



Win7 64位旗舰版下载系统工具主题
论坛汇聚Win7系统下载激活工具主题
[Win7系统下载](#)|[主题](#)|[优化](#)|[桌面](#)|[激活](#)|[小白必读](#)



Win8.1系统下载中文版平板电脑主题
Win8.1平板电脑中文版系统下载尽在Win8论坛
[Windows8.1下载](#)|[主题](#)|[激活](#)|[新手宝典](#)|[必备软件](#)



Win10正式版系统下载主题平板
重定义Modern UI,打造完美Windows全新体验
[Windows10下载](#)|[安装](#)|[新手宝典](#)|[必备软件](#)

请输入搜索内容

本版

u盘 U盘 无线网卡 网卡 移动硬盘 摄像头 cad 游戏 windows 2003
读卡器 触摸屏 侦探柯南 solidworks 古剑 升级 睡眠 唤醒 迅雷 激活

论坛

国内权威黑苹果论坛 - DIY你的苹果系统

High Sierra

关于电池显示的解决方案（笔记本）

发表新帖

回复本帖

返回列表

查看: 249 | 回复: 8

YangTuDou1220





☆☆

UID4746895

帖子231

PB币102

贡献0

技术0

活跃164

[交流] 关于电池显示的解决方案（笔记本）

发表于 2017-10-30 16:14:32 | 只看该作者 | 倒序浏览

分享到：

新浪微博

腾讯微博

开心网

人人网

更多

0

本帖最后由 YangTuDou1220 于 2017-10-30 21:21 编辑

首先声明本帖基于Rehabman的帖子的延伸，确切的来说是我自己的一点思考。本文参考贴： [外文原帖](#)。

论坛大神多，我就不班门弄斧了。大致梳理一下思路和大家做了一个模板作为参考。

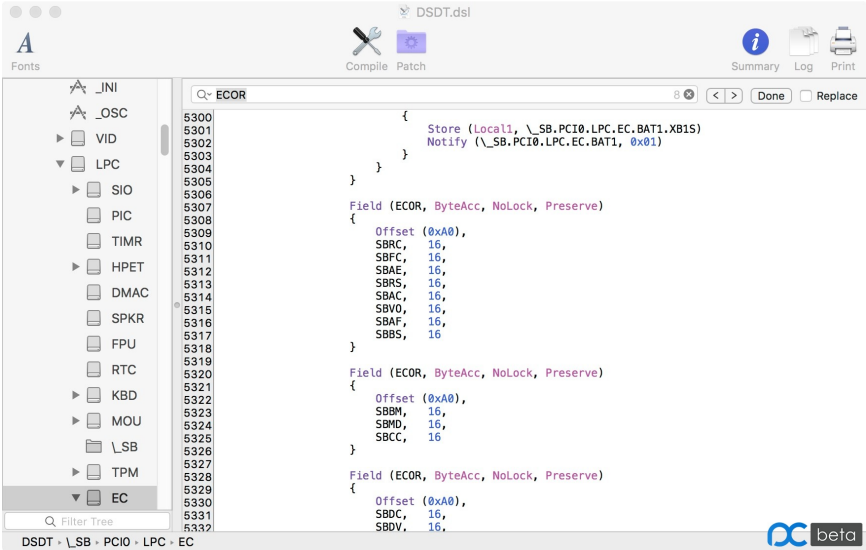
本人机器Thinkpad T440

1.提取DSDT&ssdt——clover界面F4，等待5秒左右。

2.在DSDT里面搜索关键词**EmbeddedControl**，然后找出前面自己的电池名字比方说我是**ECOR**。如图：



3.接着搜索ECOR你将发现一个或者一个以上的地方有着名字。那么在这些名字下面有16位的32位的活着的56，128位之类的，而你的目的就是让他们变成8位。



4. 下面一步涉及的是拆分为8位。

1. 罗列出所有的相同位数的名字，验证它们是否在DSDT里出现过一次，并记录他们的位置。

# 16	
SBRC	_SB.PCI0.LPC.EC.GBST
SBFC	_SB.PCI0.LPC.EC.GBIF
SBAE	这里只出现一次就不考虑了
SBRS	这里只出现一次就不考虑了
SBAC	_SB.PCI0.LPC.EC.GBST
SBVO	_SB.PCI0.LPC.EC.GBST
SBAF	这里只出现一次就不考虑了
SBBS	这里只出现一次就不考虑了
SBBM	_SB.PCI0.LPC.EC.GBIF
SBMD	这里只出现一次就不考虑了
SBCC	这里只出现一次就不考虑了
SBDC	_SB.PCI0.LPC.EC.GBIF
SBDV	_SB.PCI0.LPC.EC.GBIF
SBOM	这里只出现一次就不考虑了
SBSI	这里只出现一次就不考虑了
SBDT	这里只出现一次就不考虑了
SBSN	_SB.PCI0.LPC.EC.GBIF

