DSDT Patches

Beitrag von „Schneelöwe“ vom 7. März 2012, 23:14

Da es mir selber schwer gefallen ist, alles benötigten Patches zusammenzufinden, werde ich mal hier posten, was ich gemacht habe .
Die DSDT ist für ein medion MD 96850 mit folgenden Daten:
CPU: Intel Pentium Dual Core Mobile T2390 (Merom, 1,86 Gigahertz)
GPU: GMA X3100
Audio: ALC888
Also, fangen wir mal mit der extrahierten DSDT an, indem wir die Errors fixen .

Nächster Error:

Code

1. Error Invalid leading asterisk (*pnp0c14)/Error Non-hex letters must be upper case (pnp0c14)

Einerseits haben wir hier den Asterisk (*) der das Problem verusacht, aber wir müssen den String eben auch gross schreiben.
Also aus Name (_HID, "*pnp0c14") wird Name (_HID, "PNP0C14")

Nächste Warnung:

Code

1. 353 Warning Not all control paths return a value (_WED)
2. 353 Warning Reserved method must return a value (Integer/String/Buffer required for _WED)

Hier wird es etwas schwieriger und dazu auch noch zwei, aber nicht verzagen!

Code

1. Method (_WED, 1, NotSerialized)

2. {
3. Store (Arg0, P80H)
4. If (LEqual (Arg0, 0xB0))
5. {
6. Return (B0ED)
7. }
8. Return (Zero) <-- Das hier einfügen
9. }

Nächste Anmerkung:

Code

1. 599 Remark Use of compiler reserved name (_T_0)
2. 712 Remark Use of compiler reserved name (_T_1)
3. 755 Remark Use of compiler reserved name (_T_2)
4. 755 Remark Use of compiler reserved name (_T_2)
5. 755 Remark Use of compiler reserved name (_T_2)
6. 882 Remark Use of compiler reserved name (_T_5)
7. 882 Remark Use of compiler reserved name (_T_5)

Auch hier ist die Lösung ganz einfach!
Einfach alle _T_* durch T_* ersetzen, aber nicht nur die bei Name sondern aauch die darunter

Nächster Error:

Code

1. 1739 Error Invalid combination of Length and Min/Max fixed flags
2. 1739 Error Invalid combination of Length and Min/Max fixed flags

Jetzt müsst ihr Rechnen, und zwar Hexdezimal. Denn das Problem liegt nicht bei der linie genau.
Aufjedenfall hier ist der Code:

Code

1. 0x00000000, // Granularity
2. 0x00000000, // Range Minimum
3. 0xDFFFFFFF, // Range Maximum
4. 0x00000000, // Translation Offset
5. 0x00000000, // Length

Und jetzt subtrahiert ihr die Maximum Range von der Minimum Range und addiert 1
Leider kann man nicht von 0 subtrahieren (Zumindest nicht hier), daher nehmen wir DFFFFFFF und addieren 1. Rauskommen tut E0000000 und das kommt dann als Length rein.
SO sollte es dann aussehen:

Code

1. 0x00000000, // Granularity
2. 0x00000000, // Range Minimum
3. 0xDFFFFFFF, // Range Maximum
4. 0x00000000, // Translation Offset
5. 0xE0000000, // Length
6. ,, _Y0D, AddressRangeMemory, TypeStatic)

Nächster teil ist der hier:

Code

1. 0x00000000, // Granularity
2. 0xFED40000, // Range Minimum
3. 0xFED44FFF, // Range Maximum
4. 0x00000000, // Translation Offset
5. 0x00000000, // Length

Und wieder FED44FFF minus FED40000 + 1 was 00005000 ergibt.
Daher muss es lauten:

Code

1. 0x00000000, // Granularity
2. 0xFED40000, // Range Minimum
3. 0xFED44FFF, // Range Maximum

4. 0x00000000, // Translation Offset
5. 0x00005000, // Length

Nächste Warnung:

Code

1. 5918 Warning Not all control paths return a value (HKDS)

Das gleiche wie bei _WED

Und wir haben keine Errors mehr!
Die DSDTs habe ich mal angefügt .

