Table of Contents

OpenCore 0.5+ 部件补丁

说明

依据 OpenCore 0.5+ 的要求和建议，制作本部件补丁。

在线手册

本仓库依赖 GitBook 并使用 GitHub Actions 构建了 Pages 服务和 PDF 手册。

- https://ocbook.tlhub.cn
- OpenCore 部件库

主要内容

1. 总述
    i. ASL 语法基础
    ii. SSDT 补丁加载顺序
    iii. ACPI 表单
    iv. ASL-AML 对照表
2. 关于 `AOAC`
    i. 禁止 `S3` 睡眠
    ii. `AOAC` 禁用独显
    iii. 电源空闲管理
    iv. `AOAC` 唤醒补丁
    v. 设置 `ASPM` 工作模式
    vi. 睡眠自动关闭蓝牙 `WIFI`
3. 仿冒设备
    i. 仿冒 `EC`
    ii. RTC0
    iii. 键盘无法输入的应急解决方案 `PS2N`
    iv. 仿冒环境光传感器
4. 二进制更名与预置变量
    i. OC `I2C-GPIO` 补丁
5. 操作系统补丁
6. 注入设备
    i. 注入 X86
    ii. `PNLF` 注入方法
    iii. `SBUS(SMBU)` 补丁

7. 添加缺失的部件
8. **PS2 键盘映射及亮度快捷键 @OC-xlivans**
9. 电池补丁
    i. Thinkpad
    ii. 其它品牌
10. 禁用 **EHCx**
11. `PTSWAK` 综合扩展补丁
12. `PNP0C0E` 睡眠修正方法
13. `0D6D` 补丁
    i. 普通的 `060D` 补丁
    ii. 惠普特殊的 `060D` 补丁
14. 仿冒以太网和重置以太网 `BSD Name`
15. `CMOS` 相关
    i. `CMOS` 内存和 ***RTCMemoryFixup***
16. `ACPI` 定制 `USB` 端口
17. 禁止 `PCI` 设备
18. **ACPIDebug**
19. 品牌机器特殊补丁
    i. `Dell` 机器特殊补丁
    ii. 小新 PRO13 特殊补丁
    iii. ThinkPad 机器专用补丁
20. `I2C` 专用部件
21. `SSDT` 屏蔽独显方法

保留部件：

1. 声卡 `IRQ` 补丁
2. `CMOS` 重置补丁

常见驱动加载顺序：

1. config-1-Lilu-SMC-WEG-ALC 驱动列表
2. config-2-PS2 键盘驱动列表
3. config-3-BCM 无线和蓝牙驱动列表
4. config-4-I2C 驱动列表
5. config-5-PS2Smart 键盘驱动列表
6. config-6-Intel 无线和蓝牙驱动列表

Credits

- 特别感谢：
  - @宪武 制作的适用于 **OpenCore** 的 ACPI 部件补丁
  - @Bat.bat, @黑果小兵, @套陆, @iStar丶Forever 审核完善
- 感谢：
  - @冬瓜-X1C5th
  - @OC-xlivans
  - @Air 13 IWL-GZ-Big Orange (OC perfect)
  - @子骏oc IWL
  - @大勇-小新air13-OC-划水小白
  - .....
- **Thanks for Acidanthera maintaining OpenCorePkg**

总述

ACPI更名和补丁

- 尽可能做到不用或者少用更名和补丁。比如： `HDAS rename HDEF` 、 `EC0 rename EC` 、 `SSDT-OC-XOSI` 等等。尤其对带下划线的 `MethodObj` （如 _STA、 _OSI等）的更名要 谨慎使用 。一般情况下：
  - 无需操作系统补丁。对于受系统限制而无法正常工作的部件，根据ACPI的具体情况定制补丁。对操作系统有特别要求的 谨慎使用 《操作系统补丁》。
  - 一些机器无需使用亮度快捷键更名和补丁。使用《PS2键盘映射》能够达到同等效果。
  - 就目前而言，绝大多数机器需要《0D6D补丁》来解决 `秒醒` 问题。
  - 对于电池部分，如果必须拆分数据的话，电池的更名和补丁必不可少。
  - 绝大多数Thinkpad机器需要《PTSWAK综合扩展补丁》来解决唤醒后呼吸灯未恢复的问题。
  - 有睡眠按键【小月亮】的机器，当按下这个按键后导致系统崩溃的可参考《PNP0C0E睡眠修正方法》。
- 一些问题的解决要求启用或者禁用某些设备。启用或者禁用设备的方法有：
  - 《二进制更名与预置变量》——二进制更名对于单系统非常有效。对于多系统，应评估更名对其他系统造成的影响。谨慎使用 。 预置变量可能对其他部件或者其他系统造成影响。谨慎使用 。
  - 《仿冒设备》——这种方法非常可靠。 推荐使用 。

重要的补丁

- ***SSDT-RTC0*** ——位于《仿冒设备》

  某些机器因RTC【PNP0B00】被禁用而导致启动阶段系统崩溃，使用 ***SSDT-RTC0*** 可解决这个问题。

- ***SSDT-EC*** ——位于《仿冒EC》

  对于 **10.15+** 系统[笔记本]，如果EC控制器名称不是 `EC` 需添加 ***SSDT-EC*** ，否则启动阶段系统崩溃。

ASL 语言基础

> 本教程转载自远景论坛，发布于 2011-11-21 11:16:20，作者为：suhetao。
>
> 由 Bat.bat(williambj1) 于 2020-2-14 重新排版为 Markdown，并对部分内容进行了添加及修正。
>
> http://bbs.pcbeta.com/forum.php?mod=viewthread&tid=944566&archive=2&extra=page%3D1&page=1

写在前面的话

鄙人不是一个主板 BIOS 开发工作者，以下对 ASL 的理解仅仅来源于 ~~http://www.acpi.info/~~ [注: 已失效, 文档已转移至 https://uefi.org] 上的 ACPI Specification 文档。因此难免会出现不少错误的理解，以及错误的观点，希望大家谅解以及纠正。

简述

首先，不得不说一下 DSDT (Differentiated System Description Table Fields) 和 SSDT (Secondary System Description Table Fields) 什么是 DSDT 和 SSDT 呢？其实它们都属于 ACPI 其中的一个表格，而 ACPI 是 Advanced Configuration & Power Interface 的缩写，高级配置和电源接口，从文字上就可以理解 ACPI 是一系列的接口，这个接口包含了很多表格，所以 DSDT 和 SSDT 既是其中的表格同时也是一些接口。所以不难想象 ACPI 主要的功能就是提供操作系统一些服务以及提供一些讯息给操作系统使用。DSDT 和 SSDT 自然也不例外。ACPI 的一个特色就是专有一门语言来编写 ACPI 的那些表格。它就是 ASL (ACPI Source Language) 也就是这盘文章的主角，ASL 经过编译器编译后，就变成了 AML (ACPI Machine Language)，然后由操作系统来执行。既然 ASL 是一门语言，那就有它的准则。

ASL 准则

1. 变量命名不超过 4 个字符，且不能以数字开头。联想一下看到过的 DSDT 代码看看，绝对不会超过。
2. Scope 形成作用域，概念类似于数学中的集合 `{}` 。有且仅有一个根作用域，所以 DSDT 都以

   ```
   DefinitionBlock ("xxxx", "DSDT", 0x02, "xxxx", "xxxx", xxxx)
   {
   ```

   开始，同时以

   ```
   }
   ```

   结束。这个就是根作用域.

   `xxxx` 参数依次表示输出 `文件名` 、 `OEMID` 、 `表ID` 、 `OEM版本` 。第三个参数根据第二个参数指定，如上面所示。如果是 **`DSDT`** 就一定是 `0x02` ，其他参数都可以自由修改。

3. 以 `_` 字符开头的函数和变量都是系统保留的，这就是为什么反编译某些AML以后得到的 ASL 出现 `_T_X` ，重新编译的时候会出现警告。

4. `Method` 定义函数，函数可以定义在 `Device` 下或者 `Scope` 下，但是不能脱离 `Scope` 定义单独的函数，所以不会有 以下这种情况出现。

```
Method (xxxx, 0, NotSerialized)
{
    ...
}
DefinitionBlock ("xxxx", "DSDT", 0x02, "xxxx", "xxxx", xxxx)
{
    ...
}
```

5. 根作用域 `\` 下有 `\_GPE` ， `\_PR` ， `\_SB` ， `\_SI` ， `\_TZ` 五个作用域。

   - `\_GPE` --- ACPI 事件处理
   - `\_PR` --- 处理器
   - `\_SB` --- 所有的设备和总线
   - `\_SI` --- 系统指示灯
   - `\_TZ` --- 热区，用于读取某些温度

   > 不同属性的东西放在对应的作用域下。例如:

   - 设备 `Device (PCI0)` 放在 `Scope (\_SB)` 里面

```
Scope (\_SB)
{
    Device (PCI0)
    {
        ...
    }
    ...
}
```

   - CPU 相关的信息放在 `Scope (_PR)`

     > 不同的 CPU 所在的域会不同，常见的有 `_PR` ， `_SB` ， `SCK0` 等

```
Scope (_PR)
{
    Processor (CPU0, 0x00, 0x00000410, 0x06)
    {
        ...
    }
    ...
}
```

- `Scope (_GPE)` 放着相关的事件处理

```
Scope (_GPE)
{
    Method (_L0D, 0, NotSerialized)
    {
        ...
    }
    ...
}
```

乍一看不是函数吗？当然函数也可以放在这里。但是请注意函数名 `_` 开头的是系统保留的函数。

6. `Device (xxxx)` 也可看做是一个作用域，里面包含对设备的各种描述，例如 `_ADR` 、 `_CID` 、 `_UID` 、 `_DSM` 、 `_STA` 等对设备的说明和状态控制
7. 符号 `\` 引用根作用域， `^` 引用上级作用域，同理 `^^` 为 `^` 的上级作用域
8. 符号 `_` 本身没有任何意义，在 ASL 中只用于补齐位置，如 `_OSI` 的 `_` 用于补齐 `Method` 名称所需的四个字符
9. ACPI 为了更好理解，规范后来使用了新的 ASL 语法 ASL 2.0 (也叫做 ASL+), 新语法引入了与 C 语言等同的 `+-*/=` , `<<` , `>>` 和逻辑判断 `==` , `!=` 等等
10. 函数最多可以传递 7 个参数，在函数里用 `Arg0` ~ `Arg6` 表示，不可以自定义。
11. 函数最多可以用 8 个局部变量，用 `Local0` ~ `Local7` ，不用定义，但是需要初始化才能使用，也就是一定要有一次赋值操作。

ASL 常用的数据类型

ASL	中文
`Integer`	整数
`String`	字符串
`Event`	事件
`Buffer`	数组
`Package`	对象集合

ASL 定义变量

- 定义整数

```
Name (TEST, 0)
```

- 定义字符串

```
Name (MSTR,"ASL")
```

- 定义 Package

```
Name (_PRW, Package (0x02)
{
    0x0D,
    0x03
})
```

- 定义 Buffer Field

> Buffer Field 一共有 6 种，分别是

创建语句	大小
CreateBitField	1-Bit
CreateByteField	8-Bit
CreateWordField	16-Bit
CreateDWordField	32-Bit
CreateQWordField	64-Bit
CreateField	任意大小

```
CreateBitField (AAAA, Zero, CCCC)
CreateByteField (DDDD, 0x01, EEEE)
CreateWordField (FFFF, 0x05, GGGG)
CreateDWordField (HHHH, 0x06, IIII)
CreateQWordField (JJJJ, 0x14, KKKK)
CreateField (LLLL, Local0, 0x38, MMMM)
```

可以发现定义变量的时候不需要显式声明其类型

ASL 赋值方法

赋值方法有且仅有一个

```
Store (a,b) /* 传统 ASL */
b = a       /*   ASL+  */
```

例子:

```
Store (0, Local0)
```

```
Local0 = 0

Store (Local0, Local1)
Local1 = Local0
```

ASL 运算函数

ASL+	传统 ASL	中文含义	举例
+	Add	整数相加	`Local0 = 1 + 2` `Add (1, 2, Local0)`
-	Subtract	整数减法	`Local0 = 2 - 1` `Subtract (2, 1, Local0)`
*	Multiply	整数相乘	`Local0 = 1 * 2` `Multiply (1, 2, Local0)`
/	Divide	整数除法	`Local0 = 10 / 9` `Divide (10, 9, Local1(余数), Local0(结果))`
%	Mod	整数求余	`Local0 = 10 % 9` `Mod (10, 9, Local0)`
<<	ShiftLeft	左移	`Local0 = 1 << 20` `ShiftLeft (1, 20, Local0)`
>>	ShiftRight	右移	`Local0 = 0x10000 >> 4` `ShiftRight (0x10000, 4, Local0)`
--	Decrement	整数自减 1	`Local0--` `Decrement (Local0)`
++	Increment	整数自增 1	`Local0++` `Increment (Local0)`
&	And	整数于	`Local0 = 0x11 & 0x22` `And (0x11, 0x22, Local0)`
|	Or	或	`Local0 = 0x01 | 0x02` `Or (0x01, 0x02, Local0)`
~	Not	取反	`Local0 = ~(0x00)` `Not (0x00,Local0)`
无	Nor	异或	`Nor (0x11, 0x22, Local0)`

等等，具体请查阅 ACPI Specification

ASL 逻辑运算

ASL+	传统 ASL	中文含义	举例
&&	LAnd	逻辑与	`If (BOL1 && BOL2)` `If (LAnd(BOL1, BOL2))`
!	LNot	逻辑反	`Local0 = !0` `Store (LNot(0), Local0)`
|	LOr	逻辑或	`Local0 = (0 | 1)` `Store (LOR(0, 1), Local0)`

<	LLess	逻辑小于	`Local0 = (1 < 2)` `Store (LLess(1, 2), Local0)`
<=	LLessEqual	逻辑小于等于	`Local0 = (1 <= 2)` `Store (LLessEqual(1, 2), Local0)`
>	LGreater	逻辑大于	`Local0 = (1 > 2)` `Store (LGreater(1, 2), Local0)`
>=	LGreaterEqual	逻辑大于等于	`Local0 = (1 >= 2)` `Store (LGreaterEqual(1, 2), Local0)`
==	LEqual	逻辑相等	`Local0 = (Local0 == Local1)` `If (LEqual(Local0, Local1))`
!=	LNotEqual	逻辑不等于	`Local0 = (0 != 1)` `Store (LNotEqual(0, 1), Local0)`

不难发现逻辑运算只会有两种结果 `0` 或者 `1`

ASL 函数的定义

1. 定义函数

```
Method (TEST)
{
    ...
}
```

2. 定义有两个输入参数的函数，以及使用局部变量 `Local0` ~ `Local7`

   参数个数如果不定义默认是 `0`

```
Method (MADD, 2)
{
    Local0 = Arg0
    Local1 = Arg1
    Local0 += Local1
}
```

   实现了两个参数的加法运算，因此传入的参数一定要隐式整形数。

3. 定义带返回值的函数

```
Method (MADD, 2)
{
    Local0 = Arg0
    Local1 = Arg1
    Local0 += Local1

    Return (Local0) /* 在此处返回 */
}
```

例子为自定义加法的实现。函数实现的是如同系统函数 Add 一样的相加

```
Local0 = 1 + 2            /* ASL+ */
Store (MADD (1, 2), Local0)  /* 传统 ASL */
```

4. 定义可序列化的函数

如果不定义 `Serialized` 或者 `NotSerialized` 默认为 `NotSerialized`

```
Method (MADD, 2, Serialized)
{
    Local0 = Arg0
    Local1 = Arg1
    Local0 += Local1
    Return (Local0)
}
```

这个有点类似于多线程同步的概念，也就是说，当函数声明为 `Serialized` ，内存中仅能存在一个实例。一般应用在函数中创建一个对象。应用举例说明：

```
Method (TEST, Serialized)
{
    Name (MSTR,"I will sucess")
}
```

如果这样子声明 `TEST` 这个函数，那么在两个地方同时调用这个函数

```
Device (Dev1)
{
     TEST ()
}
Device (Dev2)
{
     TEST ()
}
```

如果先执行 `Dev1` 的 `TEST` ，`Dev2` 中的 `TEST` 将等待 `Dev1` 中的 `TEST` 函数执行完毕再执行。如果声明为

```
Method (TEST, NotSerialized)
{
    Name (MSTR, "I will sucess")
}
```

那么将在其中一个 `Devx` 调用 `TEST` 的时候，另外一个调用试图创建相同的字符串 `MSTR` 就会失败。

ACPI 预定义函数

`_OSI` (Operating System Interfaces 操作系统接口)

通过调用 `_OSI` 函数，我们可以不费吹灰之力知道当前系统运行的是什么系统，轻而易举地区分 Windows 和 macOS，从而为某个系统创建单独的补丁。

`_OSI` 函数需要输入一个参数，这个参数必须是一个操作系统定义的用于识别的字符串

操作系统	字符串
macOS	`"Darwin"`
Linux (包括基于 Linux 内核的操作系统)	`"Linux"`
FreeBSD	`"FreeBSD"`
Windows	`"Windows 20XX"`

> 与其它操作系统不同的是，每个 Windows 版本都有不同的字符串，参阅：
>
> https://docs.microsoft.com/en-us/windows-hardware/drivers/acpi/winacpi-osi

当 `_OSI` 内的参数对应了当前操作系统时，`_OSI` 就会返回 `1`，`If` 条件语句判断成立

```
If (_OSI ("Darwin")) /* 判断当前的系统是不是 macOS */
```

`_STA` (Status 状态)

注意⚠: `_STA` 分为两种，请勿与 `PowerResource` 的 `_STA` 混淆！！！

`_STA` 方法的返回值最多包含 5 个 Bit，每个 Bit 的含义如下

Bit 位	含义
Bit [0]	设备是否存在
Bit [1]	设备是否被启用且可以解码其资源
Bit [2]	设备是否在 UI 中显示
Bit [3]	设备是否正常工作
Bit [4]	设备是否存在电池

看完上面的表可能就有人要问了，“我们平时见到的 `_STA` 都是 `0x0F`，`Zero`”，和这些 Bit 有啥关系？其实只要把 16进制 值转换成 2 进制自然就一目了然了。`0x0F` 的意思是前四个 Bit 都为 `1`，等于是完全启用，反之 `Zero` 就是完全禁用。

那么有时候遇到的 `0x0B`，`0x1F` 又是什么呢？`SSDT-PNLF` 里面的 `0x0B` 转成 2 进制 是 `1011`，也就是说设备处于一个半开的状态，不允许解码其中的资源。`0x1F` 只会在笔记本的电池设备中出现，多出的那一个 Bit 用于通知 电池控制设备 (Control Method Battery Device `PNP0C0A`) 设备存在电

池。

> 这里再啰嗦一句 `PowerResource` 的 `_STA`
>
> `PowerResource` 的 `_STA` 只有两个返回值 `One` 和 `Zero`，有兴趣可以查阅 ACPI 规范，作用不再赘述

`_CRS` (Current Resource Settings 当前资源设置)

`_CRS` 函数返回的是一个 `Buffer`，在触摸设备中会返回触摸设备所需的 `GPIO Pin`，`APIC Pin` 等等，可以控制设备的中断模式。

ASL 流程控制

和常见的高级编程语言一样，ASL 也有与之对应的控制流程语句。

- Switch
  - Case
  - Default
  - BreakPoint
- While
  - Break
  - Continue
- If
  - Else
  - ElseIf
- Stall

分支控制 `If` 和 `Switch`

`If`

例如下面的语句判断一下当前系统的接口是不是 `Darwin`，如果是把 `OSYS = 0x2710`

```
    If (_OSI ("Darwin"))
    {
        OSYS = 0x2710
    }
```

`ElseIf`，`Else`

下面的例子中如果系统结构不是 `Darwin`，另外如果系统不是 `Linux`，那么 `OSYS = 0x07D0`

```
    If (_OSI ("Darwin"))
    {
        OSYS = 0x2710
    }
```

```
    ElseIf (_OSI ("Linux"))
    {
        OSYS = 0x03E8
    }
    Else
    {
        OSYS = 0x07D0
    }
```

`Switch` , `Case` , `Default` , `BreakPoint`

```
    Switch (Arg2)
    {
        Case (1) /* 条件 1 */
        {
            If (Arg1 == 1)
            {
                Return (1)
            }
            BreakPoint /* 条件判断不符合，退出 */
        }
        Case (2) /* 条件 2 */
        {
            ...
            Return (2)
        }
        Default /* 如果都不符合，则 */
        {
            BreakPoint
        }
    }
```

`Switch` 可以看做是一系列 `If...Else` 的集合。 `BreakPoint` 相当于断点，意味着退出当前 `Swtich`

循环控制

`While` 以及暂停 `Stall`

```
Local0 = 10
While (Local0 >= 0x00)
{
    Local0--
    Stall (32)
}
```

`Local0` 等于 `10` ,如果 `Local0` 逻辑不等于 `0` 不为真， `Local0` 自减 `1` ,暂停 `32μs` ,所以这段代码延时 `10 * 32 = 320 μs` 。

`For`

ASL 中的 For 循环和 `C` ， `Java` 中的如出一辙

```
for (local0 = 0, local0 < 8, local0++)
{
    ...
}
```

上面的 `For` 循环和下面的 `While` 是等效的

```
Local0 = 0
While (Local0 < 8)
{
    Local0++
}
```

外部引用 `External`

引用类型	中文(仅供参考)	外部 SSDT 引用	被引用
UnknownObj	未知	`External (\_SB.EROR, UnknownObj`	(只有 iasl 无法判断类型时才会出现，请避免使用)
IntObj	整数	`External (TEST, IntObj`	`Name (TEST, 0)`
StrObj	字符串	`External (\_PR.MSTR, StrObj`	`Name (MSTR,"ASL")`
BuffObj	缓冲	`External (\_SB.PCI0.I2C0.TPD0.SBFB, BuffObj`	`Name (SBFB, ResourceTemplate () Name (BUF0, Buffer() {"abcde"})`
PkgObj	包	`External (_SB.PCI0.RP01._PRW, PkgObj`	`Name ( PRW, Package (0x02) { 0x0D, 0x03 })`
FieldUnitObj	字段单元	`External (OSYS, FieldUnitObj`	`OSYS, 16,`
DeviceObj	设备	`External (\_SB.PCI0.I2C1.ETPD, DeviceObj`	`Device (ETPD)`
EventObj	事件	`External (XXXX, EventObj`	`Event (XXXX)`
MethodObj	函数	`External (\_SB.PCI0.GPI0._STA, MethodObj`	`Method (_STA, 0, NotSerialized)`

MutexObj	互斥体	`External (_SB.PCI0.LPCB.EC0.BATM, MutexObj`	`Mutex (BATM, 0x07)`
OpRegionObj	操作区	`External (GNVS, OpRegionObj`	`OperationRegion (GNVS, SystemMemory, 0x7A4E7000, 0x0866)`
PowerResObj	电源资源	`External (\_SB.PCI0.XDCI, PowerResObj`	`PowerResource (USBC, 0, 0)`
ProcessorObj	处理器	`External (\_SB.PR00, ProcessorObj`	`Processor (PR00, 0x01, 0x00001810, 0x06)`
ThermalZoneObj	温控区	`External (\_TZ.THRM, ThermalZoneObj`	`ThermalZone (THRM)`
BuffFieldObj	缓冲区	`External (\_SB.PCI0._CRS.BBBB, BuffFieldObj`	`CreateField (AAAA, Zero, BBBB)`

> DDBHandleObj 几乎不可能遇到因此这里不讨论

在 SSDT 中外部引用时，使用 `External (` + `路径和名称` + `,` + `引用类型)` ，把光标定位到被引用 Object 里之后 MaciASL 会在左下角显示路径

ASL 存在性判断语句 CondRefOf

`CondRefOf` 是 ASL 中一个非常实用的函数，可以用来判断所有类型 Object 的存在与否

```
Method (SSCN, 0, NotSerialized)
{
    If (_OSI ("Darwin"))
    {
        ...
    }
    ElseIf (CondRefOf (\_SB.PCI0.I2C0.XSCN))
    {
        If (USTP)
        {
            Return (\_SB.PCI0.I2C0.XSCN ())
        }
    }

    Return (Zero)
}
```

上面的代码节选自 **`SSDT-I2CxConf`** ，在启动非 macOS 的系统时，如果 `I2C0` 下面存在 `XSCN` (原为 `SSCN` ，已被被重命名成 `XSCN` ，创建一个新的函数并使用原来的名称 `SSCN`)，则返回原来函数返回的值

结语

累了，今天就写到这里，有了这些基础，鄙人觉得现在大家对着 DSDT，SSDT 的时候就不会是发呆又发呆，至少可以明白些少语句了。

SSDT 补丁加载顺序

- 通常情况下，我们的 SSDT 补丁的对象是机器的 ACPI (DSDT 或者其他 SSDT 文件)，这些原始的 ACPI 加载的时间早于我们的 SSDT 补丁。因此，我们的 SSDT 补丁在 `Add` 列表中 没有顺序要求。
- 有这样一种情况，当我们在一个 SSDT 补丁里定义了一个 `Device` 设备，又在另一个 SSDT 补丁里通过 `Scope` 引用这个 `Device` ，那么，这两个补丁在 `Add` 列表中 有顺序要求。

示例

- 补丁1：**SSDT-XXXX-1.aml**

```
External (_SB.PCI0.LPCB, DeviceObj)
Scope (_SB.PCI0.LPCB)
{
    Device (XXXX)
    {
        Name (_HID, EisaId ("ABC1111"))
    }
}
```

- 补丁2：**SSDT-XXXX-2.aml**

```
External (_SB.PCI0.LPCB.XXXX, DeviceObj)
Scope (_SB.PCI0.LPCB.XXXX)
{
      Method (YYYY, 0, NotSerialized)
     {
         /* do nothing */
     }
  }
```

- `Add` 列表要求

```
Item 1
          path    <SSDT-XXXX-1.aml>
Item 2
          path    <SSDT-XXXX-2.aml>
```

```
//Fake XXXX
DefinitionBlock ("", "SSDT", 2, "ACDT", "XXXX-1", 0)
{
    External (_SB.PCI0.LPCB, DeviceObj)
    Scope (_SB.PCI0.LPCB)
    {
        Device (XXXX)
        {
            Name (_HID, EisaId ("ABC1111"))
        }
    }
}
```

```
DefinitionBlock ("", "SSDT", 2, "ACDT", "XXXX-2", 0)
{
    External (_SB.PCI0.LPCB.XXXX, DeviceObj)
    Scope (_SB.PCI0.LPCB.XXXX)
    {
        Method (YYYY, 0, NotSerialized)
        {
            //do nothing
        }
    }
}
```

ACPI 表单

简述

- ACPI（Advanced Configuration & Power Interface）是电脑高级配置和电源接口，它是由Intel，Microsoft等厂商共同制定的一种电源管理标准，即 `ACPI规范` 。每台电脑出厂时都会提供符合 `ACPI规范` 的一组二进制文件，这些文件我们称之为 ACPI 表单。ACPI 表单的数量和内容随机器不同而不同，亦随BIOS版本不同而 可能 不同。ACPI 表单包括：
  - 1个 DSDT.aml
  - 多个 SSDT-*.aml。如： `SSDT-SataAhci.aml` 、 `SSDT-CpuSsdt.aml` 、 `SSDT-CB-01.aml` 等等
  - 其他 aml。如： `APIC.aml` 、 `BGRT.aml` 、 `DMAR.aml` 、 `ECDT.aml` 等等
- 一些工具软件和引导器可以提取机器的 ACPI 表单，像 windows 下的 Aida64，引导器 clover 等。因为 ACPI 表单是二进制文件，所以我们需要一个反编译软件来帮助我们读懂文件内容，如 MAC 系统的 MaciASL 。当用反编译软件打开这些表单时，特别是打开 DSDT.aml 时，可能会出现很多错误。需要说明的是绝大多数情况下，这些错误是软件反编译过程产生的，机器提供的 ACPI 表单并不存在这些错误。
- ACPI 表单通过 aml 语言形式来描述机器的硬件信息，它本身没有任何驱动能力。然而，某个硬件的正常工作需要正确的 ACPI，错误的描述方法会导致引导失败或者系统崩溃。比如，机器安插了博通网卡，如果 ACPI 描述为 Intel 网卡，那么系统会加载 Intel 网卡驱动，这显然是错误的。再比如，机器没有提供 `自动调节亮度` 的硬件，即使 ACPI 添加了 `SSDT-ALS0` 也无法实现亮度的自动调节功能。
- 由于 windows 系统和 MAC 系统的工作原理不同，黑苹果需要修正 ACPI 。正确的 ACPI 是黑苹果稳定工作的基础。 强烈建议 使用 `热补丁` 【HOTpatch】对 ACPI 实施补丁。 `热补丁` 可以很好的规避所谓的 DSDT 错误。
- 有关 ACPI 的详细内容请查看 `ACPI规范` ；有关 aml 语言的介绍请参阅《ASL语法基础》。
- 本章 `ACPI 补丁` 仅适用于 `OpenCore`

ACPI 补丁

- DSDT 补丁和 SSDT 补丁

  这部分内容参考《OC-little》其他章节。

- 其他表单补丁

  - 清除 **ACPI** 的 **`header fields`**
    - 补丁方法： `ACPI\Quirks\NormalizeHeaders` = `true`
    - 说明：只有 Mac 10.13 才需要这个补丁
  - 重新定位 **ACPI** 内存区域
    - 补丁方法： `ACPI\Quirks\RebaseRegions` = `true`
    - 说明：ACPI 表单的内存区域既有动态分配的地址，也有固定分配的地址。补丁的作用是重新定位 **ACPI** 内存区域 ，这个操作非常危险，除非这个补丁可以解决引导崩溃问题，否则不要选用它。

- **FACP.aml**
  - 补丁方法：`ACPI\Quirks\FadtEnableReset` = `true`
  - 说明：`ACPI规范` 以 **FADT** 来定义与配置和电源管理相关的各种静态系统信息，在机器的 ACPI 表单中以 **FACP.aml** 表单出现。 **FACP.aml** 表单表征的信息有 RTC时钟，电源和睡眠按键，电源管理等。目前和黑苹果有关的有以下几个方面：
  - 重启和关机不正常的，尝试使用本补丁
  - 按下 电源键 无法呼出 “重新启动、睡眠、取消、关机” 菜单的，尝试使用本补丁
  - **FACP.aml** 表单的 `Low Power S0 Idle` 、 `Hardware Reduced` 表征了机器类型，决定了电源管理方式。如果 `Low Power S0 Idle` = `1` 则表明机器属于 `AOAC` 。有关 `AOAC` 方面的内容参见《关于AOAC》。
- **FACS.aml**
  - 补丁方法： `ACPI\Quirks\ResetHwSig` = `true`
  - 说明：**FACS.aml** 表单的 `Hardware Signature` 项是4字节的硬件签名，是系统引导后根据基本硬件配置计算得出的。如果机器从 休眠 状态 唤醒 后这个值发生了改变系统将无法正确恢复。补丁的作用是使 `Hardware Signature` = `0` 尝试解决上述问题。
  - 注意：如果系统已经禁用了 休眠 就无需理会该补丁
- **BGRT.aml**
  - 补丁方法： `ACPI\Quirks\ResetLogoStatus` = `true`
  - 说明：**BGRT.aml** 表单是引导图形资源表。根据 `ACPI规范` ，表单的 `Displayed` 项应该 = `0` 。但是部分厂商出于某个原因 `Displayed` 项写入了非零数据，这可能导致引导阶段屏幕刷新失败。补丁的作用是使 `Displayed` = `0` 。
  - 注意：并非所有机器都有这个表单
- **DMAR.aml**
  - 补丁方法： `Kernel\Quirks\DisableIoMapper` = `true`
  - 说明：补丁的作用同 BIOS 禁止 `VT-d` 或者 Drop **DMAR.aml**
  - 注意：只有早期 Mac 系统才需要这个补丁
- **ECDT.aml**
  - 补丁方法：全局更名使所有 ACPI 表单的 `EC` 名称、路径和 `Namepath` 一致
  - 说明：个别机器（如 **Lenovo yoga-s740**）的 **ECDT.aml** 表单的 `Namepath` 与其他 ACPI 表单的 `EC` 名称不一致，这会导致机器在引导过程中出现 ACPI 错误。本补丁方法可以较好的解决 ACPI 报错问题。
  - 注意：并非所有机器都有这个表单

AOAC 总述

AOAC 技术

- 新的笔记本电脑引入了一项新技术—— `AOAC` ，即：*Always On/Always Connected* 。 `AOAC` 由 `Intel` 公司提出，旨在电脑在睡眠或者休眠模式下仍然保持网络连接及资料传输。简单的说， `AOAC` 的引入使笔记本像我们的手机一样，永不关机，永远在线。

`AOAC` 机器的判断方法

- 用 MaciASL 打开 `ACPI` 的 `FACP.aml` ，搜索 `Low Power S0 Idle` ，如果=1即属于 `AOAC` 机器。如：

```
Low Power S0 Idle (V5) : 1
```

- 有关 `AOAC` 方面的内容请百度 `AOAC` 、 `联想AOAC` 、 `AOAC网卡` 等。

AOAC 问题

睡眠失败问题

- 由于 `AOAC` 和 `S3` 本身相矛盾，采用了 `AOAC` 技术的机器不具有 `S3` 睡眠功能，如 `Lenovo PRO13` 。这样的机器一旦进入 `S3` 睡眠就会 睡眠失败。 睡眠失败 主要表现为：睡眠后无法被唤醒，呈现死机状态，只能强制关机。睡眠失败 本质是机器一直停滞在睡眠过程，始终没有睡眠成功。
- 有关 `S3` 睡眠方面内容参见《ACPI 规范》。

待机时间问题

- 禁止 **`S3`** 睡眠 可以解决 睡眠失败 问题，但是机器将不再睡眠。没有了睡眠随之而来的问题是：在电池供电模式下，机器待机时间大大缩短。比如，在"菜单睡眠"、"自动睡眠"、"盒盖睡眠"等情况下，电池耗电量较大，大约每小时耗电5%--10%。

AOAC解决方案

- 禁止 `S3` 睡眠
- 关闭独显的供电电源
- 电源空闲管理
- 选择品质较好的 SSD：SLC>MLC>TLC>QLC（不确定）
- 可能的话更新 SSD 固件以提高电源管理的效能
- 使用 NVMeFix.kext 开启 SSD 的 APST

- 启用 ASPM（BIOS 高级选项启用ASPM、补丁启用 L1）

AOAC 睡眠、唤醒

- `AOAC` 睡眠

  以上方案可以使机器睡眠，这种睡眠叫 `AOAC` 睡眠。 `AOAC` 睡眠本质是系统、硬件进入了空闲状态，非传统意义上的 `S3` 睡眠。

- `AOAC` 唤醒

  机器进入 `AOAC` 睡眠后唤醒它会比较困难，通常需要电源键唤醒。某些机器可能需要电源键 + `PNP0C0D` 方法来唤醒机器。

AOAC 补丁

- 禁止 `S3` 睡眠——参见《禁止S3睡眠》
- 禁用独显补丁——参见《AOAC禁止独显》
- 电源空闲管理补丁——参见《电源空闲管理》
- AOAC唤醒补丁——参见《AOAC唤醒方法》
- 秒醒补丁——参见《060D补丁》
- 启用设备 LI ——参见《设置ASPM工作模式》，感谢 @iStar丶Forever 提供方法
- 管控蓝牙WIFI——参见《睡眠自动关闭蓝牙WIFI》，感谢 @i5 ex900 0.66%/h 华星 OC Dreamn 提供方法

注意事项

- `AOAC` 解决方案是临时解决方案。随着 `AOAC` 技术的广泛应用，可能将来会有更好的解决方法。
- `AOAC` 睡眠、唤醒和 `S3` 睡眠、唤醒不同，以下补丁不在适用
  - 《PTSWAK综合扩展补丁》
  - 《PNP0C0E睡眠修正方法》
- 同上原因， `AOAC` 睡眠期间无法正确显示工作状态，如睡眠期间无呼吸灯闪烁。
- 非 `AOAC` 机器也可尝试本方法。

禁止 `S3` 睡眠

描述

- 禁止 `S3` 睡眠 用来解决某些机器因某种原因导致的 睡眠失败 问题。睡眠失败 是指：机器睡眠后无法被正常唤醒，表现为死机，唤醒后重启、关机等。

补丁方法

- 更名：`_S3 to XS3`

```
Find    5F53335F
Replace 5853335F
```

- 补丁
  - ***SSDT-NameS3-disable*** ：适用于 `ACPI` 描述 `S3` 的方法为 `Name` 类型的，绝大多数机器属于这种情况。
  - ***SSDT-MethodS3-disable*** ：适用于 `ACPI` 描述 `S3` 的方法为 `Method` 类型的。

注意事项

- 根据机器原始 `ACPI` 描述 `S3` 的方法选用对应的补丁。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/Pro
pertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>ACPI</key>
    <dict>
        <key>Patch</key>
        <array>
            <dict>
                <key>Comment</key>
                <string>_S3 to XS3</string>
                <key>Count</key>
                <integer>0</integer>
                <key>Enabled</key>
                <true/>
                <key>Find</key>
                <data>
                X1MzXw==
                </data>
                <key>Limit</key>
                <integer>0</integer>
                <key>Mask</key>
                <data>
                </data>
                <key>OemTableId</key>
                <data>
                </data>
                <key>Replace</key>
                <data>
                WFMzXw==
                </data>
                <key>ReplaceMask</key>
                <data>
                </data>
                <key>Skip</key>
                <integer>0</integer>
                <key>TableLength</key>
                <integer>0</integer>
                <key>TableSignature</key>
                <data>
                </data>
            </dict>
        </array>
    </dict>
</dict>
</plist>
```

```
// AOAC
// Method (_S3, ......
// In config ACPI, _S3 to XS3
// Find:     5F53335F
// Replace:  5853335F
//
DefinitionBlock("", "SSDT", 2, "OCLT", "S3-Fix", 0)
{
    External (XS3, MethodObj)

    If (_OSI ("Darwin"))
    {
        //
    }
    Else
    {
        Method (_S3, 0, NotSerialized)
        {
            Return(XS3 ())
        }
    }
}
//EOF
```

```
// AOAC
// Name (_S3, ......
// In config ACPI, _S3 to XS3
// Find:     5F53335F
// Replace:  5853335F
//
DefinitionBlock("", "SSDT", 2, "OCLT", "S3-Fix", 0)
{
    External (XS3, IntObj)

    If (_OSI ("Darwin"))
    {
        //
    }
    Else
    {
        Method (_S3, 0, NotSerialized)
        {
            Return(XS3)
        }
    }
}
//EOF
```

AOAC禁止独显

描述

- 采用了 `AOAC` 技术的笔记本建议禁用独显，以延长机器待机时间。
- 本文的补丁方法仅适用于采用了 `AOAC` 技术的机器。

`SSDT` 补丁示例

- 补丁1: ***SSDT-NDGP_OFF-AOAC***
  - 查询独显的名称和路径，确认其存在 `_ON` 和 `_OFF` 方法
  - 参考示例，修改其名称和路径同查询结果一致
- 补丁2: ***SSDT-NDGP_PS3-AOAC***
  - 查询独显的名称和路径，确认其存在 `_PS0` 、 `_PS3` 和 `_DSM` 方法
  - 参考示例，修改其名称和路径同查询结果一致
- 注意
  - 查询独显名称和路径以及 `_ON` 、 `_OFF` 、 `_PS0` 、 `_PS3` 和 `_DSM` 时，应对全部 `ACPI` 文件进行搜索，它可能存在于 `DSDT` 文件中，也可能存在于 `ACPI` 的其他 `SSDT` 文件中。
  - 以上2个示例为小新PRO13定制补丁，二选一使用。独显的名称和路径是： `_SB.PCI0.RP13.PXSX` 。

注意

- 如果 ***SSDT-NDGP_OFF*** 和 ***SSDT-NDGP_PS3*** 均满足使用要求，优先使用 ***SSDT-NDGP_OFF***
- 详细的屏蔽独显方法参考《SSDT屏蔽独显方法》

```
// Disables DGPU---lenovo PRO13
//
DefinitionBlock("", "SSDT", 2, "OCLT", "NDGP", 0)
{
    External(_SB.PCI0.RP13.PXSX._OFF, MethodObj)//lenovo PRO13

    If (_OSI ("Darwin"))
    {
        Device(DGPU)
        {
            Name(_HID, "DGPU1000")
            Method (_INI, 0, NotSerialized)
            {
                If (CondRefOf (\_SB.PCI0.RP13.PXSX._OFF))//lenovo PRO13
                {
                    \_SB.PCI0.RP13.PXSX._OFF()
                }
            }
        }
    }
}
//EOF
```

```
// Disables DGPU---lenovo PRO13
//
DefinitionBlock("", "SSDT", 2, "OCLT", "NDGP", 0)
{
    External(_SB.PCI0.RP13.PXSX._PS3, MethodObj)//lenovo PRO13
    External(_SB.PCI0.RP13.PXSX._DSM, MethodObj)//lenovo PRO13

    If (_OSI ("Darwin"))
    {
        Device(DGPU)
        {
            Name(_HID, "DGPU1000")
            Method (_INI, 0, NotSerialized)
            {
                If (CondRefOf (\_SB.PCI0.RP13.PXSX._PS3))
                {
                    \_SB.PCI0.RP13.PXSX._DSM (Buffer ()
                    {
                        /* 0000 */ 0xF8, 0xD8, 0x86, 0xA4, 0xDA, 0x0B, 0x1B, 0x47,
                        /* 0008 */ 0xA7, 0x2B, 0x60, 0x42, 0xA6, 0xB5, 0xBE, 0xE0
                    }, 0x0100, 0x1A,Buffer ()
                    { 0x01, 0x00, 0x00, 0x03 })
                    \_SB.PCI0.RP13.PXSX._PS3()
                }
            }
        }
    }
}
//EOF
```

电源空闲管理

描述

- 本补丁开启 MAC 系统自身的电源空闲管理，延长电池工作模式下的待机时间。
- 参阅：https://pikeralpha.wordpress.com/2017/01/12/debugging-sleep-issues/。

SSDT补丁

- ***SSDT-DeepIdle*** ——电源空闲管理补丁

注意事项

- ***SSDT-DeepIdle*** 和 `S3` 睡眠可能有严重冲突，使用 ***SSDT-DeepIdle*** 应避免 `S3` 睡眠，参见《禁止S3睡眠》
- ***SSDT-DeepIdle*** 可能会导致机器唤醒困难，可以通过补丁解决这个问题，参见《AOAC唤醒补丁》

备注

- ***SSDT-DeepIdle*** 主要内容来自 @Pike R.Alpha

```
// AOAC
// IORegistryExplorer
// IOService:/AppleACPIPlatformExpert/IOPMrootDomain:
// IOPMDeepIdleSupported = true
// IOPMSystemSleepType = 7
// PMStatusCode = ?
//
DefinitionBlock("", "SSDT", 2, "OCLT", "IDLE", 0)
{
    Scope (_SB)
    {
        Method (LPS0, 0, NotSerialized)
        {
            If (_OSI ("Darwin"))
            {
                Return (One)
            }
        }
    }

    Scope (_GPE)
    {
        Method (LXEN, 0, NotSerialized)
        {
            If (_OSI ("Darwin"))
            {
                Return (One)
            }
        }
    }
}
```

AOAC唤醒方法

描述

- ***SSDT-DeepIdle*** 补丁可以使机器进入深度空闲状态，延长机器待机时间。但同时也会导致唤醒机器比较困难，需要采取特殊方法来唤醒机器。有关***SSDT-DeepIdle*** 方面的内容参见《电源空闲管理》。
- 本方法 通过定制补丁协助 电源按键 自动点亮屏幕。

唤醒方法：电源按键

唤醒原理简述

- 一般情况下， 电源按键 能够唤醒机器。但有些时候，机器被唤醒后的状态并不完整，表现为：除了屏幕不亮，其他部件都能正常工作。这表明机器已经进入了 `S0` 状态。这种情况下，只要在 `LPCB` 设备下添加 `_PS0` 方法，并且在 `_PS0` 方法里添加符合 **PNP0C0D唤醒条件** 的内容，即可自动点亮屏幕。
- 建议同时使用 `_PS0` 和 `_PS3` 方法。

补丁示例

- ***SSDT-PCI0.LPCB-Wake-AOAC***

```
...
Scope (_SB.PCI0.LPCB)
{
    If (_OSI ("Darwin"))
    {
        Method (_PS0, 0, Serialized)
        {
            \_SB.PCI0.LPCB.H_EC._Q0D() /* 小新PRO13开盖方法 */
            /*
             * 定制内容：
             * 依据《附件》的《PNP0C0D唤醒条件》，制作适合于自己的开盖补丁
             * 或者使用通用开盖补丁
             */
            \_SB.PCI0.LPCB.H_EC._Q0A() /* 更新电源数据 */
        }
        Method (_PS3, 0, Serialized)
        {
        }
    }
}
...
```

说明：

1. `_SB.PCI0.LPCB.H_EC._Q0D` 为 小新PRO 开盖方法。如果使用通用开盖补丁，将 `_PS0` 部分替换为以下内容，并同时使用 ***SSDT-LIDpatch-AOAC*** ，见后文。

