DSDT Tuturial

Beitrag von „iLeopod“ vom 5. Juli 2011, 20:00

DTGP /MCDP

Als Grundlage für die DSM Methoden müssen wir die DTGP Methode deklarieren. Erst mal suchen denn in den vorgefertigten DSDT's ist diese schon vorhanden:

Code

```
1. Method (DTGP, 5, NotSerialized)
2. {
3. If (LEqual (Arg0, Buffer (0x10)
4. {
5. /* 0000 */ 0xC6, 0xB7, 0xB5, 0xA0, 0x18, 0x13, 0x1C, 0x44,
6. /* 0008 */ 0xB0, 0xC9, 0xFE, 0x69, 0x5E, 0xAF, 0x94, 0x9B
7. }))
8. {
9. If (LEqual (Arg1, One))
10. {
11. If (LEqual (Arg2, Zero))
12. {
13. Store (Buffer (One)
14. {
15. 0x03
16. }, Arg4)
17. Return (One)
18. }
19.
20.
21. If (LEqual (Arg2, One))
22. {
23. Return (One)
24. }
25. }
26. }
27.
28.
29. Store (Buffer (One)
30. {
31. 0x00
```

32. }, Arg4)
33. Return (Zero)
34. }

Alles anzeigen

Wenn man eine neue DSDT bearbeitet und wert auf einen schlanken Code legt, kann man auch anstatt der DTGP Methode die MCDP Methode verwenden:

Code

1. Method (MCDP, 2, NotSerialized) // New Method V1.1 Đ By Master Chief.
2. {
3. If (LEqual (Arg0, Zero))
4. {
5. Store (Buffer (One)
6. {
7. 0x03
8. }, Arg1)
9. }
10. }

Nach jeder DSM Methode hat man dann:

Code

1. MCDP (Arg2, RefOf (Local0))

anstatt:

Code

1. DTGP (Arg0, Arg1, Arg2, Arg3, RefOf (Local0))

Package und Buffer Größen

Code

1. Method (_DSM, 4, NotSerialized)
2. {
3. Store (Package (0x04)
4. {

5. "device-id",
6. Buffer (0x04)
7. {.....}
8. //....

Wir sehen hier beides mal 0x04 wenn wir und dem betreffenden untergeordnetem Bereich etwas ändern stimmt dieser Wert nicht mehr und der Bereich ist wirkungslos. Man kann die Längen mühsam berechnen oder es den compiler machen lassen indem man die Klammern einfach leer lässt:

Code

1. Method (_DSM, 4, NotSerialized)
2. {
3. Store (Package ()
4. {
5. "device-id",
6. Buffer ()
7. {.....}
8. //....

So können auch unnötige Fehlerquellen vermieden werden.
Compiling Errors
.....