Beitrag von „Schneelöwe“ vom 15. März 2012, 23:19

So, der nächste Teil hat jetzt aber gedauert, aber leider musste ich mich erstmal um die Grafikkarte kümmern .
Was ich in diesem Teil geplant habe ist erstmal Audio bzw. die allgemeine Device Injection per Methode _DSM und die dazu benötigte DTGP.
Zuerst fangen wir mit dem Part der direkt das HDEF-Device betrifft, und falls ihr ein solches nicht habt, habt ihr vll. ein Gerät namens "AZAL", bennent dies einfach in HDEF um .
Also, ihr habt jetzt, je nach Karte, folgenden Code für die Karte (in meinem Fall ein sehr simpler Code für die ALC888

Code

1. Method (_DSM, 4, NotSerialized)
2. {
3. Store (Package (0x04)
4. {
5. "layout-id",
6. Buffer (0x04)

```
7.  {
8.  0x78, 0x03, 0x00, 0x00
9.  },
10.
11.
12. "PinConfigurations",
13. Buffer (Zero) {}
14. }, Local0)
15. DTGP (Arg0, Arg1, Arg2, Arg3, RefOf (Local0))
16. Return (Local0)
17. }
```

Alles anzeigen

Doch wohin damit? Und was ist das überhaupt?
Mein gefestigtes Halbwissen sagt folgendes :
Methode (_DSM) ist eine Methode um Daten in IOReg über die Methode DTGP "Hochzuladen".
In diesem Fall sagt der Code ungefähr folgendes: Das Gerät "HDEF", welches die Adresse 0x001B0000 (Die steht fast immer dierekt unter dem Device in Name (_ADR, xxxxxxxxxx)) hat, hat auch die layout-id von 888 und die Pinconfiguration 0. DIese Infos werden dann per DTGP geliefert, denn wie ihr seht taucht auch DTGP dort auf.

Doch HALT! Wenn ihr aufmerksam seit ist euch vll. aufgefallen, dass ich 888 als layout-id angegeben habe, aber in _DSM steht doch 0x78, 0x03, 0x00, 0x00 ?
Tja, das hat primär was damit zu tun, dass 888 die id in Dezimal angegeben ist und 378 in Hex (Rechnet es doch um per onlinetool). Doch stimmt auch die Reihenfolge nicht.
Das was in der DSDT angegeben wird, sind "Big-Endians", das was wir lesen können sind "Little-Endians".
Wie wir vom einem zu anderen kommen? Ganz einfach! Wir spiegeln die Zahl einfach in der Mitte, lassen aber die zweierpaare intakt.
Hier ein paar Beispiele:
Big-Endian: 00 02 00 00 - 14 02 00 00
Lidle Endian: 00 00 02 00 - 00 00 02 14
Ich hoffe es wurde klar .

Also, wenn wir denn Code nun einfügen, dann kriegen wir noch folgenden Fehler:
Error: Object does not exist (DTGP)
Tja, wie ich schon sagte werden die Infos aus der _DSM-Methode per DTGP in die IORegistry hochgeladen, aber bei uns fehlt noch die DTGP. Doch keine Panik; dafür fügen wir einfach ganz

am Ende, vor der letzten geschweiften Klammer (So sieht so eine Klammer aus: { oder }) folgendes ein:

Code

```
1. Method (DTGP, 5, NotSerialized)
2. {
3. If (LEqual (Arg0, Buffer (0x10)
4. {
5. /* 0000 */ 0xC6, 0xB7, 0xB5, 0xA0, 0x18, 0x13, 0x1C, 0x44,
6. /* 0008 */ 0xB0, 0xC9, 0xFE, 0x69, 0x5E, 0xAF, 0x94, 0x9B
7. }))
8. {
9. If (LEqual (Arg1, One))
10. {
11. If (LEqual (Arg2, Zero))
12. {
13. Store (Buffer (One)
14. {
15. 0x03
16. }, Arg4)
17. Return (One)
18. }
19. If (LEqual (Arg2, One))
20. {
21. Return (One)
22. }
23. }
24. }
25. Store (Buffer (One)
26. {
27. 0x00
28. }, Arg4)
29. Return (Zero)
30. }
```

Alles anzeigen

Und fertig, der Error ist weg .
Eine Anmerkung noch, die /* */ sind Kommentare, alles was da zwischen steht wird ignoriert.
So nun ist erstmal genug hiermit .