```
...
Method (_PS0, 0, Serialized)
{
    \_SB.PCI0.LPCB.H_EC.LID0.AOAC = 1 /* 满足 PNP0C0D唤醒条件 之一 */
    Notify (\_SB.PCI0.LPCB.H_EC.LID0, 0x80) /* 满足 PNP0C0D唤醒条件 之二 */
    Sleep (200) /* 延时200 */
    \_SB.PCI0.LPCB.H_EC.LID0.AOAC = 0 /* 恢复原始状态 */
    \_SB.PCI0.LPCB.H_EC._Q0A() /* 更新电源数据 */
}
...
```

2. `_SB.PCI0.LPCB.H_EC._Q0A` 为 小新PRO 更新电源数据方法。更新电源数据 详细内容见后文。

- ***SSDT-LIDpatch-AOAC*** ——通用开盖补丁

  当无法或者很难找到适合于自己的定制方法时，使用通用开盖补丁。

  注意**1**：通用开盖补丁应和 ***SSDT-PCI0.LPCB-Wake-AOAC*** 一同使用。

  注意**2**：补丁里的设备名称、路径应与 ACPI 原始名称、路径一致。补丁要求的更名在文件的注释里。

- 更新电源数据

  在 `AOAC` 睡眠期间，如果改变电源状态（比如，在睡眠期间拔掉或者插入电源）有可能导致唤醒后电源图标无法更新。出现此问题可以采用下列方法：

  - 查找电源设备（ `_HID = ACPI0003` ）名称和路径，按电源名称搜索、记录包涵 `Notify (***电源名称, 0x80)` 的 `Method` 。将这个 `Method` 添加到 ***SSDT-PCI0.LPCB-Wake-AOAC*** 中，参考补丁示例。
  - `Notify (***电源名称, 0x80)` 可能存在于多个 `Method` 之中，这必须通过《ACPIDebug》方法加以确认。确认方法：对插、拔电源有响应的 `Method` 就是我们需要的。

附件

PNP0C0D唤醒条件:

- `_LID` 返回 `One` 。 `_LID` 是 `PNP0C0D` 设备当前状态
- 执行 `Notify(***.LID0, 0x80)` 。 `LID0` 是 `PNP0C0D` 设备名称

```
// AOAC wake
// In config ACPI, _LID to XLID
// Find:     5F4C4944 00
// Replace:  584C4944 00
//
DefinitionBlock("", "SSDT", 2, "OCLT", "LID-AOAC", 0)
{
    //path:_SB.PCI0.LPCB.H_EC.LID0._LID
    External(_SB.PCI0.LPCB.H_EC.LID0, DeviceObj)
    External(_SB.PCI0.LPCB.H_EC.LID0.XLID, MethodObj)

    Scope (_SB.PCI0.LPCB.H_EC.LID0)
    {
        Name (AOAC, Zero)
        Method (_LID, 0, NotSerialized)
        {
            If ((_OSI ("Darwin") && (AOAC == One)))
            {
                Return (One)
            }
            Else
            {
                Return (\_SB.PCI0.LPCB.H_EC.LID0.XLID())
            }
        }
    }
}
//EOF
```

```
// PCI0.LPCB-AOACWake
//
DefinitionBlock("", "SSDT", 2, "ACDT", "AOACWake", 0)
{
    External(_SB.PCI0.LPCB, DeviceObj)
    External(_SB.PCI0.LPCB.H_EC._Q0D, MethodObj)
    External(_SB.PCI0.LPCB.H_EC._Q0A, MethodObj)

    Scope (_SB.PCI0.LPCB)
    {
        If (_OSI ("Darwin"))
        {
            Method (_PS0, 0, Serialized)
            {
                \_SB.PCI0.LPCB.H_EC._Q0D()
                //
                \_SB.PCI0.LPCB.H_EC._Q0A()
            }

            Method (_PS3, 0, Serialized)
            {

            }
        }
    }
}
```

PCI设备ASPM

描述

- ASPM 即 活动状态电源管理（Active State Power Management），它是系统层面支持的一种电源链接管理方案。在 ASPM 管理下，当 PCI 设备空闲时尝试进入节电模式。
- ASPM 几种工作模式
  - L0—正常模式。
  - L0s—待机模式。L0s模式能够快速进入或退出空闲状态，进入空闲状态后，设备置于较低的功耗。
  - L1—低功耗待机模式。L1相比L0s会进一步降低功耗。但进入或退出空闲状态的时间比L0s更长。
  - L2—辅助电源模式。略。
- 对于采用了 `AOAC` 技术的机器，尝试改变 `无线网卡` 、 `SSD` 的 ASPM 模式降低机器功耗。

设置ASPM工作模式

`Properties` 注入方法【 优先使用 】

- 分别于 PCI 父设备 以及它的 子设备 注入 `pci-aspm-default`
  - 父设备
    - L0s/L1模式： `pci-aspm-default` = `03000000` 【data】
    - L1模式： `pci-aspm-default` = `02000000` 【data】
    - 禁止ASPM： `pci-aspm-default` = `00000000` 【data】
  - 子设备
    - L0s/L1模式： `pci-aspm-default` = `03010000` 【data】
    - L1模式： `pci-aspm-default` = `02010000` 【data】
    - 禁止ASPM： `pci-aspm-default` = `00000000` 【data】
- 示例

  小新PRO13的无线网卡ASPM默认是L0s/L1，设备路径是： `PciRoot(0x0)/Pci(0x1C,0x0)/Pci(0x0,0x0)` ，参照上述方法，通过注入 `pci-aspm-default` 改变ASPM为L1。如下：

```
PciRoot(0x0)/Pci(0x1C,0x0)
pci-aspm-default = 02000000
......
PciRoot(0x0)/Pci(0x1C,0x0)/Pci(0x0,0x0)
pci-aspm-default = 02010000
```

`SSDT` 补丁方法

- SSDT 补丁也可以设置 ASPM 工作模式。如：将某个设备 ASPM 设置为 L1模式，详见示例。
- 补丁原理同《禁止PCI设备》，请参阅。
- 示例：***SSDT-PCI0.RPXX-ASPM***

```
External (_SB.PCI0.RP05, DeviceObj)
Scope (_SB.PCI0.RP05)
{
    OperationRegion (LLLL, PCI_Config, 0x50, 0x01)
    Field (LLLL, AnyAcc, NoLock, Preserve)
    {
        L1,   1
    }
}

Scope (\)
{
    If (_OSI ("Darwin"))
    {
        \_SB.PCI0.RP05.L1 = Zero  /* Set ASPM = L1 */
    }
}
```

注**1**：小新 PRO13 无线网卡路径是 `_SB.PCI0.RP05` 。

注**2**： `\_SB.PCI0.RP05.L1 = 1` 时，ASPM = L0s/L1； `\_SB.PCI0.RP05.L1 = 0` 时，ASPM = L1。

注意事项

- ***Hackintool.app*** 工具可以查看设备 ASPM 工作模式。
- 改变 ASPM 后，如果发生异常情况请恢复 ASPM。

```
// Set ASPM of PCI0.RP05(WiFi) to L1
DefinitionBlock ("", "SSDT", 2, "OCLT", "ASPM", 0)
{
    External (_SB.PCI0.RP05, DeviceObj)
    Scope (_SB.PCI0.RP05)
    {
        OperationRegion (LLLL, PCI_Config, 0x50, 1)
        Field (LLLL, AnyAcc, NoLock, Preserve)
        {
            L1,   1
        }
    }

    Scope (\)
    {
        If (_OSI ("Darwin"))
        {
            \_SB.PCI0.RP05.L1 = Zero   //Set ASPM = L1
        }
    }
}
//EOF
```

简介

这是MAC下一个睡眠自动关闭蓝牙WIFI，睡醒自动开启蓝牙WIFI的脚本

只需执行一次该脚本即可实现睡眠关闭蓝牙WIFI功能

安装方法

1. 如果已经装好brew并能够正常更新软件，打开终端，将install.sh拖入终端 回车执行安装（推荐）
2. 如果未安装brew或安装失败，打开终端，将install-without-brew.sh拖入终端 回车执行安装

更新

V1.5

1.修复唤醒后WIFI无法打开的问题

2.增加卸载脚本uninstall.sh

v1.4

1. 修复蓝牙无法关闭的问题

V1.3

1. 增加无brew安装方法，未测试，自行测试

v1.2

1. 修改brew安装检测（不再自动安装，要求自己手动安装）
2. 需改覆盖为追加，清理垃圾文件

v1.1

1. 修改bluetil 来源

仿冒设备

综述

在众多 `SSDT` 补丁当中，相当数量的补丁可以归纳为仿冒设备补丁，如：

- 某些设备在ACPI中不存在，可是MAC系统需要它们。通过补丁对这些设备正确描述能够加载设备驱动。如《05-2-PNLF注入方法》、《添加缺失的部件》、《仿冒以太网》等。
- EC问题。如《仿冒EC》。
- 对于某些特殊的设备，使用禁止原始设备再仿冒它的方法，会给我们调整补丁的工作带来方便。如《OCI2C-TPXX补丁方法》。
- 某个原因造成某个设备被禁用，可是MAC系统需要它才能工作。见 `本章` 示例。
- 多数情况下，使用《二进制更名与预置变量》也可以启用设备。

仿冒设备

- 特点
  - 被仿冒的设备在ACPI中已存在，相对代码短、少、独立存在。
  - 原始设备具有规范的 `_HID` 或者 `_CID` 。
  - 即使原始设备未被禁用，使用仿冒设备的补丁也不会对ACPI造成伤害。
- 要求
  - 仿冒设备名称和ACPI的原始设备名称不同。
  - 补丁内容和原始设备主要内容相同。
  - 仿冒补丁的 `_STA` 部分应包括以下内容，确保windows系统使用原始的ACPI。

```
Method (_STA, 0, NotSerialized)
{
    If (_OSI ("Darwin"))
    {
        ...
        Return (0x0F)
    }
    Else
    {
        Return (Zero)
    }
}
```

- 示例
  - ***SSDT-RTC0*** — 仿冒RTC

- 原始设备名称：RTC
- _HID：PNP0B00

注意 ： `LPCB` 名称应和原始ACPI名称一致。

SSDT-EC 仿冒补丁

描述

OpenCore 要求不要改变 EC 控制器名称，但是为了加载 USB 电源管理，可能需要仿冒另一个 EC。

使用说明

DSDT中搜索 `PNP0C09` ，查看其所属设备名称。如果名称不是 `EC` ，使用本补丁；如果是 `EC` ，忽略本补丁。

注意

- 如果搜索到多个 `PNP0C09` ，应确认真实有效的 `PNP0C09` 设备。
- 补丁里使用了 `LPCB` ，非 `LPCB` 的请自行修改补丁内容。

```
/*
 * AppleUsbPower compatibility table for legacy hardware.
 *
 * Be warned that power supply values can be different
 * for different systems. Depending on the configuration
 * the values must be present in injected IOKitPersonalities
 * for com.apple.driver.AppleUSBMergeNub. iPad remains being
 * the most reliable device for testing USB port charging.
 *
 * Try NOT to rename EC0, H_EC, etc. to EC.
 * These devices are incompatible with macOS and may break
 * at any time. AppleACPIEC kext must NOT load.
 * See the disable code below.
 *
 * Reference USB: https://applelife.ru/posts/550233
 * Reference EC: https://applelife.ru/posts/807985
 */
DefinitionBlock ("", "SSDT", 2, "ACDT", "SsdtEC", 0x00001000)
{
    External (_SB.PCI0.LPCB, DeviceObj)

    /*
     * Uncomment replacing EC0 with your own value in case your
     * motherboard has an existing embedded controller of PNP0C09 type.
     *
     * While renaming EC0 to EC might potentially work initially,
     * it connects an incompatible driver (AppleACPIEC) to your hardware.
     * This can make your system unbootable at any time or hide bugs that
     * could trigger randomly.
     */

    /**
    External (_SB_.PCI0.LPCB.EC0, DeviceObj)

    Scope (\_SB.PCI0.LPCB.EC0)
    {
        Method (_STA, 0, NotSerialized)  // _STA: Status
        {
            If (_OSI ("Darwin"))
            {
                Return (0)
            }
            Else
            {
                Return (0x0F)
            }
        }
    }
    **/
```

```
    Scope (_SB.PCI0.LPCB)
    {
        Device (EC)
        {
            Name (_HID, "ACID0001")
            Method (_STA, 0, NotSerialized)
            {
                If (_OSI ("Darwin"))
                {
                    Return (0x0F)
                }
                Else
                {
                    Return (Zero)
                }
            }
        }
    }
}
```

仿冒RTC

综述

对于某些 300 系主板，`RTC` 设备默认被禁用的且无法通过和 `AWAC` 共用的 `STAS` 变量控制 `_STA` 的返回值来启用传统 RTC 设备，导致 ***`SSDT-AWAC`*** 无法生效，因此为了强制启用 `RTC` 设备，我们需要仿冒一个 `RTC0` 。

使用方法

> 案例

```
Device (RTC)
{
  Name (_HID, EisaId ("PNP0B00"))
  Name (_CRS, ResourceTemplate ()
  {
      IO (Decode16,
          0x0070,
          0x0070,
          0x01,
          0x08,
         )
      IRQNoFlags ()
          {8}
  })
  Method (_STA, 0, NotSerialized)
  {
    Return (0);
  }
}
```

> 上面就是 `RTC` 设备被禁用的情况，仿冒方法如下：

```
DefinitionBlock ("", "SSDT", 2, "ACDT", "RTC0", 0)
{
    External (_SB_.PCI0.LPCB, DeviceObj)

    Scope (_SB.PCI0.LPCB)
    {
        Device (RTC0)
        {
            Name (_HID, EisaId ("PNP0B00"))
            Name (_CRS, ResourceTemplate ()
            {
                IO (Decode16,
```

```
                    0x0070,
                    0x0070,
                    0x01,
                    0x08,
                    )
                IRQNoFlags ()
                    {8}
            })
            Method (_STA, 0, NotSerialized)
            {
                If (_OSI ("Darwin"))
                {
                    Return (0x0F)
                }
                Else
                {
                    Return (0);
                }
            }
        }
    }
}
```

注意

- 此部件只对 300 系主板有效。
- 此部件只在没有使用 ***SSDT-AWAC*** 且原始 `ACPI` 中 `RTC` 设备的 `_STA` 方法返回值是 `0` 时使用。
- 示例补丁的设备路径是 `LPCB` ，请结合实际情况修改。

```
//Fake RTC0
DefinitionBlock ("", "SSDT", 2, "ACDT", "RTC0", 0)
{
    External (_SB.PCI0.LPCB, DeviceObj)

    Scope (_SB.PCI0.LPCB)
    {
        Device (RTC0)
        {
            Name (_HID, EisaId ("PNP0B00"))
            Name (_CRS, ResourceTemplate ()
            {
                IO (Decode16,
                    0x0070,
                    0x0070,
                    0x01,
                    0x08,
                    )
                IRQNoFlags ()
                    {8}
            })
            Method (_STA, 0, NotSerialized)
            {
                If (_OSI ("Darwin"))
                {
                    Return (0x0F)
                }
                Else
                {
                    Return (0)
                }
            }
        }
    }
}
```

仿冒环境光传感器

综述

从 `macOS Catalina` 开始，笔记本设备需要仿冒环境光传感器 `ALS0` 来实现亮度保存和自动亮度调节，需要说明的是只有真正拥有环境光传感器硬件的机器才能实现真正意义上的自动亮度调节。

使用方法

分为两种情况，原始 `ACPI` 存在环境光传感器设备接口和不存在环境光传感器设备接口。首先在原始 `ACPI` 里面查找 `ACPI0008` ，如果可以查到相关设备，一般是 `ALSD` ，则说明存在环境光传感器设备接口，否则即为不存在环境光传感器设备接口。

存在环境光传感器设备接口

> 案例

```
Device (ALSD)
{
  Name (_HID, "ACPI0008" /* Ambient Light Sensor Device */)  // _HID: Hardware ID
  Method (_STA, 0, NotSerialized)  // _STA: Status
  {
    If ((ALSE == 0x02))
    {
      Return (0x0B)
    }

    Return (Zero)
  }

  Method (_ALI, 0, NotSerialized)  // _ALI: Ambient Light Illuminance
  {
    Return (((LHIH << 0x08) | LLOW))
  }

  Name (_ALR, Package (0x05)  // _ALR: Ambient Light Response
  {
    Package (0x02)
    {
      0x46,
      Zero
    },

    Package (0x02)
    {
      0x49,
```

```
      0x0A
    },

    Package (0x02)
    {
      0x55,
      0x50
    },

    Package (0x02)
    {
      0x64,
      0x012C
    },

    Package (0x02)
    {
      0x96,
      0x03E8
    }
  })
}
```

这种情况可以利用预置变量另其 `_STA` 方法返回 `0x0B` 以启用原始 `ACPI` 存在的环境传感器设备，方法如下：

```
DefinitionBlock ("", "SSDT", 2, "OCLT", "ALSD", 0)
{
    External (ALSE, IntObj)

    Scope (_SB)
    {
        Method (_INI, 0, NotSerialized)
        {
            If (_OSI ("Darwin"))
            {
                ALSE = 2
            }
        }
    }
}
```

不存在环境光传感器设备接口

对于这种情况我们只需要仿冒一个 `ALS0` 设备即可，方法如下：

```
DefinitionBlock ("", "SSDT", 2, "ACDT", "ALS0", 0)
{
```

```
    Scope (_SB)
    {
        Device (ALS0)
        {
            Name (_HID, "ACPI0008")
            Name (_CID, "smc-als")
            Name (_ALI, 0x012C)
            Name (_ALR, Package (0x01)
            {
                Package (0x02)
                {
                    0x64,
                    0x012C
                }
            })
            Method (_STA, 0, NotSerialized)
            {
                If (_OSI ("Darwin"))
                {
                    Return (0x0F)
                }
                Else
                {
                    Return (Zero)
                }
            }
        }
    }
}
```

注意

- 任何情况下仿冒设备都是安全且有效的，即使原始 `ACPI` 中存在环境光传感器设备接口，也可以直接仿冒 `ALS0` 。
- 被修正的 `变量` 可能存在于多个地方，对它修正后，在达到我们预期效果的同时，有可能影响到到其他部件。
- 原始 `ACPI` 中存在环境光传感器设备时，其名称可能不是 `ALSD` ，比如 `ALS0` ，不过至今没发现有其它名称。
- 原始 `ACPI` 中存在环境光传感器设备时，如果要用预置变量方法强制启用时，需要注意原始 `ACPI` 中是否存在 `_SB.INI` ，如果存在请使用仿冒 `ALS0` 方法。

```
/*
 * Starting with macOS 10.15 Ambient Light Sensor presence is required for backligh
t functioning.
 * Here we create an Ambient Light Sensor ACPI Device, which can be used by SMCLigh
tSensor kext
 * to report either dummy (when no device is present) or valid values through SMC i
nterface.
 */
DefinitionBlock ("", "SSDT", 2, "ACDT", "ALS0", 0x00000000)
{
    Scope (_SB)
    {
        Device (ALS0)
        {
            Name (_HID, "ACPI0008" /* Ambient Light Sensor Device */)  // _HID: Har
dware ID
            Name (_CID, "smc-als")  // _CID: Compatible ID
            Name (_ALI, 0x012C)  // _ALI: Ambient Light Illuminance
            Name (_ALR, Package (0x01)  // _ALR: Ambient Light Response
            {
                Package (0x02)
                {
                    0x64,
                    0x012C
                }
            })
            Method (_STA, 0, NotSerialized)  // _STA: Status
            {
                If (_OSI ("Darwin"))
                {
                    Return (0x0F)
                }
                Else
                {
                    Return (Zero)
                }
            }
        }
    }
}
```

```
DefinitionBlock ("", "SSDT", 2, "OCLT", "ALSD", 0)
{
    External (ALSE, IntObj)

    Scope (_SB)
    {
        Method (_INI, 0, NotSerialized)
        {
            If (_OSI ("Darwin"))
            {
                ALSE = 2
            }
        }
    }
}
```

二进制更名

描述

本文所描述的方法不是通常意义的对 `Device` 或者 `Method` 的更名，是通过二进制更名启用或者禁用某设备。

风险

当 OC 引导其他系统时，ACPI 二进制更名有可能对其他系统造成影响。

示例

以启用 `HPET` 为例。我们希望它的 `_STA` 返回 `0x0F` 。

二进制更名：

Find： `00 A0 08 48 50` 「注： `00` = `{` ; `A0` = `If` ……」

Replace： `00 A4 0A 0F A3` 「注： `00` = `{` ; `A4 0A 0F` = `Return(0x0F)` ; `A3` = `Noop` ，用于补齐字节数量」

- 原始代码

```
Method (_STA, 0, NotSerialized)
{
    If (HPTE)
    {
        Return (0x0F)
    }
    Return (Zero)
}
```

- 更名后代码

```
Method (_STA, 0, NotSerialized)
{
      Return (0x0F)
      Noop
      TE** ()
      Return (Zero)
}
```

解释：更名后出现了明显错误，但是这种错误不会产生危害。首先， `Return (0x0F)` 后面内容不会被执行。其次，错误位于 `{}` 内不会对其他内容产生影响。

实际情况，我们应尽可能保证更名后语法的完整性。下面是完整的 `Find`，`Replace` 数据。

Find： `00 A0 08 48 50 54 45 A4 0A 0F A4 00`

Replace： `00 A4 0A 0F A3 A3 A3 A3 A3 A3 A3 A3`

完整 `Replace` 后代码

```
Method (_STA, 0, NotSerialized)
{
    Return (0x0F)
    Noop
    Noop
    Noop
    Noop
    Noop
    Noop
    Noop
    Noop
}
```

要求

- ***ACPI*** 原始文件

  `Find` 二进制文件必须是 ***ACPI*** 原始文件，不可以被任何软件修改、保存过，也就是说它必须是机器提供的原始二进制文件。

- `Find` 唯一性、正确性

  `Find` 数量只有一个，除非 我们本意就是对多处实施 `Find` 和 `Replace` 相同操作。

  特别注意：任何重写一段代码而从中查找确认的二进制数据，非常不可信！

- `Replace` 字节数量

  `Find`，`Replace` 字节数量要求相等。比如 `Find` 10个字节，那么 `Replace` 也是 10 个字节。如果 `Replace` 少于 10 个字节就用 `A3`（空操作）补齐。

`Find` 数据查找方法

通常，用二进制软件（如 `010 Editor` ）和 `MaciASL.app` 打开同一个 `ACPI` 文件，以二进制数据方式和文本方式 `Find` 相关内容，观察上下文，相信很快能够确定 `Find` 数据。

`Replace` 内容

《要求》中说明 `Find` 时，【任何重写一段代码而从中查找确认的二进制数据，非常不可信！】，然而 `Replace` 可以这样操作。按照上面的示例，我们写一段代码：

```
DefinitionBlock ("", "SSDT", 2, "hack", "111", 0)
{
    Method (_STA, 0, NotSerialized)
    {
        Return (0x0F)
    }
}
```

编译后用二进制软件打开，发现：`XX ... 5F 53 54 41 00 A4 0A 0F` ，其中 `A4 0A 0F` 就是 `Return (0x0F)` 。

注： `Replace` 内容要遵循 ACPI 规范和 ASL 语言要求。

注意事项

- 更新BIOS有可能造成更名失效。 `Find` & `Replace` 字节数越多失效的可能性也越大。

附：TP-W530 禁止 BAT1

Find： `00 A0 4F 04 5C 48 38 44 52`

Replace： `00 A4 00 A3 A3 A3 A3 A3 A3`

- 原始代码

```
Method (_STA, 0, NotSerialized)
{
      If (\H8DR)
      {
          If (HB1A)
          {
          ...
}
```

- 更名后代码

```
Method (_STA, 0, NotSerialized)
{
      Return (Zero)
      Noop
      Noop
      Noop
      Noop
      Noop
      Noop
      If (HB1A)
      ...
}
```

预置变量法

描述

- 所谓 预置变量法 就是对ACPI的一些变量（ `Name` 类型和 `FieldUnitObj` 类型）预先赋值，达到初始化的目的。【虽然这些变量在定义时已经赋值，但在 `Method` 调用它们之前没有改变。】
- 通过第三方补丁文件在 `Scope (\)` 内对这些变量进行修正可以达到我们预期的补丁效果。

风险

- 被修正的 `变量` 可能存在于多个地方，对它修正后，在达到我们预期效果的同时，有可能影响到到其他部件。
- 被修正的 `变量` 可能来自硬件信息，只能读取不能写入。这种情况下需要 二进制更名 和 **SSDT**补丁 共同完成。应当注意，当OC引导其他系统时，有可能无法恢复被更名的 `变量` 。见 示例**4** 。

示例**1**

某设备 _STA 原文：

```
Method (_STA, 0, NotSerialized)
{
    ECTP (Zero)
    If ((SDS1 == 0x07))
    {
        Return (0x0F)
    }
    Return (Zero)
}
```

因某个原因我们需要禁用这个设备，为了达成目的 `_STA` 应返回 `Zero` 。从原文可以看出只要 `SDS1` 不等于 `0x07` 即可。采用 预置变量法 如下：

```
Scope (\)
{
    External (SDS1, FieldUnitObj)
    If (_OSI ("Darwin"))
    {
        SDS1 = 0
    }
}
```

示例**2**

官方补丁 ***SSDT-AWAC*** 用于某些 300+ 系机器强制启用 RTC 并同时禁用 AWAC。

注：启用 RTC 也可以使用 ***SSDT-RTC0*** ，见《仿冒设备》。

原文：

```
Device (RTC)
{
    ...
    Method (_STA, 0, NotSerialized)
    {
            If ((STAS == One))
            {
                Return (0x0F)
            }
            Else
            {
                Return (Zero)
            }
    }
    ...
}
Device (AWAC)
{
    ...
    Method (_STA, 0, NotSerialized)
    {
            If ((STAS == Zero))
            {
                Return (0x0F)
            }
            Else
            {
                Return (Zero)
            }
    }
    ...
}
```

从原文可以看出，只要 `STAS = 1` ，就可以启用 RTC 并同时禁用 `AWAC` 。采用 预置变量法 如下：

- 官方补丁 ***SSDT-AWAC***

```
External (STAS, IntObj)
Scope (_SB)
{
    Method (_INI, 0, NotSerialized)  /* _INI: Initialize */
    {
        If (_OSI ("Darwin"))
        {
```

```
                STAS = One
            }
        }
    }
```

注：官方补丁引入了路径 `_SB._INI` ，使用时应确认 DSDT 以及其他补丁不存在 `_SB._INI` 。

- 改进后补丁 ***SSDT-RTC_Y-AWAC_N***

```
External (STAS, IntObj)
Scope (\)
{
    If (_OSI ("Darwin"))
    {
        STAS = One
    }
}
```

示例3

使用 I2C 补丁时，可能需要启用 `GPIO` 。参见《OCI2C-GPIO补丁》的 ***SSDT-OCGPI0-GPEN***。

某原文：

```
Method (_STA, 0, NotSerialized)
{
    If ((GPEN == Zero))
    {
        Return (Zero)
    }
    Return (0x0F)
}
```

从原文可以看出，只要 `GPEN` 不等于 `0` 即可启用 `GPIO` 。采用 预置变量法 如下：

```
External(GPEN, FieldUnitObj)
Scope (\)
{
    If (_OSI ("Darwin"))
    {
        GPEN = 1
    }
}
```

示例4

当 `变量` 是只读类型时，解决方法如下：

- 对原始 `变量` 更名
- 补丁文件中重新定义一个同名 `变量`

如：某原文：

```
OperationRegion (PNVA, SystemMemory, PNVB, PNVL)
Field (PNVA, AnyAcc, Lock, Preserve)
{
    ...
    IM01,   8,
    ...
}
...
If ((IM01 == 0x02))
{
    ...
}
```

实际情况 `IM01` 不等于0x02，{...} 的内容无法被执行。为了更正错误采用 二进制更名 和 **SSDT**补丁：

更名： `IM01` rename `XM01`

```
Find:    49 4D 30 31 08
Replace: 58 4D 30 31 08
```

补丁：

```
Name (IM01, 0x02)
If (_OSI ("Darwin"))
{
    ...
}
Else
{
      IM01 = XM01 /* 同原始ACPI变量的路径 */
}
```

风险：OC引导其他系统时可能无法恢复 `XM01` 。

GPI0补丁

说明

- 如果要启用GPI0，其 `_STA` 必须 `Return (0x0F)` 。
- 样本仅供参考。使用时确认GPI0设备的 `_STA` 存在 `GPEN` 或 `GPHD` 。详见《二进制更名与预置变量》。

```
// GPI0 enable
DefinitionBlock("", "SSDT", 2, "OCLT", "GPI0", 0)
{
    External(GPEN, FieldUnitObj)

    Scope (\)
    {
        If (_OSI ("Darwin"))
        {
            GPEN = 1
        }
    }
}
//EOF
```

```
// GPI0 enable
DefinitionBlock("", "SSDT", 2, "OCLT", "GPI0", 0)
{
    External(GPHD, FieldUnitObj)


    Scope (\)
    {
        If (_OSI ("Darwin"))
        {
            GPHD = 2
        }
    }

}
//EOF
```

```
/*
 * For 300-series only. If you can't force enable Legacy RTC in BIOS GUI.
 * macOS does yet not support AWAC, so we have to force enable RTC. Do not use RTC
ACPI patch.
 *
 * The Time and Alarm device provides an alternative to the real time clock (RTC),
which is defined as a fixed feature hardware device.
 * The wake timers allow the system to transition from the S3 (or optionally S4/S5)
 state to S0 state after a time period elapses.
 * In comparison with the Real Time Clock (RTC) Alarm, the Time and Alarm device pr
ovides a larger scale of flexibility in the operation of the wake timers,
 * and allows the implementation of the time source to be abstracted from the OSPM.
 */
DefinitionBlock ("", "SSDT", 2, "ACDT", "AWAC", 0x00000000)
{
    External (STAS, IntObj)

    Scope (_SB)
    {
        Method (_INI, 0, NotSerialized)  // _INI: Initialize
        {
            If (_OSI ("Darwin"))
            {
                STAS = One
            }
        }
    }
}
```

```
//enable RTC
//disable AWAC
DefinitionBlock ("", "SSDT", 2, "OCLT", "RTCAWAC", 0x00000000)
{
    External (STAS, FieldUnitObj)

    Scope (\)
    {
        If (_OSI ("Darwin"))
        {
            STAS = One
        }
    }

}
```

操作系统补丁

描述

- 操作系统补丁 用于解除系统对某些部件的限制。通常情况下，「不建议」 使用 操作系统补丁 。对于受系统限制而无法正常工作的部件，应根据 ACPI 的具体情况定制补丁。
- 有关 操作系统补丁 方面的简要叙述见附件《操作系统补丁由来》。

更名

- **OSID to XSID**

```
Find:     4F534944
Replace:  58534944
```

- **OSIF to XSIF**

```
Find:     4F534946
Replace:  58534946
```

- **_OSI to XOSI**

```
Find:     5F4F5349
Replace:  584F5349
```

在原始 DSDT 中搜索 `OSI` ，如果除了 `_OSI` 函数以外、还有其它包含 `OSI` 的字段（如 Dell 笔记本的 `OSID` 、部分 ThinkPad 和联想笔记本的 `OSIF` ），必须先添加 对这些包含 `OSI` 字段的重命名（如 `OSID to XSID` 、 `OSIF to XSIF` ）、然后再添加 `_OSI to XOSI` 。
如果除 `_OSI` 函数以外无其它包含 `OSI` 的字段、直接添加 `_OSI to XOSI` 即可。

补丁：*SSDT-OC-XOSI*

```
Method(XOSI, 1)
{
    If (_OSI ("Darwin"))
    {
        If (Arg0 == //"Windows 2009"  //  = win7, Win Server 2008 R2
                    //"Windows 2012"  //  = Win8, Win Server 2012
                    //"Windows 2013"  //  = win8.1
                    "Windows 2015"  //  = Win10
                    //"Windows 2016"  //  = Win10 version 1607
                    //"Windows 2017"  //  = Win10 version 1703
```

```
                    //"Windows 2017.2"//  = Win10 version 1709
                    //"Windows 2018"  //  = Win10 version 1803
                    //"Windows 2018.2"//  = Win10 version 1809
                    //"Windows 2018"  //  = Win10 version 1903
            )
        {
            Return (0xFFFFFFFF)
        }
        Else
        {
            Return (Zero)
        }
    }
    Else
    {
        Return (_OSI (Arg0))
    }
}
```

使用

- 最大值

  对于单系统，可以设定操作系统参数为 DSDT 所允许的最大值。如：DSDT 的最大值是 `Windows 2018` ，则设定 `Arg0 == "Windows 2018"` 。通常 `Arg0 == "Windows 2013"` 以上就能解除系统对部件的限制。

  注：单系统不建议使用 操作系统补丁。

- 匹配值

  对于双系统，设定的操作系统参数应和Windows系统版本一致。如：Windows系统是win7，设定 Arg0 == "Windows 2009"。

注意事项

- 某些机器的 `Method` 使用了和 `_OSI` 相似的名称(如 `dell` 的 `OSID`)，当它位于 `_SB` (不足4个字符)并出现全路径时，其二进制数据和 `_OSI` 相同，导致被操作系统更名(`_OSI to XOSI`)连带更名而产生错误。这种情况下需先于 `_OSI to XOSI` 之前对它更名为其他(如 `OSID to XSID`)以避免错误。

附件：操作系统补丁由来

- 当系统加载时，ACPI 的 `_OSI` 会接收到一个参数，不同的系统，接收的参数不同，ACPI 执行的指令就不同。比如，系统是 **Win7**，这个参数是 `Windows 2009` ，系统是 **Win8**，这个参数就是 `Windows 2012` 。如：

```
If ((_OSI ("Windows 2009") || _OSI ("Windows 2013")))
{
    OperationRegion (PCF0, SystemMemory, 0xF0100000, 0x0200)
    Field (PCF0, ByteAcc, NoLock, Preserve)
    {
        HVD0,   32,
        Offset (0x160),
        TPR0,   8
    }
    ...
}
...
Method (_INI, 0, Serialized)  /* _INI: Initialize */
{
    OSYS = 0x07D0
    If (CondRefOf (\_OSI))
    {
        If (_OSI ("Windows 2001"))
        {
            OSYS = 0x07D1
        }

        If (_OSI ("Windows 2001 SP1"))
        {
            OSYS = 0x07D1
        }
        ...
        If (_OSI ("Windows 2013"))
        {
            OSYS = 0x07DD
        }

        If (_OSI ("Windows 2015"))
        {
            OSYS = 0x07DF
        }
        ...
    }
}
```

ACPI 还定义了 `OSYS` ，`OSYS` 和上述参数关系如下：

- `OSYS = 0x07D9` : Win7 系统，即 `Windows 2009`
- `OSYS = 0x07DC` : Win8 系统，即 `Windows 2012`
- `OSYS = 0x07DD` : Win8.1 系统，即 `Windows 2013`
- `OSYS = 0x07DF` : Win10 系统，即 `Windows 2015`
- `OSYS = 0x07E0` : Win10 1607，即 `Windows 2016`
- `OSYS = 0x07E1` : Win10 1703，即 `Windows 2017`
- `OSYS = 0x07E1` : Win10 1709，即 `Windows 2017.2`

    - `OSYS = 0x07E2`：Win10 1803，即 `Windows 2018`
    - `OSYS = 0x07E2`：Win10 1809，即 `Windows 2018.2`
    - `OSYS = 0x????`：Win10 1903，即 `Windows 2019`
    - ...
- 当加载的系统不被 ACPI 识别时，`OSYS` 被赋予默认值，这个值随机器不同而不同，有的代表 `Linux`，有的代表 `Windows2003`，有的是其他值。
- 不同的操作系统支持不同的硬件，比如 `Win8` 以上才支持 I2C 设备。
- 当加载 macOS 时，`_OSI` 接受的参数不会被 ACPI 识别，`OSYS` 被赋予了默认值。这个默认值通常小于 Win8 要求的值，显然 I2C 无法工作。这就需要补丁来纠正这种错误，操作系统补丁源于此。
- 某些其他部件也可能和 `OSYS` 有关。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/Pro
pertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>ACPI</key>
    <dict>
        <key>Patch</key>
        <array>
            <dict>
                <key>Comment</key>
                <string>OSID to XSID</string>
                <key>Count</key>
                <integer>0</integer>
                <key>Enabled</key>
                <true/>
                <key>Find</key>
                <data>
                T1NJRA==
                </data>
                <key>Limit</key>
                <integer>0</integer>
                <key>Mask</key>
                <data>
                </data>
                <key>OemTableId</key>
                <data>
                </data>
                <key>Replace</key>
                <data>
                WFNJRA==
                </data>
                <key>ReplaceMask</key>
                <data>
                </data>
                <key>Skip</key>
                <integer>0</integer>
                <key>TableLength</key>
                <integer>0</integer>
                <key>TableSignature</key>
                <data>
                </data>
            </dict>
            <dict>
                <key>Comment</key>
                <string>OSIF to XSIF</string>
                <key>Count</key>
                <integer>0</integer>
                <key>Enabled</key>
                <true/>
```

```
                <key>Find</key>
                <data>
                T1NJRg==
                </data>
                <key>Limit</key>
                <integer>0</integer>
                <key>Mask</key>
                <data>
                </data>
                <key>OemTableId</key>
                <data>
                </data>
                <key>Replace</key>
                <data>
                WFNJRg==
                </data>
                <key>ReplaceMask</key>
                <data>
                </data>
                <key>Skip</key>
                <integer>0</integer>
                <key>TableLength</key>
                <integer>0</integer>
                <key>TableSignature</key>
                <data>
                </data>
            </dict>
        </array>
    </dict>
</dict>
</plist>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/Pro
pertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>ACPI</key>
    <dict>
        <key>Patch</key>
        <array>
            <dict>
                <key>Comment</key>
                <string>_OSI to XOSI</string>
                <key>Count</key>
                <integer>0</integer>
                <key>Enabled</key>
                <true/>
                <key>Find</key>
                <data>
                X09TSQ==
                </data>
                <key>Limit</key>
                <integer>0</integer>
                <key>Mask</key>
                <data>
                </data>
                <key>OemTableId</key>
                <data>
                </data>
                <key>Replace</key>
                <data>
                WE9TSQ==
                </data>
                <key>ReplaceMask</key>
                <data>
                </data>
                <key>Skip</key>
                <integer>0</integer>
                <key>TableLength</key>
                <integer>0</integer>
                <key>TableSignature</key>
                <data>
                </data>
            </dict>
        </array>
    </dict>
</dict>
</plist>
```

```
//
// In config ACPI, OSID to XSID
// Find:     4F534944
// Replace:  58534944
//
// In config ACPI, OSIF to XSIF
// Find:     4F534946
// Replace:  58534946
//
// In config ACPI, _OSI to XOSI
// Find:     5F4F5349
// Replace:  584F5349
//
// Search _OSI......
//
DefinitionBlock("", "SSDT", 2, "OCLT", "OC-XOSI", 0)
{
    Method(XOSI, 1)
    {
        If (_OSI ("Darwin"))
        {
            If (Arg0 == //"Windows 2009"  //  = win7, Win Server 2008 R2
                        //"Windows 2012"  //  = Win8, Win Server 2012
                        //"Windows 2013"  //  = win8.1
                        "Windows 2015"  //  = Win10
                        //"Windows 2016"  //  = Win10 version 1607
                        //"Windows 2017"  //  = Win10 version 1703
                        //"Windows 2017.2"//  = Win10 version 1709
                        //"Windows 2018"  //  = Win10 version 1803
                        //"Windows 2018.2"//  = Win10 version 1809
                        //"Windows 2019"  //  = Win10 version 1903
                )
            {
                Return (0xFFFFFFFF)
            }


            Else
            {
                Return (Zero)
            }
        }

        Else
        {
            Return (_OSI (Arg0))
        }
    }

}
```

```
//EOF
```

注入设备

- 注入 `x86` 实现 `CPU` 电源管理
- 注入 `PNLF` 实现亮度调节
- 注入 `SBUS` 设备

注入X86

描述

注入 X86，实现 CPU 电源管理。

使用说明

- DSDT中搜索 `Processor` ，如：

```
Scope (_PR)
{
    Processor (CPU0, 0x01, 0x00001810, 0x06){}
    Processor (CPU1, 0x02, 0x00001810, 0x06){}
    Processor (CPU2, 0x03, 0x00001810, 0x06){}
    Processor (CPU3, 0x04, 0x00001810, 0x06){}
    Processor (CPU4, 0x05, 0x00001810, 0x06){}
    Processor (CPU5, 0x06, 0x00001810, 0x06){}
    Processor (CPU6, 0x07, 0x00001810, 0x06){}
    Processor (CPU7, 0x08, 0x00001810, 0x06){}
}
```

根据查询结果，选择注入文件 ***SSDT-PLUG-_PR.CPU0***

再如：

```
Scope (_SB)
{
    Processor (PR00, 0x01, 0x00001810, 0x06){}
    Processor (PR01, 0x02, 0x00001810, 0x06){}
    Processor (PR02, 0x03, 0x00001810, 0x06){}
    Processor (PR03, 0x04, 0x00001810, 0x06){}
    Processor (PR04, 0x05, 0x00001810, 0x06){}
    Processor (PR05, 0x06, 0x00001810, 0x06){}
    Processor (PR06, 0x07, 0x00001810, 0x06){}
    Processor (PR07, 0x08, 0x00001810, 0x06){}
    Processor (PR08, 0x09, 0x00001810, 0x06){}
    Processor (PR09, 0x0A, 0x00001810, 0x06){}
    Processor (PR10, 0x0B, 0x00001810, 0x06){}
    Processor (PR11, 0x0C, 0x00001810, 0x06){}
    Processor (PR12, 0x0D, 0x00001810, 0x06){}
    Processor (PR13, 0x0E, 0x00001810, 0x06){}
    Processor (PR14, 0x0F, 0x00001810, 0x06){}
    Processor (PR15, 0x10, 0x00001810, 0x06){}
}
```

