Sleep/Wake Probleme debuggen

Beitrag von „Tirom“ vom 12. Juli 2020, 21:19

Hallo apfelnico !

Danke für die gute Erklärung. Ich fasse noch mal kurz zusammen:

Ich Suche in meiner DSDT den Abschnitt _SB.PC00.RP13 und kopiere den gesamten "Scope" in eine neue .aml. In dieser nenne ich das "Device (PXSX)" nach "Device (XHC2)" um. Das sieht bei mir so aus:

Code

```
1. Scope (RP13)
2. {
3. Method (_INI, 0, NotSerialized) // _INI: Initialize
4. {
5. LTRN = LTRD /* \LTRD */
6. LMSL = PMLD /* \PMLD */
7. LNSL = PNLD /* \PNLD */
8. OBFN = OBFD /* \OBFD */
9. }
10.
11. OperationRegion (PXCS, PCI_Config, 0x00, 0x0480)
12. Field (PXCS, AnyAcc, NoLock, Preserve)
13. {
14. VDID, 32,
15. Offset (0x50),
16. L0SE, 1,
17. , 3,
18. LDIS, 1,
19. Offset (0x51),
20. Offset (0x52),
21. , 13,
22. LASX, 1,
23. Offset (0x5A),
24. ABPX, 1,
25. , 2,
26. PDCX, 1,
```

```
27. , 2,
28. PDSX, 1,
29. Offset (0x5B),
30. Offset (0x60),
31. Offset (0x62),
32. PSPX, 1,
33. PMEP, 1,
34. Offset (0xA4),
35. D3HT, 2,
36. Offset (0xD8),
37. , 30,
38. HPEX, 1,
39. PMEX, 1,
40. Offset (0xE2),
41. , 2,
42. L23E, 1,
43. L23R, 1,
44. Offset (0x324),
45. , 3,
46. LEDM, 1,
47. Offset (0x420),
48. , 30,
49. DPGE, 1
50. }
51.
52. Field (PXCS, AnyAcc, NoLock, WriteAsZeros)
53. {
54. Offset (0xDC),
55. , 30,
56. HPSX, 1,
57. PMSX, 1
58. }
59.
60. Name (LTRV, Package (0x04)
61. {
62. 0x00,
63. 0x00,
64. 0x00,
65. 0x00
66. })
67. Method (_DSM, 4, Serialized) // _DSM: Device-Specific Method
68. {
```

```
69. If ((Arg0 == ToUUID ("e5c937d0-3553-4d7a-9117-ea4d19c3434d") /* Device Labeling Interface */))
70. {
71. Switch (ToInteger (Arg2))
72. {
73. Case (0x00)
74. {
75. Name (OPTS, Buffer (0x02)
76. {
77. 0x00, 0x00 // ..
78. })
79. CreateBitField (OPTS, 0x00, FUN0)
80. CreateBitField (OPTS, 0x04, FUN4)
81. CreateBitField (OPTS, 0x06, FUN6)
82. CreateBitField (OPTS, 0x08, FUN8)
83. CreateBitField (OPTS, 0x09, FUN9)
84. If ((Arg1 >= 0x02))
85. {
86. FUN0 = 0x01
87. If (LTRE)
88. {
89. FUN6 = 0x01
90. }
91.
92. If (OBFF)
93. {
94. FUN4 = 0x01
95. }
96.
97. If ((ECR1 == 0x01))
98. {
99. If ((Arg1 >= 0x03))
100. {
101. FUN8 = 0x01
102. FUN9 = 0x01
103. }
104. }
105. }
106.
107. Return (OPTS) /* \_SB_.PC00.RP13._DSM.OPTS */
108. }
109. Case (0x04)
110. {
```

```
111. If ((Arg1 >= 0x02))
112. {
113. If (OBFN)
114. {
115. Return (Buffer (0x10)
116. {
117. /* 0000 */ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, // ........
118. /* 0008 */ 0x00, 0x00, 0x00, 0x08, 0x00, 0x00, 0x00, 0x00 // ........
119. })
120. }
121. Else
122. {
123. Return (Buffer (0x10)
124. {
125. /* 0000 */ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, // ........
126. /* 0008 */ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 // ........
127. })
128. }
129. }
130. }
131. Case (0x06)
132. {
133. If ((Arg1 >= 0x02))
134. {
135. If (LTRN)
136. {
137. If (((LMSL == 0x00) || (LNSL == 0x00)))
138. {
139. If ((PCHS == SPTH))
140. {
141. LMSL = 0x0846
142. LNSL = 0x0846
143. }
144. ElseIf ((PCHS == SPTL))
145. {
146. LMSL = 0x1003
147. LNSL = 0x1003
148. }
149. }
150.
151. LTRV [0x00] = ((LMSL >> 0x0A) & 0x07)
152. LTRV [0x01] = (LMSL & 0x03FF)
```

```
153. LTRV [0x02] = ((LNSL >> 0x0A) & 0x07)
154. LTRV [0x03] = (LNSL & 0x03FF)
155. Return (LTRV) /* \_SB_.PC00.RP13.LTRV */
156. }
157. Else
158. {
159. Return (Buffer (0x01)
160. {
161. 0x00 // .
162. })
163. }
164. }
165. }
166. Case (0x08)
167. {
168. If ((ECR1 == 0x01))
169. {
170. If ((Arg1 >= 0x03))
171. {
172. Return (0x01)
173. }
174. }
175. }
176. Case (0x09)
177. {
178. If ((ECR1 == 0x01))
179. {
180. If ((Arg1 >= 0x03))
181. {
182. Return (Package (0x05)
183. {
184. 0xC350,
185. Ones,
186. Ones,
187. 0xC350,
188. Ones
189. })
190. }
191. }
192. }
193.
194. }
```

```
195. }
196.
197. Return (Buffer (0x01)
198. {
199. 0x00 // .
200. })
201. }
202.
203. Device (XHC2)
204. {
205. Name (_ADR, 0x00) // _ADR: Address
206. Method (_PRW, 0, NotSerialized) // _PRW: Power Resources for Wake
207. {
208. Return (GPRW (0x69, 0x04))
209. }
210. }
211.
212. Method (HPME, 0, Serialized)
213. {
214. If (((VDID != 0xFFFFFFFF) && (PMSX == 0x01)))
215. {
216. Notify (PXSX, 0x02) // Device Wake
217. PMSX = 0x01
218. PSPX = 0x01
219. }
220. }
221.
222. Method (_PRW, 0, NotSerialized) // _PRW: Power Resources for Wake
223. {
224. Return (GPRW (0x69, 0x04))
225. }
226. }
```

Alles anzeigen

Jetzt habe ich aber Fragen:

Welchen Header muss ich noch hinzufügen? Nehme ich auch den aus meiner DSDT?

Wenn dann diese SSDT hinzugefügt wird, muss man schauen, dass die Info nicht doppelt vorhanden ist, richtig? Also muss der Bereich in der original DSDT entfernt werden. Daher der ACPI Delete in der config.plist.

Aber woher weiß ich, was gelöscht werden muss. 53534454 bedeutet SSDT, aber woher

bekomme ich die TableLength. Und woher weiß OC, wann der Abschnitt beginnt.

Sorry, da stehe ich etwas auf dem Schlauch.  Gibt es irgendwo eine Einführung: SSDTs für Dummies?