Beitrag von „Schneelöwe“ vom 18. März 2012, 01:03

Hrmpf Heute fällt mir natürlich auf, dass ich über so sekundäre Sachen wie die AppleHDA schon gesprochen habe, aber nicht über sowas wie die KernelPanics wegen der AppleIntelCpuPowermanagmentk.kext, oder wie denn CMOS Reset, geschweige denn über Shutdownprobleme, die einige wohl haben (Ich bisher nicht, aber man weiss ja nie).
Also, fangen wir mit dem einfachsten Patch an, dem CMOS-Reset.

Code

```
1. Device (RTC)
2. {
3. Name (_HID, EisaId ("PNP0B00"))
4. Name (_CRS, ResourceTemplate ()
5. {
6. IO (...)
7. IRQNoFlags ()
8. {8}
9. })
10. }
```

Für das Problem des Resets nehmen wir uns die Lenght vor, die muss nämlich 0x02 sein. Also so soll das dann Aussehen:

Code

```
1. Device (RTC)
2. {
3. Name (_HID, EisaId ("PNP0B00"))
4. Name (_CRS, ResourceTemplate ()
5. {
6. IO (...)
7. IRQNoFlags ()
8. {8}
9. })
10. }
```

Als nächste kommen wir zu der Problematik der KernelPanics wegen der AppleIntelCpuPowermanagmentk.kext. Diese entstehen, wenn die RTC (Real TIme Clock) und der Timer per IRQ (=Interrupt Request) denn Prozessor zum Unterbrechen auffordern. DOch wie lösen wir das Problem? Indem wir die IRQs dem Gerät zuweisen, das auch in einem echten Mac interruptet, dem HPET (High Precision Eventtimer). Was slebstverständlich ist: Wenn ihr in der DSDT kein HPET-Device habt, kann das auch keine IRQs senden, womit der Fix hinfällig wird (Womit ich auch verstanden habe, warum mein allererste Thread hier im Forum Quatsch war, hat ja nur 600 Beiträge gedauert).
Also, aus

Code

```
1. Device (RTC)
2. {
3. Name (_HID, EisaId ("PNP0B00"))
4. Name (_CRS, ResourceTemplate ()
5. {
6. IO (...)
7. IRQNoFlags ()
8. {8}
9. })
10. }
11. Device (TIMR)
12. {
13. Name (_HID, EisaId ("PNP0100"))
14. Name (_CRS, ResourceTemplate ()
15. {
16. IO (...)
17. IRQNoFlags ()
18. {0}
19. })
20. Device (IPIC)
21. {
22. Name (_HID, EisaId ("PNP0000"))
23. Name (_CRS, ResourceTemplate ()
24. {
25. IO (...)
26. IRQNoFlags ()
27. {2}
```

28. })
29. }

Alles anzeigen

entfernen wir die IRQs =

Code

1. Device (RTC)
2. {
3. Name (_HID, EisaId ("PNP0B00"))
4. Name (_CRS, ResourceTemplate ()
5. {
6. IO (...)
7. })
8. }
9. Device (TIMR)
10. {
11. Name (_HID, EisaId ("PNP0100"))
12. Name (_CRS, ResourceTemplate ()
13. {
14. IO (...)
15. })
16. }
17. Device (IPIC)
18. {
19. Name (_HID, EisaId ("PNP0000"))
20. Name (_CRS, ResourceTemplate ()
21. {
22. IO (...)
23. })
24. }

Alles anzeigen

Und beim HPET fügen wir sie ein, Daher wird aus

Code

1. Device (HPET)
2. {

```
Name (_HID, EisaId ("PNP0103"))
Name (_CID, EisaId ("PNP0C01"))
Name (BUF0, ResourceTemplate ()
{
Memory32Fixed (ReadOnly,
0xFED00000, // Address Base
0x00000400, // Address Length
_Y14)
})
Method (_STA, 0, NotSerialized)
{
If (LGreaterEqual (OSYS, 0x07D1))
{
If (HPAE)
{
Return (0x0F)
}
}
Else
{
If (HPAE)
{
Return (0x0B)
}
}
Return (Zero)
}
Method (_CRS, 0, Serialized)
{
If (HPAE)
{
CreateDWordField (BUF0, \_SB.PCI0.LPCB.HPET._Y14._BAS, HPT0)
If (LEqual (HPAS, One))
{
Store (0xFED01000, HPT0)
}
If (LEqual (HPAS, 0x02))
{
Store (0xFED02000, HPT0)
}
If (LEqual (HPAS, 0x03))
```

```
{
Store (0xFED03000, HPT0)
}
}
Return (BUF0)
}
}
```

