einmal!

Beitrag von „griven“ vom 26. März 2023, 22:35

Okay das ist dann doch aber auch eine komplett andere Kiste oder nicht? Wenn der Lüfter doch rauf und runter regelt dann ist es vermutlich eher so das die definierte Lüfterkurve "blöd" eingestellt ist oder sehe ich das falsch?

Wie sieht denn _SB.PCI0.LPCB.EC0 in Deiner DSDT aus?

Beitrag von „Holz_Michel“ vom 27. März 2023, 03:29

Hi,

habe leider erst nächstes Wochenende wieder Zugriff auf den Laptop.

Genau, ich sehe es ebenfalls so, dass nur die Kurve anders eingestellt werden muss. Nichts anderes mache ich in Linux über die oben genannten Calls, denn die WRAM Methode von Asus erlaubt ein Überschreiben der Kennlinie.

Die Kurve wird über die Werte im EC RAM definiert und zwar ab Adresse `0x537`. Die Werte entsprechen der HEX Schreibweise der jeweiligen Temperatur. In `0x537` steht standardmäßig z. B. 35 Grad als unterster Punkt, wird der Wert höher gesetzt, wird das Verhalten sofort besser. Ich finde das so eigentlich recht elegant gemacht, denn damit kann man das Management zu hoher Temperaturen einfach der Programmierung von Asus überlassen und nur „untenrum“ etwas nachjustieren.

Vielen Dank

Beitrag von „grt“ vom 27. März 2023, 08:45

dann nimm doch die wram-methode, kopier sie in eine ssdt, bearbeite die werte, und benenne die originalmethode in der dsdt per rename-patch um. sollte so eigentlich klappen.

Beitrag von „griven“ vom 27. März 2023, 09:50

Was grt schreibt war nämlich auch meine Idee

Allerdings ist es mit der WRAM Methode nicht getan denn die "schreibt" nur die Werte:

Code

```
1. Method (WRAM, 2, Serialized)
2. {
3. If (ECAV ())
4. {
5. Acquire (MU4T, 0xFFFF)
6. Local0 = Arg0
7. Local1 = (Local0 & 0xFF)
8. Local0 >>= 0x08
9. Local0 &= 0xFF
10. CMD = 0xFF
11. EDA1 = 0x81
12. EDA2 = Local0
13. EDA3 = Local1
14. EDA4 = Arg1
15. ECAC ()
16. Release (MU4T)
17. Return (One)
18. }
19.
20. Return (Ones)
21. }
```

Alles anzeigen

Man müsste sich jetzt durch die DSDT wühlen und mal gucken von wo aus die WRAM Methode aufgerufen wird. Ein heißer Kandidat hierfür scheint mir die Methode WMNB zu sein.

Beitrag von „grt“ vom 27. März 2023, 10:55

dann könnte man die WRAM in ruhe lassen, und müsste dafür den "zuspieler" der die infos/werte beinhaltet, in die ssdt stecken und original umbenennen. seh ich das richtig?

Beitrag von „griven“ vom 27. März 2023, 11:29

So würde ich das zumindest denken, ja

In der DSDT sind allerdings keine Calls auf die in Post #2 erwähnten Stellen zu finden. Möglich wäre nun das es dafür generell schon eine SSDT gibt ist aber ohne den gesamten ACPI Tabellensatz zu haben schwer bis nicht zu sagen.

Beitrag von „Holz_Michel“ vom 27. März 2023, 18:32

Hallo und erstmal herzlichen Dank für eure Antworten. Das geht wirklich in die Richtung, die Ich mir vorgestellt habe.

Mit dem „einfach in die SSDT kopieren“ hört es bei mir auf, denn mir fehlt jetzt die Ahnung, wann welcher Code in einer SSDT ausgeführt wird. Ich glaube was ihr sagt passt zu dem Gedanken von mir, dass ich noch irgend ein Init-Event oder sonst etwas brauche. WRAM ist für mein Verständnis einfach nur ein allgemeiner „Write RAM“ (es gibt auch RRAM) und wird nur glaube ich 2 mal sonst in der DSDT verwendet.

Hatte mir schon überlegt, einen der Fn-Keys zu opfern und den als „set Fan Kennlinien Button“ zu verwenden, wenn es kein Init- oder „On Boot“ Event gibt. Auch sowas müsste doch machbar und relativ einfach sein, oder?

Wie kann ich euch den vollständigen ACPI Tabellensatz zur Verfügung stellen? Die DSDT habe ich mit einem Linux Tool ausgeschrieben, dessen Name mir gerade nicht mehr einfällt.

Beitrag von „Holz_Michel“ vom 9. April 2023, 11:03

Hallo zusammen und frohe Ostern.

