Erledigt Kein Ruhezustand nach Ruhezustand

Beitrag von „mull7965“ vom 29. Juli 2013, 20:24

> Zitat von Griven
>
> Vorab vielleicht mal zum Verständnis, der Flag Darkwake ist vom Typ Boolean kennt also nur yes oder no oder eben 1 oder 0. Des weiteren hat er nichts mit der Schlaffähigkeit Deines Hacks an sich zu tun sondern legt lediglich fest ob der Hack/Mac bei bestimmten Aufwachereignissen (Wake on Lan) quasi nur halb aufwacht sprich nur die Platten, Prozessor und Ram aufwachen die Grafik aber weiter schläft oder eben nicht. Anders gesagt bewirkt Darkwake = 0 lediglich, dass der MAC/HACK immer komplett aufwacht und Darkwake=1 eben das er bei bestimmten Aufwachereignissen zwar aufwacht der Monitor aber dunkel bleibt.

Das stimmt leider nur halb. Auch wenn es noch so häufig geschrieben wird, Darkwake ist kein Boolean. Man kann das glücklicherweise selbst nachsehen, direkt im Sourcecode des Kernels von OS X. (Für alle denen das nicht bekannt ist, Apple macht schon seit langer Zeit den Kernel und einige andere "low-level" Bestandteile von OS X unter einer Open Source Lizenz verfuegbar.) Eine gute Beschreibung der Behandlung des Darkwake Flags findet sich in einem Thread in einem anderen Hackintosh Forum das hier nicht erwähnt werden darf. Wer's mit Google/Bing/etc nicht findet, oder wer lieber Deutsch liest, für den folgt eine kurze Zusammenfassung.

Der Bootloader, also zum Beispiel Chameleon, hat mit Darkwake nichts am Hut und schickt das Flag mitsamt Wert direkt an den Kernel. Dieser liest das Flag mit Hilfe der Funktion **PE_parse_boot_argn** aus (Zeile 864 in IOPMrootDomain.cpp).

Code

```
1. PE_parse_boot_argn("darkwake", &gDarkWakeFlags, sizeof(gDarkWakeFlags));
```

Geschrieben wird der Wert in eine Variable die eine Bitmaske mit verschiedenen Flags enthält:

Code

```
1. // gDarkWakeFlags
2. enum {
3. kDarkWakeFlagHIDTickleEarly = 0x01, // hid tickle before gfx suppression
4. kDarkWakeFlagHIDTickleLate = 0x02, // hid tickle after gfx suppression
5. kDarkWakeFlagHIDTickleNone = 0x03, // hid tickle is not posted
6. kDarkWakeFlagHIDTickleMask = 0x03,
7. kDarkWakeFlagIgnoreDiskIOInDark = 0x04, // ignore disk idle in DW
8. kDarkWakeFlagIgnoreDiskIOAlways = 0x08, // always ignore disk idle
9. kDarkWakeFlagIgnoreDiskIOMask = 0x0C,
10. kDarkWakeFlagAlarmIsDark = 0x0100
11. };
```

Alles anzeigen

Interessanterweise hat **PE_parse_boot_argn** überhaupt keine spezielle Behandlung der Zeichenketten *yes* bzw. *no*, sie kennt lediglich die Typen String und Number (ab Zeile 104 in bootargs.c). Warum funktioniert dann *no* für einige Leute?

Weil es eben nur Number und String gibt, wird *no* als String angesehen und die Bytes für "n" und "o" einfach in die Variable vom Typ uint32 geschrieben. Das entspricht dann der Zahl 28526 ("n" = 110, "o" = 111, 111*256+110 = 28526). Man kann sich jetzt selbst überlegen welche der o.g. Flags damit getroffen werden.

Hoffe das hilft zur Klärung.