根据查询结果，选择注入文件：***SSDT-PLUG-_SB.PR00***

- 如果查询结果和补丁文件名 无法对应 ，请任选一个文件作为样本，自行修改补丁文件相关内容。

注意

2 代，3 代机器无需注入 X86。

```
/*
 * XCPM power management compatibility table.
 */
DefinitionBlock ("", "SSDT", 2, "OCLT", "CpuPlug", 0x00003000)
{
    External (_PR.CPU0, ProcessorObj)

    Scope (_PR.CPU0)
    {
        Method (DTGP, 5, NotSerialized)
        {
            If ((Arg0 == ToUUID ("a0b5b7c6-1318-441c-b0c9-fe695eaf949b")))
            {
                If ((Arg1 == One))
                {
                    If ((Arg2 == Zero))
                    {
                        Arg4 = Buffer (One)
                            {
                                 0x03
                            }
                        Return (One)
                    }

                    If ((Arg2 == One))
                    {
                        Return (One)
                    }
                }
            }

            Arg4 = Buffer (One)
                {
                     0x00
                }
            Return (Zero)
        }

        Method (_DSM, 4, NotSerialized)
        {
            Local0 = Package (0x02)
                {
                    "plugin-type",
                    One
                }
            DTGP (Arg0, Arg1, Arg2, Arg3, RefOf (Local0))
            Return (Local0)
        }
    }
}
```

```
/*
 * XCPM power management compatibility table.
 */
DefinitionBlock ("", "SSDT", 2, "OCLT", "CpuPlug", 0x00003000)
{
    External (_PR.P000, ProcessorObj)

    Scope (_PR.P000)
    {
        Method (DTGP, 5, NotSerialized)
        {
            If ((Arg0 == ToUUID ("a0b5b7c6-1318-441c-b0c9-fe695eaf949b")))
            {
                If ((Arg1 == One))
                {
                    If ((Arg2 == Zero))
                    {
                        Arg4 = Buffer (One)
                            {
                                 0x03
                            }
                        Return (One)
                    }

                    If ((Arg2 == One))
                    {
                        Return (One)
                    }
                }
            }

            Arg4 = Buffer (One)
                {
                     0x00
                }
            Return (Zero)
        }

        Method (_DSM, 4, NotSerialized)
        {
            Local0 = Package (0x02)
                {
                    "plugin-type",
                    One
                }
            DTGP (Arg0, Arg1, Arg2, Arg3, RefOf (Local0))
            Return (Local0)
        }
    }
}
```

```
/*
 * XCPM power management compatibility table.
 */
DefinitionBlock ("", "SSDT", 2, "OCLT", "CpuPlug", 0x00003000)
{
    External (_PR.PR00, ProcessorObj)

    Scope (_PR.PR00)
    {
        Method (DTGP, 5, NotSerialized)
        {
            If ((Arg0 == ToUUID ("a0b5b7c6-1318-441c-b0c9-fe695eaf949b")))
            {
                If ((Arg1 == One))
                {
                    If ((Arg2 == Zero))
                    {
                        Arg4 = Buffer (One)
                            {
                                 0x03
                            }
                        Return (One)
                    }

                    If ((Arg2 == One))
                    {
                        Return (One)
                    }
                }
            }

            Arg4 = Buffer (One)
                {
                     0x00
                }
            Return (Zero)
        }

        Method (_DSM, 4, NotSerialized)
        {
            Local0 = Package (0x02)
                {
                    "plugin-type",
                    One
                }
            DTGP (Arg0, Arg1, Arg2, Arg3, RefOf (Local0))
            Return (Local0)
        }
    }
}
```

```
/*
 * XCPM power management compatibility table.
 */
DefinitionBlock ("", "SSDT", 2, "OCLT", "CpuPlug", 0x00003000)
{
    External (_SB.CPU0, ProcessorObj)

    Scope (_SB.CPU0)
    {
        Method (DTGP, 5, NotSerialized)
        {
            If ((Arg0 == ToUUID ("a0b5b7c6-1318-441c-b0c9-fe695eaf949b")))
            {
                If ((Arg1 == One))
                {
                    If ((Arg2 == Zero))
                    {
                        Arg4 = Buffer (One)
                            {
                                 0x03
                            }
                        Return (One)
                    }

                    If ((Arg2 == One))
                    {
                        Return (One)
                    }
                }
            }

            Arg4 = Buffer (One)
                {
                     0x00
                }
            Return (Zero)
        }

        Method (_DSM, 4, NotSerialized)
        {
            Local0 = Package (0x02)
                {
                    "plugin-type",
                    One
                }
            DTGP (Arg0, Arg1, Arg2, Arg3, RefOf (Local0))
            Return (Local0)
        }
    }
}
```

```
/*
 * XCPM power management compatibility table.
 */
DefinitionBlock ("", "SSDT", 2, "OCLT", "CpuPlug", 0x00003000)
{
    External (_SB.P000, ProcessorObj)

    Scope (_SB.P000)
    {
        Method (DTGP, 5, NotSerialized)
        {
            If ((Arg0 == ToUUID ("a0b5b7c6-1318-441c-b0c9-fe695eaf949b")))
            {
                If ((Arg1 == One))
                {
                    If ((Arg2 == Zero))
                    {
                        Arg4 = Buffer (One)
                            {
                                 0x03
                            }
                        Return (One)
                    }

                    If ((Arg2 == One))
                    {
                        Return (One)
                    }
                }
            }

            Arg4 = Buffer (One)
                {
                     0x00
                }
            Return (Zero)
        }

        Method (_DSM, 4, NotSerialized)
        {
            Local0 = Package (0x02)
                {
                    "plugin-type",
                    One
                }
            DTGP (Arg0, Arg1, Arg2, Arg3, RefOf (Local0))
            Return (Local0)
        }
    }
}
```

```
/*
 * XCPM power management compatibility table.
 */
DefinitionBlock ("", "SSDT", 2, "OCLT", "CpuPlug", 0x00003000)
{
    External (_SB.PR00, ProcessorObj)

    Scope (_SB.PR00)
    {
        Method (DTGP, 5, NotSerialized)
        {
            If ((Arg0 == ToUUID ("a0b5b7c6-1318-441c-b0c9-fe695eaf949b")))
            {
                If ((Arg1 == One))
                {
                    If ((Arg2 == Zero))
                    {
                        Arg4 = Buffer (One)
                            {
                                 0x03
                            }
                        Return (One)
                    }

                    If ((Arg2 == One))
                    {
                        Return (One)
                    }
                }
            }

            Arg4 = Buffer (One)
                {
                     0x00
                }
            Return (Zero)
        }

        Method (_DSM, 4, NotSerialized)
        {
            Local0 = Package (0x02)
                {
                    "plugin-type",
                    One
                }
            DTGP (Arg0, Arg1, Arg2, Arg3, RefOf (Local0))
            Return (Local0)
        }
    }
}
```

```
/*
 * XCPM power management compatibility table.
 */
DefinitionBlock ("", "SSDT", 2, "OCLT", "CpuPlug", 0x00003000)
{
    External (SCK0.C000, ProcessorObj)

    Scope (SCK0.C000)
    {
        Method (DTGP, 5, NotSerialized)
        {
            If ((Arg0 == ToUUID ("a0b5b7c6-1318-441c-b0c9-fe695eaf949b")))
            {
                If ((Arg1 == One))
                {
                    If ((Arg2 == Zero))
                    {
                        Arg4 = Buffer (One)
                            {
                                 0x03
                            }
                        Return (One)
                    }

                    If ((Arg2 == One))
                    {
                        Return (One)
                    }
                }
            }

            Arg4 = Buffer (One)
                {
                     0x00
                }
            Return (Zero)
        }

        Method (_DSM, 4, NotSerialized)
        {
            Local0 = Package (0x02)
                {
                    "plugin-type",
                    One
                }
            DTGP (Arg0, Arg1, Arg2, Arg3, RefOf (Local0))
            Return (Local0)
        }
    }
}
```

```
/*
 * XCPM power management compatibility table.
 */
DefinitionBlock ("", "SSDT", 2, "OCLT", "CpuPlug", 0x00003000)
{
    External (SCK0.CPU0, ProcessorObj)

    Scope (SCK0.CPU0)
    {
        Method (DTGP, 5, NotSerialized)
        {
            If ((Arg0 == ToUUID ("a0b5b7c6-1318-441c-b0c9-fe695eaf949b")))
            {
                If ((Arg1 == One))
                {
                    If ((Arg2 == Zero))
                    {
                        Arg4 = Buffer (One)
                            {
                                 0x03
                            }
                        Return (One)
                    }

                    If ((Arg2 == One))
                    {
                        Return (One)
                    }
                }
            }

            Arg4 = Buffer (One)
                {
                     0x00
                }
            Return (Zero)
        }

        Method (_DSM, 4, NotSerialized)
        {
            Local0 = Package (0x02)
                {
                    "plugin-type",
                    One
                }
            DTGP (Arg0, Arg1, Arg2, Arg3, RefOf (Local0))
            Return (Local0)
        }
    }
}
```

OC-PNLF 注入方法

亮度 (`PNLF`) 控制部分的组成

- 驱动：
  - WhateverGreen.kext 内置亮度驱动(需 Lilu.kext)

    默认情况下，WhateverGreen.kext 会加载亮度驱动。如果使用其他亮度驱动应当禁用其内置的亮度驱动。

    - 禁用方法：
      - 添加引导参数 `applbkl=0`
      - 修改驱动的 `Info.plist\IOKitPersonalities\AppleIntelPanelA\IOProbeScore=5500` 。
    - 下载地址：https://github.com/acidanthera/WhateverGreen/releases
  - IntelBacklight.kext
    - 下载地址：https://bitbucket.org/RehabMan/os-x-intel-backlight/src/master/
  - ACPIBacklight.kext
    - 下载地址：https://bitbucket.org/RehabMan/os-x-acpi-backlight/src/master/
- 补丁
  - 定制亮度补丁
    - ***SSDT-PNLF-SNB_IVY***：2, 3 代亮度补丁。
    - ***SSDT-PNLF-Haswell_Broadwell***： 4, 5 代亮度补丁。
    - ***SSDT-PNLF-SKL_KBL***：6, 7 代亮度补丁。
    - ***SSDT-PNLF-CFL***：8 代+ 亮度补丁。

      以上补丁插入于 `_SB` 。
  - RehabMan 亮度补丁
    - https://github.com/RehabMan/OS-X-Clover-Laptop-Config/blob/master/hotpatch/SSDT-PNLF.dsl
    - https://github.com/RehabMan/OS-X-Clover-Laptop-Config/blob/master/hotpatch/SSDT-PNLFCFL.dsl
    - https://github.com/RehabMan/OS-X-Clover-Laptop-Config/blob/master/hotpatch/SSDT-RMCF.dsl

      RehabMan 亮度补丁插入于 `_SB.PCI0.IGPU` ，使用时将补丁文件的 `IGPU` 重命名为 ACPI 中的原始名称（如： `GFX0` ）。

常用注入方法

- 驱动: WhateverGreen
- 补丁: 定制亮度补丁或 RehabMan 亮度补丁

ACPI注入方法

- 驱动: ACPIBacklight.kext（需禁用 WhateverGreen.kext 内置亮度驱动，见上文的禁用方法）
- 补丁: 见《ACPI 亮度补丁》方法

其他注入方法

按照驱动 + 补丁的原则自行尝试。

注意事项

- 选用某一注入方法时，应清除其他方法有关的驱动、补丁、设置等。
- 当使用定制亮度补丁时，需注意补丁都是在 `_SB` 下注入的 `PNLF` 设备，当原始 `ACPI` 中存在 `PNLF` 字段时，需将其更名，否则会影响 `Windows` 引导。也可以用 `RehabMan` 的补丁。更名如下：

```
// PNLF to XNLF
Find:    504E 4C46
Replace: 584E 4C46
```

ACPI亮度补丁

说明

常用注入方法不起作用时，尝试本方法。

补丁：*SSDT-PNLF-ACPI*

补丁可能需要修改才适合你。修改方法：

- 提取本机 ACPI
- 所有 ACPI 文件中搜索 `_BCL`，`_BCM`，`_BQC` ，记录它们所属设备名称，如 `LCD` 。
- 修改补丁文件中的 `DD1F` 为前面记录的名称（ `DD1F` 替换为 `LCD` ）。参考 《修改图示》。
- 修改补丁文件的 `IGPU` 为 ACPI 的显卡名称（如 `IGPU` 替换为 `GFX0` ）。

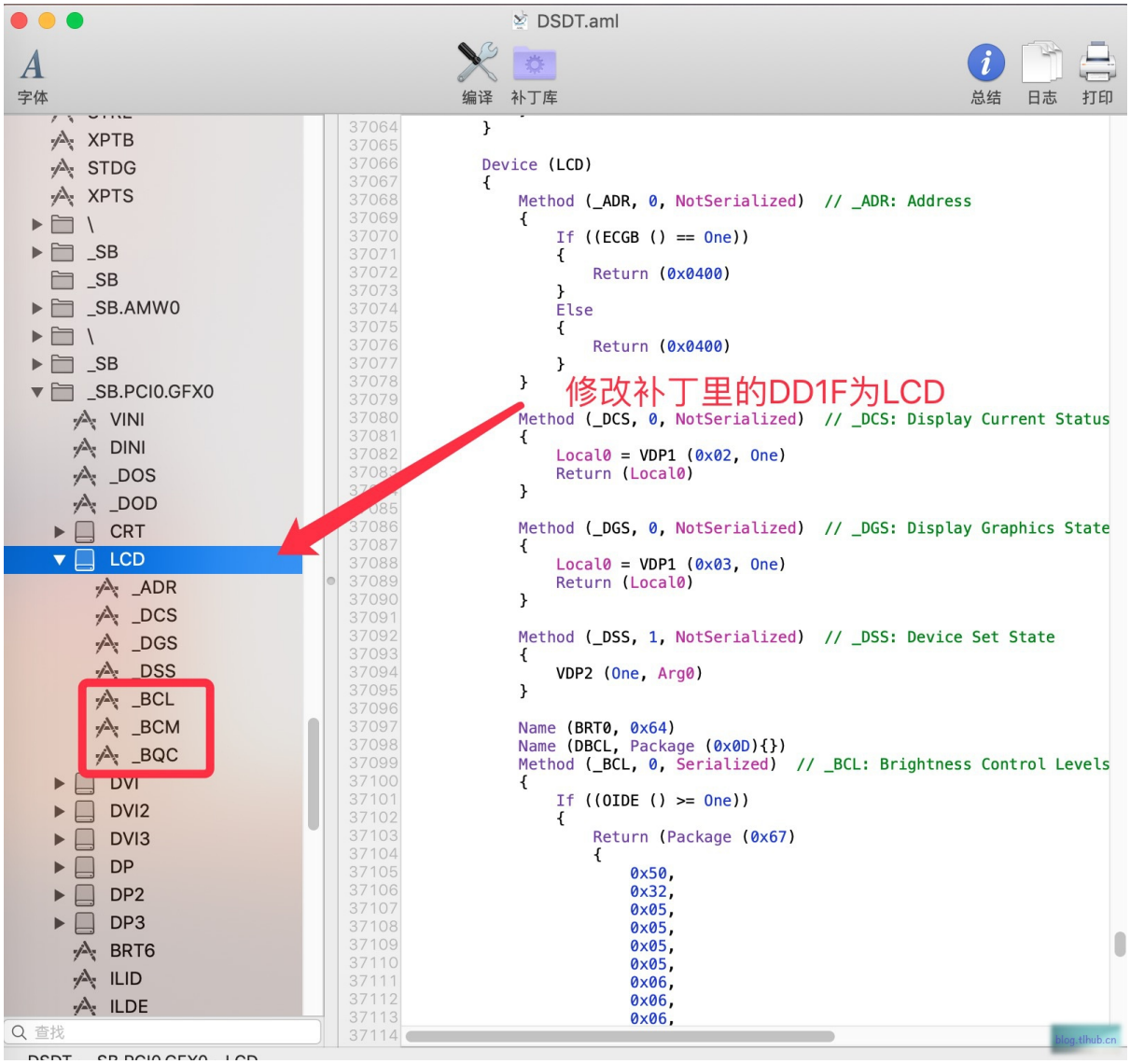
驱动

- ACPIBacklight.kext

```
//
DefinitionBlock("", "SSDT", 2, "OCLT", "PNLFACPI", 0)
{
    External(_SB.PCI0.IGPU, DeviceObj)
    External(_SB.PCI0.IGPU.DD1F._BCL, MethodObj)
    External(_SB.PCI0.IGPU.DD1F._BCM, MethodObj)
    External(_SB.PCI0.IGPU.DD1F._BQC, MethodObj)
    External(_SB.PCI0.IGPU._DOS, MethodObj)

    Scope(_SB.PCI0.IGPU)
    {
        Device(PNLF)
        {
            Name (_ADR, Zero)
            Name (_HID, EisaId ("APP0002"))
            Name (_CID, "backlight")
            Name (_UID, 10)
            // _BCM/_BQC: set/get for brightness level
            Method (_BCM, 1, NotSerialized)
            {
                ^^DD1F._BCM(Arg0)
            }
            Method (_BQC, 0, NotSerialized)
            {
                Return(^^DD1F._BQC())
            }
            Method (_BCL, 0, NotSerialized)
            {
                Return(^^DD1F._BCL())
            }
            Method (_DOS, 1, NotSerialized)
            {
                ^^_DOS(Arg0)
            }
            // extended _BCM/_BQC for setting "in between" levels
            Method (XBCM, 1, NotSerialized)
            {
                // Update backlight via existing DSDT methods
                ^^DD1F._BCM(Arg0)
            }
            Method (XBQC, 0, NotSerialized)
            {
                Return(^^DD1F._BQC())
            }
            // Use XOPT=1 to disable smooth transitions
            Name (XOPT, Zero)
            // XRGL/XRGH: defines the valid range
            Method (XRGL, 0, NotSerialized)
            {
                Store(_BCL(), Local0)
```

```
                Store(DerefOf(Index(Local0, 2)), Local0)
                Return(Local0)
            }
            Method (XRGH, 0, NotSerialized)
            {
                Store(_BCL(), Local0)
                Store(DerefOf(Index(Local0, Subtract(SizeOf(Local0), 1))), Local0)
                Return(Local0)
            }
            Method (_INI, 0, NotSerialized)
            {
                //XRGL()
                //XRGH()
            }
            Method (_STA, 0, NotSerialized)
            {
                If (_OSI ("Darwin"))
                {
                    Return (0x0B)
                }
                Else
                {
                    Return (Zero)
                }
            }
        }
    }
}
//EOF
```


DSDT.aml
字体
编译
补丁库
总结
日志
打印
XPTB
STDG
XPTS
\
_SB
_SB
_SB.AMW0
\
_SB
_SB.PCI0.GFX0
VINI
DINI
_DOS
_DOD
CRT
LCD
_ADR
_DCS
_DGS
_DSS
_BCL
_BCM
_BQC
DVI
DVI2
DVI3
DP
DP2
DP3
BRT6
ILID
ILDE
查找
修改补丁里的DD1F为LCD
37064 }
37065
37066 Device (LCD)
37067 {
37068 Method (_ADR, 0, NotSerialized) // _ADR: Address
37069 {
37070 If ((ECGB () == One))
37071 {
37072 Return (0x0400)
37073 }
37074 Else
37075 {
37076 Return (0x0400)
37077 }
37078 }
37079
37080 Method (_DCS, 0, NotSerialized) // _DCS: Display Current Status
37081 {
37082 Local0 = VDP1 (0x02, One)
37083 Return (Local0)
37084 }
37085
37086 Method (_DGS, 0, NotSerialized) // _DGS: Display Graphics State
37087 {
37088 Local0 = VDP1 (0x03, One)
37089 Return (Local0)
37090 }
37091
37092 Method (_DSS, 1, NotSerialized) // _DSS: Device Set State
37093 {
37094 VDP2 (One, Arg0)
37095 }
37096
37097 Name (BRT0, 0x64)
37098 Name (DBCL, Package (0x0D){})
37099 Method (_BCL, 0, Serialized) // _BCL: Brightness Control Levels
37100 {
37101 If ((OIDE () >= One))
37102 {
37103 Return (Package (0x67)
37104 {
37105 0x50,
37106 0x32,
37107 0x05,
37108 0x05,
37109 0x05,
37110 0x05,
37111 0x06,
37112 0x06,
37113 0x06,
37114
blog.tihub.cn

```
//
DefinitionBlock("", "SSDT", 2, "OCLT", "PNLF", 0)
{
    Scope(_SB)
    {
        Device(PNLF)
        {
            Name(_ADR, Zero)
            Name(_HID, EisaId ("APP0002"))
            Name(_CID, "backlight")
            //CoffeeLake+
            Name(_UID, 19)
            Method (_STA, 0, NotSerialized)
            {
                If (_OSI ("Darwin"))
                {
                    Return (0x0B)
                }
                Else
                {
                    Return (Zero)
                }
            }
        }
    }
}
//EOF
```

```
//
DefinitionBlock("", "SSDT", 2, "OCLT", "PNLF", 0)
{
    Scope(_SB)
    {
        Device(PNLF)
        {
            Name(_ADR, Zero)
            Name(_HID, EisaId ("APP0002"))
            Name(_CID, "backlight")
            //Haswell/Broadwell
            Name(_UID, 15)
            Method (_STA, 0, NotSerialized)
            {
                If (_OSI ("Darwin"))
                {
                    Return (0x0B)
                }
                Else
                {
                    Return (Zero)
                }
            }
        }
    }
}
//EOF
```

```
//
DefinitionBlock("", "SSDT", 2, "OCLT", "PNLF", 0)
{
    Scope(_SB)
    {
        Device(PNLF)
        {
            Name(_ADR, Zero)
            Name(_HID, EisaId ("APP0002"))
            Name(_CID, "backlight")
            //Skylake/KabyLake/KabyLake-R
            Name(_UID, 16)
            Method (_STA, 0, NotSerialized)
            {
                If (_OSI ("Darwin"))
                {
                    Return (0x0B)
                }
                Else
                {
                    Return (Zero)
                }
            }
        }
    }
}
//EOF
```

```
//
DefinitionBlock("", "SSDT", 2, "OCLT", "PNLF", 0)
{
    Scope(_SB)
    {
        Device(PNLF)
        {
            Name(_ADR, Zero)
            Name(_HID, EisaId ("APP0002"))
            Name(_CID, "backlight")
            //Sandy/Ivy
            Name(_UID, 14)
            Method (_STA, 0, NotSerialized)
            {
                If (_OSI ("Darwin"))
                {
                    Return (0x0B)
                }
                Else
                {
                    Return (Zero)
                }
            }
        }
    }
}
//EOF
```

SSDT-SBUS(SMBU) 补丁

设备名称

DSDT 中搜索 `0x001F0003` (6 代以前) 或者 `0x001F0004` (6 代及以后)，查看其所属设备名称。

补丁

- 设备名称是 `SBUS` ，使用 ***SSDT-SBUS***
- 设备名称是 `SMBU` ，使用 ***SSDT-SMBU***
- 设备名称是其他名称的，自行修改补丁相关内容

备注

TP机器多为 `SMBU` 。

```
//
DefinitionBlock ("", "SSDT", 2, "OCLT", "SBUS", 0)
{
    External (_SB_.PCI0.SBUS, DeviceObj)

    Scope (_SB.PCI0.SBUS)
    {
        Device (BUS0)
        {
            Name (_CID, "smbus")
            Name (_ADR, Zero)
            Device (DVL0)
            {
                Name (_ADR, 0x57)
                Name (_CID, "diagsvault")
                Method (_DSM, 4, NotSerialized)
                {
                    If (!Arg2)
                    {
                        Return (Buffer (One)
                        {
                             0x03
                        })
                    }

                    Return (Package (0x02)
                    {
                        "address",
                        0x57
                    })
                }
            }
            Method (_STA, 0, NotSerialized)
            {
                If (_OSI ("Darwin"))
                {
                    Return (0x0F)
                }
                Else
                {
                    Return (Zero)
                }
            }
        }
    }
}
```

```
//
DefinitionBlock ("", "SSDT", 2, "OCLT", "SMBU", 0)
{
    External (_SB_.PCI0.SMBU, DeviceObj)

    Scope (_SB.PCI0.SMBU)
    {
        Device (BUS0)
        {
            Name (_CID, "smbus")
            Name (_ADR, Zero)
            Device (DVL0)
            {
                Name (_ADR, 0x57)
                Name (_CID, "diagsvault")
                Method (_DSM, 4, NotSerialized)
                {
                    If (!Arg2)
                    {
                        Return (Buffer (One)
                        {
                             0x03
                        })
                    }

                    Return (Package (0x02)
                    {
                        "address",
                        0x57
                    })
                }
            }
            Method (_STA, 0, NotSerialized)
            {
                If (_OSI ("Darwin"))
                {
                    Return (0x0F)
                }
                Else
                {
                    Return (Zero)
                }
            }
        }
    }
}
```

添加缺失的部件

描述

添加缺失的部件只是一种完善方案，非必要！

使用说明

DSDT中:

- 搜索 `PNP0200` ，如果缺失，可添加 ***SSDT-DMAC***。
- 搜索 `MCHC` ，如果缺失，可添加 ***SSDT-MCHC***。
- 搜索 `PNP0C01` ，如果缺失，可添加 ***SSDT-MEM2***。
- 搜索 `0x00160000` ，如果缺失，可添加 ***SSDT-IMEI***。
- 6 代以上机器，搜索 `0x001F0002` ，如果缺失，可添加 ***SSDT-PPMC***。
- 6 代以上机器，搜索 `PMCR` 、 `APP9876` ，如果缺失，可添加 ***SSDT-PMCR***。

  说明：@请叫我官人 提供方法，目前已成为 OpenCore 官方的 SSDT 示例。

  > Z390 芯片组 PMC (D31:F2) 只能通过 MMIO 启动。由于 ACPI 规范中没有 PMC 设备，苹果推出了自己的命名 `APP9876` 、从 AppleIntelPCHPMC 驱动中访问这个设备。而在其它操作系统中，一般会使用 `HID: PNP0C02` 、 `UID: PCHRESV` 访问这个设备。
  > 包括 APTIO V 在内的平台，在初始化 PMC 设备之前不能读写 NVRAM（在 SMM 模式中被冻结）。
  > 虽然不知道为什么会这样，但是值得注意的是 PMC 和 SPI 位于不同的内存区域，PCHRESV 同时映射了这两个区域，但是苹果的 AppleIntelPCHPMC 只会映射 PMC 所在的区域。
  > PMC 设备与 LPC 总线之间毫无关系，这个 SSDT 纯粹是为了加快 PMC 的初始化而把该设备添加到 LPC 总线下。如果将其添加到 PCI0 总线中、PMC 只会在 PCI 配置结束后启动，对于需要读取 NVRAM 的操作来说就已经太晚了。

- 搜索 `PNP0C0C` ，如果缺失，可添加 ***SSDT-PWRB***。
- 搜索 `PNP0C0E` ，如果缺失，可添加 ***SSDT-SLPB*** ，《PNP0C0E睡眠修正方法》需要这个部件。

注意

使用以上部分补丁时，注意 `LPCB` 名称应和原始ACPI名称一致。

```
//Add DMAC
DefinitionBlock ("", "SSDT", 2, "OCLT", "DMAC", 0)
{
    External(_SB.PCI0.LPCB, DeviceObj)
    Scope (_SB.PCI0.LPCB)
    {
        Device (DMAC)
        {
            Name (_HID, EisaId ("PNP0200"))
            Name (_CRS, ResourceTemplate ()
            {
                IO (Decode16,
                    0x0000,             // Range Minimum
                    0x0000,             // Range Maximum
                    0x01,               // Alignment
                    0x20,               // Length
                    )
                IO (Decode16,
                    0x0081,             // Range Minimum
                    0x0081,             // Range Maximum
                    0x01,               // Alignment
                    0x11,               // Length
                    )
                IO (Decode16,
                    0x0093,             // Range Minimum
                    0x0093,             // Range Maximum
                    0x01,               // Alignment
                    0x0D,               // Length
                    )
                IO (Decode16,
                    0x00C0,             // Range Minimum
                    0x00C0,             // Range Maximum
                    0x01,               // Alignment
                    0x20,               // Length
                    )
                DMA (Compatibility, NotBusMaster, Transfer8_16, )
                    {4}
            })
            Method (_STA, 0, NotSerialized)
            {
                If (_OSI ("Darwin"))
                {
                    Return (0x0F)
                }
                Else
                {
                    Return (Zero)
                }
            }
        }
```

```
    }
}
```

```
/*
 * Only necessary when no IMEI device (with any name) is present in the DSDT and a custom device-id
 * is needed to be set via DeviceProperties (some Sandy Bridge or Ivy Bridge configurations).
 */
DefinitionBlock ("", "SSDT", 2, "ACDT", "IMEI", 0x00000000)
{
    External (_SB_.PCI0, DeviceObj)

    Scope (_SB.PCI0)
    {
        Device (IMEI)
        {
            Name (_ADR, 0x00160000)  // _ADR: Address
        }
    }
}
```

```
//Add MCHC
DefinitionBlock ("", "SSDT", 2, "OCLT", "MCHC", 0)
{
    External (_SB.PCI0, DeviceObj)
    Scope (_SB.PCI0)
    {
        Device (MCHC)
        {
            Name (_ADR, Zero)
            Method (_STA, 0, NotSerialized)
            {
                If (_OSI ("Darwin"))
                {
                    Return (0x0F)
                }
                Else
                {
                    Return (Zero)
                }
            }
        }
    }
}
```

```
//Add MEM2
DefinitionBlock ("", "SSDT", 2, "OCLT", "MEM2", 0)
{
    Device (MEM2)
    {
        Name (_HID, EisaId ("PNP0C01"))
        Name (_UID, 0x02)
        Name (CRS, ResourceTemplate ()
        {
            Memory32Fixed (ReadWrite,
                0x20000000,         // Address Base
                0x00200000,         // Address Length
                )
            Memory32Fixed (ReadWrite,
                0x40000000,         // Address Base
                0x00200000,         // Address Length
                )
        })
        Method (_CRS, 0, NotSerialized)
        {
            Return (CRS)
        }

        Method (_STA, 0, NotSerialized)
        {
            If (_OSI ("Darwin"))
            {
                Return (0x0F)
            }
            Else
            {
                Return (Zero)
            }
        }
    }
}
```

```
//Add PMCR
DefinitionBlock ("", "SSDT", 2, "OCLT", "PMCR", 0)
{
    External(_SB.PCI0.LPCB, DeviceObj)
    Scope (_SB.PCI0.LPCB)
    {
        Device (PMCR)
        {
            Name (_HID, EisaId ("APP9876"))
            Name (_CRS, ResourceTemplate ()
            {
                Memory32Fixed (ReadWrite,
                    0xFE000000,
                    0x00010000
                    )

            })
            Method (_STA, 0, NotSerialized)
            {
                If (_OSI ("Darwin"))
                {
                    Return (0x0B)
                }
                Else
                {
                    Return (Zero)
                }
            }
        }
    }
}
```

```
//Add PPMC
DefinitionBlock ("", "SSDT", 2, "OCLT", "PPMC", 0)
{
    External (_SB.PCI0, DeviceObj)
    Scope (_SB.PCI0)
    {
        Device (PPMC)
        {
            Name (_ADR, 0x001F0002)
            Method (_STA, 0, NotSerialized)
            {
                If (_OSI ("Darwin"))
                {
                    Return (0x0F)
                }
                Else
                {
                    Return (Zero)
                }
            }
        }
    }
}
```

```
//Add PWRB
DefinitionBlock("", "SSDT", 2, "OCLT", "PNP0C0C", 0)
{
    //search PNP0C0C
    Scope (\_SB)
    {
        Device (PWRB)
        {
            Name (_HID, EisaId ("PNP0C0C"))
            Method (_STA, 0, NotSerialized)
            {
                If (_OSI ("Darwin"))
                {
                    Return (0x0F)
                }
                Else
                {
                    Return (Zero)
                }
            }
        }
    }
}
//EOF
```

```
//Add SLPB
DefinitionBlock("", "SSDT", 2, "OCLT", "PNP0C0E", 0)
{
    //search PNP0C0E
    Scope (\_SB)
    {
        Device (SLPB)
        {
            Name (_HID, EisaId ("PNP0C0E"))
            Method (_STA, 0, NotSerialized)
            {
                If (_OSI ("Darwin"))
                {
                    Return (0x0B)
                }
                Else
                {
                    Return (Zero)
                }
            }
        }
    }
}
//EOF
```

PS2键盘映射及亮度快捷键

描述

- 通过键盘映射可以实现某个按键被按下后产生另一个按键的效果。比如，可以指定按下 `A/a` 后，打印输出的是 `Z/z` 。再比如，指定 `F2` 实现原来 `F10` 的功能。
- 新版【9月30日】 **VoodooPS2Controller.kext** 将亮度快捷键部分分离出独立驱动 **BrightnessKeys.kext** 并由它提供方法 `Notify (^^^GFX0.***, 0x86)` 和 `Notify (^^^GFX0.***, 0x87)` ，传统的亮度快捷键补丁不再需要。如果新的驱动无效请参考本章内容指定 2 个按键映射到 `F14` ， `F15` ，以实现快捷键调节亮度功能。
  - **VoodooPS2Controller.kext** ：https://github.com/acidanthera/VoodooPS2
  - **BrightnessKeys.kext** ：https://github.com/acidanthera/BrightnessKeys
- 不是所有按键都可以实现映射，只有 MAC 系统下能够捕捉到 `PS2 扫描码` 的按键才可以映射。

要求

- 使用 **VoodooPS2Controller.kext** 以及它的子驱动。
- 清除之前的、其他方法的按键映射内容。

PS2 扫描码 和 ABD 扫描码

一个按键会产生 2 种扫描码，分别是 **PS2** 扫描码 和 **ABD** 扫描码。比如 `Z/z` 键的 PS2 扫描码是 `2c` ，ABD 扫描码是 `6` 。因为扫描码的不同，对应了两种映射方法，分别是：

- `PS2 扫描码 –> PS2 扫描码`
- `PS2 扫描码 –> ADB 扫描码`

获取键盘扫描码

- 查阅头文件 `ApplePS2ToADBMap.h` ，文件中列举了大多数按键的扫描码。
- 控制台获取键盘扫描码（二选一）
  - 终端安装 `ioio`

    ```
    ioio -s ApplePS2Keyboard LogScanCodes 1
    ```

  - 修改 `VoodooPS2Keyboard.kext\info\IOKitPersonalities\Platform Profile\Default\ LogScanCodes = 1`

  打开控制台，搜索 `ApplePS2Keyboard` 。按下按键，如 `A/a` ， `Z/z` 。

  ```
  11:58:51.255023 +0800 kernel  ApplePS2Keyboard: sending key 1e=0 down
  11:58:58.636955 +0800 kernel  ApplePS2Keyboard: sending key 2c=6 down
  ```

其中：

第一行的 `1e=0` 中的 `1e` 是 `A/a` 键的 PS2 扫描码，`0` 是 ADB 扫描码。

第二行的 `2c=6` 中的 `2c` 是 `Z/z` 键的 PS2 扫描码，`6` 是 ADB 扫描码。

映射方法

通过修改 `VoodooPS2Keyboard.kext\info.plist` 文件和添加第三方补丁文件的方法可以实现键盘映射。推荐使用第三方补丁文件的方法。

示例：***SSDT-RMCF-PS2Map-AtoZ***。`A/a` 映射 `Z/z` 。

- `A/a` PS2 扫描码: `1e`
- `Z/z` PS2 扫描码: `2c`
- `Z/z` ADB 扫描码: `06`

以下两种映射方法任选其一

PS2 扫描码 —> PS2 扫描码

```
...
  "Custom PS2 Map", Package()
  {
      Package(){},
      "1e=2c",
  },
...
```

PS2 扫描码 —> ADB 扫描码

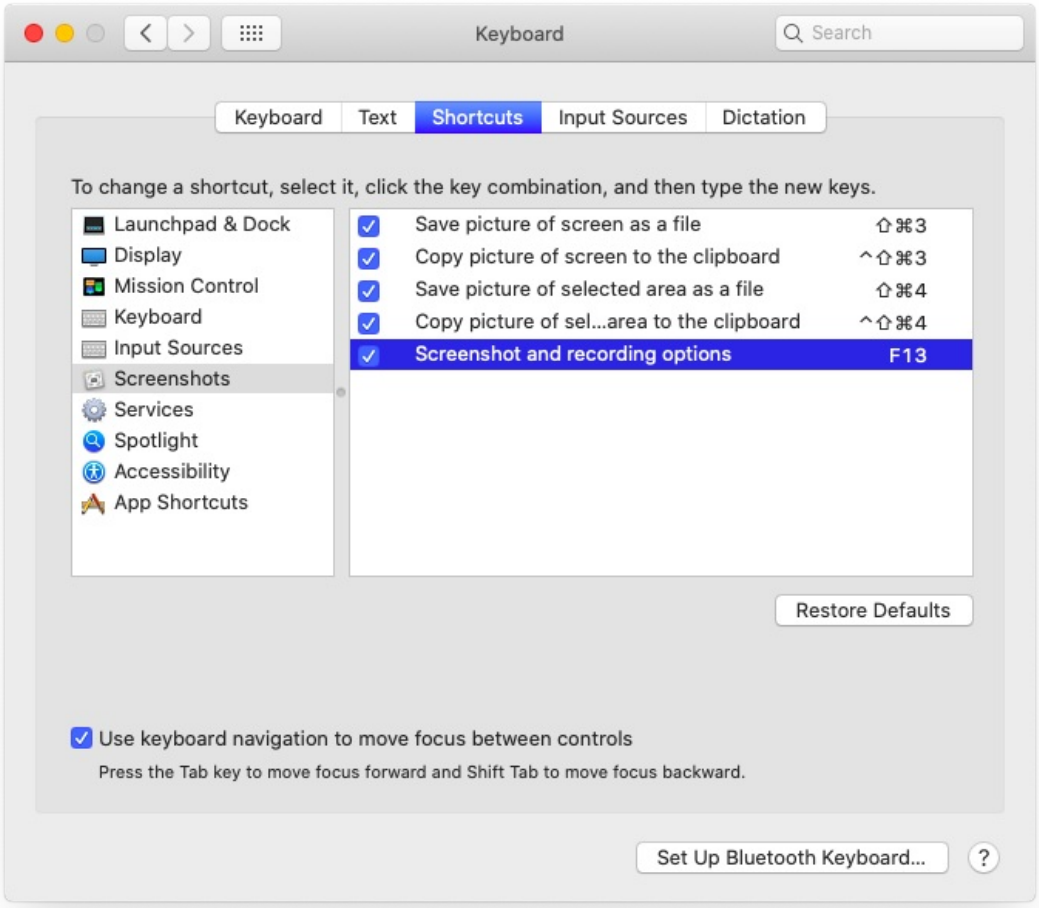
```
...
"Custom ADB Map", Package()
{
    Package(){},
    "1e=06",
}
...
```

注意事项

- 示例中的键盘路径是 `\_SB.PCI0.LPCB.PS2K`，使用时应保证其路径和 ACPI 的键盘路径一致。大多数 Thinkpad 机器的键盘路径是 `\_SB.PCI0.LPC.KBD` 或者 `\_SB.PCI0.LPCB.KBD` 。
- 补丁里使用了变量 `RMCF` ，如果其他键盘补丁里也使用了 `RMCF` ，必须合并后使用。参见 ***SSDT-RMCF-PS2Map-dell***。`注` ：***SSDT-RMCF-MouseAsTrackpad*** 用于强制开启触摸板设置选项。