Ich habe jetzt folgende SSDT erstellt, in OC unter ACPI gelegt und natürlich auch in der config.plist auf "enabled" gestellt.

Code

```
1. /*
2. * Intel ACPI Component Architecture
```

```
* AML/ASL+ Disassembler version 20200925 (64-bit version)
* Copyright (c) 2000 - 2020 Intel Corporation
*
* Disassembling to symbolic ASL+ operators
*
* Disassembly of iASLoJ221T.aml, Sun Apr 9 10:55:49 2023
*
* Original Table Header:
* Signature "SSDT"
* Length 0x00000121 (289)
* Revision 0x02
* Checksum 0x96
* OEM ID "pat"
* OEM Table ID "FanTable"
* OEM Revision 0x00000000 (0)
* Compiler ID "INTL"
* Compiler Version 0x20200925 (538970405)
*/
DefinitionBlock ("", "SSDT", 2, "pat", "FanTable", 0x00000000)
{
External (_SB_.PCI0.LPCB.EC0_.WRAM, MethodObj) // 2 Arguments

\_SB.PCI0.LPCB.EC0.WRAM (0x0537, 0x3C)
\_SB.PCI0.LPCB.EC0.WRAM (0x0538, 0x3E)
\_SB.PCI0.LPCB.EC0.WRAM (0x0539, 0x40)
\_SB.PCI0.LPCB.EC0.WRAM (0x0540, 0x44)
\_SB.PCI0.LPCB.EC0.WRAM (0x0541, 0x48)
\_SB.PCI0.LPCB.EC0.WRAM (0x0542, 0x4C)
\_SB.PCI0.LPCB.EC0.WRAM (0x0543, 0x50)
\_SB.PCI0.LPCB.EC0.WRAM (0x0544, 0x54)
}
```

Alles anzeigen

Allerdings sieht es so aus, als würde das Ganze nicht funktionieren. Compilen lies die SSDT sich mit 0 Errors. Kann es sein, dass die Calls nicht "aufgerufen" werden? Brauche ich noch irgendetwas "drum-herum"? Mein Werk sieht auch fast etwas zu einfach aus 

Ich hänge noch eine SSDT an, in der die WRAM Methode ebenfalls genutzt wird.

Vielen Dank im Voraus

Beitrag von „griven“ vom 9. April 2023, 11:58

Kommt das jetzt direkt aus der DSDT oder wo hast Du die Calls jetzt her? Weil damit das funktioniert muss das was original da ist aus dem Weg andernfalls wird das in der SSDT nicht verwendet...

Beitrag von „Holz_Michel“ vom 9. April 2023, 12:29

Ich habe nirgends in den ACPI tables etwas gefunden, das die Fan-Kurve setzt. Ich denke das kommt irgendwie schon vorher vom BIOS, sprich der EC-RAM wird mit Werten vorbefüllt, die dann in der DSDT und auch in SSDTs nicht mehr explizit gesetzt werden.
Sprich die SSDT ist komplett selber erstellt.

Ich weiß ja welche Werte ich wo hinschreiben möchte und ASUS gibt mir sogar eine Function dazu -> WRAM.

Ich habe gerade mit der gleichen OC-Config testhalber mal ein Live-Linux gestartet. Interessanterweise scheinen die Änderungen hier zu greifen, wenn ich die EC-RAM Werte auslese bekomme ich genau das, was ich in der SSDT gesetzt habe (und der Lüfter ist still).
Ob das nur Zufall war möchte ich noch prüfen, werde einfach mal andere Zahlen in die SSDT schreiben und schauen, was Linux ausliest.

EDIT: Es ist wohl tatsächlich so, dass meine SSDT funktioniert - zumindest unter Linux. Ich habe in der SSDT nun testhalber andere Zahlen eingetragen, mit OC wieder Linux gebootet und die Werte per acpi_call ausgelesen - Meine eingetragenen Werte wurden angezeigt.

Die Frage wäre dann immer noch: Was macht macOS hier anders?

UPDATE:
Habe es finally hinbekommen 

Anscheinend hat das mit der SSDT alleine für MacOS nicht ausgereicht. Ich habe jetzt von RehabMan den ACPIDebug.kext zusammen mit IOIO im Einsatz. Jetzt kann ich wie in Linux über die Befehlszeile einen ACPI Call auf Methoden innerhalb der SSDT für ACPIDebug machen. Deshalb musste ich meine Calls noch zusätzlich in die SSDT-RMDT.aml mit aufnehmen.
Mit dem call

```
ioio -s org_rehabman_ACPIDebug dbg0 1
```

kann ich jetzt ein sofortiges Schreiben meiner in der SSDT-RMDT definierten Kennline erreichen. Bei Gelegenheit und nach einer gewissen Testphase folgt eine ausführliche Doku