Was ist GPRW und wenn ja, wieviele ;-)

Beitrag von „hackopti2“ vom 11. Dezember 2021, 11:56

Spaß bei Seite, kann mir bitte mal jemand erklären was das hier macht (und warum es dafür sorgt dass mein Opitplex jetzt perfekt Sleep/Wake macht) ?

Ein Link zu einer guten Erklärung reicht auch, ich finde nur nichts.

```
DefinitionBlock ("", "SSDT", 2, "DRTNIA", "GPRW", 0x00000000)
{
External (XPRW, MethodObj) // 2 Arguments

Method (GPRW, 2, NotSerialized)
{
If (_OSI ("Darwin"))
{
If ((0x6D == Arg0))
{
Return (Package (0x02)
{
0x6D,
Zero
})
}

If ((0x0D == Arg0))
```

{

Return (Package (0x02)

{

0x0D,

Zero

})

}

}

Return (XPRW (Arg0, Arg1))

}

}

Beitrag von „5T33Z0“ vom 11. Dezember 2021, 14:01

hackopti2 Ich versuch's mal… Also das geht so:

Mit Binary Renames (unter ACPI > Patch) wird die eigentliche Methode GPRW umbenannt in XPRW, um sie neu definieren bzw. manipulierne zu können ohne die komplette DSDT bearbeiten zu müssen.

In der SSDT wird die Methode dann nur für macOS manipuliert:

External (XPRW, MethodObj) // 2 Arguments

>> Sagt: guck bitte in die Methode XPRW

Und als nächstes Steht da:

Method (GPRW, 2, NotSerialized)

Heisst: Für Methode GPRW gilt:

If OSI ("Darwin")) >> wenn macOS läuft, mache bitte, was ich dir sage.

Undzwar:

- Wenn Arg0 = x06d ist, dann ersetze es durch die Werte für das Package mit den Werten 0x6D und 0.

- Und wenn Arg0 0x0D ist, dann ersetze es durch 0x0D und 0.

Die beiden Package werden dann als **Arg0** und **Arg1** an zurückgereicht und ersetzen die originalen Packages:

Return (XPRW (**Arg0**, **Arg1**))

Wenn OSI nicht macOS ist, dann wird GPRW einfach unverändert als XPRW belassen und funktioniert dann trotzdem unter Windows etc.

Und deswegen funktioniert dann jetzt Sleep und wake unter macOS, weil für macOS dann die passenden Werte verwendet werden - wo auch immer sie herkommen und was auch immer sie bedeuten 

Geht sicher noch präziser, aber besser kann ich's nicht.

Gibt da auch einige Guides bei Github:

https://github.com/5T33Z0/OC-L...ues/060D_Instant_Wake_Fix

https://dortania.github.io/Ope...sb/misc/instant-wake.html

https://github.com/grvsh02/A-g...es-on-hackintosh-laptops/

Beitrag von „hackopti2“ vom 11. Dezember 2021, 15:56

endlich verstanden, danke 5T33Z0

mich würde aber auch noch interessieren, was verdammt ändert das, scheint ja ein üblicher Fix zu sein?

Beitrag von „5T33Z0“ vom 11. Dezember 2021, 16:25

Dazu muss man sich die ganze nethod angucken und ASL (ACPI System Language) verstehen, apfelnico kann das eventuel benatworten

---

Beitrag von „hackopti2“ vom 11. Dezember 2021, 20:15

Method (GPRW, 2, NotSerialized)

{

PRWP [Zero] = Arg0

Local0 = (SS1 << One)

Local0 |= (SS2 << 0x02)

Local0 |= (SS3 << 0x03)

Local0 |= (SS4 << 0x04)

If (((One << Arg1) & Local0))

{

PRWP [One] = Arg1

}

Else

{

Local0 >>= One

FindSetLeftBit (Local0, PRWP [One])

}

Return (PRWP) /* \PRWP */

}

das wäre dann das hier

Beitrag von „5T33Z0“ vom 11. Dezember 2021, 20:26

Benutz mal die Code Fuktion </>, damit das vernünfitig formatiert wird.

Code

1. Method (GPRW, 2, NotSerialized)
2. {
3. PRWP [Zero] = Arg0

```
4. Local0 = (SS1 << One)
5. Local0 |= (SS2 << 0x02)
6. Local0 |= (SS3 << 0x03)
7. Local0 |= (SS4 << 0x04)
8. If (((One << Arg1) & Local0))
9. {
10. PRWP [One] = Arg1
11. }
12. Else
13. {
14. Local0 >>= One
15. FindSetLeftBit (Local0, PRWP [One])
16. }
17.
18. Return (PRWP) /* \PRWP */
19. }
```

Alles anzeigen

Ich weiß halt nicht ob << "kleiner als" bzw >> "größer" als bedeutet. Aber in jedem Fall kommen da die Werte aus der SSDT für Arg0 und Arg1 an. Aber was da genau passiert übersteigt meinen Kenntnishorizont von ASL.