Alles anzeigen

das hier:

Code

```
Device (HPET)
{
Name (_HID, EisaId ("PNP0103"))
Name (_CID, EisaId ("PNP0C01"))
Name (BUF0, ResourceTemplate ()
{
IRQNoFlags ()
{0}
IRQNoFlags ()
{8}
Memory32Fixed (ReadOnly,
0xFED00000, // Address Base
0x00000400, // Address Length
_Y14)
})
Method (_STA, 0, NotSerialized)
{
If (LGreaterEqual (OSYS, 0x07D1))
{
If (HPAE)
{
Return (0x0F)
}
}
Else
{
If (HPAE)
```

```
{
Return (0x0B)
}
}
Return (Zero)
}
Method (_CRS, 0, Serialized)
{
If (HPAE)
{
CreateDWordField (BUF0, \_SB.PCI0.LPCB.HPET._Y14._BAS, HPT0)
If (LEqual (HPAS, One))
{
Store (0xFED01000, HPT0)
}
If (LEqual (HPAS, 0x02))
{
Store (0xFED02000, HPT0)
}
If (LEqual (HPAS, 0x03))
{
Store (0xFED03000, HPT0)
}
}
Return (BUF0)
}
}
```

Alles anzeigen

Achja, aus dem IPIC, muss auch der IRQNoFlag eintrag raus !

Und das beste kommt ja immer zum Schluss. Daher kommt jetzt der Shutdownpatch.
Wichtig hier ist diesmal kein Device sondern eine globale Methode. Und diese Methode heisst _PST. Unglaublicherweise heisst sie ausgeschrieben "Prepare to Shutdown", also wenn´s daran nicht liegen könnte .
Also, hier der normalcode meiner DSDT:

Code

```
1. Method (_PTS, 1, NotSerialized)
2. {
3. Store (Zero, P80D)
4. P8XH (Zero, Arg0)
5. Store (Arg0, \_SB.PCI0.LPCB.EC0.SYSC)
6. Store (One, \_SB.PCI0.LPCB.EC0.MUTE)
7. If (LEqual (Arg0, 0x04))
8. {
9. Store (\_SB.PCI0.LPCB.EC0.BNEN, \_SB.PCI0.LPCB.EC0.EBNE)
10. }
11. }
```

Alles anzeigen

Und jetzt der gepatchte:

Code

```
1. Method (_PTS, 1, NotSerialized)
2. {
3. If (LEqual (Arg0, 0x05)) {}
4. Else
5. {
6. Store (Zero, P80D)
7. P8XH (Zero, Arg0)
8. Store (Arg0, \_SB.PCI0.LPCB.EC0.SYSC)
9. Store (One, \_SB.PCI0.LPCB.EC0.MUTE)
10. If (LEqual (Arg0, 0x04))
11. {
12. Store (\_SB.PCI0.LPCB.EC0.BNEN, \_SB.PCI0.LPCB.EC0.EBNE)
13. }
14. }
15. }
```

Alles anzeigen

Und wie ihr seht, habe ich die gesamamte methode in eine Else-If anfrage untergergebracht. Aber dieser Patch soll wohl nicht für Geräte mit einer Intel Grafikkarte funktionieren.
So, das war´s für jetzt. Da wir das Problem des CMOS-Resets ja gelöst haben, gucke ich das nächste Mal, was ich bezüglich Sleep machen kann.

Beitrag von „Schneelöwe“ vom 18. März 2012, 13:29

Kommen wir nun zum Sleep, dem Patch, dem viele mit dem SleepEnabler beihelfen oder ihn gleich ignorieren. Doch hier wird nichts ignoriert, hier wird gelöst .
~~Also, zuerst einmal ein bisschen Kosmetik, nicht primär wichtig aber näer an einem echten mac.~~
~~Dazu einfach alle Geräte von USB1 bis USB5 (oder auch mehr) zu UHC1 bis UHC5 umbenennen und auch die DSDT nach eräwhnungn von USB1-5 dursuchen und ggf. ersetzen durch UHC1-5.~~
Ist Quatsch. Es ist zwar auch egal, aber auch beim Macbook 3,1 heissen sie USB1 bis USB5, und dann müssen wir ja auch nichts ändern!

