

## **Erledigt** SMBUS verhindert Sleep

**Beitrag von „Aluveitie“ vom 28. August 2020, 17:54**

Habe wieder mal etwas romprobiert und versucht SMBUS nach <https://dortania.github.io/Get...ACPI/Universal/smbus.html> zu enablen.

Zwar werden die Kexte auch ohne geladen, mit der SSDT-SBUS-MCHC.aml erscheint dann in der PCI Übersicht der Hacken bei "Driver Installed".

Sobald die SSDT jedoch drin ist geht der Rechner nicht mehr in Sleep sondern bleibt einfach am laufen.

Laut dem Guide sind die Vorteile davon minimal, mich würde trotzdem interessieren ob jemand da noch mehr Ahnung hat?

---

**Beitrag von „apfelnico“ vom 28. August 2020, 20:37**

Sobald SBUS per \_DSM-Methode deklariert wird, fällt Sleep flach.

Bei meinem X299 ist noch ein weiteres Device auf der gleichen Adresse vorhanden, welches ich unter macOS nicht haben möchte: "SMBS". Dann habe ich "SBUS" deklariert, die initiale \_DSM aber weggelassen. Möglicherweise kannst du diesen Part durch den Bootloader erledigen lassen, über "Device Properties". So sieht der Part innerhalb der SSDT für mein System aus:

Code

1. Scope (\\_SB.PC00.SMBS)
2. {
3. Method (\_STA, 0, NotSerialized) // \_STA: Status
4. {
5. If (\_OSI ("Darwin"))
6. {

```
7. Return (Zero)
8. }
9. Else
10. {
11. Return (0x0F)
12. }
13. }
14. }
15.
16. Scope (\_SB.PC00.SBUS)
17. {
18. If (_OSI ("Darwin"))
19. {
20. OperationRegion (SMBP, PCI_Config, 0x40, 0xC0)
21. Field (SMBP, DWordAcc, NoLock, Preserve)
22. {
23. , 2,
24. I2CE, 1
25. }
26.
27. OperationRegion (SMPB, PCI_Config, 0x20, 0x04)
28. Field (SMPB, DWordAcc, NoLock, Preserve)
29. {
30. , 5,
31. SBAR, 11
32. }
33.
34. OperationRegion (SMBI, SystemIO, (SBAR << 0x05), 0x10)
35. Field (SMBI, ByteAcc, NoLock, Preserve)
36. {
37. HSTS, 8,
38. Offset (0x02),
39. HCON, 8,
40. HCOM, 8,
41. TXSA, 8,
42. DAT0, 8,
43. DAT1, 8,
44. HBDR, 8,
45. PECR, 8,
46. RXSA, 8,
47. SDAT, 16
```

```
48. }
49.
50. Method (SSXB, 2, Serialized)
51. {
52. If (STRT ())
53. {
54. Return (Zero)
55. }
56.
57. I2CE = Zero
58. HSTS = 0xBF
59. TXSA = Arg0
60. HCOM = Arg1
61. HCON = 0x48
62. If (COMP ())
63. {
64. HSTS |= 0xFF
65. Return (One)
66. }
67.
68. Return (Zero)
69. }
70.
71. Method (SRXB, 1, Serialized)
72. {
73. If (STRT ())
74. {
75. Return (0xFFFF)
76. }
77.
78. I2CE = Zero
79. HSTS = 0xBF
80. TXSA = (Arg0 | One)
81. HCON = 0x44
82. If (COMP ())
83. {
84. HSTS |= 0xFF
85. Return (DAT0) /* \_SB_.PC00.SBUS.DAT0 */
86. }
87.
88. Return (0xFFFF)
89. }
```

```
90.
91. Method (SWRB, 3, Serialized)
92. {
93. If (STRT ())
94. {
95. Return (Zero)
96. }
97.
98. I2CE = Zero
99. HSTS = 0xBF
100. TXSA = Arg0
101. HCOM = Arg1
102. DAT0 = Arg2
103. HCON = 0x48
104. If (COMP ())
105. {
106. HSTS |= 0xFF
107. Return (One)
108. }
109.
110. Return (Zero)
111. }
112.
113. Method (SRDB, 2, Serialized)
114. {
115. If (STRT ())
116. {
117. Return (0xFFFF)
118. }
119.
120. I2CE = Zero
121. HSTS = 0xBF
122. TXSA = (Arg0 | One)
123. HCOM = Arg1
124. HCON = 0x48
125. If (COMP ())
126. {
127. HSTS |= 0xFF
128. Return (DAT0) /* \_SB_.PC00.SBUS.DAT0 */
129. }
130.
131. Return (0xFFFF)
```

```
132. }
133.
134. Method (SWRW, 3, Serialized)
135. {
136. If (STRT ())
137. {
138. Return (Zero)
139. }
140.
141. I2CE = Zero
142. HSTS = 0xBF
143. TXSA = Arg0
144. HCOM = Arg1
145. DAT1 = (Arg2 & 0xFF)
146. DAT0 = ((Arg2 >> 0x08) & 0xFF)
147. HCON = 0x4C
148. If (COMP ())
149. {
150. HSTS |= 0xFF
151. Return (One)
152. }
153.
154. Return (Zero)
155. }
156.
157. Method (SRDW, 2, Serialized)
158. {
159. If (STRT ())
160. {
161. Return (0xFFFF)
162. }
163.
164. I2CE = Zero
165. HSTS = 0xBF
166. TXSA = (Arg0 | One)
167. HCOM = Arg1
168. HCON = 0x4C
169. If (COMP ())
170. {
171. HSTS |= 0xFF
172. Return (((DAT0 << 0x08) | DAT1))
173. }
```

```
174.
175. Return (0xFFFFFFFF)
176. }
177.
178. Method (SBLW, 4, Serialized)
179. {
180. If (STRT ())
181. {
182. Return (Zero)
183. }
184.
185. I2CE = Arg3
186. HSTS = 0xBF
187. TXSA = Arg0
188. HCOM = Arg1
189. DAT0 = SizeOf (Arg2)
190. Local1 = Zero
191. HBDR = DerefOf (Arg2 [Zero])
192. HCON = 0x54
193. While ((SizeOf (Arg2) > Local1))
194. {
195. Local0 = 0x4E20
196. While (!(HSTS & 0x80) && Local0)
197. {
198. Local0--
199. }
200.
201. If (!Local0)
202. {
203. KILL ()
204. Return (Zero)
205. }
206.
207. Local1++
208. If ((SizeOf (Arg2) > Local1))
209. {
210. HBDR = DerefOf (Arg2 [Local1])
211. HSTS = 0x80
212. }
213. }
214.
215. HSTS = 0x80
```

```

216. If (COMP ())
217. {
218. HSTS |= 0xFF
219. Return (One)
220. }
221.
222. Return (Zero)
223. }
224.
225. Method (SBLR, 3, Serialized)
226. {
227. Name (TBUF, Buffer (0x0100){})
228. If (STRT ())
229. {
230. Return (Zero)
231. }
232.
233. I2CE = Arg2
234. HSTS = 0xBF
235. TXSA = (Arg0 | One)
236. HCOM = Arg1
237. HCON = 0x54
238. Local0 = 0x0FA0
239. While (!(HSTS & 0x80) && Local0))
240. {
241. Local0--
242. Stall (0x32)
243. }
244.
245. If (!Local0)
246. {
247. KILL ()
248. Return (Zero)
249. }
250.
251. TBUF [Zero] = DAT0 /* \_SB_.PC00.SBUS.DAT0 */
252. HSTS = 0x80
253. Local1 = One
254. While ((Local1 < DerefOf (TBUF [Zero])))
255. {
256. Local0 = 0x0FA0
257. While (!(HSTS & 0x80) && Local0))