Beitrag von „Brumbaer“ vom 11. Dezember 2021, 22:45

Arg0 Ein Wert der "durchgereicht wird"

Arg1 ist die Nummer des Sleep States der abgefragt werden soll - gibt man etwas an was es nicht gibt bekommt man den höchsten möglichen Sleep State zurück

Das Ergebnis landet in PRWP Feld aus 2 Werten, das am Ende ausgegeben wird.

Zuerst wird der erste Wert von PRWP auf den Wert des ersten Argumentes gesetzt.

SS1 bis SS4 sind Bits in denen das BIOS angibt welche der 4 Sleep States enabled sind. Die Bits werden zu einem Bitfeld gemacht, das in local 0 gespeichert wird und zwar landet das Enable

Bit des SSx im xten Bit von local0.

SS1 wird geholt um 1 verschoben. SS2 wird um 2 verschoben usw. und alle Bits werden in einem Wert zusammengefasst (geodert)

Das alles nur damit man mit nur einer if Abfrage feststellen kann ob der Sleep State mit der Nummer die dem Wert in Arg1 entspricht gesetzt ist. Sprich ob dieser Sleep State enabled it.

Ist dem so wird die Nummer des Sleep States im 2 Wert von PRWP abgelegt.

Ist dieser Sleep State nicht gesetzt, wird bestimmt welches das am höchsten gesetzte Bit ist, als was der höchste enabledete Sleep State ist und dessen Wert im 2 Wert von PRWP abgelegt. FindSetLeftBit zählt von der 1 aus nicht von der 0 wie Arg1, deshalb wird vorher local0 einmal nach rechts geschoben damit das Ergebnis mit der Zählweise von Arg1 kompatibel ist.

Dann wird PRWP ausgegeben.

Beitrag von „hackopti2“ vom 12. Dezember 2021, 15:50

vielen Dank für die Erklärung. Rein aus Interesse, was bedeutet das praktisch: Die Geräte bekommen also bessere Infos zu Sleep/Wake-States?

Beitrag von „Brumbaer“ vom 13. Dezember 2021, 00:24

ACPI kennt GPEs General Purpose Events. Die Events können durch alles mögliche ausgelöst werden u.a. Power Taste, Serielle Schnittstelle.

Jeder dieser Events hat eine Nummer. Die Nummern sind nicht standardisiert, aber es gibt gebräuchliche Nummern.

0x6D wird u.a. von LAN und USB verwendet.

0x0D müsste ich nachschauen.

Geräten sind _PWR Methoden zugeordnet, die bestimmen aus welchem Sleep State ein Gerät ein Wake auslösen kann.

Die _PWR wird nicht in der DSDT oder SSDT aufgerufen sondern vom Host Computer aus. Was der mit der Info macht ist seine Sache.

Diese _PWR Methode ruft für gewöhnlich GPRW mit der Eventnummer und Sleep State als Argumenten auf.

Beim Mac muss, bei Abfragen für bestimmte Events, 0 als zweiter Wert zurückkommen.

Wir haben aber gesehen, dass in der Originalversion, die über die ich geschrieben habe, solange irgendein Sleep State enabled ist, immer ein Wert zwischen 1 und 4 im zweiten Wert zurück kommt (entweder der Wert in Argument1 oder der höchste enabelte Sleep State).

Die neue Version, die im ersten Post, schaut nach ob das Argument0 einer dieser Events ist und ob es sich um einen Rechner mit Darwin handelt.

Trifft beides zu gibt es Argument0 im ersten Wert (wie sonst auch) und 0 im zweiten Wert (Statt dem abgefragtem, bzw. dem höchsten möglichen Sleep Wert) zurück.

Statt eine extra Variable (PRWP in der Originalversion) anzulegen gibt die neue Funktion die Werte direkt als Package zurück - das ist nur Kosmetik.

Ist Argument0 keiner der beiden Eventnummern oder ist es kein Rechner mit Darwin, dann wir die Originalfunktion verwendet.

Beitrag von „5T33Z0“ vom 13. Dezember 2021, 10:57

Brumbaer Vielen Dank für die Erläuterungen. Denn ansonsten findet man nicht viel dazu, um es zu verstehen. DIe ACPI Dokumentation allein hilft bei solchen Sonderfällen auch nur bedingt weiter.