Nachdem wir das jetzt haben wird es etwas schwieriger. Dank der Tatsache, das wir denn CMOS-reset der durch Sleep manchmal ausgelöst wird, schon im vorherigen Beitrag ausgemerzt haben, können wir gefahrlos SLeep erstmal ausprobieren. Aber denkt daran: Öffnet vorher die Konsole und stellt auf "Alle meldungen".
Bei mir schläft der Rechner gut ein, wacht aber sofort wieder auf. Ein Blick in die Konsole und wir sehen: Wake reason = LANC EHC1.
Meine Vermutung war, dass das Erwachen primär durch EHC1 ausgelöst wird.
Also fix einen Fix dafür gesucht (Blödes Wortspiel, ich weiss). Der sieht so aus (Für EHCI1, für EHCI2 ist di clock-id einfach 0x02):

Code

```
1. Method (_DSM, 4, NotSerialized)
2. {
3. Store (Package (0x0B)
4. {
5. "AAPL,clock-id",
6. Buffer (One)
7. {
8. 0x01
9. },
10. "device_type",
11. Buffer (0x05)
12. {
13. "EHCI"
```

```
14. },
15. "AAPL,current-available",
16. 0x05dc,
17. "AAPL,current-extra",
18. 0x03e8,
19. "AAPL,current-in-sleep",
20. 0x05dc,
21. Buffer (One)
22. {
23. 0x00
24. }
25. }, Local0)
26. DTGP (Arg0, Arg1, Arg2, Arg3, RefOf (Local0))
27. Return (Local0)
28. }
```

Alles anzeigen

Doch ich war mir bezüglich der AAPL Einträge etwas unsicher, da die so systemspezifisch sind. Also habe ich mir einen IOreg dump von einem echten Macbook 3,1 geholt, das ich auch als SMBios.plist benutze. Ausserdem hat es eine sehr ähnliche Hardware, insbesondere der Chipsatz ist gleich und damit auch der EHCI-Controller, da dieser mit im Chipsatz steckt.
Ein Blick in denn IOReg dump verrät mir, das eben diese Macbook folgende Einträge bei denn EHCI-Controllern hat:

AAPL,current-available 0x04b0
AAPL,current-extra 0x02bc
AAPL,current-in-sleep 0x03e8
womit wir die betreffenden Passagen einfach ändern:

Code

```
1. Method (_DSM, 4, NotSerialized)
2. {
3. Store (Package (0x0B)
4. {
5. "AAPL,clock-id",
6. Buffer (One)
```

```
{
0x01
},
"device_type",
Buffer (0x05)
{
"EHCI"
},
"AAPL,current-available",
0x04b0,
"AAPL,current-extra",
0x02bc,
"AAPL,current-in-sleep",
0x03e8,
Buffer (One)
{
0x00
}
}, Local0)
DTGP (Arg0, Arg1, Arg2, Arg3, RefOf (Local0))
Return (Local0)
}
```

Alles anzeigen

So, das machen wir für beide EHCIs, wobei bei EHCI1 halt die Clock-ID 1 ist und bei EHCI2 ist die Clock-ID 1.
DSDT unter Extra kopiert, neustart, Ruhezustand ausgewählt..... Und siehe da, der Recher schläft, wie wunderbar !
Ein Druck auf denn Powerknopf und er ist wieder wach, und die Konsole sagt uns auch, dass der Wake reason diesmal der power-button ist. Mission Sleep ist damit erfolgreich abgeschlossen .

Beitrag von „Schneelöwe“ vom 18. März 2012, 17:54

Kleinupdate:
Damit der AC-Adapter unter OSX erkannt wird, fehlt es insgesamt an 4 Zeilen in der DSDT.
Sucht nach dem AC-Adapter in der DSDT (Sucht nach Name (_HID, "ACPI0003")).

Dann gebt ihr nach dem Name folgendes ein:

Code

```
1. Name (_PRW, Package (0x02)
2. {
3. 0x18,
4. 0x03
5. })
```

Also so soll es dann aussehen:

Code

```
1. Device (ADP1)
2. {
3. Name (_HID, "ACPI0003")
4. Name (_PRW, Package (0x02)
5. {
6. 0x18,
7. 0x03
8. })
9. // Code geht weiter ist aber unwichtig für uns :)
```

So viel dazu, mal gucken was noch wichtig ist  .