```

```
258. {
259. Local0--
260. Stall (0x32)
261. }
262.
263. If (!Local0)
264. {
265. KILL ()
266. Return (Zero)
267. }
268.
269. TBUF [Local1] = HBDR /* \_SB_.PC00.SBUS.HBDR */
270. HSTS = 0x80
271. Local1++
272. }
273.
274. If (COMP ())
275. {
276. HSTS |= 0xFF
277. Return (TBUF) /* \_SB_.PC00.SBUS.SBLR.TBUF */
278. }
279.
280. Return (Zero)
281. }
282.
283. Method (STRT, 0, Serialized)
284. {
285. Local0 = 0xC8
286. While (Local0)
287. {
288. If ((HSTS & 0x40))
289. {
290. Local0--
291. Sleep (One)
292. If ((Local0 == Zero))
293. {
294. Return (One)
295. }
296. }
297. Else
298. {
299. Local0 = Zero
```



```
300. }
301. }
302.
303. Local0 = 0x0FA0
304. While (Local0)
305. {
306. If ((HSTS & One))
307. {
308. Local0--
309. Stall (0x32)
310. If ((Local0 == Zero))
311. {
312. KILL ()
313. }
314. }
315. Else
316. {
317. Return (Zero)
318. }
319. }
320.
321. Return (One)
322. }
323.
324. Method (COMP, 0, Serialized)
325. {
326. Local0 = 0x0FA0
327. While (Local0)
328. {
329. If ((HSTS & 0x02))
330. {
331. Return (One)
332. }
333. Else
334. {
335. Local0--
336. Stall (0x32)
337. If ((Local0 == Zero))
338. {
339. KILL ()
340. }
```

```
341. }
342. }
343.
344. Return (Zero)
345. }
346.
347. Method (KILL, 0, Serialized)
348. {
349. HCON |= 0x02
350. HSTS |= 0xFF
351. }
352.
353. Device (BUS0)
354. {
355. Name (_CID, "smbus") // _CID: Compatible ID
356. Name (_ADR, Zero) // _ADR: Address
357. Device (BLC0)
358. {
359. Name (_ADR, Zero) // _ADR: Address
360. Name (_CID, "smbus-blc") // _CID: Compatible ID
361. If (_OSI ("Darwin"))
362. {
363. Method (_DSM, 4, NotSerialized) // _DSM: Device-Specific Method
364. {
365. If ((Arg0 == ToUUID ("a0b5b7c6-1318-441c-b0c9-fe695eaf949b")))
366. {
367. Local0 = Package (0x02)
368. {
369. "refnum",
370. Zero
371. }
372. DTGP (Arg0, Arg1, Arg2, Arg3, RefOf (Local0))
373. Return (Local0)
374. }
375.
376. Return (Zero)
377. }
378. }
379. }
380. }
381.
382. Device (BUS1)
```

```

383. {
384. Name (_CID, "smbus") // _CID: Compatible ID
385. Name (_ADR, One) // _ADR: Address
386. }
387. }
388. }