- 在 VoodooPS2 中，`PrtSc` 按键对应的 PS2 扫描码是 `e037` ，即触摸板（以及 ThinkPad 机器的小红点）的开关。可以将该按键映射到 `F13` 、并在「系统偏好设置」中将 `F13` 绑定到截图功能上：

```
...
"Custom ADB Map", Package()
{
    Package(){},
    "e037=64", // PrtSc -> F13
}
...
```



```
/*
 * Copyright (c) 1998-2000 Apple Computer, Inc. All rights reserved.
 *
 * @APPLE_LICENSE_HEADER_START@
 *
 * The contents of this file constitute Original Code as defined in and
 * are subject to the Apple Public Source License Version 1.1 (the
 * "License").  You may not use this file except in compliance with the
 * License.  Please obtain a copy of the License at
 * http://www.apple.com/publicsource and read it before using this file.
 *
 * This Original Code and all software distributed under the License are
 * distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, EITHER
 * EXPRESS OR IMPLIED, AND APPLE HEREBY DISCLAIMS ALL SUCH WARRANTIES,
 * INCLUDING WITHOUT LIMITATION, ANY WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT.  Please see the
 * License for the specific language governing rights and limitations
 * under the License.
 *
 * @APPLE_LICENSE_HEADER_END@
 */

#ifndef _APPLEPS2TOADBMAP_H
#define _APPLEPS2TOADBMAP_H

#define PROBOOK

#define DEADKEY                 0x80

#if 0
// These ADB codes are for normal NX key brightness (broken in 10.12)
#define BRIGHTNESS_DOWN         0x91
#define BRIGHTNESS_UP           0x90
#else
// These ADB codes are for F14/F15 (works in 10.12)
#define BRIGHTNESS_DOWN         0x6b
#define BRIGHTNESS_UP           0x71
#endif

#define ADB_CONVERTER_LEN       256 * 2     // 0x00~0xff : normal key , 0x100~0x1ff
 : extended key
#define ADB_CONVERTER_EX_START  256

// PS/2 scancode reference : USB HID to PS/2 Scan Code Translation Table PS/2 Set 1
 columns
// http://download.microsoft.com/download/1/6/1/161ba512-40e2-4cc9-843a-923143f3456
c/translate.pdf
static const UInt8 PS2ToADBMapStock[ADB_CONVERTER_LEN] =
{
/*  ADB        AT  ANSI Key-Legend
```

```
    ======================== */
    DEADKEY,// 00
    0x35,   // 01  Escape
    0x12,   // 02  1!
    0x13,   // 03  2@
    0x14,   // 04  3#
    0x15,   // 05  4$
    0x17,   // 06  5%
    0x16,   // 07  6^
    0x1a,   // 08  7&
    0x1c,   // 09  8*
    0x19,   // 0a  9(
    0x1d,   // 0b  0)
    0x1b,   // 0c  -_
    0x18,   // 0d  =+
    0x33,   // 0e  Backspace
    0x30,   // 0f  Tab
    0x0c,   // 10  qQ
    0x0d,   // 11  wW
    0x0e,   // 12  eE
    0x0f,   // 13  rR
    0x11,   // 14  tT
    0x10,   // 15  yY
    0x20,   // 16  uU
    0x22,   // 17  iI
    0x1f,   // 18  oO
    0x23,   // 19  pP
    0x21,   // 1a  [{
    0x1e,   // 1b  ]}
    0x24,   // 1c  Return
    0x3b,   // 1d  Left Control
    0x00,   // 1e  aA
    0x01,   // 1f  sS
    0x02,   // 20  dD
    0x03,   // 21  fF
    0x05,   // 22  gG
    0x04,   // 23  hH
    0x26,   // 24  jJ
    0x28,   // 25  kK
    0x25,   // 26  lL
    0x29,   // 27  ;:
    0x27,   // 28  '"
    0x32,   // 29  `~
    0x38,   // 2a  Left Shift
    0x2a,   // 2b  \| , Europe 1(ISO)
    0x06,   // 2c  zZ
    0x07,   // 2d  xX
    0x08,   // 2e  cC
    0x09,   // 2f  vV
    0x0b,   // 30  bB
    0x2d,   // 31  nN
```

```
    0x2e,   // 32  mM
    0x2b,   // 33  ,<
    0x2f,   // 34  .>
    0x2c,   // 35  /?
    0x3c,   // 36  Right Shift
    0x43,   // 37  Keypad *
    0x3a,   // 38  Left Alt
    0x31,   // 39  Space
    0x39,   // 3a  Caps Lock
    0x7a,   // 3b  F1
    0x78,   // 3c  F2
    0x63,   // 3d  F3
    0x76,   // 3e  F4
    0x60,   // 3f  F5
    0x61,   // 40  F6
    0x62,   // 41  F7
    0x64,   // 42  F8
    0x65,   // 43  F9
    0x6d,   // 44  F10
    0x47,   // 45  Num Lock
    0x6b,   // 46  Scroll Lock
    0x59,   // 47  Keypad 7 Home
    0x5b,   // 48  Keypad 8 Up
    0x5c,   // 49  Keypad 9 PageUp
    0x4e,   // 4a  Keypad -
    0x56,   // 4b  Keypad 4 Left
    0x57,   // 4c  Keypad 5
    0x58,   // 4d  Keypad 6 Right
    0x45,   // 4e  Keypad +
    0x53,   // 4f  Keypad 1 End
    0x54,   // 50  Keypad 2 Down
    0x55,   // 51  Keypad 3 PageDn
    0x52,   // 52  Keypad 0 Insert
    0x41,   // 53  Keypad . Delete
    0x44,   // 54  SysReq
    0x46,   // 55
    0x0a,   // 56  Europe 2(ISO)
    0x67,   // 57  F11
    0x6f,   // 58  F12
    0x51,   // 59  Keypad =
    DEADKEY,// 5a
    DEADKEY,// 5b
    0x5f,   // 5c  Keyboard Int'l 6 (PC9800 Keypad , )
    DEADKEY,// 5d
    DEADKEY,// 5e
    DEADKEY,// 5f
    DEADKEY,// 60
    DEADKEY,// 61
    DEADKEY,// 62
    DEADKEY,// 63
    0x69,   // 64  F13
```

```
    0x6b,   // 65  F14
    0x71,   // 66  F15
    0x6a,   // 67  F16
    0x40,   // 68  F17
    0x4f,   // 69  F18
    0x50,   // 6a  F19
    0x5a,   // 6b  F20
    DEADKEY,// 6c  F21
    DEADKEY,// 6d  F22
    DEADKEY,// 6e  F23
    DEADKEY,// 6f
    0x68,   // 70  Keyboard Intl'2 (Japanese Katakana/Hiragana)
    DEADKEY,// 71
    DEADKEY,// 72
    0x5e,   // 73  Keyboard Int'l 1 (Japanese Ro)
    DEADKEY,// 74
    DEADKEY,// 75
    DEADKEY,// 76  F24 , Keyboard Lang 5 (Japanese Zenkaku/Hankaku)
    0x68,   // 77  Keyboard Lang 4 (Japanese Hiragana)
    0x68,   // 78  Keyboard Lang 3 (Japanese Katakana)
    0x68,   // 79  Keyboard Int'l 4 (Japanese Henkan)
    DEADKEY,// 7a
    0x66,   // 7b  Keyboard Int'l 5 (Japanese Muhenkan)
    DEADKEY,// 7c
    0x5d,   // 7d  Keyboard Int'l 3 (Japanese Yen)
    0x5f,   // 7e  Keypad , (Brazilian Keypad .)
    DEADKEY,// 7f
    DEADKEY,// 80
    DEADKEY,// 81
    DEADKEY,// 82
    DEADKEY,// 83
    DEADKEY,// 84
    DEADKEY,// 85
    DEADKEY,// 86
    DEADKEY,// 87
    DEADKEY,// 88
    DEADKEY,// 89
    DEADKEY,// 8a
    DEADKEY,// 8b
    DEADKEY,// 8c
    DEADKEY,// 8d
    DEADKEY,// 8e
    DEADKEY,// 8f
    DEADKEY,// 90
    DEADKEY,// 91
    DEADKEY,// 92
    DEADKEY,// 93
    DEADKEY,// 94
    DEADKEY,// 95
    DEADKEY,// 96
    DEADKEY,// 97
```

```
    DEADKEY,// 98
    DEADKEY,// 99
    DEADKEY,// 9a
    DEADKEY,// 9b
    DEADKEY,// 9c
    DEADKEY,// 9d
    DEADKEY,// 9e
    DEADKEY,// 9f
    DEADKEY,// a0
    DEADKEY,// a1
    DEADKEY,// a2
    DEADKEY,// a3
    DEADKEY,// a4
    DEADKEY,// a5
    DEADKEY,// a6
    DEADKEY,// a7
    DEADKEY,// a8
    DEADKEY,// a9
    DEADKEY,// aa
    DEADKEY,// ab
    DEADKEY,// ac
    DEADKEY,// ad
    DEADKEY,// ae
    DEADKEY,// af
    DEADKEY,// b0
    DEADKEY,// b1
    DEADKEY,// b2
    DEADKEY,// b3
    DEADKEY,// b4
    DEADKEY,// b5
    DEADKEY,// b6
    DEADKEY,// b7
    DEADKEY,// b8
    DEADKEY,// b9
    DEADKEY,// ba
    DEADKEY,// bb
    DEADKEY,// bc
    DEADKEY,// bd
    DEADKEY,// be
    DEADKEY,// bf
    DEADKEY,// c0
    DEADKEY,// c1
    DEADKEY,// c2
    DEADKEY,// c3
    DEADKEY,// c4
    DEADKEY,// c5
    DEADKEY,// c6
    DEADKEY,// c7
    DEADKEY,// c8
    DEADKEY,// c9
    DEADKEY,// ca
```

```
    DEADKEY,// cb
    DEADKEY,// cc
    DEADKEY,// cd
    DEADKEY,// ce
    DEADKEY,// cf
    DEADKEY,// d0
    DEADKEY,// d1
    DEADKEY,// d2
    DEADKEY,// d3
    DEADKEY,// d4
    DEADKEY,// d5
    DEADKEY,// d6
    DEADKEY,// d7
    DEADKEY,// d8
    DEADKEY,// d9
    DEADKEY,// da
    DEADKEY,// db
    DEADKEY,// dc
    DEADKEY,// dd
    DEADKEY,// de
    DEADKEY,// df
    DEADKEY,// e0
    DEADKEY,// e1
    DEADKEY,// e2
    DEADKEY,// e3
    DEADKEY,// e4
    DEADKEY,// e5
    DEADKEY,// e6
    DEADKEY,// e7
    DEADKEY,// e8
    DEADKEY,// e9
    DEADKEY,// ea
    DEADKEY,// eb
    DEADKEY,// ec
    DEADKEY,// ed
    DEADKEY,// ee
    DEADKEY,// ef
    DEADKEY,// f0
    0x66,   // f1*  Keyboard Lang 2 (Korean Hanja)
    0x68,   // f2*  Keyboard Lang 1 (Korean Hangul)
    DEADKEY,// f3
    DEADKEY,// f4
    DEADKEY,// f5
    DEADKEY,// f6
    DEADKEY,// f7
    DEADKEY,// f8
    DEADKEY,// f9
    DEADKEY,// fa
    DEADKEY,// fb
    DEADKEY,// fc
    DEADKEY,// fd
```

```
    DEADKEY,// fe
    DEADKEY,// ff
    DEADKEY,// e0 00
    DEADKEY,// e0 01
    DEADKEY,// e0 02
    DEADKEY,// e0 03
    DEADKEY,// e0 04
    BRIGHTNESS_DOWN,    // e0 05 dell down
    BRIGHTNESS_UP,      // e0 06 dell up
    DEADKEY,// e0 07
#ifndef PROBOOK
    BRIGHTNESS_UP,      // e0 08 samsung up
    BRIGHTNESS_DOWN,    // e0 09 samsung down
#else
    DEADKEY,// e0 08
    0x83,   // e0 09 Launchpad (hp Fn+F6)
#endif
    0xa0,   // e0 0a Mission Control (hp Fn+F5)
    DEADKEY,// e0 0b
    DEADKEY,// e0 0c
    DEADKEY,// e0 0d
    DEADKEY,// e0 0e
    DEADKEY,// e0 0f
    0x4d,   // e0 10  Scan Previous Track (hp Fn+F10)
    DEADKEY,// e0 11
    BRIGHTNESS_DOWN,    // e0 12 hp down (Fn+F2)
    DEADKEY,// e0 13
    DEADKEY,// e0 14
    DEADKEY,// e0 15
    DEADKEY,// e0 16
    BRIGHTNESS_UP,      // e0 17 hp up (Fn+F3)
    DEADKEY,// e0 18
    0x42,   // e0 19  Scan Next Track (hp Fn+F12)
    DEADKEY,// e0 1a
    DEADKEY,// e0 1b
    0x4c,   // e0 1c  Keypad Enter
    0x3e,   // e0 1d  Right Control
    DEADKEY,// e0 1e
    DEADKEY,// e0 1f
    0x4a,   // e0 20  Mute (hp Fn+F7)
    DEADKEY,// e0 21  Calculator
    0x34,   // e0 22  Play/Pause (hp Fn+F11)
    DEADKEY,// e0 23
    DEADKEY,// e0 24  Stop
    DEADKEY,// e0 25
    DEADKEY,// e0 26
    DEADKEY,// e0 27
    DEADKEY,// e0 28
    DEADKEY,// e0 29
    DEADKEY,// e0 2a
    DEADKEY,// e0 2b
```

```
    DEADKEY,// e0 2c
    DEADKEY,// e0 2d
    0x49,   // e0 2e  Volume Down (hp Fn+F8)
    DEADKEY,// e0 2f
    0x48,   // e0 30  Volume Up (hp Fn+F9)
    DEADKEY,// e0 31
    DEADKEY,// e0 32  WWW Home
    DEADKEY,// e0 33
    DEADKEY,// e0 34
    0x4b,   // e0 35  Keypad /
    DEADKEY,// e0 36
    0x69,   // e0 37  Print Screen
    0x3d,   // e0 38  Right Alt
    DEADKEY,// e0 39
    DEADKEY,// e0 3a
    DEADKEY,// e0 3b
    DEADKEY,// e0 3c
    DEADKEY,// e0 3d
    DEADKEY,// e0 3e
    DEADKEY,// e0 3f
    DEADKEY,// e0 40
    DEADKEY,// e0 41
    DEADKEY,// e0 42
    DEADKEY,// e0 43
    DEADKEY,// e0 44
    0x71,   // e0 45* Pause
    DEADKEY,// e0 46* Break(Ctrl-Pause)
    0x73,   // e0 47  Home
    0x7e,   // e0 48  Up Arrow
    0x74,   // e0 49  Page Up
    DEADKEY,// e0 4a
    0x7b,   // e0 4b  Left Arrow
    DEADKEY,// e0 4c
    0x7c,   // e0 4d  Right Arrow
    BRIGHTNESS_UP,      // e0 4e acer up
    0x77,   // e0 4f  End
    0x7d,   // e0 50  Down Arrow
    0x79,   // e0 51  Page Down
    0x92,   // e0 52  Insert = Eject
    0x75,   // e0 53  Delete
    DEADKEY,// e0 54
    DEADKEY,// e0 55
    DEADKEY,// e0 56
    DEADKEY,// e0 57
    DEADKEY,// e0 58
    BRIGHTNESS_UP,      // e0 59 acer up for my acer
    DEADKEY,// e0 5a
    0x37,   // e0 5b  Left GUI(Windows)
    0x36,   // e0 5c  Right GUI(Windows)
    0x6e,   // e0 5d  App( Windows context menu key )
    0x7f,   // e0 5e  System Power / Keyboard Power
```

```
    DEADKEY,// e0 5f  System Sleep (hp Fn+F1)
    DEADKEY,// e0 60
    DEADKEY,// e0 61
    DEADKEY,// e0 62
    DEADKEY,// e0 63  System Wake
    DEADKEY,// e0 64
    DEADKEY,// e0 65  WWW Search
    DEADKEY,// e0 66  WWW Favorites
    DEADKEY,// e0 67  WWW Refresh
    DEADKEY,// e0 68  WWW Stop
    DEADKEY,// e0 69  WWW Forward
    DEADKEY,// e0 6a  WWW Back
    DEADKEY,// e0 6b  My Computer
    DEADKEY,// e0 6c  Mail
    DEADKEY,// e0 6d  Media Select
#ifndef PROBOOK
    BRIGHTNESS_UP,      // e0 6e acer up
    BRIGHTNESS_DOWN,    // e0 6f acer down
#else
    0x70,   // e0 6e  Video Mirror = hp Fn+F4
    DEADKEY,// e0 6f  Fn+Home
#endif
    DEADKEY,// e0 70
    DEADKEY,// e0 71
    DEADKEY,// e0 72
    DEADKEY,// e0 73
    DEADKEY,// e0 74
    DEADKEY,// e0 75
    DEADKEY,// e0 76
#ifndef PROBOOK
    BRIGHTNESS_DOWN,    // e0 77 lg down
    BRIGHTNESS_UP,      // e0 78 lg up
#else
    DEADKEY,// e0 77
    DEADKEY,// e0 78 WiFi on/off button on HP ProBook
#endif
    DEADKEY,// e0 79
    DEADKEY,// e0 7a
    DEADKEY,// e0 7b
    DEADKEY,// e0 7c
    DEADKEY,// e0 7d
    DEADKEY,// e0 7e
    DEADKEY,// e0 7f
    DEADKEY,// e0 80
    DEADKEY,// e0 81
    DEADKEY,// e0 82
    DEADKEY,// e0 83
    DEADKEY,// e0 84
    DEADKEY,// e0 85
    DEADKEY,// e0 86
    DEADKEY,// e0 87
```

```
    DEADKEY,// e0 88
    DEADKEY,// e0 89
    DEADKEY,// e0 8a
    DEADKEY,// e0 8b
    DEADKEY,// e0 8c
    DEADKEY,// e0 8d
    DEADKEY,// e0 8e
    DEADKEY,// e0 8f
    DEADKEY,// e0 90
    DEADKEY,// e0 91
    DEADKEY,// e0 92
    DEADKEY,// e0 93
    DEADKEY,// e0 94
    DEADKEY,// e0 95
    DEADKEY,// e0 96
    DEADKEY,// e0 97
    DEADKEY,// e0 98
    DEADKEY,// e0 99
    DEADKEY,// e0 9a
    DEADKEY,// e0 9b
    DEADKEY,// e0 9c
    DEADKEY,// e0 9d
    DEADKEY,// e0 9e
    DEADKEY,// e0 9f
    DEADKEY,// e0 a0
    DEADKEY,// e0 a1
    DEADKEY,// e0 a2
    DEADKEY,// e0 a3
    DEADKEY,// e0 a4
    DEADKEY,// e0 a5
    DEADKEY,// e0 a6
    DEADKEY,// e0 a7
    DEADKEY,// e0 a8
    DEADKEY,// e0 a9
    DEADKEY,// e0 aa
    DEADKEY,// e0 ab
    DEADKEY,// e0 ac
    DEADKEY,// e0 ad
    DEADKEY,// e0 ae
    DEADKEY,// e0 af
    DEADKEY,// e0 b0
    DEADKEY,// e0 b1
    DEADKEY,// e0 b2
    DEADKEY,// e0 b3
    DEADKEY,// e0 b4
    DEADKEY,// e0 b5
    DEADKEY,// e0 b6
    DEADKEY,// e0 b7
    DEADKEY,// e0 b8
    DEADKEY,// e0 b9
    DEADKEY,// e0 ba
```

```
    DEADKEY,// e0 bb
    DEADKEY,// e0 bc
    DEADKEY,// e0 bd
    DEADKEY,// e0 be
    DEADKEY,// e0 bf
    DEADKEY,// e0 c0
    DEADKEY,// e0 c1
    DEADKEY,// e0 c2
    DEADKEY,// e0 c3
    DEADKEY,// e0 c4
    DEADKEY,// e0 c5
    DEADKEY,// e0 c6
    DEADKEY,// e0 c7
    DEADKEY,// e0 c8
    DEADKEY,// e0 c9
    DEADKEY,// e0 ca
    DEADKEY,// e0 cb
    DEADKEY,// e0 cc
    DEADKEY,// e0 cd
    DEADKEY,// e0 ce
    DEADKEY,// e0 cf
    DEADKEY,// e0 d0
    DEADKEY,// e0 d1
    DEADKEY,// e0 d2
    DEADKEY,// e0 d3
    DEADKEY,// e0 d4
    DEADKEY,// e0 d5
    DEADKEY,// e0 d6
    DEADKEY,// e0 d7
    DEADKEY,// e0 d8
    DEADKEY,// e0 d9
    DEADKEY,// e0 da
    DEADKEY,// e0 db
    DEADKEY,// e0 dc
    DEADKEY,// e0 dd
    DEADKEY,// e0 de
    DEADKEY,// e0 df
    DEADKEY,// e0 e0
    DEADKEY,// e0 e1
    DEADKEY,// e0 e2
    DEADKEY,// e0 e3
    DEADKEY,// e0 e4
    DEADKEY,// e0 e5
    DEADKEY,// e0 e6
    DEADKEY,// e0 e7
    DEADKEY,// e0 e8
    DEADKEY,// e0 e9
    DEADKEY,// e0 ea
    DEADKEY,// e0 eb
    DEADKEY,// e0 ec
    DEADKEY,// e0 ed
```

```
    DEADKEY,// e0 ee
    DEADKEY,// e0 ef
    DEADKEY,// e0 f0 // Note: codes e0f0 through e0ff are reserved for ACPI callbac
k
    DEADKEY,// e0 f1
    DEADKEY,// e0 f2
    DEADKEY,// e0 f3
    DEADKEY,// e0 f4
    DEADKEY,// e0 f5
    DEADKEY,// e0 f6
    DEADKEY,// e0 f7
    DEADKEY,// e0 f8
    DEADKEY,// e0 f9
    DEADKEY,// e0 fa
    DEADKEY,// e0 fb
    DEADKEY,// e0 fc
    DEADKEY,// e0 fd
    DEADKEY,// e0 fe
    DEADKEY // e0 ff // End reserved
};

////////////////////////////////////////////////////////////////////////////////
//
//
// high-byte of flags are (bit number + 1) for modifier key tracking
//  1: left control
//  2: right control
//  3: left shift
//  4: right shift
//  5: left alt
//  6: right alt
//  7: left windows
//  8: right windows
//  9: left Fn (e0 63 on Lenovo u430)
// 10: windows context menu (usually on right)
//
// low-byte is used for other purposes
//  bit 0: breakless bit (set by "PS2 Breakless"
//

#define kMaskLeftControl    0x0001
#define kMaskRightControl   0x0002
#define kMaskLeftShift      0x0004
#define kMaskRightShift     0x0008
#define kMaskLeftAlt        0x0010
#define kMaskRightAlt       0x0020
#define kMaskLeftWindows    0x0040
#define kMaskRightWindows   0x0080
#define kMaskLeftFn         0x0100
#define kMaskWindowsContext 0x0200
```

```
static const UInt16 _PS2flagsStock[ADB_CONVERTER_LEN] =
{
    // flags/modifier key        AT  ANSI Key-Legend
    0x00,   // 00
    0x00,   // 01  Escape
    0x00,   // 02  1!
    0x00,   // 03  2@
    0x00,   // 04  3#
    0x00,   // 05  4$
    0x00,   // 06  5%
    0x00,   // 07  6^
    0x00,   // 08  7&
    0x00,   // 09  8*
    0x00,   // 0a  9(
    0x00,   // 0b  0)
    0x00,   // 0c  -_
    0x00,   // 0d  =+
    0x00,   // 0e  Backspace
    0x00,   // 0f  Tab
    0x00,   // 10  qQ
    0x00,   // 11  wW
    0x00,   // 12  eE
    0x00,   // 13  rR
    0x00,   // 14  tT
    0x00,   // 15  yY
    0x00,   // 16  uU
    0x00,   // 17  iI
    0x00,   // 18  oO
    0x00,   // 19  pP
    0x00,   // 1a  [{
    0x00,   // 1b  ]}
    0x00,   // 1c  Return
    0x0100, // 1d  Left Control
    0x00,   // 1e  aA
    0x00,   // 1f  sS
    0x00,   // 20  dD
    0x00,   // 21  fF
    0x00,   // 22  gG
    0x00,   // 23  hH
    0x00,   // 24  jJ
    0x00,   // 25  kK
    0x00,   // 26  lL
    0x00,   // 27  ;:
    0x00,   // 28  '"
    0x00,   // 29  `~
    0x0300, // 2a  Left Shift
    0x00,   // 2b  \| , Europe 1(ISO)
    0x00,   // 2c  zZ
    0x00,   // 2d  xX
    0x00,   // 2e  cC
    0x00,   // 2f  vV
```

```
    0x00,   // 30  bB
    0x00,   // 31  nN
    0x00,   // 32  mM
    0x00,   // 33  ,<
    0x00,   // 34  .>
    0x00,   // 35  /?
    0x0400, // 36  Right Shift
    0x00,   // 37  Keypad *
    0x0500, // 38  Left Alt
    0x00,   // 39  Space
    0x00,   // 3a  Caps Lock
    0x00,   // 3b  F1
    0x00,   // 3c  F2
    0x00,   // 3d  F3
    0x00,   // 3e  F4
    0x00,   // 3f  F5
    0x00,   // 40  F6
    0x00,   // 41  F7
    0x00,   // 42  F8
    0x00,   // 43  F9
    0x00,   // 44  F10
    0x00,   // 45  Num Lock
    0x00,   // 46  Scroll Lock
    0x00,   // 47  Keypad 7 Home
    0x00,   // 48  Keypad 8 Up
    0x00,   // 49  Keypad 9 PageUp
    0x00,   // 4a  Keypad -
    0x00,   // 4b  Keypad 4 Left
    0x00,   // 4c  Keypad 5
    0x00,   // 4d  Keypad 6 Right
    0x00,   // 4e  Keypad +
    0x00,   // 4f  Keypad 1 End
    0x00,   // 50  Keypad 2 Down
    0x00,   // 51  Keypad 3 PageDn
    0x00,   // 52  Keypad 0 Insert
    0x00,   // 53  Keypad . Delete
    0x00,   // 54  SysReq
    0x00,   // 55
    0x00,   // 56  Europe 2(ISO)
    0x00,   // 57  F11
    0x00,   // 58  F12
    0x00,   // 59  Keypad =
    0x00,   // 5a
    0x00,   // 5b
    0x00,   // 5c  Keyboard Int'l 6 (PC9800 Keypad , )
    0x00,   // 5d
    0x00,   // 5e
    0x00,   // 5f
    0x00,   // 60
    0x00,   // 61
    0x00,   // 62
```

```
    0x00,   // 63
    0x00,   // 64  F13
    0x00,   // 65  F14
    0x00,   // 66  F15
    0x00,   // 67  F16
    0x00,   // 68  F17
    0x00,   // 69  F18
    0x00,   // 6a  F19
    0x00,   // 6b  F20
    0x00,   // 6c  F21
    0x00,   // 6d  F22
    0x00,   // 6e  F23
    0x00,   // 6f
    0x00,   // 70  Keyboard Intl'2 (Japanese Katakana/Hiragana)
    0x00,   // 71
    0x00,   // 72
    0x00,   // 73  Keyboard Int'l 1 (Japanese Ro)
    0x00,   // 74
    0x00,   // 75
    0x00,   // 76  F24 , Keyboard Lang 5 (Japanese Zenkaku/Hankaku)
    0x00,   // 77  Keyboard Lang 4 (Japanese Hiragana)
    0x00,   // 78  Keyboard Lang 3 (Japanese Katakana)
    0x00,   // 79  Keyboard Int'l 4 (Japanese Henkan)
    0x00,   // 7a
    0x00,   // 7b  Keyboard Int'l 5 (Japanese Muhenkan)
    0x00,   // 7c
    0x00,   // 7d  Keyboard Int'l 3 (Japanese Yen)
    0x00,   // 7e  Keypad , (Brazilian Keypad .)
    0x00,   // 7f
    0x00,   // 80
    0x00,   // 81
    0x00,   // 82
    0x00,   // 83
    0x00,   // 84
    0x00,   // 85
    0x00,   // 86
    0x00,   // 87
    0x00,   // 88
    0x00,   // 89
    0x00,   // 8a
    0x00,   // 8b
    0x00,   // 8c
    0x00,   // 8d
    0x00,   // 8e
    0x00,   // 8f
    0x00,   // 90
    0x00,   // 91
    0x00,   // 92
    0x00,   // 93
    0x00,   // 94
    0x00,   // 95
```

```
    0x00,   // 96
    0x00,   // 97
    0x00,   // 98
    0x00,   // 99
    0x00,   // 9a
    0x00,   // 9b
    0x00,   // 9c
    0x00,   // 9d
    0x00,   // 9e
    0x00,   // 9f
    0x00,   // a0
    0x00,   // a1
    0x00,   // a2
    0x00,   // a3
    0x00,   // a4
    0x00,   // a5
    0x00,   // a6
    0x00,   // a7
    0x00,   // a8
    0x00,   // a9
    0x00,   // aa
    0x00,   // ab
    0x00,   // ac
    0x00,   // ad
    0x00,   // ae
    0x00,   // af
    0x00,   // b0
    0x00,   // b1
    0x00,   // b2
    0x00,   // b3
    0x00,   // b4
    0x00,   // b5
    0x00,   // b6
    0x00,   // b7
    0x00,   // b8
    0x00,   // b9
    0x00,   // ba
    0x00,   // bb
    0x00,   // bc
    0x00,   // bd
    0x00,   // be
    0x00,   // bf
    0x00,   // c0
    0x00,   // c1
    0x00,   // c2
    0x00,   // c3
    0x00,   // c4
    0x00,   // c5
    0x00,   // c6
    0x00,   // c7
    0x00,   // c8
```

```
    0x00,   // c9
    0x00,   // ca
    0x00,   // cb
    0x00,   // cc
    0x00,   // cd
    0x00,   // ce
    0x00,   // cf
    0x00,   // d0
    0x00,   // d1
    0x00,   // d2
    0x00,   // d3
    0x00,   // d4
    0x00,   // d5
    0x00,   // d6
    0x00,   // d7
    0x00,   // d8
    0x00,   // d9
    0x00,   // da
    0x00,   // db
    0x00,   // dc
    0x00,   // dd
    0x00,   // de
    0x00,   // df
    0x00,   // e0
    0x00,   // e1
    0x00,   // e2
    0x00,   // e3
    0x00,   // e4
    0x00,   // e5
    0x00,   // e6
    0x00,   // e7
    0x00,   // e8
    0x00,   // e9
    0x00,   // ea
    0x00,   // eb
    0x00,   // ec
    0x00,   // ed
    0x00,   // ee
    0x00,   // ef
    0x00,   // f0
    0x00,   // f1*  Keyboard Lang 2 (Korean Hanja)
    0x00,   // f2*  Keyboard Lang 1 (Korean Hangul)
    0x00,   // f3
    0x00,   // f4
    0x00,   // f5
    0x00,   // f6
    0x00,   // f7
    0x00,   // f8
    0x00,   // f9
    0x00,   // fa
    0x00,   // fb
```

```
    0x00,   // fc
    0x00,   // fd
    0x00,   // fe
    0x00,   // ff
    0x00,   // e0 00
    0x00,   // e0 01
    0x00,   // e0 02
    0x00,   // e0 03
    0x00,   // e0 04
    0x00,   // e0 05 dell down
    0x00,   // e0 06 dell up
    0x00,   // e0 07
#ifndef PROBOOK
    0x00,   // e0 08 samsung up
    0x00,   // e0 09 samsung down
#else
    0x00,   // e0 08
    0x00,   // e0 09 Launchpad (hp Fn+F6)
#endif
    0x00,   // e0 0a Mission Control (hp Fn+F5)
    0x00,   // e0 0b
    0x00,   // e0 0c
    0x00,   // e0 0d
    0x00,   // e0 0e
    0x00,   // e0 0f
    0x00,   // e0 10  Scan Previous Track (hp Fn+F10)
    0x00,   // e0 11
    0x00,   // e0 12 hp down (Fn+F2)
    0x00,   // e0 13
    0x00,   // e0 14
    0x00,   // e0 15
    0x00,   // e0 16
    0x00,   // e0 17 hp up (Fn+F3)
    0x00,   // e0 18
    0x00,   // e0 19  Scan Next Track (hp Fn+F12)
    0x00,   // e0 1a
    0x00,   // e0 1b
    0x00,   // e0 1c  Keypad Enter
    0x0200, // e0 1d  Right Control
    0x00,   // e0 1e
    0x00,   // e0 1f
    0x00,   // e0 20  Mute (hp Fn+F7)
    0x00,   // e0 21  Calculator
    0x00,   // e0 22  Play/Pause (hp Fn+F11)
    0x00,   // e0 23
    0x00,   // e0 24  Stop
    0x00,   // e0 25
    0x00,   // e0 26
    0x00,   // e0 27  Fn+fkeys/fkeys toggle alternate (default Ctrl+e037)
    0x00,   // e0 28
    0x00,   // e0 29
```

```
    0x00,   // e0 2a
    0x00,   // e0 2b
    0x00,   // e0 2c
    0x00,   // e0 2d
    0x00,   // e0 2e  Volume Down (hp Fn+F8)
    0x00,   // e0 2f
    0x00,   // e0 30  Volume Up (hp Fn+F9)
    0x00,   // e0 31
    0x00,   // e0 32  WWW Home
    0x00,   // e0 33
    0x00,   // e0 34
    0x00,   // e0 35  Keypad /
    0x00,   // e0 36
    0x00,   // e0 37  Print Screen
    0x0600, // e0 38  Right Alt
    0x00,   // e0 39
    0x00,   // e0 3a
    0x00,   // e0 3b
    0x00,   // e0 3c
    0x00,   // e0 3d
    0x00,   // e0 3e
    0x00,   // e0 3f
    0x00,   // e0 40
    0x00,   // e0 41
    0x00,   // e0 42
    0x00,   // e0 43
    0x00,   // e0 44
    0x00,   // e0 45* Pause
    0x00,   // e0 46* Break(Ctrl-Pause)
    0x00,   // e0 47  Home
    0x00,   // e0 48  Up Arrow
    0x00,   // e0 49  Page Up
    0x00,   // e0 4a
    0x00,   // e0 4b  Left Arrow
    0x00,   // e0 4c
    0x00,   // e0 4d  Right Arrow
    0x00,   // e0 4e acer up
    0x00,   // e0 4f  End
    0x00,   // e0 50  Down Arrow
    0x00,   // e0 51  Page Down
    0x00,   // e0 52  Insert = Eject
    0x00,   // e0 53  Delete
    0x00,   // e0 54
    0x00,   // e0 55
    0x00,   // e0 56
    0x00,   // e0 57
    0x00,   // e0 58
    0x00,   // e0 59 acer up for my acer
    0x00,   // e0 5a
    0x0700, // e0 5b  Left GUI(Windows)
    0x0800, // e0 5c  Right GUI(Windows)
```

```
    0x0a00, // e0 5d  App( Windows context menu key )
    0x00,   // e0 5e  System Power / Keyboard Power
    0x00,   // e0 5f  System Sleep (hp Fn+F1)
    0x00,   // e0 60
    0x00,   // e0 61
    0x00,   // e0 62
    0x0900, // e0 63  System Wake (Fn on Lenovo u430)
    0x00,   // e0 64
    0x00,   // e0 65  WWW Search
    0x00,   // e0 66  WWW Favorites
    0x00,   // e0 67  WWW Refresh
    0x00,   // e0 68  WWW Stop
    0x00,   // e0 69  WWW Forward
    0x00,   // e0 6a  WWW Back
    0x00,   // e0 6b  My Computer
    0x00,   // e0 6c  Mail
    0x00,   // e0 6d  Media Select
#ifndef PROBOOK
    0x00,   // e0 6e acer up
    0x00,   // e0 6f acer down
#else
    0x00,   // e0 6e  Video Mirror = hp Fn+F4
    0x00,   // e0 6f  Fn+Home
#endif
    0x00,   // e0 70
    0x00,   // e0 71
    0x00,   // e0 72
    0x00,   // e0 73
    0x00,   // e0 74
    0x00,   // e0 75
    0x00,   // e0 76
#ifndef PROBOOK
    0x00,   // e0 77 lg down
    0x00,   // e0 78 lg up
#else
    0x00,   // e0 77
    0x00,   // e0 78 WiFi on/off button on HP ProBook
#endif
    0x00,   // e0 79
    0x00,   // e0 7a
    0x00,   // e0 7b
    0x00,   // e0 7c
    0x00,   // e0 7d
    0x00,   // e0 7e
    0x00,   // e0 7f
    0x00,   // e0 80
    0x00,   // e0 81
    0x00,   // e0 82
    0x00,   // e0 83
    0x00,   // e0 84
    0x00,   // e0 85
```

```
    0x00,   // e0 86
    0x00,   // e0 87
    0x00,   // e0 88
    0x00,   // e0 89
    0x00,   // e0 8a
    0x00,   // e0 8b
    0x00,   // e0 8c
    0x00,   // e0 8d
    0x00,   // e0 8e
    0x00,   // e0 8f
    0x00,   // e0 90
    0x00,   // e0 91
    0x00,   // e0 92
    0x00,   // e0 93
    0x00,   // e0 94
    0x00,   // e0 95
    0x00,   // e0 96
    0x00,   // e0 97
    0x00,   // e0 98
    0x00,   // e0 99
    0x00,   // e0 9a
    0x00,   // e0 9b
    0x00,   // e0 9c
    0x00,   // e0 9d
    0x00,   // e0 9e
    0x00,   // e0 9f
    0x00,   // e0 a0
    0x00,   // e0 a1
    0x00,   // e0 a2
    0x00,   // e0 a3
    0x00,   // e0 a4
    0x00,   // e0 a5
    0x00,   // e0 a6
    0x00,   // e0 a7
    0x00,   // e0 a8
    0x00,   // e0 a9
    0x00,   // e0 aa
    0x00,   // e0 ab
    0x00,   // e0 ac
    0x00,   // e0 ad
    0x00,   // e0 ae
    0x00,   // e0 af
    0x00,   // e0 b0
    0x00,   // e0 b1
    0x00,   // e0 b2
    0x00,   // e0 b3
    0x00,   // e0 b4
    0x00,   // e0 b5
    0x00,   // e0 b6
    0x00,   // e0 b7
    0x00,   // e0 b8
```

```
    0x00,   // e0 b9
    0x00,   // e0 ba
    0x00,   // e0 bb
    0x00,   // e0 bc
    0x00,   // e0 bd
    0x00,   // e0 be
    0x00,   // e0 bf
    0x00,   // e0 c0
    0x00,   // e0 c1
    0x00,   // e0 c2
    0x00,   // e0 c3
    0x00,   // e0 c4
    0x00,   // e0 c5
    0x00,   // e0 c6
    0x00,   // e0 c7
    0x00,   // e0 c8
    0x00,   // e0 c9
    0x00,   // e0 ca
    0x00,   // e0 cb
    0x00,   // e0 cc
    0x00,   // e0 cd
    0x00,   // e0 ce
    0x00,   // e0 cf
    0x00,   // e0 d0
    0x00,   // e0 d1
    0x00,   // e0 d2
    0x00,   // e0 d3
    0x00,   // e0 d4
    0x00,   // e0 d5
    0x00,   // e0 d6
    0x00,   // e0 d7
    0x00,   // e0 d8
    0x00,   // e0 d9
    0x00,   // e0 da
    0x00,   // e0 db
    0x00,   // e0 dc
    0x00,   // e0 dd
    0x00,   // e0 de
    0x00,   // e0 df
    0x00,   // e0 e0
    0x00,   // e0 e1
    0x00,   // e0 e2
    0x00,   // e0 e3
    0x00,   // e0 e4
    0x00,   // e0 e5
    0x00,   // e0 e6
    0x00,   // e0 e7
    0x00,   // e0 e8
    0x00,   // e0 e9
    0x00,   // e0 ea
    0x00,   // e0 eb
```

```
    0x00,   // e0 ec
    0x00,   // e0 ed
    0x00,   // e0 ee
    0x00,   // e0 ef
    0x00,   // e0 f0 // Note: codes e0f0 through e0ff are reserved for ACPI callbac
k
    0x00,   // e0 f1
    0x00,   // e0 f2
    0x00,   // e0 f3
    0x00,   // e0 f4
    0x00,   // e0 f5
    0x00,   // e0 f6
    0x00,   // e0 f7
    0x00,   // e0 f8
    0x00,   // e0 f9
    0x00,   // e0 fa
    0x00,   // e0 fb
    0x00,   // e0 fc
    0x00,   // e0 fd
    0x00,   // e0 fe
    0x00,   // e0 ff // End reserved
};

#endif /* !_APPLEPS2TOADBMAP_H */
```

```
//
DefinitionBlock ("", "SSDT", 2, "ACDT", "RMCF", 0)
{
    External(_SB.PCI0.LPCB.PS2K, DeviceObj)
    Scope (_SB.PCI0.LPCB.PS2K)
    {
        Name (RMCF,Package()
        {
            "Mouse", Package ()
            {
                "ActLikeTrackpad",
                ">y"
            }
        })
    }
}
//EOF
```

```
//
DefinitionBlock ("", "SSDT", 2, "ACDT", "RMCF", 0)
{
    External(_SB.PCI0.LPCB.PS2K, DeviceObj)
    Scope (_SB.PCI0.LPCB.PS2K)
    {
        Name (RMCF,Package()
        {
            "Keyboard", Package()
            {
                "Custom PS2 Map", Package()
                {
                    Package(){},
                    "1e=2c",
                },

                //or
                /*
                "Custom ADB Map", Package()
                {
                    Package(){},
                    "1e=06",
                }
                */
            },
        })
    }
}
//EOF
```

```
//
DefinitionBlock ("", "SSDT", 2, "ACDT", "RMCF", 0)
{
    External(_SB.PCI0.LPCB.PS2K, DeviceObj)
    Scope (_SB.PCI0.LPCB.PS2K)
    {
        Name (RMCF,Package()
        {
            "Mouse", Package ()
            {
                "ActLikeTrackpad",
                ">y"
            },

            "Keyboard", Package()
            {
                "Custom PS2 Map", Package()
                {
                    Package(){},
                    "46=0",    // disable Fn+S and F6
                    "e045=0",  // disable Fn+B
                    "e037=64", // PrtSc=F13
                    "57=65",   // F11=F14
                    "58=66",   // F12=F15
                },
            }
        })
    }
}
//EOF
```

```
//
DefinitionBlock ("", "SSDT", 2, "ACDT", "RMCF", 0)
{
    External(_SB.PCI0.LPCB.PS2K, DeviceObj)
    Scope (_SB.PCI0.LPCB.PS2K)
    {
        Name (RMCF,Package()
        {
            "Keyboard", Package()
            {
                "Custom PS2 Map", Package()
                {
                    Package(){},
                    "e037=e022"
                },

                "Custom ADB Map", Package()
                {
                    Package(){},
                    "46=80",
                    "e045=80",
                    "e04e=4d",
                    "e003=42",
                    "e04c=6b",
                    "e054=71"
                }
            }
        })
    }
}
//EOF
```

电池热补丁收录及方法指导

电池热补丁收录

本项目收录了部分笔记本的 OpenCore 电池热补丁(如 ThinkPad 系列笔记本)，供大家参考使用

更多已经制作好的电池热补丁可参考 GZ小白 的收录项目

项目页面：https://github.com/GZXiaoBai/Hackintosh-Battery-Hotpatch

电池热补丁教程

如果你想了解和学习电池热补丁的制作方法，可前往 XStarDev 的教程网站学习，有问题请去 `Issues` 页面提出（请不要问太过低级的问题，否则一律视为 Invalid）

- 教程页面：
  - https://xstar-dev.github.io/hackintosh_advanced/Guide_For_Battery_Hotpatch.html
  - http://yqp7js.coding-pages.com/2020/05/16/%E8%BF%9B%E9%98%B6%EF%BC%9A%E7%94%B5%E6%B1%A0%E7%83%AD%E8%A1%A5%E4%B8%81%EF%BC%88Battery-Hotpatch%EF%BC%89%E4%B9%8B%E8%B7%AF/
- 问题解答：https://github.com/XStar-Dev/xstar-dev.github.io/issues

Thanks

- Rehabman（编写早期电池补丁教程并开发了 ACPIBatteryManager 电池驱动项目）
- 宪武（ThinkPad 系列笔记本电池补丁示例）
- GZ小白（电池热补丁收录、电池热补丁教程）
- XStarDev（电池热补丁教程）

ThinkPad电池补丁

说明

- 请阅读 `附件` 《ThinkPad 电池系统》。
- 确认 `主电池` 路径是 `\_SB.PCI0. LPC .EC.BAT0` 或者 `\_SB.PCI0. LPCB .EC.BAT0` ，非两者，本章内容仅供参考。
- 以下分 三种情况 说明电池补丁的使用方法。

单电池系统更名和补丁

- 更名：
  - TP 电池基本更名
  - TP 电池 `Mutex` 置 `0` 更名
- 补丁
  - `主电池` 补丁 -- ***SSDT-OCBAT0-TP****

双电池系统一块物理电池更名和补丁

- 更名
  - TP 电池基本更名
  - TP 电池 `Mutex` 置 `0` 更名
  - `BAT1` 禁用更名 `_STA to XSTA`

    注意：请正确使用 `Count` ， `Skip` ， `TableSignature` ，并通过system-DSDT验证 `_STA to XSTA` 位置是否正确
- 补丁
  - `主电池` 补丁 -- ***SSDT-OCBAT0-TP****
  - `BAT1` 禁用补丁 -- ***SSDT-OCBAT1-disable-*** ***`LPC`*** 【或者 ***SSDT-OCBAT1-disable-*** ***`LPCB`*** 】

双电池系统二块物理电池更名和补丁

- 更名
  - TP 电池基本更名
  - TP 电池 `Mutex` 置 `0` 更名
  - `Notify` 更名
- 补丁
  - `主电池` 补丁 --***SSDT-OCBAT0-TP****
  - `BATC` 补丁 -- ***SSDT-OCBATC-TP-*** ***`LPC`*** 【或者 ***SSDT-OCBATC-TP-*** ***`LPCB`*** 、***SSDT-OCBATC-TP-*** ***`_BIX`*** 】
  - `Notify` 补丁 -- ***SSDT-Notify-*** ***`LPC`*** 【或者 ***SSDT-Notify-*** ***`LPCB`*** 】

注意：

- 选用 `BATC 补丁时，7代+机器使用` ***SSDT-OCBATC-TP-`_BIX`***
- 选用 `Notify 补丁时，应当 仔细 检查补丁里的 _Q22 ， _Q24 ， _Q25 ， _Q4A ， _Q4B ， _Q4C ， _Q4D ， BATW ， BFCC 等内容是否与原始 ACPI 对应内容一致，` 如果不一致请修正补丁相应内容。比如：3代机器的 `_Q4C` 的内容和样本内容不同；4代、5代、6代、7代机器无 `_Q4C` ；7代+机器有 `BFCC` 。等等……。样本文件 ***SSDT-Notify-`LPCB`*** 仅适用于T580 。

- 加载顺序
  - `主电池` 补丁
  - `BATC` 补丁
  - `Notify` 补丁

注意事项

- ***SSDT-OCBAT0-TP**** 是 `主电池` 补丁。选用时根据机器型号选择对应的补丁。
- 选用补丁时，应注意 `LPC` 和 `LPCB` 的区别。
- 是否需要 `TP 电池 Mutex 置 0 更名` ，自行尝试。

`Notify` 补丁示例【仅 `Method (_Q22` …部分】

> T580 原文

```
Method (_Q22, 0, NotSerialized)  /* _Qxx: EC Query, xx=0x00-0xFF */
{
    CLPM ()
    If (HB0A)
    {
        Notify (BAT0, 0x80) /* Status Change */
    }

    If (HB1A)
    {
        Notify (BAT1, 0x80) /* Status Change */
    }
}
```

> 重写

```
/*
 * For ACPI Patch:
 * _Q22 to XQ22:
 * Find:    5f51 3232
 * Replace: 5851 3232
 */
Method (_Q22, 0, NotSerialized)  /* _Qxx: EC Query, xx=0x00-0xFF */
```

```
{
    If (_OSI ("Darwin"))
    {
        CLPM ()
        If (HB0A)
        {
            Notify (BATC, 0x80) /* Status Change */
        }

        If (HB1A)
        {
            Notify (BATC, 0x80) /* Status Change */
        }
    }
    Else
    {
        \_SB.PCI0.LPCB.EC.XQ22 ()
    }
}
```

详见 `Notify` 补丁 -- ***SSDT-Notify-*** **`BFCC`**

> 已验证机器为 `ThinkPad T580` ，补丁和更名如下：

- **SSDT-OCBAT0-TP_tx80_x1c6th**
- **SSDT-OCBATC-TP-** **`LPCB`**
- **SSDT-Notify-** **`LPCB`**
- **TP**电池基本更名
- **Notify**更名

附件：ThinkPad电池系统

概述

Thinkpad 电池系统分为单电池系统和双电池系统。

- 双电池系统就是机器配备了二块电池。或者，虽然机器只装备了一块电池，但提供了第二块电池的物理接口以及对应的 ACPI。第二块电池作为可选件，可以后期安装。双电池系统的机器可能有一块电池，也可能有二块电池。
- 单电池系统是机器配备了一块电池，且只有一块电池的 ACPI。
- 比如，T470, T470s 的电池结构属于双电池系统，T470P 的电池结构属于单电池系统。再比如，T430 系列属于双电池系统，机器本身只有一块电池，但可以通过光驱位安装第二块电池。

