ALSD _ALI returns 0 (SMCLightSensor)

Beitrag von „BlvckBytes“ vom 15. Dezember 2021, 21:26

Hey!

Ich weis, das ist eigentlich bloß eine Spielerei... aber ich hätte echt gerne einen funktionierenden Ambient Light Sensor, nachdem dieser auf Windoof 10 auch sehr gute Dienste leistet. Mein System ist das Dell XPS 9700 mit i7-10875H und 4K-Display. Hab' mal den aktuellen EFI-folder angehangen.

Um mal kurz meinen bisherigen Ansatz zu beschreiben, zitiere ich das zu patchende Gerät aus meinen ACPI tables:

Code

```
1. Scope (_SB.PCI0.LPCB)
2. {
3. Device (ALSD)
4. {
5. Name (_HID, "ACPI0008" /* Ambient Light Sensor Device */) // _HID: Hardware ID
6. Method (_STA, 0, NotSerialized) // _STA: Status
7. {
8. If ((ALSE == 0x02))
9. {
10. Return (0x0B)
11. }
12.
13. Return (Zero)
14. }
15.
16. Method (_ALI, 0, NotSerialized) // _ALI: Ambient Light Illuminance
17. {
18. Return (((LHIH << 0x08) | LLOW))
19. }
20.
21. Name (_ALR, Package (0x05) // _ALR: Ambient Light Response
22. {
```

```
23. Package (0x02)
24. {
25. 0x46,
26. Zero
27. },
28.
29. Package (0x02)
30. {
31. 0x49,
32. 0x0A
33. },
34.
35. Package (0x02)
36. {
37. 0x55,
38. 0x50
39. },
40.
41. Package (0x02)
42. {
43. 0x64,
44. 0x012C
45. },
46.
47. Package (0x02)
48. {
49. 0x96,
50. 0x03E8
51. }
52. })
53. }
54. }
```

Alles anzeigen

Damit das Gerät in der IOReg aufscheint, musste ich natürlich ALSE auf 0x02 setzen, mittels SSDT. Gesagt getan, und schon mounted auch der LMU-Controller von Apple sowie der Eintrag in den Bildschirmeinstellungen (siehe Anhänge). Leider bewegt sich der Slider nicht, weshalb ich kurzerhand das Repo "VirtualSMC" geklont habe, um weitere Debugs einzufügen. Und zwar wollte ich wissen, was _ALI zurückgibt, nachdem diese Schnittstelle von SMCLightManager gepollt wird.

Code

```
1. bool SMCLightSensor::refreshSensor(bool post) {
2. uint32_t lux = 0;
3. auto ret = alsdDevice->evaluateInteger("_ALI", &lux);
4. if (ret != kIOReturnSuccess)
5. lux = 0xFFFFFFFF; // ACPI invalid
6.
7. DBGLOG("alsd", "Lux currently is: %llu", lux);
8.
9. atomic_store_explicit(&currentLux, lux, memory_order_release);
10.
11. if (post) {
12. VirtualSMCAPI::postInterrupt(SmcEventALSChange);
13. poller->setTimeoutMS(SensorUpdateTimeoutMS);
14. }
15.
16. return ret == kIOReturnSuccess;
17. }
```

Alles anzeigen

Danach konnte ich mittels dmesg den Wert auslesen, welcher recht enttäuschend ausfällt:

Code

```
1. [ 791.886212]: SMCLightSensor alsd: @ (DBG) Lux currently is: 0
2. [ 792.887578]: SMCLightSensor alsd: @ (DBG) Lux currently is: 0
3. [ 793.888480]: SMCLightSensor alsd: @ (DBG) Lux currently is: 0
4. [ 794.890898]: SMCLightSensor alsd: @ (DBG) Lux currently is: 0
5. [ 795.893362]: SMCLightSensor alsd: @ (DBG) Lux currently is: 0
6. [ 796.895817]: SMCLightSensor alsd: @ (DBG) Lux currently is: 0
7. [ 797.896634]: SMCLightSensor alsd: @ (DBG) Lux currently is: 0
8. [ 798.899057]: SMCLightSensor alsd: @ (DBG) Lux currently is: 0
```

Nun machte ich mich auf die Suche nach LHIH und LLOW, nachdem diese zwei Variablen den Output bestimmen.

Code

```
1. blvckbytes@BlvckBook origin % grep -E 'LHIH|LLOW' *.dsl
2. DSDT.dsl: External (LHIH, UnknownObj) // (from opcode)
3. DSDT.dsl: External (LLOW, UnknownObj) // (from opcode)
4. DSDT.dsl: Return (((LHIH << 0x08) | LLOW))
5. SSDT-2-SaSsdt.dsl: LLOW, 8,
6. SSDT-2-SaSsdt.dsl: LHIH, 8,
```

Die Variablen befinden sich in einer OperationRegion auf dem SystemMemory, im globalen Scope. Ansonsten gibts wohl keine weiteren Zugriffe mehr darauf...

Code

```
1. OperationRegion (SANV, SystemMemory, 0x55DA3018, 0x01F4)
2. Field (SANV, AnyAcc, Lock, Preserve)
3. {
4. <omitted>
5. }
```

Und nun steh ich an. Die Werte werden wohl von einer ganz anderen Stelle im System über ihre bekannten Speicheradressen beschrieben (oder auch nicht, wie die 0 zeigt...). Hat jemand von Euch Ideen, wie ich weitermachen könnte?