```

Alles anzeigen

Edit:

sobald ich das dazu gepackt hatte, funktioniert Sleep ebenfalls nicht mehr, also "Jacke wie Hose". Nur der Part in der SSDT macht was es soll ...

▼ DeviceProperties	Dictionary	◇ 2 Schlüssel/Wert-Paare
▼ Add	Dictionary	◇ 6 Schlüssel/Wert-Paare
▶ PciRoot(0x0)/Pci(0x14,0x0)	Dictionary	◇ 6 Schlüssel/Wert-Paare
▶ PciRoot(0x0)/Pci(0x17,0x0)	Dictionary	◇ 5 Schlüssel/Wert-Paare
▶ PciRoot(0x0)/Pci(0x1F,0x2)	Dictionary	◇ 5 Schlüssel/Wert-Paare
▶ PciRoot(0x0)/Pci(0x1F,0x3)	Dictionary	◇ 12 Schlüssel/Wert-Paare
▼ PciRoot(0x0)/Pci(0x1F,0x4)	Dictionary	◇ 5 Schlüssel/Wert-Paare
AAPL_slot-name	String	◇ Built In
built-in	Daten	◇ 1 Bytes: 00
device_type	String	◇ SBUS-Controller
model	String	◇ Intel X299 Series System Management Bus
name	String	◇ SBUS

### Beitrag von „Aluveitie“ vom 28. August 2020, 21:11

Danke für die ausführliche Antwort, interessant. Der SMBUS Controller ist auf meinem System nicht in der DSDT drin, ohne die SSDT ist er nur als PCI device gelistet.

Muss mal probieren wie sich das verhält wenn ich die \_DSM weglasse [SSDT-SBUS-MCHC.aml](#)

### Beitrag von „Medallo“ vom 30. August 2020, 03:20

Device	Type	Driver installed	Name
7 Series/2790 Series Chipset Family 8 port SATA Controller [AHCI mode]	SATA controller	0	InternalSATA
7 Series/2790 Series Chipset Family USB eHCI Host Controller	USB controller	0	InternalUSB
7 Series/2790 Chipset Family High Definition Audio Controller	Audio device	0	InternalHDA
7 Series/2790 Chipset Family LAN Controller #1	Communication controller	0	InternalLME
7 Series/2790 Chipset Family SMBus Controller	SMBus	0	InternalSMB
7 Series/2790 Chipset Family USB Enhanced Host Controller #1	USB controller	0	InternalUEH1
7 Series/2790 Chipset Family USB Enhanced Host Controller #2	USB controller	0	InternalUEH2
Intel 82579L PCI Express Gigabit Ethernet	Ethernet controller	0	InternalE1G
Intel HD Graphics 4000	Video controller	0	InternalVGA
ASUS SuperM 32 TB	Controller de partiții	0	InternalSATA
pci1049d	Supravoltaj de audio	0	InternalSMB

same problem.

Driver not installed, but if I put SSDT-SBUS-MCHC.aml in EFI-EFI-OC-ACPI = "Driver Yes Installed".

But problems with sleep, it does not turn off cpu fans, chassis, etc.

## Beitrag von „Aluveitie“ vom 7. Oktober 2021, 11:00

Nachtrag: Injecten von Device Properties damit der SMBus Controller unter PCIe Devices aufgelistet wird war schuld, dass Sleep nicht mehr funktioniert. Ohne diese funktioniert alles.