# 32	
SBCH	_SB.PCI0.LPC.EC.GBIF

# 128	
SBMN	_SB.PCI0.LPC.EC.GBIF
SBDN	_SB.PCI0.LPC.EC.GBIF

2. 整理一下，不用的删掉，把同一地址的放在一起。

_SB.PCI0.LPC.EC.GBIF	_SB.PCI0.LPC.EC.GBST
SBFC (16)	SBRC (16)
SBBM (16)	SBAC (16)
SBDC (16)	SBVO (16)
SBDV (16)	
SBSN (16)	
SBCH (32)	
SBMN&SBDN (128)	

3. 初步拆解，遵循的原理是16——2x8， 32——4x8， 128——nx8等

至于名字怎么起，看心情。

这里要注意一点的是拆完后的名字只能在DSDT里出现过一次！

SBFC	BFC0, BFC1
SBBM	BBM0, BBM1

SBDC	BDC0, BDC1
SBDV	BDV0, BDV1
SBSN	BSN0, BSN1
SBRC	BRC0, BRC1
SBAC	BAC0, BAC1
SBVO	BVO0, BVO1
SBCH	BCH0, BCH1, BCH2, BCH3
SBMN&SBDN (128)	这个比较特殊后面会有提起

4.拆解代码
ps:这里加粗的字都是要按照自己电脑的名字写的。

SBFC	<div>01. into device label EC code_regex SBFC,\s+16 replace_matched begin BFC0,8,BFC1,8 end; 复制代码</div>	
SBBM	<div>01. into device label EC code_regex SBBM,\s+16 replace_matched begin BBM0,8,BBM1,8 end; 复制代码</div>	
SBDC	<div>01. into device label EC code_regex SBDC,\s+16 replace_matched begin BDC0,8,BDC1,8 end; 复制代码</div>	
SBDV	<div>01. into device label EC code_regex SBDV,\s+16 replace_matched begin BDV0,8,BDV1,8 end; 复制代码</div>	
SBSN	<div>01. into device label EC code_regex SBSN,\s+16 replace_matched begin BSN0,8,BSN1,8 end; 复制代码</div>	
SBRC	<div>01. into device label EC code_regex SBRC,\s+16 replace_matched begin BRC0,8,BRC1,8 end; 复制代码</div>	
SBAC	<div>01. into device label EC code_regex SBAC,\s+16 replace_matched begin BAC0,8,BAC1,8 end; 复制代码</div>	
SBVO		

	<div>01. into device label EC code_regex SBV0,\s+16 replace_matched begin BV00,8,BV01,8 end; 复制代码</div>	
SBCH	<div>01. Into device label EC code_regex SBCH,\s+32 replace_matched begin BCH0,8,BCH1,8,BCH2,8,BCH3,8 end; 复制代码</div>	
SBMN&SBDN (128)	<div>01. into device label EC code_regex (SBMN/SBDN,)\s+(128) replace_matched begin BMNX,%2,/%1%2/BDNX,%2,/%1%2 end; 复制代码</div>	/的意思都懂得吧

5.拆分完自然是要用这些拆分好的，下一步就是调用这些拆分好的。

SBFC	<div>01. Into method label GBIF code_regex \ (SBFC, replaceall_matched begin (B1B2(BFC0,BFC1), end; 复制代码</div>	
SBBM	<div>01. Into method label GBIF code_regex \ (SBBM, replaceall_matched begin (B1B2(BBM0,BBM1), end; 复制代码</div>	
SBDC	<div>01. Into method label GBIF code_regex \ (SBDC, replaceall_matched begin (B1B2(BDC0,BDC1), end; 复制代码</div>	
SBDV	<div>01. Into method label GBST code_regex \ (SBDV, replaceall_matched begin (B1B2(BDV0,BDV1), end; 复制代码</div>	
SBSN	<div>01. Into method label GBIF code_regex \ </div>	