单，双电池系统判定

- 双电池系统：ACPI 中同时存在 `BAT0` 和 `BAT1`
- 单电池系统：ACPI 中只有 `BAT0` ，无 `BAT1`

```
//
// For ACPI Patch:
// _Q22 to XQ22:
// Find:    5f51 3232
// Replace: 5851 3232
//
// _Q24 to XQ24:
// Find:    5f51 3234
// Replace: 5851 3234
//
// _Q25 to XQ25:
// Find:    5f51 3235
// Replace: 5851 3235
//
// _Q4A to XQ4A:
// Find:    5f51 3441
// Replace: 5851 3441
//
// _Q4B to XQ4B:
// Find:    5f51 3442
// Replace: 5851 3442
//
// _Q4C to XQ4C:
// Find:    5f51 3443
// Replace: 5851 3443
//
// _Q4D to XQ4D:
// Find:    5f51 3444
// Replace: 5851 3444
//
// BATW to XATW:
// Find:    4241 545701
// Replace: 5841 545701
//
DefinitionBlock ("", "SSDT", 2, "OCLT", "NTFY", 0)
{
    External (\_SB.PCI0.LPC.EC, DeviceObj)
    External (\_SB.PCI0.LPC.EC.BATC, DeviceObj)
    //
    External (\_SB.PCI0.LPC.EC.BAT1.XB1S, IntObj)
    External (\_SB.PCI0.LPC.EC.BAT1.B1ST, IntObj)
    External (\_SB.PCI0.LPC.EC.BAT1.SBLI, IntObj)
    //
    External (\_SB.PCI0.LPC.EC.CLPM, MethodObj)
    External (\_SB.PCI0.LPC.EC.HKEY.MHKQ, MethodObj)
    //
    External (\BT2T, FieldUnitObj)
    External (\_SB.PCI0.LPC.EC.SLUL, FieldUnitObj)
    External (\_SB.PCI0.LPC.EC.HB0A, FieldUnitObj)
    External (\_SB.PCI0.LPC.EC.HB1A, FieldUnitObj)
```

```
    //
    External (\_SB.PCI0.LPC.EC.XQ22, MethodObj)
    External (\_SB.PCI0.LPC.EC.XQ24, MethodObj)
    External (\_SB.PCI0.LPC.EC.XQ25, MethodObj)
    External (\_SB.PCI0.LPC.EC.XQ4A, MethodObj)
    External (\_SB.PCI0.LPC.EC.XQ4B, MethodObj)
    External (\_SB.PCI0.LPC.EC.XQ4C, MethodObj)
    External (\_SB.PCI0.LPC.EC.XQ4D, MethodObj)
    External (\_SB.PCI0.LPC.EC.XATW, MethodObj)

    Scope (\_SB.PCI0.LPC.EC)
    {
        Method (_Q22, 0, NotSerialized)  // _Qxx: EC Query, xx=0x00-0xFF
        {
            If (_OSI ("Darwin"))
            {
                CLPM ()
                If (HB0A)
                {
                    Notify (BATC, 0x80) // Status Change
                }

                If (HB1A)
                {
                    Notify (BATC, 0x80) // Status Change
                }
            }
            Else
            {
                \_SB.PCI0.LPC.EC.XQ22 ()
            }
        }

        Method (_Q24, 0, NotSerialized)  // _Qxx: EC Query, xx=0x00-0xFF
        {
            If (_OSI ("Darwin"))
            {
                CLPM ()
                Notify (BATC, 0x80) // Status Change
            }
            Else
            {
                \_SB.PCI0.LPC.EC.XQ24 ()
            }
        }

        Method (_Q25, 0, NotSerialized)  // _Qxx: EC Query, xx=0x00-0xFF
        {
            If (_OSI ("Darwin"))
            {
                If ((^BAT1.B1ST & ^BAT1.XB1S))
```

```
                {
                    CLPM ()
                    Notify (BATC, 0x80) // Status Change
                }
            }
            Else
            {
                \_SB.PCI0.LPC.EC.XQ25 ()
            }
        }

        Method (_Q4A, 0, NotSerialized)  // _Qxx: EC Query, xx=0x00-0xFF
        {
            If (_OSI ("Darwin"))
            {
                CLPM ()
                Notify (BATC, 0x81) // Information Change
            }
            Else
            {
                \_SB.PCI0.LPC.EC.XQ4A ()
            }
        }

        Method (_Q4B, 0, NotSerialized)  // _Qxx: EC Query, xx=0x00-0xFF
        {
            If (_OSI ("Darwin"))
            {
                CLPM ()
                Notify (BATC, 0x80) // Status Change
            }
            Else
            {
                \_SB.PCI0.LPC.EC.XQ4B ()
            }
        }

        Method (_Q4C, 0, NotSerialized)  // _Qxx: EC Query, xx=0x00-0xFF
        {
            If (_OSI ("Darwin"))
            {
                \_SB.PCI0.LPC.EC.CLPM ()
                If (\_SB.PCI0.LPC.EC.HB1A)
                {
                    \_SB.PCI0.LPC.EC.HKEY.MHKQ (0x4010)
                    Notify (\_SB.PCI0.LPC.EC.BATC, 0x01) // Device Check
                }
                Else
                {
                    \_SB.PCI0.LPC.EC.HKEY.MHKQ (0x4011)
                    If (\_SB.PCI0.LPC.EC.BAT1.XB1S)
```

```
                {
                    Notify (\_SB.PCI0.LPC.EC.BATC, 0x03) // Eject Request
                }
            }
        }
        Else
        {
            \_SB.PCI0.LPC.EC.XQ4C ()
        }
    }

    Method (_Q4D, 0, NotSerialized)  // _Qxx: EC Query, xx=0x00-0xFF
    {
        If (_OSI ("Darwin"))
        {
            CLPM ()
            If (\BT2T)
            {
                If ((^BAT1.SBLI == 0x01))
                {
                    Sleep (0x0A)
                    If ((HB1A && (SLUL == 0x00)))
                    {
                        ^BAT1.XB1S = 0x01
                        Notify (\_SB.PCI0.LPC.EC.BATC, 0x01) // Device Check
                    }
                }
                ElseIf ((SLUL == 0x01))
                {
                    ^BAT1.XB1S = 0x00
                    Notify (\_SB.PCI0.LPC.EC.BATC, 0x03) // Eject Request
                }
            }

            If ((^BAT1.B1ST & ^BAT1.XB1S))
            {
                Notify (BATC, 0x80) // Status Change
            }
        }
        Else
        {
            \_SB.PCI0.LPC.EC.XQ4D ()
        }
    }

    Method (BATW, 1, NotSerialized)
    {
        If (_OSI ("Darwin"))
        {
            If (\BT2T)
            {
```

```
                    Local0 = \_SB.PCI0.LPC.EC.BAT1.XB1S
                    If ((HB1A && !SLUL))
                    {
                        Local1 = 0x01
                    }
                    Else
                    {
                        Local1 = 0x00
                    }

                    If ((Local0 ^ Local1))
                    {
                        \_SB.PCI0.LPC.EC.BAT1.XB1S = Local1
                        Notify (\_SB.PCI0.LPC.EC.BATC, 0x01) // Device Check
                    }
                }
            }
            Else
            {
                \_SB.PCI0.LPC.EC.XATW (Arg0)
            }
        }
    }
}
```

```
//
// For ACPI Patch:
// _Q22 to XQ22:
// Find:    5f51 3232
// Replace: 5851 3232
//
// _Q24 to XQ24:
// Find:    5f51 3234
// Replace: 5851 3234
//
// _Q25 to XQ25:
// Find:    5f51 3235
// Replace: 5851 3235
//
// _Q4A to XQ4A:
// Find:    5f51 3441
// Replace: 5851 3441
//
// _Q4B to XQ4B:
// Find:    5f51 3442
// Replace: 5851 3442
//
// _Q4C to XQ4C:
// Find:    5f51 3443
// Replace: 5851 3443
//
// _Q4D to XQ4D:
// Find:    5f51 3444
// Replace: 5851 3444
//
// BATW to XATW:
// Find:    4241 545701
// Replace: 5841 545701
//
// BFCC to XFCC:
// Find:    42464343 00
// Replace: 58464343 00
//
DefinitionBlock ("", "SSDT", 2, "OCLT", "NTFY", 0)
{
    External (\_SB.PCI0.LPCB.EC, DeviceObj)
    External (\_SB.PCI0.LPCB.EC.BATC, DeviceObj)
    //
    External (\_SB.PCI0.LPCB.EC.BAT1.XB1S, IntObj)
    External (\_SB.PCI0.LPCB.EC.BAT1.SBLI, IntObj)
    External (\_SB.PCI0.LPCB.EC.BAT0.B0ST, IntObj)
    External (\_SB.PCI0.LPCB.EC.BAT1.B1ST, IntObj)
    //
    External (\_SB.PCI0.LPCB.EC.CLPM, MethodObj)
    External (\_SB.PCI0.LPCB.EC.HKEY.MHKQ, MethodObj)
```

```
    //
    External (\BT2T, FieldUnitObj)
    External (\_SB.PCI0.LPCB.EC.SLUL, FieldUnitObj)
    External (\_SB.PCI0.LPCB.EC.HB0A, FieldUnitObj)
    External (\_SB.PCI0.LPCB.EC.HB1A, FieldUnitObj)
    //
    External (\_SB.PCI0.LPCB.EC.XQ22, MethodObj)
    External (\_SB.PCI0.LPCB.EC.XQ24, MethodObj)
    External (\_SB.PCI0.LPCB.EC.XQ25, MethodObj)
    External (\_SB.PCI0.LPCB.EC.XQ4A, MethodObj)
    External (\_SB.PCI0.LPCB.EC.XQ4B, MethodObj)
    External (\_SB.PCI0.LPCB.EC.XQ4C, MethodObj)
    External (\_SB.PCI0.LPCB.EC.XQ4D, MethodObj)
    External (\_SB.PCI0.LPCB.EC.XATW, MethodObj)
    External (\_SB.PCI0.LPCB.EC.XFCC, MethodObj)

    Scope (\_SB.PCI0.LPCB.EC)
    {
        Method (_Q22, 0, NotSerialized)  // _Qxx: EC Query, xx=0x00-0xFF
        {
            If (_OSI ("Darwin"))
            {
                CLPM ()
                If (HB0A)
                {
                    Notify (BATC, 0x80) // Status Change
                }

                If (HB1A)
                {
                    Notify (BATC, 0x80) // Status Change
                }
            }
            Else
            {
                \_SB.PCI0.LPCB.EC.XQ22 ()
            }
        }

        Method (_Q24, 0, NotSerialized)  // _Qxx: EC Query, xx=0x00-0xFF
        {
            If (_OSI ("Darwin"))
            {
                CLPM ()
                Notify (BATC, 0x80) // Status Change
            }
            Else
            {
                \_SB.PCI0.LPCB.EC.XQ24 ()
            }
        }
```

```
        Method (_Q25, 0, NotSerialized)  // _Qxx: EC Query, xx=0x00-0xFF
        {
            If (_OSI ("Darwin"))
            {
                If ((^BAT1.B1ST & ^BAT1.XB1S))
                {
                    CLPM ()
                    Notify (BATC, 0x80) // Status Change
                }
            }
            Else
            {
                \_SB.PCI0.LPCB.EC.XQ25 ()
            }
        }

        Method (_Q4A, 0, NotSerialized)  // _Qxx: EC Query, xx=0x00-0xFF
        {
            If (_OSI ("Darwin"))
            {
                CLPM ()
                Notify (BATC, 0x81) // Information Change
            }
            Else
            {
                \_SB.PCI0.LPCB.EC.XQ4A ()
            }
        }

        Method (_Q4B, 0, NotSerialized)  // _Qxx: EC Query, xx=0x00-0xFF
        {
            If (_OSI ("Darwin"))
            {
                CLPM ()
                Notify (BATC, 0x80) // Status Change
            }
            Else
            {
                \_SB.PCI0.LPCB.EC.XQ4B ()
            }
        }

        Method (_Q4C, 0, NotSerialized)  // _Qxx: EC Query, xx=0x00-0xFF
        {
            If (_OSI ("Darwin"))
            {
                \_SB.PCI0.LPCB.EC.CLPM ()
                If (\_SB.PCI0.LPCB.EC.HB1A)
                {
                    \_SB.PCI0.LPCB.EC.HKEY.MHKQ (0x4010)
```

```
                    Notify (\_SB.PCI0.LPCB.EC.BATC, 0x01) // Device Check
                }
                Else
                {
                    \_SB.PCI0.LPCB.EC.HKEY.MHKQ (0x4011)
                    If (\_SB.PCI0.LPCB.EC.BAT1.XB1S)
                    {
                        Notify (\_SB.PCI0.LPCB.EC.BATC, 0x03) // Eject Request
                    }
                }
            }
            Else
            {
                \_SB.PCI0.LPCB.EC.XQ4C ()
            }
        }

        Method (_Q4D, 0, NotSerialized)  // _Qxx: EC Query, xx=0x00-0xFF
        {
            If (_OSI ("Darwin"))
            {
                CLPM ()
                If (\BT2T)
                {
                    If ((^BAT1.SBLI == 0x01))
                    {
                        Sleep (0x0A)
                        If ((HB1A && (SLUL == 0x00)))
                        {
                            ^BAT1.XB1S = 0x01
                            Notify (\_SB.PCI0.LPCB.EC.BATC, 0x01) // Device Check
                        }
                    }
                    ElseIf ((SLUL == 0x01))
                    {
                        ^BAT1.XB1S = 0x00
                        Notify (\_SB.PCI0.LPCB.EC.BATC, 0x03) // Eject Request
                    }
                }

                If ((^BAT1.B1ST & ^BAT1.XB1S))
                {
                    Notify (BATC, 0x80) // Status Change
                }
            }
            Else
            {
                \_SB.PCI0.LPCB.EC.XQ4D ()
            }
        }
```

```
        Method (BATW, 1, NotSerialized)
        {
            If (_OSI ("Darwin"))
            {
                If (\BT2T)
                {
                    Local0 = \_SB.PCI0.LPCB.EC.BAT1.XB1S
                    If ((HB1A && !SLUL))
                    {
                        Local1 = 0x01
                    }
                    Else
                    {
                        Local1 = 0x00
                    }

                    If ((Local0 ^ Local1))
                    {
                        \_SB.PCI0.LPCB.EC.BAT1.XB1S = Local1
                        Notify (\_SB.PCI0.LPCB.EC.BATC, 0x01) // Device Check
                    }
                }
            }
            Else
            {
                \_SB.PCI0.LPCB.EC.XATW (Arg0)
            }
        }

        Method (BFCC, 0, NotSerialized)
        {
            If (_OSI ("Darwin"))
            {
                If (\_SB.PCI0.LPCB.EC.BAT0.B0ST)
                {
                    Notify (BATC, 0x81)
                }

                If (\_SB.PCI0.LPCB.EC.BAT1.B1ST)
                {
                    Notify (BATC, 0x81)
                }
            }
            Else
            {
                \_SB.PCI0.LPCB.EC.XFCC ()
            }
        }
    }
}
```

```
// battery
DefinitionBlock ("", "SSDT", 2, "OCLT", "BAT0", 0)
{
    External(_SB.PCI0.LPCB.EC0, DeviceObj)
    External(_SB.PCI0.LPCB.EC0.BATM, MutexObj)
    External(_SB.PCI0.LPCB.EC0.HIID, FieldUnitObj)
    External(_SB.PCI0.LPCB.EC0.SBCM, FieldUnitObj)
    //
    External(_SB.PCI0.LPCB.EC0.XBIF, MethodObj)
    External(_SB.PCI0.LPCB.EC0.XBST, MethodObj)

    Method (B1B2, 2, NotSerialized)
    {
        ShiftLeft (Arg1, 8, Local0)
        Or (Arg0, Local0, Local0)
        Return (Local0)

    }
    Method (B1B4, 4, NotSerialized)
    {
        Store (Arg3, Local0)
        Or (Arg2, ShiftLeft (Local0, 0x08), Local0)
        Or (Arg1, ShiftLeft (Local0, 0x08), Local0)
        Or (Arg0, ShiftLeft (Local0, 0x08), Local0)
        Return (Local0)
    }
    Scope(\_SB.PCI0.LPCB.EC0)
    {
        Method (RE1B, 1, NotSerialized)
        {
            OperationRegion(ERAM, EmbeddedControl, Arg0, 1)
            Field(ERAM, ByteAcc, NoLock, Preserve) { BYTE, 8 }
            Return(BYTE)
        }
        Method (RECB, 2, Serialized)
        {
            ShiftRight(Arg1, 3, Arg1)
            Name(TEMP, Buffer(Arg1) { })
            Add(Arg0, Arg1, Arg1)
            Store(0, Local0)
            While (LLess(Arg0, Arg1))
            {
                Store(RE1B(Arg0), Index(TEMP, Local0))
                Increment(Arg0)
                Increment(Local0)
            }
            Return(TEMP)
        }
        OperationRegion (BRAM, EmbeddedControl, 0x00, 0x0100)
        Field (BRAM, ByteAcc, NoLock, Preserve)
```

```
        {
            Offset (0xA0),
            BRCA,8,BRCB,8,      //SBRC,   16,
            BFC0,8,BFC1,8,      //SBFC,   16,s2
                                //SBAE,   16,
                                //SBRS,   16,
            Offset (0xA8),
            BAC0,8,BAC1,8,      //SBAC,   16,s2
            BVO0,8,BVO1,8,      //SBVO,   16,
                                //SBAF,   16,
                                //SBBS,   16
        }
        Field (BRAM, ByteAcc, NoLock, Preserve)
        {
            Offset (0xA0),
            BDC0,8,BDC1,8,      //SBDC,   16,s2
            BDV0,8,BDV1,8,      //SBDV,   16,s2
                                //SBOM,   16,
                                //SBSI,   16,
                                //SBDT,   16,
            Offset (0xAA),
            BSN0,8,BSN1,8       //SBSN,   16,s2
        }
        Field (BRAM, ByteAcc, NoLock, Preserve)
        {
            Offset (0xA0),
            BCH0,8,BCH1,8,BCH2,8,BCH3,8 //SBCH,   32
        }
        Method (GBIF, 3, NotSerialized)
        {
            If (_OSI ("Darwin"))
            {
                Acquire (BATM, 0xFFFF)
                If (Arg2)
                {
                Or (Arg0, 0x01, ^HIID)
                Sleep (0x14)
                Store (^SBCM, Local7)
                XOr (Local7, 0x01, Index (Arg1, 0x00))
                Store (Arg0, ^HIID)
                Sleep (0x14)
                If (Local7)
                {
                    Multiply (B1B2 (BFC0, BFC1), 0x0A, Local1)
                }
                Else
                {
                    Store (B1B2 (BFC0, BFC1), Local1)
                }

                Store (Local1, Index (Arg1, 0x02))
```

```
                Or (Arg0, 0x02, ^HIID)
                Sleep (0x14)
                If (Local7)
                {
                    Multiply (B1B2 (BDC0, BDC1), 0x0A, Local0)
                }
                Else
                {
                    Store (B1B2 (BDC0, BDC1), Local0)
                }

                Store (Local0, Index (Arg1, 0x01))
                Divide (Local1, 0x14, Local2, Index (Arg1, 0x05))
                If (Local7)
                {
                    Store (0xC8, Index (Arg1, 0x06))
                }
                ElseIf (B1B2 (BDV0, BDV1))
                {
                    Divide (0x00030D40, B1B2 (BDV0, BDV1), Local2, Index (Arg1, 0x0
6))
                }
                Else
                {
                    Store (0x00, Index (Arg1, 0x06))
                }

                Store (B1B2 (BDV0, BDV1), Index (Arg1, 0x04))
                Store (B1B2 (BSN0, BSN1), Local0)
                Name (SERN, Buffer (0x06)
                {
                    "     "
                })
                Store (0x04, Local2)
                While (Local0)
                {
                    Divide (Local0, 0x0A, Local1, Local0)
                    Add (Local1, 0x30, Index (SERN, Local2))
                    Decrement (Local2)
                }

                Store (SERN, Index (Arg1, 0x0A))
                Or (Arg0, 0x06, ^HIID)
                Sleep (0x14)
                Store (RECB(0xA0,128), Index (Arg1, 0x09))
                Or (Arg0, 0x04, ^HIID)
                Sleep (0x14)
                Name (BTYP, Buffer (0x05)
                {
                     0x00, 0x00, 0x00, 0x00, 0x00
                })
```

```
                Store (B1B4 (BCH0, BCH1, BCH2, BCH3), BTYP)
                Store (BTYP, Index (Arg1, 0x0B))
                Or (Arg0, 0x05, ^HIID)
                Sleep (0x14)
                Store (RECB(0xA0,128), Index (Arg1, 0x0C))
                }
                Else
                {
                Store (0xFFFFFFFF, Index (Arg1, 0x01))
                Store (0x00, Index (Arg1, 0x05))
                Store (0x00, Index (Arg1, 0x06))
                Store (0xFFFFFFFF, Index (Arg1, 0x02))
                }

                Release (BATM)
                Return (Arg1)
            }
            Else
            {
                Return (\_SB.PCI0.LPCB.EC0.XBIF(Arg0, Arg1, Arg2))
            }
        }

        Method (GBST, 4, NotSerialized)
        {
            If (_OSI ("Darwin"))
            {
                Acquire (BATM, 0xFFFF)
                If (And (Arg1, 0x20))
                {
                    Store (0x02, Local0)
                }
                ElseIf (And (Arg1, 0x40))
                {
                    Store (0x01, Local0)
                }
                Else
                {
                    Store (0x00, Local0)
                }

                If (And (Arg1, 0x07)) {}
                Else
                {
                    Or (Local0, 0x04, Local0)
                }

                If (LEqual (And (Arg1, 0x07), 0x07))
                {
                    Store (0x04, Local0)
                    Store (0x00, Local1)
```

```
                    Store (0x00, Local2)
                    Store (0x00, Local3)
                }
                Else
                {
                    Sleep (0x32)
                    Store (Arg0, ^HIID)
                    Sleep (0x32)
                    Store (^HIID,Local6)
                    If (LNotEqual (Arg0, Local6))
                    {
                        Release (BATM)
                        Return (Arg3)
                    }

                    Store (B1B2 (BVO0, BVO1), Local3)
                    If (Arg2)
                    {
                        Multiply (B1B2 (BRCA, BRCB), 0x0A, Local2)
                    }
                    Else
                    {
                        Store (B1B2 (BRCA, BRCB), Local2)
                    }

                    Store (B1B2 (BAC0, BAC1), Local1)
                    If (LGreaterEqual (Local1, 0x8000))
                    {
                        If (And (Local0, 0x01))
                        {
                            Subtract (0x00010000, Local1, Local1)
                        }
                        Else
                        {
                            Store (0x00, Local1)
                        }
                    }
                    ElseIf (LNot (And (Local0, 0x02)))
                    {
                        Store (0x00, Local1)
                    }

                    If (Arg2)
                    {
                        Multiply (Local3, Local1, Local1)
                        Divide (Local1, 0x03E8, Local7, Local1)
                        Store (Local0, Local7)
                        Store (Local7, Local0)
                    }
                }
```

```
                Store (Local0, Index (Arg3, 0x00))
                Store (Local1, Index (Arg3, 0x01))
                Store (Local2, Index (Arg3, 0x02))
                Store (Local3, Index (Arg3, 0x03))
                Release (BATM)
                Return (Arg3)
            }
            Else
            {
                Return (\_SB.PCI0.LPCB.EC0.XBST(Arg0, Arg1, Arg2, Arg3))
            }
        }
    }
}
//EOF
```

```
// battery
DefinitionBlock ("", "SSDT", 2, "OCLT", "BAT0", 0)
{
    External(_SB.PCI0.LPCB.EC, DeviceObj)
    External(_SB.PCI0.LPCB.EC.AC._PSR, MethodObj)
    External(_SB.PCI0.LPCB.EC.BSWR, IntObj)
    External(_SB.PCI0.LPCB.EC.BSWA, IntObj)
    External(_SB.PCI0.LPCB.EC.BATM, MutexObj)
    External(_SB.PCI0.LPCB.EC.HIID, FieldUnitObj)
    External(_SB.PCI0.LPCB.EC.B0I0, IntObj)
    External(_SB.PCI0.LPCB.EC.B0I1, IntObj)
    External(_SB.PCI0.LPCB.EC.B0I2, IntObj)
    External(_SB.PCI0.LPCB.EC.B0I3, IntObj)
    External(_SB.PCI0.LPCB.EC.B1I0, IntObj)
    External(_SB.PCI0.LPCB.EC.B1I1, IntObj)
    External(_SB.PCI0.LPCB.EC.B1I2, IntObj)
    External(_SB.PCI0.LPCB.EC.B1I3, IntObj)
    //
    External(_SB.PCI0.LPCB.EC.XBIF, MethodObj)
    External(_SB.PCI0.LPCB.EC.XBIX, MethodObj)
    External(_SB.PCI0.LPCB.EC.XBST, MethodObj)

    Method (B1B2, 2, NotSerialized)
    {
        ShiftLeft (Arg1, 8, Local0)
        Or (Arg0, Local0, Local0)
        Return (Local0)

    }
    Method (B1B4, 4, NotSerialized)
    {
        Store (Arg3, Local0)
        Or (Arg2, ShiftLeft (Local0, 0x08), Local0)
        Or (Arg1, ShiftLeft (Local0, 0x08), Local0)
        Or (Arg0, ShiftLeft (Local0, 0x08), Local0)
        Return (Local0)
    }
    Scope(\_SB.PCI0.LPCB.EC)
    {
        Method (RE1B, 1, NotSerialized)
        {
            OperationRegion(ERAM, EmbeddedControl, Arg0, 1)
            Field(ERAM, ByteAcc, NoLock, Preserve) { BYTE, 8 }
            Return(BYTE)
        }
        Method (RECB, 2, Serialized)
        {
            ShiftRight(Arg1, 3, Arg1)
            Name(TEMP, Buffer(Arg1) { })
            Add(Arg0, Arg1, Arg1)
```

```
            Store(0, Local0)
            While (LLess(Arg0, Arg1))
            {
                Store(RE1B(Arg0), Index(TEMP, Local0))
                Increment(Arg0)
                Increment(Local0)
            }
            Return(TEMP)
        }
        OperationRegion (BRAM, EmbeddedControl, 0x00, 0x0100)
        Field (BRAM, ByteAcc, NoLock, Preserve)
        {
            Offset (0xA0),
            BRCA,8,BRCB,8,      //SBRC,   16,
            BFC0,8,BFC1,8,      //SBFC,   16,s2
                                //SBAE,   16,
                                //SBRS,   16,
            Offset (0xA8),
            BAC0,8,BAC1,8,      //SBAC,   16,s2
            BVO0,8,BVO1,8,      //SBVO,   16,
                                //SBAF,   16,
                                //SBBS,   16
        }
        Field (BRAM, ByteAcc, NoLock, Preserve)
        {
            Offset (0xA0),
            BBM0,8,BBM1,8,      //SBBM,   16,
                                //SBMD,   16,
            Offset (0xA4),
            BC00,8,BC01,8,      //SBCC,   16 //E470,T470S

        }
        Field (BRAM, ByteAcc, NoLock, Preserve)
        {
            Offset (0xA0),
            BDC0,8,BDC1,8,      //SBDC,   16,s2
            BDV0,8,BDV1,8,      //SBDV,   16,s2
                                //SBOM,   16,
                                //SBSI,   16,
                                //SBDT,   16,
            Offset (0xAA),
            BSN0,8,BSN1,8       //SBSN,   16,s2
        }
        Field (BRAM, ByteAcc, NoLock, Preserve)
        {
            Offset (0xA0),
            BCH0,8,BCH1,8,BCH2,8,BCH3,8 //SBCH,   32
        }

        Method (GBIF, 3, NotSerialized)
        {
```

```
            If (_OSI ("Darwin"))
            {
                Acquire (BATM, 0xFFFF)
                If (Arg2)
                {
                    Or (Arg0, 0x01, HIID)
                    Store (B1B2 (BBM0, BBM1), Local7)
                    ShiftRight (Local7, 0x0F, Local7)
                    XOr (Local7, 0x01, Index (Arg1, 0x00))
                    Store (Arg0, HIID)
                    If (Local7)
                    {
                        Multiply (B1B2 (BFC0, BFC1), 0x0A, Local1)
                    }
                    Else
                    {
                        Store (B1B2 (BFC0, BFC1), Local1)
                    }

                    Store (Local1, Index (Arg1, 0x02))
                    Or (Arg0, 0x02, HIID)
                    If (Local7)
                    {
                        Multiply (B1B2 (BDC0, BDC1), 0x0A, Local0)
                    }
                    Else
                    {
                        Store (B1B2 (BDC0, BDC1), Local0)
                    }

                    Store (Local0, Index (Arg1, 0x01))
                    Divide (Local1, 0x14, Local2, Index (Arg1, 0x05))
                    If (Local7)
                    {
                        Store (0xC8, Index (Arg1, 0x06))
                    }
                    ElseIf (B1B2 (BDV0, BDV1))
                    {
                        Divide (0x00030D40, B1B2 (BDV0, BDV1), Local2, Index (Arg1,
0x06))
                    }
                    Else
                    {
                        Store (0x00, Index (Arg1, 0x06))
                    }

                    Store (B1B2 (BDV0, BDV1), Index (Arg1, 0x04))
                    Store (B1B2 (BSN0, BSN1), Local0)
                    Name (SERN, Buffer (0x06)
                    {
                        "     "
```

```
                    })
                    Store (0x04, Local2)
                    While (Local0)
                    {
                        Divide (Local0, 0x0A, Local1, Local0)
                        Add (Local1, 0x30, Index (SERN, Local2))
                        Decrement (Local2)
                    }

                    Store (SERN, Index (Arg1, 0x0A))
                    Or (Arg0, 0x06, HIID)
                    //Arg1 [0x09] = SBDN
                    Arg1 [0x09] = RECB(0xA0,128)
                    Or (Arg0, 0x04, HIID)
                    Name (BTYP, Buffer (0x05)
                    {
                         0x00, 0x00, 0x00, 0x00, 0x00
                    })
                    Store (B1B4 (BCH0, BCH1, BCH2, BCH3), BTYP)
                    Store (BTYP, Index (Arg1, 0x0B))
                    Or (Arg0, 0x05, HIID)
                    //Arg1 [0x0C] = SBMN
                    Arg1 [0x0C] = RECB(0xA0,128)
                }
                Else
                {
                    Arg1 [One] = 0xFFFFFFFF
                    Arg1 [0x05] = Zero
                    Arg1 [0x06] = Zero
                    Arg1 [0x02] = 0xFFFFFFFF
                }

                Release (BATM)
                Return (Arg1)
            }
            Else
            {
                Return (\_SB.PCI0.LPCB.EC.XBIF(Arg0, Arg1, Arg2))
            }
        }

        Method (GBIX, 3, NotSerialized)
        {
            If (_OSI ("Darwin"))
            {
                Acquire (BATM, 0xFFFF)
                If (Arg2)
                {
                    Or (Arg0, 0x01, HIID)
                    Store (B1B2 (BC00, BC01), Local7)
                    Store (Local7, Index (Arg1, 0x08))
```

```
                    Store (B1B2 (BBM0, BBM1), Local7)
                    ShiftRight (Local7, 0x0F, Local7)
                    XOr (Local7, 0x01, Index (Arg1, 0x01))
                    Store (Arg0, HIID)
                    If (Local7)
                    {
                        Multiply (B1B2 (BFC0, BFC1), 0x0A, Local1)
                    }
                    Else
                    {
                        Store (B1B2 (BFC0, BFC1), Local1)
                    }

                    Store (Local1, Index (Arg1, 0x03))
                    Or (Arg0, 0x02, HIID)
                    If (Local7)
                    {
                        Multiply (B1B2 (BDC0, BDC1), 0x0A, Local0)
                    }
                    Else
                    {
                        Store (B1B2 (BDC0, BDC1), Local0)
                    }

                    Store (Local0, Index (Arg1, 0x02))
                    Divide (Local1, 0x14, Local2, Index (Arg1, 0x06))
                    If (Local7)
                    {
                        Store (0xC8, Index (Arg1, 0x07))
                    }
                    ElseIf (B1B2 (BDV0, BDV1))
                    {
                        Divide (0x00030D40, B1B2 (BDV0, BDV1), Local2, Index (Arg1,
0x07))
                    }
                    Else
                    {
                        Store (0x00, Index (Arg1, 0x07))
                    }

                    Store (B1B2 (BDV0, BDV1), Index (Arg1, 0x05))
                    Store (B1B2 (BSN0, BSN1), Local0)
                    Name (SERN, Buffer (0x06)
                    {
                        "     "
                    })
                    Store (0x04, Local2)
                    While (Local0)
                    {
                        Divide (Local0, 0x0A, Local1, Local0)
                        Add (Local1, 0x30, Index (SERN, Local2))
```

```
                    Decrement (Local2)
                }

                Store (SERN, Index (Arg1, 0x11))
                Or (Arg0, 0x06, HIID)
                //Arg1 [0x10] = SBDN
                Arg1 [0x10] = RECB(0xA0,128)
                Or (Arg0, 0x04, HIID)
                Name (BTYP, Buffer (0x05)
                {
                     0x00, 0x00, 0x00, 0x00, 0x00
                })
                Store (B1B4 (BCH0, BCH1, BCH2, BCH3), BTYP)
                Store (BTYP, Index (Arg1, 0x12))
                Or (Arg0, 0x05, HIID)
                //Arg1 [0x13] = SBMN
                Arg1 [0x13] = RECB(0xA0,128)
            }
            Else
            {
                Arg1 [0x02] = 0xFFFFFFFF
                Arg1 [0x06] = Zero
                Arg1 [0x07] = Zero
                Arg1 [0x03] = 0xFFFFFFFF
            }

            Release (BATM)
            Return (Arg1)
        }
        Else
        {
            Return (\_SB.PCI0.LPCB.EC.XBIX(Arg0, Arg1, Arg2))
        }

    }

    Method (GBST, 4, NotSerialized)
    {
        If (_OSI ("Darwin"))
        {
            Acquire (BATM, 0xFFFF)
            If (And (Arg1, 0x20))
            {
                Store (0x02, Local0)
            }
            ElseIf (And (Arg1, 0x40))
            {
                Store (0x01, Local0)
            }
            Else
            {
```

```
                    Store (0x00, Local0)
                }

                If (And (Arg1, 0x07)) {}
                Else
                {
                    Or (Local0, 0x04, Local0)
                }

                If (LEqual (And (Arg1, 0x07), 0x07))
                {
                    Store (0x04, Local0)
                    Store (0x00, Local1)
                    Store (0x00, Local2)
                    Store (0x00, Local3)
                }
                Else
                {
                    Store (Arg0, HIID)
                    Store (B1B2 (BVO0, BVO1), Local3)
                    If (Arg2)
                    {
                        Multiply (B1B2 (BRCA, BRCB), 0x0A, Local2)
                    }
                    Else
                    {
                        Store (B1B2 (BRCA, BRCB), Local2)
                    }

                    Store (B1B2 (BAC0, BAC1), Local1)
                    If (LGreaterEqual (Local1, 0x8000))
                    {
                        If (And (Local0, 0x01))
                        {
                            Subtract (0x00010000, Local1, Local1)
                        }
                        Else
                        {
                            Store (0x00, Local1)
                        }
                    }
                    ElseIf (LNot (And (Local0, 0x02)))
                    {
                        Store (0x00, Local1)
                    }

                    If (Arg2)
                    {
                        Multiply (Local3, Local1, Local1)
                        Divide (Local1, 0x03E8, Local7, Local1)
                        Store (Local0, Local7)
```

```
                        Store (Local7, Local0)
                    }
                }

                Local5 = (One << (Arg0 >> 0x04))
                BSWA |= BSWR /* \_SB_.PCI0.LPCB.EC__.BSWR */
                If (((BSWA & Local5) == Zero))
                {
                    Arg3 [Zero] = Local0
                    Arg3 [One] = Local1
                    Arg3 [0x02] = Local2
                    Arg3 [0x03] = Local3
                    If ((Arg0 == Zero))
                    {
                        B0I0 = Local0
                        B0I1 = Local1
                        B0I2 = Local2
                        B0I3 = Local3
                    }
                    Else
                    {
                        B1I0 = Local0
                        B1I1 = Local1
                        B1I2 = Local2
                        B1I3 = Local3
                    }
                }
                Else
                {
                    If (\_SB.PCI0.LPCB.EC.AC._PSR ())
                    {
                        If ((Arg0 == Zero))
                        {
                            Arg3 [Zero] = \_SB.PCI0.LPCB.EC.B0I0
                            Arg3 [One] = \_SB.PCI0.LPCB.EC.B0I1
                            Arg3 [0x02] = \_SB.PCI0.LPCB.EC.B0I2
                            Arg3 [0x03] = \_SB.PCI0.LPCB.EC.B0I3
                        }
                        Else
                        {
                            Arg3 [Zero] = \_SB.PCI0.LPCB.EC.B1I0
                            Arg3 [One] = \_SB.PCI0.LPCB.EC.B1I1
                            Arg3 [0x02] = \_SB.PCI0.LPCB.EC.B1I2
                            Arg3 [0x03] = \_SB.PCI0.LPCB.EC.B1I3
                        }
                    }
                    Else
                    {
                        Arg3 [Zero] = Local0
                        Arg3 [One] = Local1
                        Arg3 [0x02] = Local2
```

```
                        Arg3 [0x03] = Local3
                    }

                    If ((((Local0 & 0x04) == Zero) && ((Local2 > Zero) &&
                        (Local3 > Zero))))
                    {
                        BSWA &= ~Local5
                        Arg3 [Zero] = Local0
                        Arg3 [One] = Local1
                        Arg3 [0x02] = Local2
                        Arg3 [0x03] = Local3
                    }
                }

                Release (BATM)
                Return (Arg3)
            }
            Else
            {
                Return (\_SB.PCI0.LPCB.EC.XBST(Arg0, Arg1, Arg2, Arg3))
            }
        }
    }
}
//EOF
```

```
// battery
DefinitionBlock ("", "SSDT", 2, "OCLT", "BATT", 0)
{
    External (_SB.PCI0.LPCB.EC, DeviceObj)
    External (_SB.PCI0.LPCB.EC.AC._PSR, MethodObj)
    External (_SB.PCI0.LPCB.EC.B0I0, IntObj)
    External (_SB.PCI0.LPCB.EC.B0I1, IntObj)
    External (_SB.PCI0.LPCB.EC.B0I2, IntObj)
    External (_SB.PCI0.LPCB.EC.B0I3, IntObj)
    External (_SB.PCI0.LPCB.EC.B1I0, IntObj)
    External (_SB.PCI0.LPCB.EC.B1I1, IntObj)
    External (_SB.PCI0.LPCB.EC.B1I2, IntObj)
    External (_SB.PCI0.LPCB.EC.B1I3, IntObj)
    External (_SB.PCI0.LPCB.EC.BATM, MutexObj)
    External (_SB.PCI0.LPCB.EC.BATW, MethodObj)
    External (_SB.PCI0.LPCB.EC.BSWA, IntObj)
    External (_SB.PCI0.LPCB.EC.BSWR, IntObj)
    External (_SB.PCI0.LPCB.EC.HB0S, FieldUnitObj)
    External (_SB.PCI0.LPCB.EC.HB1S, FieldUnitObj)
    External (_SB.PCI0.LPCB.EC.HIID, FieldUnitObj)
    //
    External(_SB.PCI0.LPCB.EC.XBIF, MethodObj)
    External(_SB.PCI0.LPCB.EC.XBIX, MethodObj)
    External(_SB.PCI0.LPCB.EC.XBST, MethodObj)
    External(_SB.PCI0.LPCB.EC.XJTP, MethodObj)

    Method (B1B2, 2, NotSerialized)
    {
        Local0 = (Arg1 << 0x08)
        Local0 |= Arg0
        Return (Local0)
    }

    Method (B1B4, 4, NotSerialized)
    {
        Local0 = Arg3
        Local0 = (Arg2 | (Local0 << 0x08))
        Local0 = (Arg1 | (Local0 << 0x08))
        Local0 = (Arg0 | (Local0 << 0x08))
        Return (Local0)
    }

    Scope (\_SB.PCI0.LPCB.EC)
    {
        Method (RE1B, 1, NotSerialized)
        {
            OperationRegion (ECOR, EmbeddedControl, Arg0, One)
            Field (ECOR, ByteAcc, NoLock, Preserve)
            {
                BYTE,   8
```

```
            }

            Return (BYTE)
        }

        Method (RECB, 2, Serialized)
        {
            Arg1 >>= 0x03
            Name (TEMP, Buffer (Arg1){})
            Arg1 += Arg0
            Local0 = Zero
            While ((Arg0 < Arg1))
            {
                TEMP [Local0] = RE1B (Arg0)
                Arg0++
                Local0++
            }

            Return (TEMP)
        }

        OperationRegion (BRAM, EmbeddedControl, Zero, 0x0100)
        Field (BRAM, ByteAcc, NoLock, Preserve)
        {
            Offset (0xA0),
            RC00,   8, RC01,   8,      //SBRC,   16,
            FC00,   8, FC01,   8,      //SBFC,   16,
            ,   16,
            ,   16,
            AC00,   8, AC01,   8,      //SBAC,   16,
            BV00,   8, BV01,   8,      //SBVO,   16,
            ,   16,
            ,   16
        }

        Field (BRAM, ByteAcc, NoLock, Preserve)
        {
            Offset (0xA0),
            SB00,   8, SB01,   8,      //SBBM,   16,
            ,   16,
            CC00,   8, CC01,   8,     //SBCC,   16 //E470,T470S
        }

        Field (BRAM, ByteAcc, NoLock, Preserve)
        {
            Offset (0xA0),
            DC00,   8, DC01,   8,      //SBDC,   16,
            DV00,   8, DV01,   8,      //SBDV,   16,
            ,   16,
            ,   16,
            ,   16,
```

```
            SN00,   8, SN01,   8       //SBSN,   16
        }

        Field (BRAM, ByteAcc, NoLock, Preserve)
        {
            Offset (0xA0),
            CH00,   8, CH01,   8, CH02,   8, CH03,   8 //SBCH,   32
        }

        Method (GBIF, 3, NotSerialized)
        {
            If (_OSI ("Darwin"))
            {
                Acquire (BATM, 0xFFFF)
                If (Arg2)
                {
                    HIID = (Arg0 | One)
                    Local7 = B1B2 (SB00, SB01)
                    Local7 >>= 0x0F
                    Arg1 [Zero] = (Local7 ^ One)
                    HIID = Arg0
                    If (Local7)
                    {
                        Local1 = (B1B2 (FC00, FC01) * 0x0A)
                    }
                    Else
                    {
                        Local1 = B1B2 (FC00, FC01)
                    }

                    Arg1 [0x02] = Local1
                    HIID = (Arg0 | 0x02)
                    If (Local7)
                    {
                        Local0 = (B1B2 (DC00, DC01) * 0x0A)
                    }
                    Else
                    {
                        Local0 = B1B2 (DC00, DC01)
                    }

                    Arg1 [One] = Local0
                    Divide (Local1, 0x14, Local2, Arg1 [0x05])
                    If (Local7)
                    {
                        Arg1 [0x06] = 0xC8
                    }
                    ElseIf (B1B2 (DV00, DV01))
                    {
                        Divide (0x00030D40, B1B2 (DV00, DV01), Local2, Arg1 [0x06])
                    }
```

```
                    Else
                    {
                        Arg1 [0x06] = Zero
                    }

                    Arg1 [0x04] = B1B2 (DV00, DV01)
                    Local0 = B1B2 (SN00, SN01)
                    Name (SERN, Buffer (0x06)
                    {
                        "     "
                    })
                    Local2 = 0x04
                    While (Local0)
                    {
                        Divide (Local0, 0x0A, Local1, Local0)
                        SERN [Local2] = (Local1 + 0x30)
                        Local2--
                    }

                    Arg1 [0x0A] = SERN
                    HIID = (Arg0 | 0x06)
                    Arg1 [0x09] = RECB (0xA0, 0x80)
                    HIID = (Arg0 | 0x04)
                    Name (BTYP, Buffer (0x05)
                    {
                         0x00, 0x00, 0x00, 0x00, 0x00
                    })
                    BTYP = B1B4 (CH00, CH01, CH02, CH03)
                    Arg1 [0x0B] = BTYP
                    HIID = (Arg0 | 0x05)
                    Arg1 [0x0C] = RECB (0xA0, 0x80)
                }
                Else
                {
                    Arg1 [One] = 0xFFFFFFFF
                    Arg1 [0x05] = Zero
                    Arg1 [0x06] = Zero
                    Arg1 [0x02] = 0xFFFFFFFF
                }

                Release (BATM)
                Return (Arg1)
            }
            Else
            {
                Return (\_SB.PCI0.LPCB.EC.XBIF(Arg0, Arg1, Arg2))
            }
        }

        Method (GBIX, 3, NotSerialized)
        {
```

```
            If (_OSI ("Darwin"))
            {
                Acquire (BATM, 0xFFFF)
                If (Arg2)
                {
                    HIID = (Arg0 | One)
                    Local7 = B1B2 (CC00, CC01)
                    Arg1 [0x08] = Local7
                    Local7 = B1B2 (SB00, SB01)
                    Local7 >>= 0x0F
                    Arg1 [One] = (Local7 ^ One)
                    HIID = Arg0
                    If (Local7)
                    {
                        Local1 = (B1B2 (FC00, FC01) * 0x0A)
                    }
                    Else
                    {
                        Local1 = B1B2 (FC00, FC01)
                    }

                    Arg1 [0x03] = Local1
                    HIID = (Arg0 | 0x02)
                    If (Local7)
                    {
                        Local0 = (B1B2 (DC00, DC01) * 0x0A)
                    }
                    Else
                    {
                        Local0 = B1B2 (DC00, DC01)
                    }

                    Arg1 [0x02] = Local0
                    Divide (Local1, 0x14, Local2, Arg1 [0x06])
                    If (Local7)
                    {
                        Arg1 [0x07] = 0xC8
                    }
                    ElseIf (B1B2 (DV00, DV01))
                    {
                        Divide (0x00030D40, B1B2 (DV00, DV01), Local2, Arg1 [0x07])
                    }
                    Else
                    {
                        Arg1 [0x07] = Zero
                    }

                    Arg1 [0x05] = B1B2 (DV00, DV01)
                    Local0 = B1B2 (SN00, SN01)
                    Name (SERN, Buffer (0x06)
                    {
```

```
                    "     "
                })
                Local2 = 0x04
                While (Local0)
                {
                    Divide (Local0, 0x0A, Local1, Local0)
                    SERN [Local2] = (Local1 + 0x30)
                    Local2--
                }

                Arg1 [0x11] = SERN
                HIID = (Arg0 | 0x06)
                Arg1 [0x10] = RECB (0xA0, 0x80)
                HIID = (Arg0 | 0x04)
                Name (BTYP, Buffer (0x05)
                {
                     0x00, 0x00, 0x00, 0x00, 0x00
                })
                BTYP = B1B4 (CH00, CH01, CH02, CH03)
                Arg1 [0x12] = BTYP
                HIID = (Arg0 | 0x05)
                Arg1 [0x13] = RECB (0xA0, 0x80)
            }
            Else
            {
                Arg1 [0x02] = 0xFFFFFFFF
                Arg1 [0x06] = Zero
                Arg1 [0x07] = Zero
                Arg1 [0x03] = 0xFFFFFFFF
            }

            Release (BATM)
            Return (Arg1)
        }
        Else
        {
            Return (\_SB.PCI0.LPCB.EC.XBIX(Arg0, Arg1, Arg2))
        }

    }

    Method (GBST, 4, NotSerialized)
    {
        If (_OSI ("Darwin"))
        {
            Acquire (BATM, 0xFFFF)
            If ((Arg1 & 0x20))
            {
                Local0 = 0x02
            }
            ElseIf ((Arg1 & 0x40))
```

```
                {
                    Local0 = One
                }
                Else
                {
                    Local0 = Zero
                }

                If ((Arg1 & 0x07)){}
                Else
                {
                    Local0 |= 0x04
                }

                If (((Arg1 & 0x07) == 0x07))
                {
                    Local0 = 0x04
                    Local1 = Zero
                    Local2 = Zero
                    Local3 = Zero
                }
                Else
                {
                    HIID = Arg0
                    Local3 = B1B2 (BV00, BV01)
                    If (Arg2)
                    {
                        Local2 = (B1B2 (RC00, RC01) * 0x0A)
                    }
                    Else
                    {
                        Local2 = B1B2 (RC00, RC01)
                    }

                    Local1 = B1B2 (AC00, AC01)
                    If ((Local1 >= 0x8000))
                    {
                        If ((Local0 & One))
                        {
                            Local1 = (0x00010000 - Local1)
                        }
                        Else
                        {
                            Local1 = Zero
                        }
                    }
                    ElseIf (!(Local0 & 0x02))
                    {
                        Local1 = Zero
                    }
```

```
                    If (Arg2)
                    {
                        Local1 *= Local3
                        Divide (Local1, 0x03E8, Local7, Local1)