Beitrag von „Schneelöwe“ vom 24. August 2013, 19:34

Update 3 für Heute:
Da mein Notebook jetzt ja super schläft (dank denn DSDT-Patches), dachte ich mir, ich gucke mal wie es mit Sleep und Wake per Lid Close/open läuft. Also ob der Rechner beim Schliessen des Deckels schläft und beim Öffnen wieder aufwacht. Ergebniss: Aufwachen beim Öffnen tut er, nur nicht einschlafen.

Also habe ich kurz Tante Google befragt und bin von zehntausend Threads über Sleep on LidClose erschlagen worden. Und das schlimme ist, überall wird etwas anderes gesagt und zum

Teil wird auch gesagt "ersetzt euer Lid0-Device einfach mit dem hier". Mein Normaler Menschenverstand hat mir aber gesagt, dass es wahrscheinlich eher schlechter wird, wenn ich mein Lid-Device mit etwas ersetzte, was auf Register verweist, die ich nicht habe.
Aber dann fande ich doch eine Lösung, die mir sinnig erschien. Und zwar ein Device namens PNLF in die DSDT einzüfgen.

Langer Rede, kurzer Sinn: An dem Lid-Device machen wir nichts, wir fügen unter Scope (_SB) einfach folgendes ein:

Code

```
1. Device (PNLF)
2. {
3. Name (_HID, EisaId ("APP0002"))
4. Name (_CID, "backlight")
5. Name (_UID, 0x0A)
6. Name (_STA, 0x0B)
7. }
```

Und siehe da, er schläft .
Aber was ich an dieser Lösung am besten finde: Sie ist auf allen Systemen gleich! Desweitern gibt sie einigen Systemen auch die Möglichkeit ihre Hintergrundbeleuchtung zu Regeln, da durch diese Device der interne Monitor als "AppleBacklightDisplay" erkannt wird und man somit unter Systemeinstellung/Monitore einen Helligkeitsschieberegler hat.
~~Die Helligkeit kann ich zwar auch per FN+Pfeiltasten regeln, der Schieberegler wirkt bei mir jedoch davon unbeeindruckt, auch kriege ich keine Anzeige über die Helligkeit, wenn ich sie ändere. Soll heissen: Ein Problem gelöst ein neus (kleineres) Problem gefunden~~ . Unter Lion tritt das Problem nicht mehr auf.

Beitrag von „Schneelöwe“ vom 6. September 2013, 19:33

Diesmal ein ganz anderer Fall, mein N61JQ
Error:

Code

1. Error Object does not exist

Aber das viele male.
DIe Lösung ist realtiv einfach. Wir löschen folgenden Teil:

Code

1. If (CondRefOf (FPED))
2. {
3. FPED ()
4. }

und viele viele Fehler verschwinden.
Nächster ist das Gegenteil von dem, was wir oben gemacht haben:

Code

1. Warning Reserved method should not return a value

Daher aus dieser Methode, vor der letzten Klammer das Return (One) raus.
So wird aus

Code

1. Method (_Q0F, 0, NotSerialized)
2. {
3. [...]
4. Return (One)
5. }

das hier:

Code

1. Method (_Q0F, 0, NotSerialized)
2. {
3. [...]
4. }

Und nächster:

Code

1. Warning Result is not used, operator has no effect

Bei mir sieht der betroffene Code so aus:

Code

1. And (CTRL, 0x1E)

Und so muss er richtig aussehen:

Code

1. And (CTRL, 0x1E, CTRL)

Und einen hab ich noch:

Code

1. Warning Result is not used, possible operator timeout will be missed

Da machen wir einfach aus

Code

1. Acquire (MUTE, 0x03E8)

den hier:

Code

1. Acquire (MUTE, 0xFFFF)

Und das war es auch erstmal wieder .

Beitrag von „Griven“ vom 6. September 2013, 22:17

Schneelöwe, ich muss einfach mal sagen "Hut ab" ich finde gut und lehrreich was Du machst. Ich selbst bin ja auch kein ganz dummer in Sachen DSDT aber in dem Post hier kann man einiges lernen, wenn man sich auf die Materie mal eingelassen hat.

Mach so weiter, finde es immer wieder spannend die Sachen zu lesen

---

Beitrag von „TuRock“ vom 6. September 2013, 22:22

Ich steige langsam tiefer in die "Materie" ein und mache so manche Erfahrungen !