	<div><div>(SBSN, replaceall_matched begin (B1B2(BSN0,BSN1), end; 复制代码</div></div>	
SBRC	<div><div>01. Into method label GBST code_regex \ (SBRC, replaceall_matched begin (B1B2(BRC0,BRC1), end; 复制代码</div></div>	
SBAC	<div><div>01. Into method label GBST code_regex \ (SBAC, replaceall_matched begin (B1B2(BAC0,BAC1), end; 复制代码</div></div>	
SBVO	<div><div>01. Into method label GBST code_regex \ (SBV0, replaceall_matched begin (B1B2(BV00,BV01), end; 复制代码</div></div>	
SBCH	<div><div>01. into method label GBIF code_regex \ (SBCH, replaceall_matched begin (B1B4(BCH0,BCH1,BCH2,BCH3), end; 复制代码</div></div>	
SBMN&SBDN (128)	<div><div>01. into method label GBIF code_regex \ (SBMN, replaceall_matched begin (RECB(0xA0,128), end; 复制代码</div></div>	<div><div>01. into method label GBIF code_regex \ (SBDN, replaceall_matched begin (RECB(0xA0,128), end; 复制代码</div></div>

6.这些都准备好了后细心的朋友应该已经发现多了一些东西**B1B2**，**B1B4**，**RECB**这些，那么这些怎么定义呢，这里我直接抄袭了RM的//

创建B1B2 基本都用这个

01. Method
02.
03. into
04. method label B1B2 remove_entry;
05.
06. into

```
07. definitionblock code_regex . insert
08.
09. begin
10.
11. Method
12. (B1B2, 2, NotSerialized) { Return(Or(Arg0, ShiftLeft(Arg1, 8))) }
13.
14. \n
15.
16. end;
```

[复制代码](#)

#Create B1B4 Method 添加B1B4 函数（32位）

```
01. into
02. method label B1B4 remove_entry;
03.
04. into
05. definitionblock code_regex . insert
06.
07. begin
08.
09. Method
10. (B1B4, 4, NotSerialized)\n
11.
12. {\n
13.
14.     Store(Arg3, Local0)\n
15.
16.     Or(Arg2, ShiftLeft(Local0, 8), Local0)\n
17.
18.     Or(Arg1, ShiftLeft(Local0, 8), Local0)\n
19.
20.     Or(Arg0, ShiftLeft(Local0, 8), Local0)\n
21.
22.     Return(Local0)\n
23.
24. }\n
25.
26. end;
```

[复制代码](#)

还有 56位什么的修改方式类似

超过32位调用，不添加会有RECB问题

utility methods to read/write buffers from/to EC

```
01. into
02. method label RE1B parent_label EC remove_entry;
03.
04. into method
05. label RECB parent_label EC remove_entry;
06.
07. into
08. device label EC insert
09.
10. begin
11.
12. Method
13. (RE1B, 1, NotSerialized)\n
14.
15. //
16. Arg0 - offset in bytes from zero-based EC\n
17.
18. {\n
19.
20.     OperationRegion(ERAM,
21.     EmbeddedControl, Arg0, 1)\n
22.
23.     Field(ERAM, ByteAcc, NoLock, Preserve) {
```

```
24. BYTE, 8 } \n
25.
26.     Return(BYTE) \n
27.
28. } \n
29.
30. Method
31. (RECB, 2, Serialized) \n
32.
33. //
34. Arg0 - offset in bytes from zero-based EC \n
35.
36. //
37. Arg1 - size of buffer in bits \n
38.
39. { \n
40.
41.     ShiftRight(Arg1,
42.     3, Arg1) \n
43.
44.     Name(TEMP,
45. Buffer(Arg1) { }) \n
46.
47.     Add(Arg0,
48. Arg1, Arg1) \n
49.
50.     Store(0,
51. Local0) \n
52.
53.     While
54. (LLess(Arg0, Arg1)) \n
55.
56.     { \n
57.
58.         Store(RE1B(Arg0),
59. Index(TEMP, Local0)) \n
60.
61. Increment(Arg0) \n
62.
63. Increment(Local0) \n
64.
65.     } \n
66.
67.     Return(TEMP) \n
68.
69. } \n
70.
71. end;
复制代码
```

7.最后把这些全部写完就可以用了，我这里我把写的贴上作参考。

// 下一个更新用hotpatch的方式来搞定这个

 [T440 电池显示 dsdt补丁代码.zip](#) (15.66 KB, 下载次数: 0)