                    }
                }

                Local5 = (One << (Arg0 >> 0x04))
                BSWA |= BSWR
                If (((BSWA & Local5) == Zero))
                {
                    Arg3 [Zero] = Local0
                    Arg3 [One] = Local1
                    Arg3 [0x02] = Local2
                    Arg3 [0x03] = Local3
                    If ((Arg0 == Zero))
                    {
                        B0I0 = Local0
                        B0I1 = Local1
                        B0I2 = Local2
                        B0I3 = Local3
                    }
                    Else
                    {
                        B1I0 = Local0
                        B1I1 = Local1
                        B1I2 = Local2
                        B1I3 = Local3
                    }
                }
                Else
                {
                    If (\_SB.PCI0.LPCB.EC.AC._PSR ())
                    {
                        If ((Arg0 == Zero))
                        {
                            Arg3 [Zero] = B0I0
                            Arg3 [One] = B0I1
                            Arg3 [0x02] = B0I2
                            Arg3 [0x03] = B0I3
                        }
                        Else
                        {
                            Arg3 [Zero] = B1I0
                            Arg3 [One] = B1I1
                            Arg3 [0x02] = B1I2
                            Arg3 [0x03] = B1I3
                        }
                    }
                    Else
                    {
```

```
                    Arg3 [Zero] = Local0
                    Arg3 [One] = Local1
                    Arg3 [0x02] = Local2
                    Arg3 [0x03] = Local3
                }

                If ((((Local0 & 0x04) == Zero) && ((Local2 > Zero) &&
                    (Local3 > Zero))))
                {
                    BSWA &= ~Local5
                    Arg3 [Zero] = Local0
                    Arg3 [One] = Local1
                    Arg3 [0x02] = Local2
                    Arg3 [0x03] = Local3
                }
            }

            Release (BATM)
            Return (Arg3)
        }
        Else
        {
            Return (\_SB.PCI0.LPCB.EC.XBST(Arg0, Arg1, Arg2, Arg3))
        }
    }

    Method (AJTP, 3, NotSerialized)
    {
        If (_OSI ("Darwin"))
        {
            Local0 = Arg1
            Acquire (BATM, 0xFFFF)
            HIID = Arg0
            Local1 = B1B2 (RC00, RC01)
            Release (BATM)
            If ((Arg0 == Zero))
            {
                Local2 = HB0S
            }
            Else
            {
                Local2 = HB1S
            }

            If ((Local2 & 0x20))
            {
                If ((Arg2 > Zero))
                {
                    Local0 += One
                }
```

```
                    If ((Local0 <= Local1))
                    {
                        Local0 = (Local1 + One)
                    }
                }
                ElseIf ((Local2 & 0x40))
                {
                    If ((Local0 >= Local1))
                    {
                        Local0 = (Local1 - One)
                    }
                }

                Return (Local0)
            }
            Else
            {
                Return (\_SB.PCI0.LPCB.EC.XJTP(Arg0, Arg1, Arg2))
            }
        }
    }
}
```

```
// battery
DefinitionBlock ("", "SSDT", 2, "OCLT", "BAT0", 0)
{
    External(_SB.PCI0.LPC.EC, DeviceObj)
    External(_SB.PCI0.LPC.EC.BATM, MutexObj)
    External(_SB.PCI0.LPC.EC.HIID, FieldUnitObj)
    //
    External(_SB.PCI0.LPC.EC.XBIF, MethodObj)
    External(_SB.PCI0.LPC.EC.XBST, MethodObj)

    Method (B1B2, 2, NotSerialized)
    {
        ShiftLeft (Arg1, 8, Local0)
        Or (Arg0, Local0, Local0)
        Return (Local0)

    }
    Method (B1B4, 4, NotSerialized)
    {
        Store (Arg3, Local0)
        Or (Arg2, ShiftLeft (Local0, 0x08), Local0)
        Or (Arg1, ShiftLeft (Local0, 0x08), Local0)
        Or (Arg0, ShiftLeft (Local0, 0x08), Local0)
        Return (Local0)
    }
    Scope(\_SB.PCI0.LPC.EC)
    {
        Method (RE1B, 1, NotSerialized)
        {
            OperationRegion(ERAM, EmbeddedControl, Arg0, 1)
            Field(ERAM, ByteAcc, NoLock, Preserve) { BYTE, 8 }
            Return(BYTE)
        }
        Method (RECB, 2, Serialized)
        {
            ShiftRight(Arg1, 3, Arg1)
            Name(TEMP, Buffer(Arg1) { })
            Add(Arg0, Arg1, Arg1)
            Store(0, Local0)
            While (LLess(Arg0, Arg1))
            {
                Store(RE1B(Arg0), Index(TEMP, Local0))
                Increment(Arg0)
                Increment(Local0)
            }
            Return(TEMP)
        }
        OperationRegion (BRAM, EmbeddedControl, 0x00, 0x0100)
        Field (BRAM, ByteAcc, NoLock, Preserve)
        {
```

```
            Offset (0xA0),
            BRCA,8,BRCB,8,      //SBRC,   16,
            BFC0,8,BFC1,8,      //SBFC,   16,
                                //SBAE,   16,
                                //SBRS,   16,
            Offset (0xA8),
            BAC0,8,BAC1,8,      //SBAC,   16,
            BVO0,8,BVO1,8,      //SBVO,   16,
                                //SBAF,   16,
                                //SBBS,   16
        }
        Field (BRAM, ByteAcc, NoLock, Preserve)
        {
            Offset (0xA0),
            BBM0,8,BBM1,8,      //SBBM,   16,
                                //SBMD,   16,

        }
        Field (BRAM, ByteAcc, NoLock, Preserve)
        {
            Offset (0xA0),
            BDC0,8,BDC1,8,      //SBDC,   16,
            BDV0,8,BDV1,8,      //SBDV,   16,
                                //SBOM,   16,
                                //SBSI,   16,
                                //SBDT,   16,
            Offset (0xAA),
            BSN0,8,BSN1,8       //SBSN,   16,s2
        }
        Field (BRAM, ByteAcc, NoLock, Preserve)
        {
            Offset (0xA0),
            BCH0,8,BCH1,8,BCH2,8,BCH3,8 //SBCH,   32
        }
        Method (GBIF, 3, NotSerialized)
        {
            If (_OSI ("Darwin"))
            {
            Acquire (BATM, 0xFFFF)
            If (Arg2)
            {
                Or (Arg0, 0x01, HIID)
                Store (B1B2 (BBM0, BBM1), Local7)
                ShiftRight (Local7, 0x0F, Local7)
                XOr (Local7, 0x01, Index (Arg1, 0x00))
                Store (Arg0, HIID)
                If (Local7)
                {
                    Multiply (B1B2 (BFC0, BFC1), 0x0A, Local1)
                }
                Else
```

```
                {
                    Store (B1B2 (BFC0, BFC1), Local1)
                }

                Store (Local1, Index (Arg1, 0x02))
                Or (Arg0, 0x02, HIID)
                If (Local7)
                {
                    Multiply (B1B2 (BDC0, BDC1), 0x0A, Local0)
                }
                Else
                {
                    Store (B1B2 (BDC0, BDC1), Local0)
                }

                Store (Local0, Index (Arg1, 0x01))
                Divide (Local1, 0x14, Local2, Index (Arg1, 0x05))
                If (Local7)
                {
                    Store (0xC8, Index (Arg1, 0x06))
                }
                ElseIf (B1B2 (BDV0, BDV1))
                {
                    Divide (0x00030D40, B1B2 (BDV0, BDV1), Local2, Index (Arg1, 0x0
6))
                }
                Else
                {
                    Store (0x00, Index (Arg1, 0x06))
                }

                Store (B1B2 (BDV0, BDV1), Index (Arg1, 0x04))
                Store (B1B2 (BSN0, BSN1), Local0)
                Name (SERN, Buffer (0x06)
                {
                    "     "
                })
                Store (0x04, Local2)
                While (Local0)
                {
                    Divide (Local0, 0x0A, Local1, Local0)
                    Add (Local1, 0x30, Index (SERN, Local2))
                    Decrement (Local2)
                }

                Store (SERN, Index (Arg1, 0x0A))
                Or (Arg0, 0x06, HIID)
                Store (RECB(0xA0,128), Index (Arg1, 0x09))
                Or (Arg0, 0x04, HIID)
                Name (BTYP, Buffer (0x05)
                {
```

```
                    0x00, 0x00, 0x00, 0x00, 0x00
                })
                Store (B1B4 (BCH0, BCH1, BCH2, BCH3), BTYP)
                Store (BTYP, Index (Arg1, 0x0B))
                Or (Arg0, 0x05, HIID)
                Store (RECB(0xA0,128), Index (Arg1, 0x0C))
            }
            Else
            {
                Store (0xFFFFFFFF, Index (Arg1, 0x01))
                Store (0x00, Index (Arg1, 0x05))
                Store (0x00, Index (Arg1, 0x06))
                Store (0xFFFFFFFF, Index (Arg1, 0x02))
            }

            Release (BATM)
            Return (Arg1)
            }
            Else
            {
                Return (\_SB.PCI0.LPC.EC.XBIF(Arg0, Arg1, Arg2))
            }
        }

        Method (GBST, 4, NotSerialized)
        {
            If (_OSI ("Darwin"))
            {
            Acquire (BATM, 0xFFFF)
            If (And (Arg1, 0x20))
            {
                Store (0x02, Local0)
            }
            ElseIf (And (Arg1, 0x40))
            {
                Store (0x01, Local0)
            }
            Else
            {
                Store (0x00, Local0)
            }

            If (And (Arg1, 0x07)) {}
            Else
            {
                Or (Local0, 0x04, Local0)
            }

            If (LEqual (And (Arg1, 0x07), 0x07))
            {
                Store (0x04, Local0)
```

```
                Store (0x00, Local1)
                Store (0x00, Local2)
                Store (0x00, Local3)
            }
            Else
            {
                Store (Arg0, HIID)
                Store (B1B2 (BVO0, BVO1), Local3)
                If (Arg2)
                {
                    Multiply (B1B2 (BRCA, BRCB), 0x0A, Local2)
                }
                Else
                {
                    Store (B1B2 (BRCA, BRCB), Local2)
                }

                Store (B1B2 (BAC0, BAC1), Local1)
                If (LGreaterEqual (Local1, 0x8000))
                {
                    If (And (Local0, 0x01))
                    {
                        Subtract (0x00010000, Local1, Local1)
                    }
                    Else
                    {
                        Store (0x00, Local1)
                    }
                }
                ElseIf (LNot (And (Local0, 0x02)))
                {
                    Store (0x00, Local1)
                }

                If (Arg2)
                {
                    Multiply (Local3, Local1, Local1)
                    Divide (Local1, 0x03E8, Local7, Local1)
                    Store (Local0, Local7)
                    Store (Local7, Local0)
                }
            }

            Store (Local0, Index (Arg3, 0x00))
            Store (Local1, Index (Arg3, 0x01))
            Store (Local2, Index (Arg3, 0x02))
            Store (Local3, Index (Arg3, 0x03))
            Release (BATM)
            Return (Arg3)
            }
            Else
```

```
            {
                Return (\_SB.PCI0.LPC.EC.XBST(Arg0, Arg1, Arg2, Arg3))
            }
        }
    }
}
//EOF
```

```
// battery for WTX30,WTX40,X260,X1C2014,X1C2016
//
// In config ACPI, _STA to XSTA
// Count:          1
// Find:           5F 53 54 41
// Replace:        58 53 54 41
// Skip:           12
// TableSignature: 44 53 44 54
//
// battery for WTX50,WTX60,X1C2015
//
// In config ACPI, _STA to XSTA
// Count:          1
// Find:           5F 53 54 41
// Replace:        58 53 54 41
// Skip:           11
// TableSignature: 44 53 44 54
//
DefinitionBlock ("", "SSDT", 2, "OCLT", "BAT1-off", 0)
{
    External(_SB.PCI0.LPC.EC.BAT1, DeviceObj)
    External(_SB.PCI0.LPC.EC.BAT1.XSTA, MethodObj)

    Scope(\_SB.PCI0.LPC.EC.BAT1)
    {
        Method(_STA)
        {
            If (_OSI ("Darwin"))
            {
                Return (0)
            }
            Else
            {
                Return (\_SB.PCI0.LPC.EC.BAT1.XSTA ())
            }
        }
    }
}
// EOF
```

```
// battery for T470,T470s
//
// In config ACPI, _STA to XSTA
// Count:          1
// Find:           5F 53 54 41
// Replace:        58 53 54 41
// Skip:           36
// TableSignature: 44 53 44 54
//
// battery for X270
//
// In config ACPI, _STA to XSTA
// Count:          1
// Find:           5F 53 54 41
// Replace:        58 53 54 41
// Skip:           35
// TableSignature: 44 53 44 54
//
DefinitionBlock ("", "SSDT", 2, "OCLT", "BAT1-off", 0)
{
    External(_SB.PCI0.LPCB.EC.BAT1, DeviceObj)
    External(_SB.PCI0.LPCB.EC.BAT1.XSTA, MethodObj)

    Scope(\_SB.PCI0.LPCB.EC.BAT1)
    {
        Method(_STA)
        {
            If (_OSI ("Darwin"))
            {
                Return (0)
            }
            Else
            {
                Return (\_SB.PCI0.LPCB.EC.BAT1.XSTA ())
            }
        }
    }
}
// EOF
```

```
//
DefinitionBlock ("", "SSDT", 2, "ACDT", "BATC", 0x00000000)
{
    External (_SB_.PCI0.LPCB.EC, DeviceObj)
    External (_SB_.PCI0.LPCB.EC.BAT0, DeviceObj)
    External (_SB_.PCI0.LPCB.EC.BAT0._BIF, MethodObj)
    External (_SB_.PCI0.LPCB.EC.BAT0._BIX, MethodObj)
    External (_SB_.PCI0.LPCB.EC.BAT0._BST, MethodObj)
    External (_SB_.PCI0.LPCB.EC.BAT0._HID, IntObj)
    External (_SB_.PCI0.LPCB.EC.BAT0._STA, MethodObj)
    External (_SB_.PCI0.LPCB.EC.BAT1, DeviceObj)
    External (_SB_.PCI0.LPCB.EC.BAT1._BIF, MethodObj)
    External (_SB_.PCI0.LPCB.EC.BAT1._BIX, MethodObj)
    External (_SB_.PCI0.LPCB.EC.BAT1._BST, MethodObj)
    External (_SB_.PCI0.LPCB.EC.BAT1._HID, IntObj)
    External (_SB_.PCI0.LPCB.EC.BAT1._STA, MethodObj)

    Scope (\_SB.PCI0.LPCB.EC)
    {
        Device (BATC)
        {
            Name (_HID, EisaId ("PNP0C0A"))
            Name (_UID, 0x02)

            Method (_INI)
            {
                // disable original battery objects by setting invalid _HID
                ^^BAT0._HID = 0
                ^^BAT1._HID = 0
            }

            Method (_STA)
            {
               // call original _STA for BAT0 and BAT1
               // result is bitwise OR between them
               If (_OSI ("Darwin"))
               {
                   Return(^^BAT0._STA() | ^^BAT1._STA())
               }
               Else
               {
                   Return (0)
               }
            }

            Method (_BIF)
            {
                // Local0 BAT0._BIF
                // Local1 BAT1._BIF
                // Local2 BAT0._STA
```

```
                // Local3 BAT1._STA
                // Local4/Local5 scratch

                // gather and validate data from BAT0
                Local0 = ^^BAT0._BIF ()
                Local2 = ^^BAT0._STA ()
                If (0x1f == Local2)
                {
                    // check for invalid design capacity
                    Local4 = DerefOf (Local0 [1])
                    If (!Local4 || Ones == Local4) { Local2 = 0; }
                    // check for invalid last full charge capacity
                    Local4 = DerefOf (Local0 [2])
                    If (!Local4 || Ones == Local4) { Local2 = 0; }
                    // check for invalid design voltage
                    Local4 = DerefOf (Local0 [4])
                    If (!Local4 || Ones == Local4) { Local2 = 0; }
                }
                // gather and validate data from BAT1
                Local1 = ^^BAT1._BIF ()
                Local3 = ^^BAT1._STA ()
                If (0x1f == Local3)
                {
                    // check for invalid design capacity
                    Local4 = DerefOf (Local1 [1])
                    If (!Local4 || Ones == Local4) { Local3 = 0; }
                    // check for invalid last full charge capacity
                    Local4 = DerefOf (Local1 [2])
                    If (!Local4 || Ones == Local4) { Local3 = 0; }
                    // check for invalid design voltage
                    Local4 = DerefOf (Local1 [4])
                    If (!Local4 || Ones == Local4) { Local3 = 0; }
                }
                // find primary and secondary battery
                If (0x1f != Local2 && 0x1f == Local3)
                {
                    // make primary use BAT1 data
                    Local0 = Local1 // BAT1._BIF result
                    Local2 = Local3 // BAT1._STA result
                    Local3 = 0  // no secondary battery
                }
                // combine batteries into Local0 result if possible
                If (0x1f == Local2 && 0x1f == Local3)
                {
                    // _BIF 0 Power Unit - leave BAT0 value
                    // _BIF 1 Design Capacity - add BAT0 and BAT1 values
                    Local4 = DerefOf (Local0 [1])
                    Local5 = DerefOf (Local1 [1])
                    If (0xffffffff != Local4 && 0xffffffff != Local5)
                    {
                        Local0 [1] = Local4 + Local5
```

```
                    }
                    // _BIF 2 Last Full Charge Capacity - add BAT0 and BAT1 values
                    Local4 = DerefOf (Local0 [2])
                    Local5 = DerefOf (Local1 [2])
                    If (0xffffffff != Local4 && 0xffffffff != Local5)
                    {
                        Local0 [2] = Local4 + Local5
                    }
                    // _BIF 3 Battery Technology - leave BAT0 value
                    // _BIF 4 Design Voltage - average between BAT0 and BAT1 value
s
                    Local4 = DerefOf (Local0 [4])
                    Local5 = DerefOf (Local1 [4])
                    If (0xffffffff != Local4 && 0xffffffff != Local5)
                    {
                        Local0 [4] = (Local4 + Local5) / 2
                    }
                    // _BIF 5 Design Capacity of Warning - add BAT0 and BAT1 value
s
                    Local0 [5] = DerefOf (Local0 [5]) + DerefOf (Local1 [5])
                    // _BIF 6 Design Capacity of Low - add BAT0 and BAT1 values
                    Local0 [6] = DerefOf (Local0 [6]) + DerefOf (Local1 [6])
                    // _BIF 7 Battery Capacity Granularity 1 - add BAT0 and BAT1 v
alues
                    Local4 = DerefOf (Local0 [7])
                    Local5 = DerefOf (Local1 [7])
                    If (0xffffffff != Local4 && 0xffffffff != Local5)
                    {
                        Local0 [7] = Local4 + Local5
                    }
                    // _BIF 8 Battery Capacity Granularity 2 - add BAT0 and BAT1 v
alues
                    Local4 = DerefOf (Local0 [8])
                    Local5 = DerefOf (Local1 [8])
                    If (0xffffffff != Local4 && 0xffffffff != Local5)
                    {
                        Local0 [8] = Local4 + Local5
                    }
                    // _BIF 9 Model Number - concatenate BAT0 and BAT1 values
                    Local0 [0x09] = Concatenate (Concatenate (DerefOf (Local0 [0x0
9]), " / "), DerefOf (Local1 [0x09]))
                    // _BIF a Serial Number - concatenate BAT0 and BAT1 values
                    Local0 [0x0a] = Concatenate (Concatenate (DerefOf (Local0 [0x0
a]), " / "), DerefOf (Local1 [0x0a]))
                    // _BIF b Battery Type - concatenate BAT0 and BAT1 values
                    Local0 [0x0b] = Concatenate (Concatenate (DerefOf (Local0 [0x0
b]), " / "), DerefOf (Local1 [0x0b]))
                    // _BIF c OEM Information - concatenate BAT0 and BAT1 values
                    Local0 [0x0c] = Concatenate (Concatenate (DerefOf (Local0 [0x0
c]), " / "), DerefOf (Local1 [0x0c]))
                }
```

```
        Return (Local0)
    } // _BIF

    Method (_BIX)
    {
        // Local0 BAT0._BIX
        // Local1 BAT1._BIX
        // Local2 BAT0._STA
        // Local3 BAT1._STA
        // Local4/Local5 scratch

        // gather and validate data from BAT0
        Local0 = ^^BAT0._BIX ()
        Local2 = ^^BAT0._STA ()
        If (0x1f == Local2)
        {
            // check for invalid design capacity
            Local4 = DerefOf (Local0 [2])
            If (!Local4 || Ones == Local4) { Local2 = 0; }
            // check for invalid last full charge capacity
            Local4 = DerefOf (Local0 [3])
            If (!Local4 || Ones == Local4) { Local2 = 0; }
            // check for invalid design voltage
            Local4 = DerefOf (Local0 [5])
            If (!Local4 || Ones == Local4) { Local2 = 0; }
        }
        // gather and validate data from BAT1
        Local1 = ^^BAT1._BIX ()
        Local3 = ^^BAT1._STA ()
        If (0x1f == Local3)
        {
            // check for invalid design capacity
            Local4 = DerefOf (Local1 [2])
            If (!Local4 || Ones == Local4) { Local3 = 0; }
            // check for invalid last full charge capacity
            Local4 = DerefOf (Local1 [3])
            If (!Local4 || Ones == Local4) { Local3 = 0; }
            // check for invalid design voltage
            Local4 = DerefOf (Local1 [5])
            If (!Local4 || Ones == Local4) { Local3 = 0; }
        }
        // find primary and secondary battery
        If (0x1f != Local2 && 0x1f == Local3)
        {
            // make primary use BAT1 data
            Local0 = Local1 // BAT1._BIX result
            Local2 = Local3 // BAT1._STA result
            Local3 = 0  // no secondary battery
        }
        // combine batteries into Local0 result if possible
```

```
                If (0x1f == Local2 && 0x1f == Local3)
                {
                    // _BIX 0 Revision - leave BAT0 value
                    // _BIX 1 Power Unit - leave BAT0 value
                    // _BIX 2 Design Capacity - add BAT0 and BAT1 values
                    Local4 = DerefOf (Local0 [2])
                    Local5 = DerefOf (Local1 [2])
                    If (0xffffffff != Local4 && 0xffffffff != Local5)
                    {
                        Local0 [2] = Local4 + Local5
                    }
                    // _BIX 3 Last Full Charge Capacity - add BAT0 and BAT1 values
                    Local4 = DerefOf (Local0 [3])
                    Local5 = DerefOf (Local1 [3])
                    If (0xffffffff != Local4 && 0xffffffff != Local5)
                    {
                        Local0 [3] = Local4 + Local5
                    }
                    // _BIX 4 Battery Technology - leave BAT0 value
                    // _BIX 5 Design Voltage - average between BAT0 and BAT1 value
s
                    Local4 = DerefOf (Local0 [5])
                    Local5 = DerefOf (Local1 [5])
                    If (0xffffffff != Local4 && 0xffffffff != Local5)
                    {
                        Local0 [5] = (Local4 + Local5) / 2
                    }
                    // _BIX 6 Design Capacity of Warning - add BAT0 and BAT1 value
s
                    Local0 [6] = DerefOf (Local0 [6]) + DerefOf (Local1 [6])
                    // _BIX 7 Design Capacity of Low - add BAT0 and BAT1 values
                    Local0 [7] = DerefOf (Local0 [7]) + DerefOf (Local1 [7])
                    // _BIX 8 Cycle Count - average between BAT0 and BAT1 values
                    Local4 = DerefOf (Local0 [8])
                    Local5 = DerefOf (Local1 [8])
                    If (0xffffffff != Local4 && 0xffffffff != Local5)
                    {
                        Local0 [8] = (Local4 + Local5) / 2
                    }
                    // _BIX 9 Measurement Accuracy - average between BAT0 and BAT1
 values
                    Local0 [9] = (DerefOf (Local0 [9]) + DerefOf (Local1 [9])) / 2
                    // _BIX 0xa Max Sampling Time - average between BAT0 and BAT1
values
                    Local4 = DerefOf (Local0 [0xa])
                    Local5 = DerefOf (Local1 [0xa])
                    If (0xffffffff != Local4 && 0xffffffff != Local5)
                    {
                        Local0 [0xa] = (Local4 + Local5) / 2
                    }
                    // _BIX 0xb Min Sampling Time - average between BAT0 and BAT1
```

```
values
                    Local4 = DerefOf (Local0 [0xb])
                    Local5 = DerefOf (Local1 [0xb])
                    If (0xffffffff != Local4 && 0xffffffff != Local5)
                    {
                        Local0 [0xb] = (Local4 + Local5) / 2
                    }
                    // _BIX 0xc Max Averaging Interval - average between BAT0 and
BAT1 values
                    Local0 [0xc] = (DerefOf (Local0 [0xc]) + DerefOf (Local1 [0xc]
)) / 2
                    // _BIX 0xd Min Averaging Interval - average between BAT0 and
BAT1 values
                    Local0 [0xd] = (DerefOf (Local0 [0xd]) + DerefOf (Local1 [0xd]
)) / 2
                    // _BIX 0xe Battery Capacity Granularity 1 - add BAT0 and BAT1
 values
                    Local4 = DerefOf (Local0 [0xe])
                    Local5 = DerefOf (Local1 [0xe])
                    If (0xffffffff != Local4 && 0xffffffff != Local5)
                    {
                        Local0 [0xe] = Local4 + Local5
                    }
                    // _BIX 0xf Battery Capacity Granularity 2 - add BAT0 and BAT1
 values
                    Local4 = DerefOf (Local0 [0xf])
                    Local5 = DerefOf (Local1 [0xf])
                    If (0xffffffff != Local4 && 0xffffffff != Local5)
                    {
                        Local0 [0xf] = Local4 + Local5
                    }
                    // _BIX 10 Model Number - concatenate BAT0 and BAT1 values
                    Local0 [0x10] = Concatenate (Concatenate (DerefOf (Local0 [0x1
0]), " / "), DerefOf (Local1 [0x10]))
                    // _BIX 11 Serial Number - concatenate BAT0 and BAT1 values
                    Local0 [0x11] = Concatenate (Concatenate (DerefOf (Local0 [0x1
1]), " / "), DerefOf (Local1 [0x11]))
                    // _BIX 12 Battery Type - concatenate BAT0 and BAT1 values
                    Local0 [0x12] = Concatenate (Concatenate (DerefOf (Local0 [0x1
2]), " / "), DerefOf (Local1 [0x12]))
                    // _BIX 13 OEM Information - concatenate BAT0 and BAT1 values
                    Local0 [0x13] = Concatenate (Concatenate (DerefOf (Local0 [0x1
3]), " / "), DerefOf (Local1 [0x13]))
                    // _BIX 14 Battery Swapping Capability - leave BAT0 value for
now
                }
                Return (Local0)
            } // _BIX

            Method (_BST)
            {
```

```
                    // Local0 BAT0._BST
                    // Local1 BAT1._BST
                    // Local2 BAT0._STA
                    // Local3 BAT1._STA
                    // Local4/Local5 scratch

                    // gather battery data from BAT0
                    Local0 = ^^BAT0._BST ()
                    Local2 = ^^BAT0._STA ()
                    If (0x1f == Local2)
                    {
                        // check for invalid remaining capacity
                        Local4 = DerefOf (Local0 [2])
                        If (!Local4 || Ones == Local4) { Local2 = 0; }
                    }
                    // gather battery data from BAT1
                    Local1 = ^^BAT1._BST ()
                    Local3 = ^^BAT1._STA ()
                    If (0x1f == Local3)
                    {
                        // check for invalid remaining capacity
                        Local4 = DerefOf (Local1 [2])
                        If (!Local4 || Ones == Local4) { Local3 = 0; }
                    }
                    // find primary and secondary battery
                    If (0x1f != Local2 && 0x1f == Local3)
                    {
                        // make primary use BAT1 data
                        Local0 = Local1 // BAT1._BST result
                        Local2 = Local3 // BAT1._STA result
                        Local3 = 0  // no secondary battery
                    }
                    // combine batteries into Local0 result if possible
                    If (0x1f == Local2 && 0x1f == Local3)
                    {
                        // _BST 0 - Battery State - if one battery is charging, then c
harging, else discharging
                        Local4 = DerefOf (Local0 [0])
                        Local5 = DerefOf (Local1 [0])
                        If (Local4 != Local5)
                        {
                            If (Local4 == 2 || Local5 == 2)
                            {
                                // 2 = charging
                                Local0 [0] = 2
                            }
                            ElseIf (Local4 == 1 || Local5 == 1)
                            {
                                // 1 = discharging
                                Local0 [0] = 1
                            }
```

```
                        ElseIf (Local4 == 3 || Local5 == 3)
                        {
                            Local0 [0] = 3
                        }
                        ElseIf (Local4 == 4 || Local5 == 4)
                        {
                            // critical
                            Local0 [0] = 4
                        }
                        ElseIf (Local4 == 5 || Local5 == 5)
                        {
                            // critical and discharging
                            Local0 [0] = 5
                        }
                        // if none of the above, just leave as BAT0 is
                    }

                    // _BST 1 - Battery Present Rate - add BAT0 and BAT1 values
                    Local0 [1] = DerefOf (Local0 [1]) + DerefOf (Local1 [1])
                    // _BST 2 - Battery Remaining Capacity - add BAT0 and BAT1 val
ues
                    Local0 [2] = DerefOf (Local0 [2]) + DerefOf (Local1 [2])
                    // _BST 3 - Battery Present Voltage - average between BAT0 and
 BAT1 values
                    Local0 [3] = (DerefOf (Local0 [3]) + DerefOf (Local1 [3])) / 2
                }
                Return (Local0)
            } // _BST
        } // BATC
    } // Scope (...)
}
//EOF
```

```
// ThinkPad-LPC
DefinitionBlock ("", "SSDT", 2, "OCLT", "BATC", 0)
{
    External(\_SB.PCI0.LPC.EC, DeviceObj)

    Scope(\_SB.PCI0.LPC.EC)
    {
        External(BAT0._HID, IntObj)
        External(BAT0._STA, MethodObj)
        External(BAT0._BIF, MethodObj)
        External(BAT0._BST, MethodObj)
        External(BAT1, DeviceObj)
        External(BAT1._HID, IntObj)
        External(BAT1._STA, MethodObj)
        External(BAT1._BIF, MethodObj)
        External(BAT1._BST, MethodObj)
        Device(BATC)
        {
            Name(_HID, EisaId ("PNP0C0A"))
            Name(_UID, 0x02)

            Method(_INI)
            {
                // disable original battery objects by setting invalid _HID
                ^^BAT0._HID = 0
                ^^BAT1._HID = 0
            }

            Method(CVWA, 3)
            // Convert mW to mA (or mWh to mAh)
            // Arg0 is mW or mWh (or mA/mAh in the case Arg2==0)
            // Arg1 is mV (usually design voltage)
            // Arg2 is whether conversion is needed (non-zero for convert)
            // return is mA or mAh
            {
                If (Arg2)
                {
                    Arg0 = (Arg0 * 1000) / Arg1
                }
                Return(Arg0)
            }

            Method(_STA)
            {
                // call original _STA for BAT0 and BAT1
                // result is bitwise OR between them

                If (_OSI ("Darwin"))
                {
                    Return(^^BAT0._STA() | ^^BAT1._STA())
```

```
                }
                Else
                {
                    Return (0)
                }
            }

            Name(B0CO, 0x00) // BAT0 0/1 needs conversion to mAh
            Name(B1CO, 0x00) // BAT1 0/1 needs conversion to mAh
            Name(B0DV, 0x00) // BAT0 design voltage
            Name(B1DV, 0x00) // BAT1 design voltage

            Method(_BST)
            {
                // Local0 BAT0._BST
                // Local1 BAT1._BST
                // Local2 BAT0._STA
                // Local3 BAT1._STA
                // Local4/Local5 scratch

                // gather battery data from BAT0
                Local0 = ^^BAT0._BST()
                Local2 = ^^BAT0._STA()
                If (0x1f == Local2)
                {
                    // check for invalid remaining capacity
                    Local4 = DerefOf(Local0[2])
                    If (!Local4 || Ones == Local4) { Local2 = 0; }
                }
                // gather battery data from BAT1
                Local1 = ^^BAT1._BST()
                Local3 = ^^BAT1._STA()
                If (0x1f == Local3)
                {
                    // check for invalid remaining capacity
                    Local4 = DerefOf(Local1[2])
                    If (!Local4 || Ones == Local4) { Local3 = 0; }
                }
                // find primary and secondary battery
                If (0x1f != Local2 && 0x1f == Local3)
                {
                    // make primary use BAT1 data
                    Local0 = Local1 // BAT1._BST result
                    Local2 = Local3 // BAT1._STA result
                    Local3 = 0  // no secondary battery
                }
                // combine batteries into Local0 result if possible
                If (0x1f == Local2 && 0x1f == Local3)
                {
                    // _BST 0 - Battery State - if one battery is charging, then ch
arging, else discharging
```

```
                    Local4 = DerefOf(Local0[0])
                    Local5 = DerefOf(Local1[0])
                    If (Local4 == 2 || Local5 == 2)
                    {
                        // 2 = charging
                        Local0[0] = 2
                    }
                    ElseIf (Local4 == 1 || Local5 == 1)
                    {
                        // 1 = discharging
                        Local0[0] = 1
                    }
                    ElseIf (Local4 == 5 || Local5 == 5)
                    {
                        // critical and discharging
                        Local0[0] = 5
                    }
                    ElseIf (Local4 == 4 || Local5 == 4)
                    {
                        // critical
                        Local0[0] = 4
                    }
                    // if none of the above, just leave as BAT0 is

                    // Note: Following code depends on _BIF being called before _BS
T to set B0CO and B1CO

                    // _BST 1 - Battery Present Rate - Add BAT0 and BAT1 values
                    Local0[1] = CVWA(DerefOf(Local0[1]), B0DV, B0CO) + CVWA(DerefOf
(Local1[1]), B1DV, B1CO)
                    // _BST 2 - Battery Remaining Capacity - Add BAT0 and BAT1 valu
es
                    Local0[2] = CVWA(DerefOf(Local0[2]), B0DV, B0CO) + CVWA(DerefOf
(Local1[2]), B1DV, B1CO)
                    // _BST 3 - Battery Present Voltage - Average BAT0 and BAT1 val
ues
                    Local0[3] = (DerefOf(Local0[3]) + DerefOf(Local1[3])) / 2
                }
                Return(Local0)
            } // _BST

            Method(_BIF)
            {
                // Local0 BAT0._BIF
                // Local1 BAT1._BIF
                // Local2 BAT0._STA
                // Local3 BAT1._STA
                // Local4/Local5 scratch

                // gather and validate data from BAT0
                Local0 = ^^BAT0._BIF()
```

```
                Local2 = ^^BAT0._STA()
                If (0x1f == Local2)
                {
                    // check for invalid design capacity
                    Local4 = DerefOf(Local0[1])
                    If (!Local4 || Ones == Local4) { Local2 = 0; }
                    // check for invalid max capacity
                    Local4 = DerefOf(Local0[2])
                    If (!Local4 || Ones == Local4) { Local2 = 0; }
                    // check for invalid design voltage
                    Local4 = DerefOf(Local0[4])
                    If (!Local4 || Ones == Local4) { Local2 = 0; }
                }
                // gather and validate data from BAT1
                Local1 = ^^BAT1._BIF()
                Local3 = ^^BAT1._STA()
                If (0x1f == Local3)
                {
                    // check for invalid design capacity
                    Local4 = DerefOf(Local1[1])
                    If (!Local4 || Ones == Local4) { Local3 = 0; }
                    // check for invalid max capacity
                    Local4 = DerefOf(Local1[2])
                    If (!Local4 || Ones == Local4) { Local3 = 0; }
                    // check for invalid design voltage
                    Local4 = DerefOf(Local1[4])
                    If (!Local4 || Ones == Local4) { Local3 = 0; }
                }
                // find primary and secondary battery
                If (0x1f != Local2 && 0x1f == Local3)
                {
                    // make primary use BAT1 data
                    Local0 = Local1 // BAT1._BIF result
                    Local2 = Local3 // BAT1._STA result
                    Local3 = 0  // no secondary battery
                }
                // combine batteries into Local0 result if possible
                If (0x1f == Local2 && 0x1f == Local3)
                {
                    // _BIF 0 - Power Unit - 0 = mWh | 1 = mAh
                    // set B0CO/B1CO if convertion to amps needed
                    B0CO = !DerefOf(Local0[0])
                    B1CO = !DerefOf(Local1[0])
                    // set _BIF[0] = 1 => mAh
                    Local0[0] = 1
                    // _BIF 4 - Design Voltage - store value for each Battery in mV
                    B0DV = DerefOf(Local0[4]) // cache BAT0 voltage
                    B1DV = DerefOf(Local1[4]) // cache BAT1 voltage
                    // _BIF 1 - Design Capacity - add BAT0 and BAT1 values
                    Local0[1] = CVWA(DerefOf(Local0[1]), B0DV, B0CO) + CVWA(DerefOf
(Local1[1]), B1DV, B1CO)
```

```
                    // _BIF 2 - Last Full Charge Capacity - add BAT0 and BAT1 value
s
                    Local0[2] = CVWA(DerefOf(Local0[2]), B0DV, B0CO) + CVWA(DerefOf
(Local1[2]), B1DV, B1CO)
                    // _BIF 3 - Battery Technology - leave BAT0 value
                    // _BIF 4 - Design Voltage - average BAT0 and BAT1 values
                    Local0[4] = (B0DV + B1DV) / 2
                    // _BIF 5 - Design Capacity Warning - add BAT0 and BAT1 values
                    Local0[5] = CVWA(DerefOf(Local0[5]), B0DV, B0CO) + CVWA(DerefOf
(Local1[5]), B1DV, B1CO)
                    // _BIF 6 - Design Capacity of Low - add BAT0 and BAT1 values
                    Local0[6] = CVWA(DerefOf(Local0[6]), B0DV, B0CO) + CVWA(DerefOf
(Local1[6]), B1DV, B1CO)
                    // _BIF 7+ - Leave BAT0 values for now
                }
                Return(Local0)
            } // _BIF
        } // BATC
    } // Scope(...)
}
// EOF
```

```
// ThinkPad-LPCB
DefinitionBlock ("", "SSDT", 2, "OCLT", "BATC", 0)
{
    External(\_SB.PCI0.LPCB.EC, DeviceObj)

    Scope(\_SB.PCI0.LPCB.EC)
    {
        External(BAT0._HID, IntObj)
        External(BAT0._STA, MethodObj)
        External(BAT0._BIF, MethodObj)
        External(BAT0._BST, MethodObj)
        External(BAT1, DeviceObj)
        External(BAT1._HID, IntObj)
        External(BAT1._STA, MethodObj)
        External(BAT1._BIF, MethodObj)
        External(BAT1._BST, MethodObj)
        Device(BATC)
        {
            Name(_HID, EisaId ("PNP0C0A"))
            Name(_UID, 0x02)

            Method(_INI)
            {
                // disable original battery objects by setting invalid _HID
                ^^BAT0._HID = 0
                ^^BAT1._HID = 0
            }

            Method(CVWA, 3)
            // Convert mW to mA (or mWh to mAh)
            // Arg0 is mW or mWh (or mA/mAh in the case Arg2==0)
            // Arg1 is mV (usually design voltage)
            // Arg2 is whether conversion is needed (non-zero for convert)
            // return is mA or mAh
            {
                If (Arg2)
                {
                    Arg0 = (Arg0 * 1000) / Arg1
                }
                Return(Arg0)
            }

            Method(_STA)
            {
                // call original _STA for BAT0 and BAT1
                // result is bitwise OR between them

                If (_OSI ("Darwin"))
                {
                    Return(^^BAT0._STA() | ^^BAT1._STA())
```

```
                }
                Else
                {
                    Return (0)
                }
            }

            Name(B0CO, 0x00) // BAT0 0/1 needs conversion to mAh
            Name(B1CO, 0x00) // BAT1 0/1 needs conversion to mAh
            Name(B0DV, 0x00) // BAT0 design voltage
            Name(B1DV, 0x00) // BAT1 design voltage

            Method(_BST)
            {
                // Local0 BAT0._BST
                // Local1 BAT1._BST
                // Local2 BAT0._STA
                // Local3 BAT1._STA
                // Local4/Local5 scratch

                // gather battery data from BAT0
                Local0 = ^^BAT0._BST()
                Local2 = ^^BAT0._STA()
                If (0x1f == Local2)
                {
                    // check for invalid remaining capacity
                    Local4 = DerefOf(Local0[2])
                    If (!Local4 || Ones == Local4) { Local2 = 0; }
                }
                // gather battery data from BAT1
                Local1 = ^^BAT1._BST()
                Local3 = ^^BAT1._STA()
                If (0x1f == Local3)
                {
                    // check for invalid remaining capacity
                    Local4 = DerefOf(Local1[2])
                    If (!Local4 || Ones == Local4) { Local3 = 0; }
                }
                // find primary and secondary battery
                If (0x1f != Local2 && 0x1f == Local3)
                {
                    // make primary use BAT1 data
                    Local0 = Local1 // BAT1._BST result
                    Local2 = Local3 // BAT1._STA result
                    Local3 = 0  // no secondary battery
                }
                // combine batteries into Local0 result if possible
                If (0x1f == Local2 && 0x1f == Local3)
                {
                    // _BST 0 - Battery State - if one battery is charging, then ch
arging, else discharging
```

```
                    Local4 = DerefOf(Local0[0])
                    Local5 = DerefOf(Local1[0])
                    If (Local4 == 2 || Local5 == 2)
                    {
                        // 2 = charging
                        Local0[0] = 2
                    }
                    ElseIf (Local4 == 1 || Local5 == 1)
                    {
                        // 1 = discharging
                        Local0[0] = 1
                    }
                    ElseIf (Local4 == 5 || Local5 == 5)
                    {
                        // critical and discharging
                        Local0[0] = 5
                    }
                    ElseIf (Local4 == 4 || Local5 == 4)
                    {
                        // critical
                        Local0[0] = 4
                    }
                    // if none of the above, just leave as BAT0 is

                    // Note: Following code depends on _BIF being called before _BS
T to set B0CO and B1CO

                    // _BST 1 - Battery Present Rate - Add BAT0 and BAT1 values
                    Local0[1] = CVWA(DerefOf(Local0[1]), B0DV, B0CO) + CVWA(DerefOf
(Local1[1]), B1DV, B1CO)
                    // _BST 2 - Battery Remaining Capacity - Add BAT0 and BAT1 valu
es
                    Local0[2] = CVWA(DerefOf(Local0[2]), B0DV, B0CO) + CVWA(DerefOf
(Local1[2]), B1DV, B1CO)
                    // _BST 3 - Battery Present Voltage - Average BAT0 and BAT1 val
ues
                    Local0[3] = (DerefOf(Local0[3]) + DerefOf(Local1[3])) / 2
                }
                Return(Local0)
            } // _BST

            Method(_BIF)
            {
                // Local0 BAT0._BIF
                // Local1 BAT1._BIF
                // Local2 BAT0._STA
                // Local3 BAT1._STA
                // Local4/Local5 scratch

                // gather and validate data from BAT0
                Local0 = ^^BAT0._BIF()
```

```
                Local2 = ^^BAT0._STA()
                If (0x1f == Local2)
                {
                    // check for invalid design capacity
                    Local4 = DerefOf(Local0[1])
                    If (!Local4 || Ones == Local4) { Local2 = 0; }
                    // check for invalid max capacity
                    Local4 = DerefOf(Local0[2])
                    If (!Local4 || Ones == Local4) { Local2 = 0; }
                    // check for invalid design voltage
                    Local4 = DerefOf(Local0[4])
                    If (!Local4 || Ones == Local4) { Local2 = 0; }
                }
                // gather and validate data from BAT1
                Local1 = ^^BAT1._BIF()
                Local3 = ^^BAT1._STA()
                If (0x1f == Local3)
                {
                    // check for invalid design capacity
                    Local4 = DerefOf(Local1[1])
                    If (!Local4 || Ones == Local4) { Local3 = 0; }
                    // check for invalid max capacity
                    Local4 = DerefOf(Local1[2])
                    If (!Local4 || Ones == Local4) { Local3 = 0; }
                    // check for invalid design voltage
                    Local4 = DerefOf(Local1[4])
                    If (!Local4 || Ones == Local4) { Local3 = 0; }
                }
                // find primary and secondary battery
                If (0x1f != Local2 && 0x1f == Local3)
                {
                    // make primary use BAT1 data
                    Local0 = Local1 // BAT1._BIF result
                    Local2 = Local3 // BAT1._STA result
                    Local3 = 0  // no secondary battery
                }
                // combine batteries into Local0 result if possible
                If (0x1f == Local2 && 0x1f == Local3)
                {
                    // _BIF 0 - Power Unit - 0 = mWh | 1 = mAh
                    // set B0CO/B1CO if convertion to amps needed
                    B0CO = !DerefOf(Local0[0])
                    B1CO = !DerefOf(Local1[0])
                    // set _BIF[0] = 1 => mAh
                    Local0[0] = 1
                    // _BIF 4 - Design Voltage - store value for each Battery in mV
                    B0DV = DerefOf(Local0[4]) // cache BAT0 voltage
                    B1DV = DerefOf(Local1[4]) // cache BAT1 voltage
                    // _BIF 1 - Design Capacity - add BAT0 and BAT1 values
                    Local0[1] = CVWA(DerefOf(Local0[1]), B0DV, B0CO) + CVWA(DerefOf
(Local1[1]), B1DV, B1CO)
```

```
                    // _BIF 2 - Last Full Charge Capacity - add BAT0 and BAT1 value
s
                    Local0[2] = CVWA(DerefOf(Local0[2]), B0DV, B0CO) + CVWA(DerefOf
(Local1[2]), B1DV, B1CO)
                    // _BIF 3 - Battery Technology - leave BAT0 value
                    // _BIF 4 - Design Voltage - average BAT0 and BAT1 values
                    Local0[4] = (B0DV + B1DV) / 2
                    // _BIF 5 - Design Capacity Warning - add BAT0 and BAT1 values
                    Local0[5] = CVWA(DerefOf(Local0[5]), B0DV, B0CO) + CVWA(DerefOf
(Local1[5]), B1DV, B1CO)
                    // _BIF 6 - Design Capacity of Low - add BAT0 and BAT1 values
                    Local0[6] = CVWA(DerefOf(Local0[6]), B0DV, B0CO) + CVWA(DerefOf
(Local1[6]), B1DV, B1CO)
                    // _BIF 7+ - Leave BAT0 values for now
                }
                Return(Local0)
            } // _BIF
        } // BATC
    } // Scope(...)
}
// EOF
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/Pro
pertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>ACPI</key>
    <dict>
        <key>Patch</key>
        <array>
            <dict>
                <key>Comment</key>
                <string>TP-BAT:GBIF03 to XBIF03</string>
                <key>Count</key>
                <integer>0</integer>
                <key>Enabled</key>
                <true/>
                <key>Find</key>
                <data>
                R0JJRgM=
                </data>
                <key>Limit</key>
                <integer>0</integer>
                <key>Mask</key>
                <data>
                </data>
                <key>OemTableId</key>
                <data>
                </data>
                <key>Replace</key>
                <data>
                WEJJRgM=
                </data>
                <key>ReplaceMask</key>
                <data>
                </data>
                <key>Skip</key>
                <integer>0</integer>
                <key>TableLength</key>
                <integer>0</integer>
                <key>TableSignature</key>
                <data>
                </data>
            </dict>
            <dict>
                <key>Comment</key>
                <string>TP-BAT:GBIX03 to XBIX03</string>
                <key>Count</key>
                <integer>0</integer>
                <key>Enabled</key>
                <true/>
```

```
                <key>Find</key>
                <data>
                R0JJWAM=
                </data>
                <key>Limit</key>
                <integer>0</integer>
                <key>Mask</key>
                <data>
                </data>
                <key>OemTableId</key>
                <data>
                </data>
                <key>Replace</key>
                <data>
                WEJJWAM=
                </data>
                <key>ReplaceMask</key>
                <data>
                </data>
                <key>Skip</key>
                <integer>0</integer>
                <key>TableLength</key>
                <integer>0</integer>
                <key>TableSignature</key>
                <data>
                </data>
            </dict>
            <dict>
                <key>Comment</key>
                <string>TP-BAT:GBST04 to XBST04</string>
                <key>Count</key>
                <integer>0</integer>
                <key>Enabled</key>
                <true/>
                <key>Find</key>
                <data>
                R0JTVAQ=
                </data>
                <key>Limit</key>
                <integer>0</integer>
                <key>Mask</key>
                <data>
                </data>
                <key>OemTableId</key>
                <data>
                </data>
                <key>Replace</key>
                <data>
                WEJTVAQ=
                </data>
                <key>ReplaceMask</key>
```

```
                <data>
                </data>
                <key>Skip</key>
                <integer>0</integer>
                <key>TableLength</key>
                <integer>0</integer>
                <key>TableSignature</key>
                <data>
                </data>
            </dict>
            <dict>
                <key>Comment</key>
                <string>TP-BAT:AJTP03 to XJTP03</string>
                <key>Count</key>
                <integer>0</integer>
                <key>Enabled</key>
                <true/>
                <key>Find</key>
                <data>QUpUUAM=</data>
                <key>Limit</key>
                <integer>0</integer>
                <key>Mask</key>
                <data></data>
                <key>OemTableId</key>
                <data></data>
                <key>Replace</key>
                <data>WEpUUAM=</data>
                <key>ReplaceMask</key>
                <data></data>
                <key>Skip</key>
                <integer>0</integer>
                <key>TableLength</key>
                <integer>0</integer>
                <key>TableSignature</key>
                <data></data>
            </dict>
        </array>
    </dict>
</dict>
</plist>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/Pro
pertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>ACPI</key>
    <dict>
        <key>Patch</key>
        <array>
            <dict>
                <key>Comment</key>
                <string>Mutex:MDGS=0</string>
                <key>Count</key>
                <integer>0</integer>
                <key>Enabled</key>
                <true/>
                <key>Find</key>
                <data>
                AU1ER1MH
                </data>
                <key>Limit</key>
                <integer>0</integer>
                <key>Mask</key>
                <data>
                </data>
                <key>OemTableId</key>
                <data>
                </data>
                <key>Replace</key>
                <data>
                AU1ER1MA
                </data>
                <key>ReplaceMask</key>
                <data>
                </data>
                <key>Skip</key>
                <integer>0</integer>
                <key>TableLength</key>
                <integer>0</integer>
                <key>TableSignature</key>
                <data>
                </data>
            </dict>
            <dict>
                <key>Comment</key>
                <string>Mutex:MCPU=0</string>
                <key>Count</key>
                <integer>0</integer>
                <key>Enabled</key>
                <true/>
```

```
                <key>Find</key>
                <data>
                AU1DUFUH
                </data>
                <key>Limit</key>
                <integer>0</integer>
                <key>Mask</key>
                <data>
                </data>
                <key>OemTableId</key>
                <data>
                </data>
                <key>Replace</key>
                <data>
                AU1DUFUA
                </data>
                <key>ReplaceMask</key>
                <data>
                </data>
                <key>Skip</key>
                <integer>0</integer>
                <key>TableLength</key>
                <integer>0</integer>
                <key>TableSignature</key>
                <data>
                </data>
            </dict>
            <dict>
                <key>Comment</key>
                <string>Mutex:BATM=0</string>
                <key>Count</key>
                <integer>0</integer>
                <key>Enabled</key>
                <true/>
                <key>Find</key>
                <data>
                AUJBVE0H
                </data>
                <key>Limit</key>
                <integer>0</integer>
                <key>Mask</key>
                <data>
                </data>
                <key>OemTableId</key>
                <data>
                </data>
                <key>Replace</key>
                <data>
                AUJBVE0A
                </data>
                <key>ReplaceMask</key>
```

```
                <data>
                </data>
                <key>Skip</key>
                <integer>0</integer>
                <key>TableLength</key>
                <integer>0</integer>
                <key>TableSignature</key>
                <data>
                </data>
            </dict>
            <dict>
                <key>Comment</key>
                <string>Mutex:MSMI=0</string>
                <key>Count</key>
                <integer>0</integer>
                <key>Enabled</key>
                <true/>
                <key>Find</key>
                <data>
                AU1TTUkH
                </data>
                <key>Limit</key>
                <integer>0</integer>
                <key>Mask</key>
                <data>
                </data>
                <key>OemTableId</key>
                <data>
                </data>
                <key>Replace</key>
                <data>
                AU1TTUkA
                </data>
                <key>ReplaceMask</key>
                <data>
                </data>
                <key>Skip</key>
                <integer>0</integer>
                <key>TableLength</key>
                <integer>0</integer>
                <key>TableSignature</key>
                <data>
                </data>
            </dict>
            <dict>
                <key>Comment</key>
                <string>Mutex:BFWM=0</string>
                <key>Count</key>
                <integer>0</integer>
                <key>Enabled</key>
                <true/>
```

```
                <key>Find</key>
                <data>
                AUJGV00H
                </data>
                <key>Limit</key>
                <integer>0</integer>
                <key>Mask</key>
                <data>
                </data>
                <key>OemTableId</key>
                <data>
                </data>
                <key>Replace</key>
                <data>
                AUJGV00A
                </data>
                <key>ReplaceMask</key>
                <data>
                </data>
                <key>Skip</key>
                <integer>0</integer>
                <key>TableLength</key>
                <integer>0</integer>
                <key>TableSignature</key>
                <data>
                </data>
            </dict>
            <dict>
                <key>Comment</key>
                <string>Mutex:XDHK=0</string>
                <key>Count</key>
                <integer>0</integer>
                <key>Enabled</key>
                <true/>
                <key>Find</key>
                <data>
                AVhESEsH
                </data>
                <key>Limit</key>
                <integer>0</integer>
                <key>Mask</key>
                <data>
                </data>
                <key>OemTableId</key>
                <data>
                </data>
                <key>Replace</key>
                <data>
                AVhESEsA
                </data>
                <key>ReplaceMask</key>
```

```
                <data>
                </data>
                <key>Skip</key>
                <integer>0</integer>
                <key>TableLength</key>
                <integer>0</integer>
                <key>TableSignature</key>
                <data>
                </data>
            </dict>
            <dict>
                <key>Comment</key>
                <string>Mutex:MWMI=0</string>
                <key>Count</key>
                <integer>0</integer>
                <key>Enabled</key>
                <true/>
                <key>Find</key>
                <data>
                AU1XTUkH
                </data>
                <key>Limit</key>
                <integer>0</integer>
                <key>Mask</key>
                <data>
                </data>
                <key>OemTableId</key>
                <data>
                </data>
                <key>Replace</key>
                <data>
                AU1XTUkA
                </data>
                <key>ReplaceMask</key>
                <data>
                </data>
                <key>Skip</key>
                <integer>0</integer>
                <key>TableLength</key>
                <integer>0</integer>
                <key>TableSignature</key>
                <data>
                </data>
            </dict>
            <dict>
                <key>Comment</key>
                <string>Mutex:NPWM=0</string>
                <key>Count</key>
                <integer>0</integer>
                <key>Enabled</key>
                <true/>
```

```
                <key>Find</key>
                <data>
                AU5QV00H
                </data>
                <key>Limit</key>
                <integer>0</integer>
                <key>Mask</key>
                <data>
                </data>
                <key>OemTableId</key>
                <data>
                </data>
                <key>Replace</key>
                <data>
                AU5QV00A
                </data>
                <key>ReplaceMask</key>
                <data>
                </data>
                <key>Skip</key>
                <integer>0</integer>
                <key>TableLength</key>
                <integer>0</integer>
                <key>TableSignature</key>
                <data>
                </data>
            </dict>
        </array>
    </dict>
</dict>
</plist>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/Pro
pertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>ACPI</key>
    <dict>
        <key>Patch</key>
        <array>
            <dict>
                <key>Comment</key>
                <string>disableBAT1:WTX30,WTX40,X260,X1C2014,X1C2016</string>
                <key>Count</key>
                <integer>1</integer>
                <key>Enabled</key>
                <false/>
                <key>Find</key>
                <data>
                X1NUQQ==
                </data>
                <key>Limit</key>
                <integer>0</integer>
                <key>Mask</key>
                <data>
                </data>
                <key>OemTableId</key>
                <data>
                </data>
                <key>Replace</key>
                <data>
                WFNUQQ==
                </data>
                <key>ReplaceMask</key>
                <data>
                </data>
                <key>Skip</key>
                <integer>12</integer>
                <key>TableLength</key>
                <integer>0</integer>
                <key>TableSignature</key>
                <data>
                RFNEVA==
                </data>
            </dict>
            <dict>
                <key>Comment</key>
                <string>disableBAT1:WTX50,WTX60,X1C2015</string>
                <key>Count</key>
                <integer>1</integer>
                <key>Enabled</key>
```

```
                <false/>
                <key>Find</key>
                <data>
                X1NUQQ==
                </data>
                <key>Limit</key>
                <integer>0</integer>
                <key>Mask</key>
                <data>
                </data>
                <key>OemTableId</key>
                <data>
                </data>
                <key>Replace</key>
                <data>
                WFNUQQ==
                </data>
                <key>ReplaceMask</key>
                <data>
                </data>
                <key>Skip</key>
                <integer>11</integer>
                <key>TableLength</key>
                <integer>0</integer>
                <key>TableSignature</key>
                <data>
                RFNEVA==
                </data>
            </dict>
            <dict>
                <key>Comment</key>
                <string>disableBAT1:T470,T470s</string>
                <key>Count</key>
                <integer>1</integer>
                <key>Enabled</key>
                <false/>
                <key>Find</key>
                <data>
                X1NUQQ==
                </data>
                <key>Limit</key>
                <integer>0</integer>
                <key>Mask</key>
                <data>
                </data>
                <key>OemTableId</key>
                <data>
                </data>
                <key>Replace</key>
                <data>
                WFNUQQ==
```

```
                <key>ReplaceMask</key>
                <data>
                </data>
                <key>Skip</key>
                <integer>36</integer>
                <key>TableLength</key>
                <integer>0</integer>
                <key>TableSignature</key>
                <data>
                RFNEVA==
                </data>
            </dict>
            <dict>
                <key>Comment</key>
                <string>disableBAT1:X270</string>
                <key>Count</key>
                <integer>1</integer>
                <key>Enabled</key>
                <false/>
                <key>Find</key>
                <data>
                X1NUQQ==
                </data>
                <key>Limit</key>
                <integer>0</integer>
                <key>Mask</key>
                <data>
                </data>
                <key>OemTableId</key>
                <data>
                </data>
                <key>Replace</key>
                <data>
                WFNUQQ==
                </data>
                <key>ReplaceMask</key>
                <data>
                </data>
                <key>Skip</key>
                <integer>35</integer>
                <key>TableLength</key>
                <integer>0</integer>
                <key>TableSignature</key>
                <data>
                RFNEVA==
                </data>
            </dict>
        </array>
    </dict>
</dict>
```

```
</plist>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/Pro
pertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>ACPI</key>
    <dict>
        <key>Patch</key>
        <array>
            <dict>
                <key>Comment</key>
                <string>_Q22 to XQ22</string>
                <key>Count</key>
                <integer>0</integer>
                <key>Enabled</key>
                <true/>
                <key>Find</key>
                <data>
                X1EyMg==
                </data>
                <key>Limit</key>
                <integer>0</integer>
                <key>Mask</key>
                <data>
                </data>
                <key>OemTableId</key>
                <data>
                </data>
                <key>Replace</key>
                <data>
                WFEyMg==
                </data>
                <key>ReplaceMask</key>
                <data>
                </data>
                <key>Skip</key>
                <integer>0</integer>
                <key>TableLength</key>
                <integer>0</integer>
                <key>TableSignature</key>
                <data>
                </data>
            </dict>
            <dict>
                <key>Comment</key>
                <string>_Q24 to XQ24</string>
                <key>Count</key>
                <integer>0</integer>
                <key>Enabled</key>
                <true/>
```

```
                <key>Find</key>
                <data>
                X1EyNA==
                </data>
                <key>Limit</key>
                <integer>0</integer>
                <key>Mask</key>
                <data>
                </data>
                <key>OemTableId</key>
                <data>
                </data>
                <key>Replace</key>
                <data>
                WFEyNA==
                </data>
                <key>ReplaceMask</key>
                <data>
                </data>
                <key>Skip</key>
                <integer>0</integer>
                <key>TableLength</key>
                <integer>0</integer>
                <key>TableSignature</key>
                <data>
                </data>
            </dict>
            <dict>
                <key>Comment</key>
                <string>_Q25 to XQ25</string>
                <key>Count</key>
                <integer>0</integer>
                <key>Enabled</key>
                <true/>
                <key>Find</key>
                <data>
                X1EyNQ==
                </data>
                <key>Limit</key>
                <integer>0</integer>
                <key>Mask</key>
                <data>
                </data>
                <key>OemTableId</key>
                <data>
                </data>
                <key>Replace</key>
                <data>
                WFEyNQ==
                </data>
                <key>ReplaceMask</key>
```

```
                <data>
                </data>
                <key>Skip</key>
                <integer>0</integer>
                <key>TableLength</key>
                <integer>0</integer>
                <key>TableSignature</key>
                <data>
                </data>
            </dict>
            <dict>
                <key>Comment</key>
                <string>_Q4A to XQ4A</string>
                <key>Count</key>
                <integer>0</integer>
                <key>Enabled</key>
                <true/>
                <key>Find</key>
                <data>
                X1E0QQ==
                </data>
                <key>Limit</key>
                <integer>0</integer>
                <key>Mask</key>
                <data>
                </data>
                <key>OemTableId</key>
                <data>
                </data>
                <key>Replace</key>
                <data>
                WFE0QQ==
                </data>
                <key>ReplaceMask</key>
                <data>
                </data>
                <key>Skip</key>
                <integer>0</integer>
                <key>TableLength</key>
                <integer>0</integer>
                <key>TableSignature</key>
                <data>
                </data>
            </dict>
            <dict>
                <key>Comment</key>
                <string>_Q4B to XQ4B</string>
                <key>Count</key>
                <integer>0</integer>
                <key>Enabled</key>
                <true/>
```

```
                <key>Find</key>
                <data>
                X1E0Qg==
                </data>
                <key>Limit</key>
                <integer>0</integer>
                <key>Mask</key>
                <data>
                </data>
                <key>OemTableId</key>
                <data>
                </data>
                <key>Replace</key>
                <data>
                WFE0Qg==
                </data>
                <key>ReplaceMask</key>
                <data>
                </data>
                <key>Skip</key>
                <integer>0</integer>
                <key>TableLength</key>
                <integer>0</integer>
                <key>TableSignature</key>
                <data>
                </data>
            </dict>
            <dict>
                <key>Comment</key>
                <string>_Q4C to XQ4C</string>
                <key>Count</key>
                <integer>0</integer>
                <key>Enabled</key>
                <true/>
                <key>Find</key>
                <data>
                X1E0Qw==
                </data>
                <key>Limit</key>
                <integer>0</integer>
                <key>Mask</key>
                <data>
                </data>
                <key>OemTableId</key>
                <data>
                </data>
                <key>Replace</key>
                <data>
                WFE0Qw==
                </data>
                <key>ReplaceMask</key>
```

```
                <data>
                </data>
                <key>Skip</key>
                <integer>0</integer>
                <key>TableLength</key>
                <integer>0</integer>
                <key>TableSignature</key>
                <data>
                </data>
            </dict>
            <dict>
                <key>Comment</key>
                <string>_Q4D to XQ4D</string>
                <key>Count</key>
                <integer>0</integer>
                <key>Enabled</key>
                <true/>
                <key>Find</key>
                <data>
                X1E0RA==
                </data>
                <key>Limit</key>
                <integer>0</integer>
                <key>Mask</key>
                <data>
                </data>
                <key>OemTableId</key>
                <data>
                </data>
                <key>Replace</key>
                <data>
                WFE0RA==
                </data>
                <key>ReplaceMask</key>
                <data>
                </data>
                <key>Skip</key>
                <integer>0</integer>
                <key>TableLength</key>
                <integer>0</integer>
                <key>TableSignature</key>
                <data>
                </data>
            </dict>
            <dict>
                <key>Comment</key>
                <string>BATW to XATW</string>
                <key>Count</key>
                <integer>0</integer>
                <key>Enabled</key>
                <true/>
```

```
                <key>Find</key>
                <data>
                QkFUVwE=
                </data>
                <key>Limit</key>
                <integer>0</integer>
                <key>Mask</key>
                <data>
                </data>
                <key>OemTableId</key>
                <data>
                </data>
                <key>Replace</key>
                <data>
                WEFUVwE=
                </data>
                <key>ReplaceMask</key>
                <data>
                </data>
                <key>Skip</key>
                <integer>0</integer>
                <key>TableLength</key>
                <integer>0</integer>
                <key>TableSignature</key>
                <data>
                </data>
            </dict>
            <dict>
                <key>Comment</key>
                <string>BFCC to XFCC</string>
                <key>Count</key>
                <integer>0</integer>
                <key>Enabled</key>
                <true/>
                <key>Find</key>
                <data>
                QkZDQwA=
                </data>
                <key>Limit</key>
                <integer>0</integer>
                <key>Mask</key>
                <data>
                </data>
                <key>OemTableId</key>
                <data>
                </data>
                <key>Replace</key>
                <data>
                WEZDQwA=
                </data>
                <key>ReplaceMask</key>
```

```
                <data>
                </data>
                <key>Skip</key>
                <integer>0</integer>
                <key>TableLength</key>
                <integer>0</integer>
                <key>TableSignature</key>
                <data>
                </data>
            </dict>
        </array>
    </dict>
</dict>
</plist>
```

```
// battery
// In config ACPI, _BIX renamed XBIX
// Find:     5F 42 49 58
// Replace:  58 42 49 58
//
// In config ACPI, SMBR renamed XMBR
// Find:     53 4D 42 52 0B
// Replace:  58 4D 42 52 0B
//
// In config ACPI, SMBW renamed XMBW
// Find:     53 4D 42 57 0D
// Replace:  58 4D 42 57 0D
//
// In config ACPI, ECSB renamed XCSB
// Find:     45 43 53 42 07
// Replace:  58 43 53 42 07
//
DefinitionBlock ("", "SSDT", 2, "OCLT", "BAT0", 0)
{
    External(_SB.PCI0.BAT0, DeviceObj)
    External(_SB.PCI0.LPCB.EC0, DeviceObj)
    External(_SB.PCI0.LPCB.EC0.BATP, MethodObj)
    External(_SB.PCI0.LPCB.EC0.GBTT, MethodObj)
    External(_SB.PCI0.BAT0._BIF, MethodObj)
    External(_SB.PCI0.BAT0.XBIX, MethodObj)
    External(_SB.PCI0.LPCB.EC0.XMBR, MethodObj)
    External(_SB.PCI0.LPCB.EC0.XMBW, MethodObj)
    External(_SB.PCI0.LPCB.EC0.XCSB, MethodObj)

    External(_SB.PCI0.BAT0.NBIX, PkgObj)
    External(_SB.PCI0.BAT0.PBIF, PkgObj)
    External(_SB.PCI0.BAT0.BIXT, PkgObj)
    //
    External(_SB.PCI0.LPCB.EC0.ECAV, MethodObj)
    External(_SB.PCI0.LPCB.EC0.RDBL, IntObj)
    External(_SB.PCI0.LPCB.EC0.RDWD, IntObj)
    External(_SB.PCI0.LPCB.EC0.RDBT, IntObj)
    External(_SB.PCI0.LPCB.EC0.RCBT, IntObj)
    External(_SB.PCI0.LPCB.EC0.RDQK, IntObj)
    External(_SB.PCI0.LPCB.EC0.WRBL, IntObj)
    External(_SB.PCI0.LPCB.EC0.WRWD, IntObj)
    External(_SB.PCI0.LPCB.EC0.WRBT, IntObj)
    External(_SB.PCI0.LPCB.EC0.SDBT, IntObj)
    External(_SB.PCI0.LPCB.EC0.WRQK, IntObj)
    External(_SB.PCI0.LPCB.EC0.SBBY, IntObj)
    //
    External(_SB.PCI0.LPCB.EC0.MUEC, MutexObj)
    External(_SB.PCI0.LPCB.EC0.PRTC, FieldUnitObj)
    External(_SB.PCI0.LPCB.EC0.ADDR, FieldUnitObj)
    External(_SB.PCI0.LPCB.EC0.CMDB, FieldUnitObj)
```

```
    External(_SB.PCI0.LPCB.EC0.BCNT, FieldUnitObj)
    External(_SB.PCI0.LPCB.EC0.DAT0, FieldUnitObj)
    External(_SB.PCI0.LPCB.EC0.DAT1, FieldUnitObj)

    External(_SB.PCI0.LPCB.EC0.PRT2, FieldUnitObj)
    External(_SB.PCI0.LPCB.EC0.ADD2, FieldUnitObj)
    External(_SB.PCI0.LPCB.EC0.CMD2, FieldUnitObj)
    External(_SB.PCI0.LPCB.EC0.BCN2, FieldUnitObj)
    External(_SB.PCI0.LPCB.EC0.DA20, FieldUnitObj)
    External(_SB.PCI0.LPCB.EC0.DA21, FieldUnitObj)

    External(_SB.PCI0.LPCB.EC0.SSTS, FieldUnitObj)
    External(_SB.PCI0.LPCB.EC0.SST2, FieldUnitObj)
    External(_SB.PCI0.LPCB.EC0.SWTC, MethodObj)
    //

    Method (B1B2, 2, NotSerialized)
    {
        ShiftLeft (Arg1, 8, Local0)
        Or (Arg0, Local0, Local0)
        Return (Local0)
    }

    Scope(_SB.PCI0.LPCB.EC0)
    {
        Method (RE1B, 1, NotSerialized)
        {
            OperationRegion(ERAM, EmbeddedControl, Arg0, 1)
            Field(ERAM, ByteAcc, NoLock, Preserve) { BYTE, 8 }
            Return(BYTE)
        }
        Method (RECB, 2, Serialized)
        {
            ShiftRight(Arg1, 3, Arg1)
            Name(TEMP, Buffer(Arg1) { })
            Add(Arg0, Arg1, Arg1)
            Store(0, Local0)
            While (LLess(Arg0, Arg1))
            {
                Store(RE1B(Arg0), Index(TEMP, Local0))
                Increment(Arg0)
                Increment(Local0)
            }
            Return(TEMP)
        }

        Method (WE1B, 2, NotSerialized)
        {
            OperationRegion(ERAM, EmbeddedControl, Arg0, 1)
            Field(ERAM, ByteAcc, NoLock, Preserve) { BYTE, 8 }
            Store(Arg1, BYTE)
```

```
        }
        Method (WECB, 3, Serialized)
        {
            ShiftRight(Arg1, 3, Arg1)
            Name(TEMP, Buffer(Arg1) { })
            Store(Arg2, TEMP)
            Add(Arg0, Arg1, Arg1)
            Store(0, Local0)
            While (LLess(Arg0, Arg1))
            {
                WE1B(Arg0, DerefOf(Index(TEMP, Local0)))
                Increment(Arg0)
                Increment(Local0)
            }
        }

        OperationRegion (BAM0, EmbeddedControl, 0x00, 0xFF)
        Field (BAM0, ByteAcc, Lock, Preserve)
        {
            Offset (0xBE),
                ,16,
                ,16,
                ,16,
            B030,8,B031,8,    //B0C3,   16,
        }
        OperationRegion (BAM1, EmbeddedControl, 0x18, 0x28)
        Field (BAM1, ByteAcc, NoLock, Preserve)
        {
            Offset (0x04),
            XXXX,    256,     //BDAT,   256,
        }
        Field (BAM1, ByteAcc, NoLock, Preserve)
        {
            Offset (0x04),
            DT20,8,DT21,8,    //DT2B,   16
        }

        OperationRegion (BAM2, EmbeddedControl, 0x40, 0x28)
        Field (BAM2, ByteAcc, NoLock, Preserve)
        {
            Offset (0x04),
            YYYY,    256,     //BDA2,   256,
        }
    }

    Scope(_SB.PCI0.BAT0)
    {
        Method (_BIX, 0, NotSerialized)
        {
          If (_OSI ("Darwin"))
          {
```

```
            If (LNot (^^LPCB.EC0.BATP (Zero)))
            {
                Return (\_SB.PCI0.BAT0.NBIX)
            }

            If (LEqual (^^LPCB.EC0.GBTT (Zero), 0xFF))
            {
                Return (\_SB.PCI0.BAT0.NBIX)
            }

            _BIF ()
                BIXT [One] = DerefOf (PBIF [Zero])
                BIXT [0x02] = DerefOf (PBIF [One])
                BIXT [0x03] = DerefOf (PBIF [0x02])
                BIXT [0x04] = DerefOf (PBIF [0x03])
                BIXT [0x05] = DerefOf (PBIF [0x04])
                BIXT [0x06] = DerefOf (PBIF [0x05])
                BIXT [0x07] = DerefOf (PBIF [0x06])
                BIXT [0x0E] = DerefOf (PBIF [0x07])
                BIXT [0x0F] = DerefOf (PBIF [0x08])
                BIXT [0x10] = DerefOf (PBIF [0x09])
                BIXT [0x11] = DerefOf (PBIF [0x0A])
                BIXT [0x12] = DerefOf (PBIF [0x0B])
                BIXT [0x13] = DerefOf (PBIF [0x0C])
            If ((DerefOf (BIXT [One]) == One))
            {
                    BIXT [One] = Zero
                    Local0 = DerefOf (BIXT [0x05])
                    BIXT [0x02] = (DerefOf (BIXT [0x02]) * Local0)
                    BIXT [0x03] = (DerefOf (BIXT [0x03]) * Local0)
                    BIXT [0x06] = (DerefOf (BIXT [0x06]) * Local0)
                    BIXT [0x07] = (DerefOf (BIXT [0x07]) * Local0)
                    BIXT [0x0E] = (DerefOf (BIXT [0x0E]) * Local0)
                    BIXT [0x0F] = (DerefOf (BIXT [0x0F]) * Local0)
                    Divide (DerefOf (BIXT [0x02]), 0x03E8, Local0, BIXT [0x02])
                    Divide (DerefOf (BIXT [0x03]), 0x03E8, Local0, BIXT [0x03])
                    Divide (DerefOf (BIXT [0x06]), 0x03E8, Local0, BIXT [0x06])
                    Divide (DerefOf (BIXT [0x07]), 0x03E8, Local0, BIXT [0x07])
                    Divide (DerefOf (BIXT [0x0E]), 0x03E8, Local0, BIXT [0x0E])
                    Divide (DerefOf (BIXT [0x0F]), 0x03E8, Local0, BIXT [0x0F])
            }

            BIXT [0x08] = B1B2 (^^LPCB.EC0.B030, ^^LPCB.EC0.B031)
            BIXT [0x09] = 0x0001869F
            Return (\_SB.PCI0.BAT0.BIXT)
          }
          Else
          {
            Return (\_SB.PCI0.BAT0.XBIX())
          }
        }
```

```
    }
    //
    Scope(\_SB.PCI0.LPCB.EC0)
    {
        Method (SMBR, 3, Serialized)
        {
          If (_OSI ("Darwin"))
          {
            Store (Package (0x03)
                {
                    0x07,
                    Zero,
                    Zero
                }, Local0)
            If (LNot (ECAV ()))
            {
                Return (Local0)
            }

            If (LNotEqual (Arg0, RDBL))
            {
                If (LNotEqual (Arg0, RDWD))
                {
                    If (LNotEqual (Arg0, RDBT))
                    {
                        If (LNotEqual (Arg0, RCBT))
                        {
                            If (LNotEqual (Arg0, RDQK))
                            {
                                Return (Local0)
                            }
                        }
                    }
                }
            }

            Acquire (MUEC, 0xFFFF)
            Store (\_SB.PCI0.LPCB.EC0.PRTC, Local1)
            Store (Zero, Local2)
            While (LNotEqual (Local1, Zero))
            {
                Stall (0x0A)
                Increment (Local2)
                If (LGreater (Local2, 0x03E8))
                {
                    Store (\_SB.PCI0.LPCB.EC0.SBBY, Index (Local0, Zero))
                    Store (Zero, Local1)
                }
                Else
                {
                    Store (\_SB.PCI0.LPCB.EC0.PRTC, Local1)
```

```
                }
            }

            If (LLessEqual (Local2, 0x03E8))
            {
                ShiftLeft (Arg1, One, Local3)
                Or (Local3, One, Local3)
                Store (Local3, ADDR)
                If (LNotEqual (Arg0, RDQK))
                {
                    If (LNotEqual (Arg0, RCBT))
                    {
                        Store (Arg2, CMDB)
                    }
                }

                WECB (0x04, 0x0100, Zero) //BDAT
                Store (Arg0, \_SB.PCI0.LPCB.EC0.PRTC)
                Store (SWTC (Arg0), Index (Local0, Zero))
                If (LEqual (DerefOf (Index (Local0, Zero)), Zero))
                {
                    If (LEqual (Arg0, RDBL))
                    {
                        Store (\_SB.PCI0.LPCB.EC0.BCNT, Index (Local0, One))
                        Store (RECB(0x04,0x100), Index (Local0, 0x02))//BDAT
                    }

                    If (LEqual (Arg0, RDWD))
                    {
                        Store (0x02, Index (Local0, One))
                        Store (B1B2 (\_SB.PCI0.LPCB.EC0.DT20, \_SB.PCI0.LPCB.EC0.DT
21), Index (Local0, 0x02))
                        //DT2B
                    }

                    If (LEqual (Arg0, RDBT))
                    {
                        Store (One, Index (Local0, One))
                        Store (\_SB.PCI0.LPCB.EC0.DAT0, Index (Local0, 0x02))
                    }

                    If (LEqual (Arg0, RCBT))
                    {
                        Store (One, Index (Local0, One))
                        Store (\_SB.PCI0.LPCB.EC0.DAT0, Index (Local0, 0x02))
                    }
                }
            }

            Release (MUEC)
            Return (Local0)
```

```
          }
          Else
          {
            Return (\_SB.PCI0.LPCB.EC0.XMBR(Arg0, Arg1, Arg2))
          }
        }

        Method (SMBW, 5, Serialized)
        {
          If (_OSI ("Darwin"))
          {
            Store (Package (0x01)
                {
                    0x07
                }, Local0)
            If (LNot (ECAV ()))
            {
                Return (Local0)
            }

            If (LNotEqual (Arg0, WRBL))
            {
                If (LNotEqual (Arg0, WRWD))
                {
                    If (LNotEqual (Arg0, WRBT))
                    {
                        If (LNotEqual (Arg0, SDBT))
                        {
                            If (LNotEqual (Arg0, WRQK))
                            {
                                Return (Local0)
                            }
                        }
                    }
                }
            }

            Acquire (MUEC, 0xFFFF)
            Store (\_SB.PCI0.LPCB.EC0.PRTC, Local1)
            Store (Zero, Local2)
            While (LNotEqual (Local1, Zero))
            {
                Stall (0x0A)
                Increment (Local2)
                If (LGreater (Local2, 0x03E8))
                {
                    Store (\_SB.PCI0.LPCB.EC0.SBBY, Index (Local0, Zero))
                    Store (Zero, Local1)
                }
                Else
                {
```

```
                    Store (\_SB.PCI0.LPCB.EC0.PRTC, Local1)
                }
            }

            If (LLessEqual (Local2, 0x03E8))
            {
                WECB (0x04, 0x0100, Zero) //BDAT
                ShiftLeft (Arg1, One, Local3)
                Store (Local3, ADDR)
                If (LNotEqual (Arg0, WRQK))
                {
                    If (LNotEqual (Arg0, SDBT))
                    {
                        Store (Arg2, CMDB)
                    }
                }

                If (LEqual (Arg0, WRBL))
                {
                    Store (Arg3, \_SB.PCI0.LPCB.EC0.BCNT)
                    WECB (0x04, 0x0100, Arg4) //BDAT
                }

                If (LEqual (Arg0, WRWD))
                {
                    Store (Arg4, B1B2 (\_SB.PCI0.LPCB.EC0.DT20, \_SB.PCI0.LPCB.EC0.
DT21))
                    //DT2B
                }

                If (LEqual (Arg0, WRBT))
                {
                    Store (Arg4, \_SB.PCI0.LPCB.EC0.DAT0)
                }

                If (LEqual (Arg0, SDBT))
                {
                    Store (Arg4, \_SB.PCI0.LPCB.EC0.DAT0)
                }

                Store (Arg0, \_SB.PCI0.LPCB.EC0.PRTC)
                Store (SWTC (Arg0), Index (Local0, Zero))
            }

            Release (MUEC)
            Return (Local0)
          }
          Else
          {
            Return (\_SB.PCI0.LPCB.EC0.XMBW(Arg0, Arg1, Arg2, Arg3, Arg4))
          }
```

```
        }

        Method (ECSB, 7, NotSerialized)
        {
          If (_OSI ("Darwin"))
          {
            Local1 = Package (0x05)
                {
                    0x11,
                    Zero,
                    Zero,
                    Zero,
                    Buffer (0x20){}
                }
            If ((Arg0 > One))
            {
                Return (Local1)
            }

            If (ECAV ())
            {
                Acquire (MUEC, 0xFFFF)
                If ((Arg0 == Zero))
                {
                    Local0 = \_SB.PCI0.LPCB.EC0.PRTC /* \_SB_.PCI0.LPCB.EC0_.PRTC *
/
                }
                Else
                {
                    Local0 = \_SB.PCI0.LPCB.EC0.PRT2 /* \_SB_.PCI0.LPCB.EC0_.PRT2 *
/
                }

                Local2 = Zero
                While ((Local0 != Zero))
                {
                    Stall (0x0A)
                    Local2++
                    If ((Local2 > 0x03E8))
                    {
                        Local1 [Zero] = \_SB.PCI0.LPCB.EC0.SBBY /* \_SB_.PCI0.LPCB.
EC0_.SBBY */
                        Local0 = Zero
                    }
                    ElseIf ((Arg0 == Zero))
                    {
                        Local0 = \_SB.PCI0.LPCB.EC0.PRTC /* \_SB_.PCI0.LPCB.EC0_.PR
TC */
                    }
                    Else
                    {
```

```
                        Local0 = \_SB.PCI0.LPCB.EC0.PRT2 /* \_SB_.PCI0.LPCB.EC0_.PR
T2 */
                    }
                }

                If ((Local2 <= 0x03E8))
                {
                    If ((Arg0 == Zero))
                    {
                        ADDR = Arg2
                        CMDB = Arg3
                        If (((Arg1 == 0x0A) || (Arg1 == 0x0B)))
                        {
                            \_SB.PCI0.LPCB.EC0.BCNT = DerefOf (Arg6 [Zero])
                            WECB (0x04, 0x0100, DerefOf (Arg6 [One])) //BDAT
                        }
                        Else
                        {
                            \_SB.PCI0.LPCB.EC0.DAT0 = Arg4
                            \_SB.PCI0.LPCB.EC0.DAT1 = Arg5
                        }

                        \_SB.PCI0.LPCB.EC0.PRTC = Arg1
                    }
                    Else
                    {
                        ADD2 = Arg2
                        CMD2 = Arg3
                        If (((Arg1 == 0x0A) || (Arg1 == 0x0B)))
                        {
                            \_SB.PCI0.LPCB.EC0.BCN2 = DerefOf (Arg6 [Zero])
                            WECB (0x04, 0x0100, DerefOf (Arg6 [One])) //BDA2
                        }
                        Else
                        {
                            \_SB.PCI0.LPCB.EC0.DA20 = Arg4
                            \_SB.PCI0.LPCB.EC0.DA21 = Arg5
                        }

                        \_SB.PCI0.LPCB.EC0.PRT2 = Arg1
                    }

                    Local0 = 0x7F
                    If ((Arg0 == Zero))
                    {
                        While (\_SB.PCI0.LPCB.EC0.PRTC)
                        {
                            Sleep (One)
                            Local0--
                        }
                    }
```

```
                    Else
                    {
                        While (\_SB.PCI0.LPCB.EC0.PRT2)
                        {
                            Sleep (One)
                            Local0--
                        }
                    }

                    If (Local0)
                    {
                        If ((Arg0 == Zero))
                        {
                            Local0 = \_SB.PCI0.LPCB.EC0.SSTS /* \_SB_.PCI0.LPCB.EC0
_.SSTS */
                            Local1 [One] = \_SB.PCI0.LPCB.EC0.DAT0 /* \_SB_.PCI0.LP
CB.EC0_.DAT0 */
                            Local1 [0x02] = \_SB.PCI0.LPCB.EC0.DAT1 /* \_SB_.PCI0.L
PCB.EC0_.DAT1 */
                            Local1 [0x03] = \_SB.PCI0.LPCB.EC0.BCNT /* \_SB_.PCI0.L
PCB.EC0_.BCNT */
                            Local1 [0x04] = RECB(0x04,0x100) //BDAT

                        }
                        Else
                        {
                            Local0 = \_SB.PCI0.LPCB.EC0.SST2 /* \_SB_.PCI0.LPCB.EC0
_.SST2 */
                            Local1 [One] = \_SB.PCI0.LPCB.EC0.DA20 /* \_SB_.PCI0.LP
CB.EC0_.DA20 */
                            Local1 [0x02] = \_SB.PCI0.LPCB.EC0.DA21 /* \_SB_.PCI0.L
PCB.EC0_.DA21 */
                            Local1 [0x03] = \_SB.PCI0.LPCB.EC0.BCN2 /* \_SB_.PCI0.L
PCB.EC0_.BCN2 */
                            Store (RECB(0x04,0x100), Local1 [0x04])//BDA2
                        }

                        Local0 &= 0x1F
                        If (Local0)
                        {
                            Local0 += 0x10
                        }

                        Local1 [Zero] = Local0
                    }
                    Else
                    {
                        Local1 [Zero] = 0x10
                    }
                }
```

```
                Release (MUEC)
            }

            Return (Local1)
          }
          Else
          {
            Return (\_SB.PCI0.LPCB.EC0.XCSB(Arg0, Arg1, Arg2, Arg3, Arg4, Arg5, Arg
6))
          }
        }
    }
}
//EOF
```

```
// battery
// In config ACPI, BTIF renamed XTIF
// Find:     42544946 09
// Replace:  58544946 09
//
// In config ACPI, BTST renamed XTST
// Find:     42545354 0A
// Replace:  58545354 0A
//
// In config ACPI, ITLB renamed XTLB
// Find:     49544C42 00
// Replace:  58544C42 00
//
// In config ACPI, GBTI renamed XBTI
// Find:     47425449 01
// Replace:  58425449 01
//
// In config ACPI, GBTC renamed XBTC
// Find:     47425443 00
// Replace:  58425443 00
//
// In config ACPI, SBTC renamed XSTC
// Find:     53425443 03
// Replace:  58535443 03
//
// In config ACPI, GCGC renamed XCGC
// Find:     47434743 08
// Replace:  58434743 08
//
DefinitionBlock ("", "SSDT", 2, "OCLT", "BAT0", 0)
{
    External(_SB.PCI0.LPCB.EC0, DeviceObj)
    External(_SB.NDBS, PkgObj)
    External(_SB.NBST, PkgObj)
    External(_SB.NBTI, PkgObj)
    External(_SB.PCI0.LPCB.EC0.NGBF, IntObj)
    External(_SB.PCI0.LPCB.EC0.NGBT, IntObj)
    External(_SB.PCI0.LPCB.EC0.NLB1, IntObj)
    External(_SB.PCI0.LPCB.EC0.NLB2, IntObj)
    External(_SB.PCI0.LPCB.EC0.NLO2, IntObj)
    External(_SB.PCI0.LPCB.EC0.NDCB, IntObj)
    External(_SB.PCI0.LPCB.EC0.ECRG, IntObj)
    External(_SB.PCI0.LPCB.EC0.BTMX, MutexObj)
    External(_SB.PCI0.LPCB.EC0.ECMX, MutexObj)
    External(_SB.PCI0.LPCB.EC0.BSEL, FieldUnitObj)
    External(_SB.PCI0.LPCB.EC0.BST, FieldUnitObj)
    External(_SB.PCI0.LPCB.EC0.LB1, FieldUnitObj)
    External(_SB.PCI0.LPCB.EC0.LB2, FieldUnitObj)
    External(_SB.PCI0.LPCB.EC0.BATP, FieldUnitObj)
    External(_SB.PCI0.LPCB.EC0.BRCC, FieldUnitObj)
```

```
    External(_SB.PCI0.LPCB.EC0.BRCV, FieldUnitObj)
    External(_SB.PCI0.LPCB.EC0.BATN, FieldUnitObj)
    External(_SB.PCI0.LPCB.EC0.INAC, FieldUnitObj)
    External(_SB.PCI0.LPCB.EC0.INCH, FieldUnitObj)
    External(_SB.PCI0.LPCB.EC0.IDIS, FieldUnitObj)
    External(_SB.PCI0.LPCB.EC0.PSSB, FieldUnitObj)
    External(_SB.PCI0.LPCB.EC0.BTDR, MethodObj)
    External(_SB.PCI0.LPCB.EC0.BSTA, MethodObj)
    External(_SB.PCI0.LPCB.EC0.GBSS, MethodObj)
    External(_SB.PCI0.LPCB.EC0.GACS, MethodObj)
    External(_SB.PCI0.LPCB.EC0.GBMF, MethodObj)
    External(_SB.PCI0.LPCB.EC0.GCTL, MethodObj)
    External(_SB.PCI0.LPCB.EC0.GDNM, MethodObj)
    External(_SB.PCI0.LPCB.EC0.GDCH, MethodObj)
    External(_SB.PCI0.LPCB.EC0.XTIF, MethodObj)
    External(_SB.PCI0.LPCB.EC0.XTST, MethodObj)
    External(_SB.PCI0.LPCB.EC0.XTLB, MethodObj)
    External(_SB.PCI0.LPCB.EC0.XBTC, MethodObj)
    External(_SB.PCI0.LPCB.EC0.XSTC, MethodObj)
    External(_SB.PCI0.LPCB.EC0.XBTI, MethodObj)
    External(_TZ.XCGC, MethodObj)

    Method (B1B2, 2, NotSerialized)
    {
        ShiftLeft (Arg1, 8, Local0)
        Or (Arg0, Local0, Local0)
        Return (Local0)
    }

    Scope(_SB.PCI0.LPCB.EC0)
    {
        OperationRegion (ERM0, EmbeddedControl, 0x00, 0xFF)
        Field (ERM0, ByteAcc, Lock, Preserve)
        {
            Offset (0x87),
            ,    8,
            ,    8,
            BDC0,8,BDC1,8, //BDC,    16,
            Offset (0x8D),
            BFC0,8,BFC1,8, //BFC,    16,
            RTE0,8,RTE1,8, //BRTE,   16,
            ,    1,
            Offset (0x92),
            BME0,8,BME1,8, //BME,    16,
            ,    8,
            BDV0,8,BDV1,8, //BDV,    16,
            CV10,8,CV11,8, //BCV1,   16,
            ,    4,
            Offset (0x9B),
            ATE0,8,ATE1,8, //BATE,   16,
            BPR0,8,BPR1,8, //BPR,    16,
```

```
            BCR0,8,BCR1,8, //BCR,     16,
            BRC0,8,BRC1,8, //BRC,     16,
            BCC0,8,BCC1,8, //BCC,     16,
            BPV0,8,BPV1,8, //BPV,     16,
            CV20,8,CV21,8, //BCV2,    16,
            CV30,8,CV31,8, //BCV3,    16,
            CV40,8,CV41,8, //BCV4,    16,
            ,    16,
            ATF0,8,ATF1,8, //BATF,    16,
            ,    16,
            AXC0,8,AXC1,8, //MAXC,    16,
            ,   8,
            ,   1,
            ,   1,
                ,   2,
            ,   4,
            STS0,8,STS1,8, //BSTS,    16,
            ,   8,
            Offset (0xC9),
            BSN0,8,BSN1,8, //BSN,     16,
            DAT0,8,DAT1,8, //BDAT,    16,
            ,    8,
            Offset (0xD5),
            ,   8,
            ,   8,
            ,   8,
            ,   8,
            ,   8,
            ,   8,
            ,   8,
            ,   8,
            ,   4,
            ,    4,
            ,   16,
            CBT0,8,CBT1,8, //CBT,     16,
        }
    }


    Scope(\_SB.PCI0.LPCB.EC0)
    {
        Method (BTIF, 1, Serialized)
        {
            If (_OSI ("Darwin"))
            {
                Local7 = (One << Arg0)
                BTDR (One)
                If ((BSTA (Local7) == 0x0F))
                {
                    Return (0xFF)
                }
```

```
                Acquire (BTMX, 0xFFFF)
                Local0 = NGBF /* \_SB_.PCI0.LPCB.EC0_.NGBF */
                Release (BTMX)
                If (((Local0 & Local7) == Zero))
                {
                    Return (Zero)
                }

                NBST [Arg0] = NDBS /* \_SB_.NDBS */
                Acquire (BTMX, 0xFFFF)
                NGBT |= Local7
                Release (BTMX)
                Acquire (ECMX, 0xFFFF)
                If (ECRG)
                {
                    BSEL = Arg0
                    Local0 = B1B2 (BFC0,BFC1) /* \_SB_.PCI0.LPCB.EC0_.BFC_ */
                    DerefOf (NBTI [Arg0]) [One] = Local0
                    DerefOf (NBTI [Arg0]) [0x02] = Local0
                    DerefOf (NBTI [Arg0]) [0x04] = B1B2 (BDV0,BDV1) /* \_SB_.PCI0.L
PCB.EC0_.BDV_ */
                    Local0 = (B1B2 (BFC0,BFC1) * NLB1) /* \_SB_.PCI0.LPCB.EC0_.NLB1
 */
                    Divide (Local0, 0x64, Local3, Local4)
                    DerefOf (NBTI [Arg0]) [0x05] = Local4
                    Local0 = (B1B2 (BFC0,BFC1) * NLO2) /* \_SB_.PCI0.LPCB.EC0_.NLO2
 */
                    Divide (Local0, 0x64, Local3, Local4)
                    DerefOf (NBTI [Arg0]) [0x06] = Local4
                    Local0 = B1B2 (BSN0,BSN1) /* \_SB_.PCI0.LPCB.EC0_.BSN_ */
                    Local1 = B1B2 (DAT0,DAT1) /* \_SB_.PCI0.LPCB.EC0_.BDAT */
                }

                Release (ECMX)
                Local2 = GBSS (Local0, Local1)
                DerefOf (NBTI [Arg0]) [0x0A] = Local2
                Acquire (BTMX, 0xFFFF)
                NGBF &= ~Local7
                Release (BTMX)
                Return (Zero)
            }
            Else
            {
                Return (\_SB.PCI0.LPCB.EC0.XTIF (Arg0))
            }
        }

        Method (BTST, 2, Serialized)
        {
            If (_OSI ("Darwin"))
```

```
            {
                Local7 = (One << Arg0)
                BTDR (One)
                If ((BSTA (Local7) == 0x0F))
                {
                    NBST [Arg0] = Package (0x04)
                        {
                            Zero,
                            0xFFFFFFFF,
                            0xFFFFFFFF,
                            0xFFFFFFFF
                        }
                    Return (0xFF)
                }

                Acquire (BTMX, 0xFFFF)
                If (Arg1)
                {
                    NGBT = 0xFF
                }

                Local0 = NGBT /* \_SB_.PCI0.LPCB.EC0_.NGBT */
                Release (BTMX)
                If (((Local0 & Local7) == Zero))
                {
                    Return (Zero)
                }

                Acquire (ECMX, 0xFFFF)
                If (ECRG)
                {
                    BSEL = Arg0
                    Local0 = BST /* \_SB_.PCI0.LPCB.EC0_.BST_ */
                    Local3 = B1B2 (BPR0,BPR1) /* \_SB_.PCI0.LPCB.EC0_.BPR_ */
                    DerefOf (NBST [Arg0]) [0x02] = B1B2 (BRC0,BRC1) /* \_SB_.PCI0.L
PCB.EC0_.BRC_ */
                    DerefOf (NBST [Arg0]) [0x03] = B1B2 (BPV0,BPV1) /* \_SB_.PCI0.L
PCB.EC0_.BPV_ */
                }

                Release (ECMX)
                If ((GACS () == One))
                {
                    Local0 &= 0xFFFFFFFFFFFFFFFE
                }
                Else
                {
                    Local0 &= 0xFFFFFFFFFFFFFFFD
                }

                If ((Local0 & One))
```

```
                {
                    Acquire (BTMX, 0xFFFF)
                    NDCB = Local7
                    Release (BTMX)
                }

                DerefOf (NBST [Arg0]) [Zero] = Local0
                If ((Local0 & One))
                {
                    If (((Local3 < 0x0190) || (Local3 > 0x1964)))
                    {
                        Local5 = DerefOf (DerefOf (NBST [Arg0]) [One])
                        If (((Local5 < 0x0190) || (Local5 > 0x1964)))
                        {
                            Local3 = 0x0D7A
                        }
                        Else
                        {
                            Local3 = Local5
                        }
                    }
                }
                ElseIf (((Local0 & 0x02) == Zero))
                {
                    Local3 = Zero
                }

                DerefOf (NBST [Arg0]) [One] = Local3
                Acquire (BTMX, 0xFFFF)
                NGBT &= ~Local7
                Release (BTMX)
                Return (Zero)
            }
            Else
            {
                Return (\_SB.PCI0.LPCB.EC0.XTST (Arg0, Arg1))
            }
        }

        Method (ITLB, 0, NotSerialized)
        {
            If (_OSI ("Darwin"))
            {
                Local0 = (B1B2 (BFC0,BFC1) * NLB1) /* \_SB_.PCI0.LPCB.EC0_.NLB1 */
                Divide (Local0, 0x64, Local3, Local4)
                Divide ((Local4 + 0x09), 0x0A, Local0, Local1)
                Local0 = (B1B2 (BFC0,BFC1) * NLB2) /* \_SB_.PCI0.LPCB.EC0_.NLB2 */
                Divide (Local0, 0x64, Local3, Local4)
                Divide ((Local4 + 0x09), 0x0A, Local0, Local2)
                If (ECRG)
                {
```

```
                    LB1 = Local1
                    LB2 = Local2
                }
            }
            Else
            {
                \_SB.PCI0.LPCB.EC0.XTLB ()
            }
        }

        Method (GBTI, 1, NotSerialized)
        {
            If (_OSI ("Darwin"))
            {
                Debug = "Enter getbattinfo"
                Acquire (ECMX, 0xFFFF)
                If (ECRG)
                {
                    If ((BATP & (One << Arg0)))
                    {
                        BSEL = Arg0
                        Local0 = Package (0x02)
                            {
                                Zero,
                                Buffer (0x6B){}
                            }
                        DerefOf (Local0 [One]) [Zero] = B1B2 (BDC0,BDC1) /* \_SB_.P
CI0.LPCB.EC0_.BDC_ */
                        DerefOf (Local0 [One]) [One] = (B1B2 (BDC0,BDC1) >> 0x08
                            )
                        DerefOf (Local0 [One]) [0x02] = B1B2 (BFC0,BFC1) /* \_SB_.P
CI0.LPCB.EC0_.BFC_ */
                        DerefOf (Local0 [One]) [0x03] = (B1B2 (BFC0,BFC1) >> 0x08
                            )
                        DerefOf (Local0 [One]) [0x04] = B1B2 (BRC0,BRC1) /* \_SB_.P
CI0.LPCB.EC0_.BRC_ */
                        DerefOf (Local0 [One]) [0x05] = (B1B2 (BRC0,BRC1) >> 0x08
                            )
                        DerefOf (Local0 [One]) [0x06] = B1B2 (BME0,BME1) /* \_SB_.P
CI0.LPCB.EC0_.BME_ */
                        DerefOf (Local0 [One]) [0x07] = (B1B2 (BME0,BME1) >> 0x08
                            )
                        DerefOf (Local0 [One]) [0x08] = B1B2 (BCC0,BCC1) /* \_SB_.P
CI0.LPCB.EC0_.BCC_ */
                        DerefOf (Local0 [One]) [0x09] = (B1B2 (BCC0,BCC1) >> 0x08
                            )
                        Local1 = B1B2 (CBT0,CBT1) /* \_SB_.PCI0.LPCB.EC0_.CBT_ */
                        Local1 -= 0x0AAC
                        Divide (Local1, 0x0A, Local2, Local3)
                        DerefOf (Local0 [One]) [0x0A] = Local3
                        DerefOf (Local0 [One]) [0x0B] = (Local3 >> 0x08
```

```
                        )
                    DerefOf (Local0 [One]) [0x0C] = B1B2 (BPV0,BPV1) /* \_SB_.P
CI0.LPCB.EC0_.BPV_ */
                    DerefOf (Local0 [One]) [0x0D] = (B1B2 (BPV0,BPV1) >> 0x08
                        )
                    Local1 = B1B2 (BPR0,BPR1) /* \_SB_.PCI0.LPCB.EC0_.BPR_ */
                    If (Local1)
                    {
                        If ((B1B2 (STS0,STS1) & 0x40))
                        {
                            Local1 = (~Local1 + One)
                            Local1 &= 0xFFFF
                        }
                    }

                    DerefOf (Local0 [One]) [0x0E] = Local1
                    DerefOf (Local0 [One]) [0x0F] = (Local1 >> 0x08
                        )
                    DerefOf (Local0 [One]) [0x10] = B1B2 (BDV0,BDV1) /* \_SB_.P
CI0.LPCB.EC0_.BDV_ */
                    DerefOf (Local0 [One]) [0x11] = (B1B2 (BDV0,BDV1) >> 0x08
                        )
                    DerefOf (Local0 [One]) [0x12] = B1B2 (STS0,STS1) /* \_SB_.P
CI0.LPCB.EC0_.BSTS */
                    DerefOf (Local0 [One]) [0x13] = (B1B2 (STS0,STS1) >> 0x08
                        )
                    DerefOf (Local0 [One]) [0x14] = B1B2 (CV10,CV11) /* \_SB_.P
CI0.LPCB.EC0_.BCV1 */
                    DerefOf (Local0 [One]) [0x15] = (B1B2 (CV10,CV11) >> 0x08
                        )
                    DerefOf (Local0 [One]) [0x16] = B1B2 (CV20,CV21) /* \_SB_.P
CI0.LPCB.EC0_.BCV2 */
                    DerefOf (Local0 [One]) [0x17] = (B1B2 (CV20,CV21) >> 0x08
                        )
                    DerefOf (Local0 [One]) [0x18] = B1B2 (CV30,CV31) /* \_SB_.P
CI0.LPCB.EC0_.BCV3 */
                    DerefOf (Local0 [One]) [0x19] = (B1B2 (CV30,CV31) >> 0x08
                        )
                    DerefOf (Local0 [One]) [0x1A] = B1B2 (CV40,CV41) /* \_SB_.P
CI0.LPCB.EC0_.BCV4 */
                    DerefOf (Local0 [One]) [0x1B] = (B1B2 (CV40,CV41) >> 0x08
                        )
                    CreateField (DerefOf (Local0 [One]), 0xE0, 0x80, BTSN)
                    BTSN = GBSS (B1B2 (BSN0,BSN1), B1B2 (DAT0,DAT1))
                    Local1 = GBMF ()
                    Local2 = SizeOf (Local1)
                    CreateField (DerefOf (Local0 [One]), 0x0160, (Local2 * 0x08
), BMAN)
                    BMAN = Local1
                    Local2 += 0x2C
                    CreateField (DerefOf (Local0 [One]), (Local2 * 0x08), 0x80,
```

```
    CLBL)
                        CLBL = GCTL (Zero)
                        Local2 += 0x11
                        CreateField (DerefOf (Local0 [One]), (Local2 * 0x08), 0x38,
    DNAM)
                        DNAM = GDNM (Zero)
                        Local2 += 0x07
                        CreateField (DerefOf (Local0 [One]), (Local2 * 0x08), 0x20,
    DCHE)
                        DCHE = GDCH (Zero)
                        Local2 += 0x04
                        CreateField (DerefOf (Local0 [One]), (Local2 * 0x08), 0x10,
    BMAC)
                        BMAC = Zero
                        Local2 += 0x02
                        CreateField (DerefOf (Local0 [One]), (Local2 * 0x08), 0x10,
    BMAD)
                        BMAD = B1B2 (DAT0,DAT1) /* \_SB_.PCI0.LPCB.EC0_.BDAT */
                        Local2 += 0x02
                        CreateField (DerefOf (Local0 [One]), (Local2 * 0x08), 0x10,
    BCCU)
                        BCCU = BRCC /* \_SB_.PCI0.LPCB.EC0_.BRCC */
                        Local2 += 0x02
                        CreateField (DerefOf (Local0 [One]), (Local2 * 0x08), 0x10,
    BCVO)
                        BCVO = BRCV /* \_SB_.PCI0.LPCB.EC0_.BRCV */
                        Local2 += 0x02
                        CreateField (DerefOf (Local0 [One]), (Local2 * 0x08), 0x10,
    BAVC)
                        Local1 = B1B2 (BCR0,BCR1) /* \_SB_.PCI0.LPCB.EC0_.BCR_ */
                        If (Local1)
                        {
                            If ((B1B2 (STS0,STS1) & 0x40))
                            {
                                Local1 = (~Local1 + One)
                                Local1 &= 0xFFFF
                            }
                        }

                        BAVC = Local1
                        Local2 += 0x02
                        CreateField (DerefOf (Local0 [One]), (Local2 * 0x08), 0x10,
    RTTE)
                        RTTE = B1B2 (RTE0,RTE1) /* \_SB_.PCI0.LPCB.EC0_.BRTE */
                        Local2 += 0x02
                        CreateField (DerefOf (Local0 [One]), (Local2 * 0x08), 0x10,
    ATTE)
                        RTTE = B1B2 (ATE0,ATE1) /* \_SB_.PCI0.LPCB.EC0_.BATE */
                        Local2 += 0x02
                        CreateField (DerefOf (Local0 [One]), (Local2 * 0x08), 0x10,
    ATTF)
```

```
                        RTTE = B1B2 (ATF0,ATF1) /* \_SB_.PCI0.LPCB.EC0_.BATF */
                        Local2 += 0x02
                        CreateField (DerefOf (Local0 [One]), (Local2 * 0x08), 0x08,
NOBS)
                        NOBS = BATN /* \_SB_.PCI0.LPCB.EC0_.BATN */
                    }
                    Else
                    {
                        Local0 = Package (0x01)
                            {
                                0x34
                            }
                    }
                }
                Else
                {
                    Local0 = Package (0x01)
                        {
                            0x0D
                        }
                }

                Release (ECMX)
                Return (Local0)
            }
            Else
            {
                Return (\_SB.PCI0.LPCB.EC0.XBTI (Arg0))
            }
        }

        Method (GBTC, 0, NotSerialized)
        {
            If (_OSI ("Darwin"))
            {
                Debug = "Enter GetBatteryControl"
                Acquire (ECMX, 0xFFFF)
                If (ECRG)
                {
                    Local0 = Package (0x02)
                        {
                            Zero,
                            Buffer (0x04){}
                        }
                    If ((BATP & One))
                    {
                        BSEL = Zero
                        DerefOf (Local0 [One]) [Zero] = Zero
                        If ((((INAC == Zero) && (INCH == Zero)) && (IDIS == Zero)))
                        {
                            DerefOf (Local0 [One]) [Zero] = Zero
```

```
                        }
                        ElseIf (((((INAC == Zero) && (INCH == 0x02)) && (
                            IDIS == One)) && (B1B2 (AXC0,AXC1) == Zero)))
                        {
                            DerefOf (Local0 [One]) [Zero] = One
                        }
                        ElseIf (((INAC == One) && (IDIS == 0x02)))
                        {
                            DerefOf (Local0 [One]) [Zero] = 0x02
                        }
                        ElseIf (((((INAC == Zero) && (INCH == 0x02)) && (
                            IDIS == One)) && (B1B2 (AXC0,AXC1) == 0xFA)))
                        {
                            DerefOf (Local0 [One]) [Zero] = 0x03
                        }
                        ElseIf (((INAC == Zero) && (INCH == 0x03)))
                        {
                            DerefOf (Local0 [One]) [Zero] = 0x04
                        }
                    }
                    Else
                    {
                        DerefOf (Local0 [One]) [Zero] = 0xFF
                    }

                    If ((BATP & 0x02))
                    {
                        BSEL = One
                        DerefOf (Local0 [One]) [One] = Zero
                        If ((((INAC == Zero) && (INCH == Zero)) && (IDIS == Zero)))
                        {
                            DerefOf (Local0 [One]) [One] = Zero
                        }
                        ElseIf (((((INAC == Zero) && (INCH == One)) && (
                            IDIS == 0x02)) && (B1B2 (AXC0,AXC1) == Zero)))
                        {
                            DerefOf (Local0 [One]) [One] = One
                        }
                        ElseIf (((INAC == One) && (IDIS == One)))
                        {
                            DerefOf (Local0 [One]) [One] = 0x02
                        }
                        ElseIf (((((INAC == Zero) && (INCH == One)) && (
                            IDIS == 0x02)) && (B1B2 (AXC0,AXC1) == 0xFA)))
                        {
                            DerefOf (Local0 [One]) [One] = 0x03
                        }
                        ElseIf (((INAC == Zero) && (INCH == 0x03)))
                        {
                            DerefOf (Local0 [One]) [One] = 0x04
                        }
```

```
                }
                Else
                {
                    DerefOf (Local0 [One]) [One] = 0xFF
                }
            }
            Else
            {
                Local0 = Package (0x02)
                    {
                        0x35,
                        Zero
                    }
            }

            Release (ECMX)
            Return (Local0)
        }
        Else
        {
            Return (\_SB.PCI0.LPCB.EC0.XBTC ())
        }
    }

    Method (SBTC, 3, NotSerialized)
    {
        If (_OSI ("Darwin"))
        {
            Debug = "Enter SetBatteryControl"
            Acquire (ECMX, 0xFFFF)
            If (ECRG)
            {
                Local0 = Arg2
                Debug = Local0
                Local4 = Package (0x01)
                    {
                        0x06
                    }
                Local1 = Zero
                Local2 = Zero
                Local1 = DerefOf (Local0 [Zero])
                If ((Local1 == Zero))
                {
                    Debug = "battery 0"
                    If ((BATP & One))
                    {
                        Local2 = DerefOf (Local0 [One])
                        If ((Local2 == Zero))
                        {
                            INCH = Zero
                            IDIS = Zero
```

```
                    INAC = Zero
                    B1B2 (AXC0, AXC1) = Package (){Zero}
                    PSSB = One
                    Local4 = Package (0x01)
                        {
                            Zero
                        }
                }

                If ((Local2 == One))
                {
                    INAC = Zero
                    INCH = 0x02
                    IDIS = One
                    B1B2 (AXC0,AXC1) = Package (){Zero}
                    PSSB = Zero
                    Local4 = Package (0x01)
                        {
                            Zero
                        }
                }

                If ((Local2 == 0x02))
                {
                    INAC = One
                    INCH = One
                    IDIS = 0x02
                    PSSB = Zero
                    Local4 = Package (0x01)
                        {
                            Zero
                        }
                }

                If ((Local2 == 0x03))
                {
                    INCH = 0x02
                    IDIS = One
                    INAC = Zero
                    B1B2 (AXC0,AXC1) = Package (){0xFA}
                    PSSB = Zero
                    Local4 = Package (0x01)
                        {
                            Zero
                        }
                }

                If ((Local2 == 0x04))
                {
                    B1B2 (AXC0,AXC1) = Package (){0xFA}
                    Local4 = Package (0x01)
```

```
                            {
                                Zero
                            }
                        }

                        If ((Local2 == 0x05))
                        {
                            INAC = Zero
                            INCH = 0x03
                            Local4 = Package (0x01)
                                {
                                    Zero
                                }
                        }
                    }
                    Else
                    {
                        Local4 = Package (0x01)
                            {
                                0x34
                            }
                    }
                }

                If ((Local1 == One))
                {
                    If ((BATP & 0x02))
                    {
                        Debug = "battery 1"
                        Local2 = DerefOf (Local0 [One])
                        If ((Local2 == Zero))
                        {
                            INCH = Zero
                            IDIS = Zero
                            INAC = Zero
                            B1B2 (AXC0,AXC1) = Package (){Zero}
                            PSSB = One
                            Local4 = Package (0x01)
                                {
                                    Zero
                                }
                        }

                        If ((Local2 == One))
                        {
                            INAC = Zero
                            INCH = One
                            IDIS = 0x02
                            B1B2 (AXC0,AXC1) = Package (){Zero}
                            PSSB = Zero
                            Local4 = Package (0x01)
```

```
                {
                    Zero
                }
            }

            If ((Local2 == 0x02))
            {
                INAC = One
                INCH = 0x02
                IDIS = One
                PSSB = Zero
                Local4 = Package (0x01)
                    {
                        Zero
                    }
            }

            If ((Local2 == 0x03))
            {
                INCH = One
                IDIS = 0x02
                INAC = Zero
                B1B2 (AXC0,AXC1) = Package (){0xFA}
                PSSB = Zero
                Local4 = Package (0x01)
                    {
                        Zero
                    }
            }

            If ((Local2 == 0x04))
            {
                INCH = Zero
                IDIS = Zero
                INAC = Zero
                Local4 = Package (0x01)
                    {
                        Zero
                    }
            }

            If ((Local2 == 0x05))
            {
                INAC = Zero
                INCH = 0x03
                Local4 = Package (0x01)
                    {
                        Zero
                    }
            }
        }
```

```
                        Else
                        {
                            Local4 = Package (0x01)
                                {
                                    0x34
                                }
                        }
                    }
                }

                Release (ECMX)
                Return (Local4)
            }
            Else
            {
                Return (\_SB.PCI0.LPCB.EC0.XSTC (Arg0, Arg1, Arg2))
            }
        }
    }
    Scope (_TZ)
    {
        Method (GCGC, 0, Serialized)
        {
            If (_OSI ("Darwin"))
            {
                Name (LTMP, Buffer (0x02){})
                If (\_SB.PCI0.LPCB.EC0.ECRG)
                {
                    Acquire (\_SB.PCI0.LPCB.EC0.ECMX, 0xFFFF)
                    LTMP = B1B2 (\_SB.PCI0.LPCB.EC0.BPR0, \_SB.PCI0.LPCB.EC0.BPR1)
                    Release (\_SB.PCI0.LPCB.EC0.ECMX)
                }

                Return (LTMP) /* \_TZ_.GCGC.LTMP */
            }
            Else
            {
                Return (\_TZ.XCGC ())
            }
        }
    }
}
```

```
/* battery
 * In config, ADJT to XDJT
 * Find:    41444A54 08
 * Replace: 58444A54 08
 *
 * In config, CLRI to XLRI
 * Find:    434C5249 08
 * Replace: 584C5249 08
 *
 * In config, _BST to XBST
 * Find:    5F425354 00
 * Replace: 58425354 00
 *
 * In config, UPBI to XPBI
 * Find:    55504249 00
 * Replace: 58504249 00
 *
 * In config, UPBS to XPBS
 * Find:    55504253 00
 * Replace: 58504253 00
 *
 * In config, SMRD to XMRD
 * Find:    534D5244 04
 * Replace: 584D5244 04
 *
 * In config, SMWR to XMWR
 * Find:    534D5752 04
 * Replace: 584D5752 04
 */
DefinitionBlock ("", "SSDT", 2, "OCLT", "BATT", 0)
{
    External (_SB.BAT0, DeviceObj)
    External (_SB.BAT0._STA, MethodObj)
    External (_SB.BAT0.FABL, IntObj)
    External (_SB.BAT0.IVBS, MethodObj)
    External (_SB.BAT0.PBIF, PkgObj)
    External (_SB.BAT0.PBST, PkgObj)
    External (_SB.BAT0.UPUM, MethodObj)
    External (_SB.BAT0.XBST, MethodObj)
    External (_SB.BAT0.XPBI, MethodObj)
    External (_SB.BAT0.XPBS, MethodObj)
    External (_SB.GBFE, MethodObj)
    External (_SB.LID0._LID, MethodObj)
    External (_SB.PBFE, MethodObj)
    External (_SB.PCI0.ACEL, DeviceObj)
    External (_SB.PCI0.ACEL._STA, MethodObj)
    External (_SB.PCI0.ACEL.CNST, IntObj)
    External (_SB.PCI0.ACEL.XDJT, MethodObj)
    External (_SB.PCI0.ACEL.XLRI, MethodObj)
    External (_SB.PCI0.LPCB.EC0, DeviceObj)
```

```
    External (_SB.PCI0.LPCB.EC0.BACR, FieldUnitObj)
    External (_SB.PCI0.LPCB.EC0.BCNT, FieldUnitObj)
    External (_SB.PCI0.LPCB.EC0.BVHB, FieldUnitObj)
    External (_SB.PCI0.LPCB.EC0.BVLB, FieldUnitObj)
    External (_SB.PCI0.LPCB.EC0.ECOK, IntObj)
    External (_SB.PCI0.LPCB.EC0.MBNH, FieldUnitObj)
    External (_SB.PCI0.LPCB.EC0.MBST, FieldUnitObj)
    External (_SB.PCI0.LPCB.EC0.MBTS, FieldUnitObj)
    External (_SB.PCI0.LPCB.EC0.MUT0, MutexObj)
    External (_SB.PCI0.LPCB.EC0.PLGS, FieldUnitObj)
    External (_SB.PCI0.LPCB.EC0.SMAD, FieldUnitObj)
    External (_SB.PCI0.LPCB.EC0.SMB0, FieldUnitObj)
    External (_SB.PCI0.LPCB.EC0.SMCM, FieldUnitObj)
    External (_SB.PCI0.LPCB.EC0.SMPR, FieldUnitObj)
    External (_SB.PCI0.LPCB.EC0.SMST, FieldUnitObj)
    External (_SB.PCI0.LPCB.EC0.SW2S, FieldUnitObj)
    External (_SB.PCI0.LPCB.EC0.XMRD, MethodObj)
    External (_SB.PCI0.LPCB.EC0.XMWR, MethodObj)
    External (PWRS, FieldUnitObj)

    Method (B1B2, 2, NotSerialized)
    {
        Local0 = (Arg1 << 0x08)
        Local0 |= Arg0
        Return (Local0)
    }

    Scope (\_SB.PCI0.LPCB.EC0)
    {
        OperationRegion (ERM2, EmbeddedControl, Zero, 0xFF)
        Field (ERM2, ByteAcc, Lock, Preserve)
        {
            Offset (0x70),
            ADC0,   8,
            ADC1,   8,
            FCC0,   8,
            FCC1,   8,
            Offset (0x82),
            Offset (0x83),
            CUR0,   8,
            CUR1,   8,
            BRM0,   8,
            BRM1,   8,
            BCV0,   8,
            BCV1,   8
        }

        Field (ERM2, ByteAcc, NoLock, Preserve)
        {
            Offset (0x04),
            MW00,   8,
```

```
            MW01,   8
        }

        Method (RE1B, 1, NotSerialized)
        {
            OperationRegion (ERM2, EmbeddedControl, Arg0, One)
            Field (ERM2, ByteAcc, NoLock, Preserve)
            {
                BYTE,   8
            }

            Return (BYTE) /* \_SB.PCI0.LPCB.EC0.RE1B.BYTE */
        }

        Method (RECB, 2, Serialized)
        {
            Arg1 = ((Arg1 + 0x07) >> 0x03)
            Name (TEMP, Buffer (Arg1){})
            Arg1 += Arg0
            Local0 = Zero
            While ((Arg0 < Arg1))
            {
                TEMP [Local0] = RE1B (Arg0)
                Arg0++
                Local0++
            }

            Return (TEMP) /* \_SB.PCI0.LPCB.EC0.RECB.TEMP */
        }

        Method (WE1B, 2, NotSerialized)
        {
            OperationRegion (ERM2, EmbeddedControl, Arg0, One)
            Field (ERM2, ByteAcc, NoLock, Preserve)
            {
                BYTE,   8
            }

            BYTE = Arg1
        }

        Method (WECB, 3, Serialized)
        {
            Arg1 = ((Arg1 + 0x07) >> 0x03)
            Name (TEMP, Buffer (Arg1){})
            TEMP = Arg2
            Arg1 += Arg0
            Local0 = Zero
            While ((Arg0 < Arg1))
            {
                WE1B (Arg0, DerefOf (TEMP [Local0]))
```

```
                Arg0++
                Local0++
            }
        }

        Method (SMRD, 4, NotSerialized)
        {
            If (_OSI ("Darwin"))
            {
                If (!ECOK)
                {
                    Return (0xFF)
                }

                If ((Arg0 != 0x07))
                {
                    If ((Arg0 != 0x09))
                    {
                        If ((Arg0 != 0x0B))
                        {
                            If ((Arg0 != 0x47))
                            {
                                If ((Arg0 != 0xC7))
                                {
                                    Return (0x19)
                                }
                            }
                        }
                    }
                }

                Acquire (MUT0, 0xFFFF)
                Local0 = 0x04
                While ((Local0 > One))
                {
                    SMST &= 0x40
                    SMCM = Arg2
                    SMAD = Arg1
                    SMPR = Arg0
                    Local3 = Zero
                    While (!Local1 = (SMST & 0xBF))
                    {
                        Sleep (0x02)
                        Local3++
                        If ((Local3 == 0x32))
                        {
                            SMST &= 0x40
                            SMCM = Arg2
                            SMAD = Arg1
                            SMPR = Arg0
                            Local3 = Zero
```

```
                        }
                    }

                    If ((Local1 == 0x80))
                    {
                        Local0 = Zero
                    }
                    Else
                    {
                        Local0--
                    }
                }

                If (Local0)
                {
                    Local0 = (Local1 & 0x1F)
                }
                Else
                {
                    If ((Arg0 == 0x07))
                    {
                        Arg3 = SMB0 /* External reference */
                    }

                    If ((Arg0 == 0x47))
                    {
                        Arg3 = SMB0 /* External reference */
                    }

                    If ((Arg0 == 0xC7))
                    {
                        Arg3 = SMB0 /* External reference */
                    }

                    If ((Arg0 == 0x09))
                    {
                        Arg3 = B1B2 (MW00, MW00)
                    }

                    If ((Arg0 == 0x0B))
                    {
                        Local3 = BCNT /* External reference */
                        Local2 = 0x20
                        If ((Local3 > Local2))
                        {
                            Local3 = Local2
                        }

                        If ((Local3 < 0x09))
                        {
                            Local2 = RECB (0x04, 0x40)
```

```
                    }
                    ElseIf ((Local3 < 0x11))
                    {
                        Local2 = RECB (0x04, 0x80)
                    }
                    ElseIf ((Local3 < 0x19))
                    {
                        Local2 = RECB (0x04, 0xC0)
                    }
                    Else
                    {
                        Local2 = RECB (0x04, 0x0100)
                    }

                    Local3++
                    Local4 = Buffer (Local3){}
                    Local3--
                    Local5 = Zero
                    Name (OEMS, Buffer (0x46){})
                    ToBuffer (Local2, OEMS) /* \_SB.PCI0.LPCB.EC0.SMRD.OEMS */
                    While ((Local3 > Local5))
                    {
                        GBFE (OEMS, Local5, RefOf (Local6))
                        PBFE (Local4, Local5, Local6)
                        Local5++
                    }

                    PBFE (Local4, Local5, Zero)
                    Arg3 = Local4
                }
            }

            Release (MUT0)
            Return (Local0)
        }
        Else
        {
            Return (\_SB.PCI0.LPCB.EC0.XMRD (Arg0, Arg1, Arg2, Arg3))
        }
    }

    Method (SMWR, 4, NotSerialized)
    {
        If (_OSI ("Darwin"))
        {
            If (!ECOK)
            {
                Return (0xFF)
            }

            If ((Arg0 != 0x06))
```

```
                {
                    If ((Arg0 != 0x08))
                    {
                        If ((Arg0 != 0x0A))
                        {
                            If ((Arg0 != 0x46))
                            {
                                If ((Arg0 != 0xC6))
                                {
                                    Return (0x19)
                                }
                            }
                        }
                    }
                }

                Acquire (MUT0, 0xFFFF)
                Local0 = 0x04
                While ((Local0 > One))
                {
                    If ((Arg0 == 0x06))
                    {
                        SMB0 = Arg3
                    }

                    If ((Arg0 == 0x46))
                    {
                        SMB0 = Arg3
                    }

                    If ((Arg0 == 0xC6))
                    {
                        SMB0 = Arg3
                    }

                    If ((Arg0 == 0x08))
                    {
                        MW00 = Arg3
                        MW01 = (Arg3 >> 0x08)
                    }

                    If ((Arg0 == 0x0A))
                    {
                        WECB (0x04, 0x0100, Arg3)
                    }

                    SMST &= 0x40
                    SMCM = Arg2
                    SMAD = Arg1
                    SMPR = Arg0
                    Local3 = Zero
```

```
                    While (!Local1 = (SMST & 0xBF))
                    {
                        Sleep (0x02)
                        Local3++
                        If ((Local3 == 0x32))
                        {
                            SMST &= 0x40
                            SMCM = Arg2
                            SMAD = Arg1
                            SMPR = Arg0
                            Local3 = Zero
                        }
                    }

                    If ((Local1 == 0x80))
                    {
                        Local0 = Zero
                    }
                    Else
                    {
                        Local0--
                    }
                }

                If (Local0)
                {
                    Local0 = (Local1 & 0x1F)
                }

                Release (MUT0)
                Return (Local0)
            }
            Else
            {
                Return (\_SB.PCI0.LPCB.EC0.XMWR (Arg0, Arg1, Arg2, Arg3))
            }
        }
    }

    Scope (\_SB.BAT0)
    {
        Method (_BST, 0, NotSerialized)  // _BST: Battery Status
        {
            If (_OSI ("Darwin"))
            {
                If ((^^PCI0.LPCB.EC0.ECOK == One))
                {
                    If (^^PCI0.LPCB.EC0.MBTS)
                    {
                        UPBS ()
                    }
```

```
                    Else
                    {
                        IVBS ()
                    }
                }
                Else
                {
                    IVBS ()
                }

                Return (PBST) /* External reference */
            }
            Else
            {
                Return (\_SB.BAT0.XBST ())
            }
        }

        Method (UPBI, 0, NotSerialized)
        {
            If (_OSI ("Darwin"))
            {
                Local5 = B1B2 (^^PCI0.LPCB.EC0.FCC0, ^^PCI0.LPCB.EC0.FCC1)
                If ((Local5 && !(Local5 & 0x8000)))
                {
                    Local5 >>= 0x05
                    Local5 <<= 0x05
                    PBIF [One] = Local5
                    PBIF [0x02] = Local5
                    Local2 = (Local5 / 0x64)
                    Local2 += One
                    If ((B1B2 (^^PCI0.LPCB.EC0.ADC0, ^^PCI0.LPCB.EC0.ADC1) < 0x0C80
))
                    {
                        Local4 = (Local2 * 0x0E)
                        PBIF [0x05] = (Local4 + 0x02)
                        Local4 = (Local2 * 0x09)
                        PBIF [0x06] = (Local4 + 0x02)
                        Local4 = (Local2 * 0x0C)
                    }
                    Else
                    {
                        Local4 = (Local2 * 0x0C)
                        PBIF [0x05] = (Local4 + 0x02)
                        Local4 = (Local2 * 0x07)
                        PBIF [0x06] = (Local4 + 0x02)
                        Local4 = (Local2 * 0x0A)
                    }

                    FABL = (Local4 + 0x02)
                }
```

```
            If (^^PCI0.LPCB.EC0.MBNH)
            {
                Local0 = ^^PCI0.LPCB.EC0.BVLB /* External reference */
                Local1 = ^^PCI0.LPCB.EC0.BVHB /* External reference */
                Local1 <<= 0x08
                Local0 |= Local1
                PBIF [0x04] = Local0
                PBIF [0x09] = "OANI$"
                PBIF [0x0B] = "NiMH"
            }
            Else
            {
                Local0 = ^^PCI0.LPCB.EC0.BVLB /* External reference */
                Local1 = ^^PCI0.LPCB.EC0.BVHB /* External reference */
                Local1 <<= 0x08
                Local0 |= Local1
                PBIF [0x04] = Local0
                Sleep (0x32)
                PBIF [0x0B] = "LION"
            }

            PBIF [0x09] = "Primary"
            UPUM ()
            PBIF [Zero] = One
        }
        Else
        {
            \_SB.BAT0.XPBI ()
        }
    }

    Method (UPBS, 0, NotSerialized)
    {
        If (_OSI ("Darwin"))
        {
            Local0 = B1B2 (^^PCI0.LPCB.EC0.CUR0, ^^PCI0.LPCB.EC0.CUR1)
            If ((Local0 & 0x8000))
            {
                If ((Local0 == 0xFFFF))
                {
                    PBST [One] = 0xFFFFFFFF
                }
                Else
                {
                    Local1 = ~Local0
                    Local1++
                    Local3 = (Local1 & 0xFFFF)
                    PBST [One] = Local3
                }
            }
```

```
                Else
                {
                    PBST [One] = Local0
                }

                Local5 = B1B2 (^^PCI0.LPCB.EC0.BRM0, ^^PCI0.LPCB.EC0.BRM1)
                If (!(Local5 & 0x8000))
                {
                    Local5 >>= 0x05
                    Local5 <<= 0x05
                    If ((Local5 != DerefOf (PBST [0x02])))
                    {
                        PBST [0x02] = Local5
                    }
                }

                If ((!^^PCI0.LPCB.EC0.SW2S && (^^PCI0.LPCB.EC0.BACR == One)))
                {
                    PBST [0x02] = FABL /* External reference */
                }

                PBST [0x03] = B1B2 (^^PCI0.LPCB.EC0.BCV0, ^^PCI0.LPCB.EC0.BCV1)
                PBST [Zero] = ^^PCI0.LPCB.EC0.MBST /* External reference */
            }
            Else
            {
                \_SB.BAT0.XPBS ()
            }
        }
    }

    Scope (\_SB.PCI0.ACEL)
    {
        Method (ADJT, 0, Serialized)
        {
            If (_OSI ("Darwin"))
            {
                If (_STA ())
                {
                    If ((^^LPCB.EC0.ECOK == One))
                    {
                        Local0 = ^^LPCB.EC0.SW2S /* External reference */
                    }
                    Else
                    {
                        Local0 = PWRS /* External reference */
                    }

                    If (((^^^LID0._LID () == Zero) && (Local0 == Zero)))
                    {
                        If ((CNST != One))
```

```
                    {
                        CNST = One
                        Sleep (0x0BB8)
                        ^^LPCB.EC0.SMWR (0xC6, 0x50, 0x36, 0x14)
                        ^^LPCB.EC0.SMWR (0xC6, 0x50, 0x37, 0x10)
                        ^^LPCB.EC0.SMWR (0xC6, 0x50, 0x34, 0x2A)
                        ^^LPCB.EC0.SMWR (0xC6, 0x50, 0x24, Zero)
                        ^^LPCB.EC0.PLGS = Zero
                        ^^LPCB.EC0.SMWR (0xC6, 0x50, 0x22, 0x20)
                    }
                }
                ElseIf ((CNST != Zero))
                {
                    CNST = Zero
                    ^^LPCB.EC0.SMWR (0xC6, 0x50, 0x36, One)
                    ^^LPCB.EC0.SMWR (0xC6, 0x50, 0x37, 0x50)
                    ^^LPCB.EC0.SMWR (0xC6, 0x50, 0x34, 0x7F)
                    ^^LPCB.EC0.SMWR (0xC6, 0x50, 0x24, 0x02)
                    ^^LPCB.EC0.PLGS = One
                    ^^LPCB.EC0.SMWR (0xC6, 0x50, 0x22, 0x40)
                }
            }
        }
        Else
        {
            \_SB.PCI0.ACEL.XDJT ()
        }
    }

    Method (CLRI, 0, Serialized)
    {
        If (_OSI ("Darwin"))
        {
            Local0 = Zero
            If ((^^LPCB.EC0.ECOK == One))
            {
                If ((^^LPCB.EC0.SW2S == Zero))
                {
                    If ((^^^BAT0._STA () == 0x1F))
                    {
                        If ((B1B2 (^^LPCB.EC0.BRM0, ^^LPCB.EC0.BRM1) <= 0x96))
                        {
                            Local0 = One
                        }
                    }
                }
            }

            Return (Local0)
        }
        Else
```

```
            {
                Return (\_SB.PCI0.ACEL.XLRI ())
            }
        }
    }
}
```

```
//// battery
// In config ACPI, GACW to XGAC
// Find:     14254741 4357
// Replace:  14255847 4143
//
// In config ACPI, GBAW  to XGAB
// Find:     45044742 4157
// Replace:  45045847 4142
//
// In config ACPI, BTIF to XTIF
// Find:     42544946 0979
// Replace:  58544946 0979
//
// In config ACPI, BTST to XTST
// Find:     42545354 0A
// Replace:  58545354 0A
//
// In config ACPI, ITLB to XITL
// Find:     4D044954 4C42
// Replace:  4D045849 544C
//
//
//
// In config ACPI, GBTI to XBTI
// Find:     47425449 01
// Replace:  58425449 01
//
// In config ACPI, GBTC to XGBU
// Find:     42204742 5443
// Replace:  42205847 4255
//
// In config ACPI, SBTC to XSBT
// Find:     49265342 5443
// Replace:  49265853 4254
//
// In config ACPI, GCGC to XGCG
// Find:     4F074743 4743
// Replace:  4F075847 4347
//
DefinitionBlock ("", "SSDT", 2, "OCLT", "BAT0", 0x00000000)
{
    External (_SB.PCI0.LPCB.EC0, DeviceObj)
    External (_SB.PCI0.LPCB.EC0.BATN, FieldUnitObj)
    External (_SB.PCI0.LPCB.EC0.BATP, FieldUnitObj)
    External (_SB.PCI0.LPCB.EC0.BRCC, FieldUnitObj)
    External (_SB.PCI0.LPCB.EC0.BRCV, FieldUnitObj)
    External (_SB.PCI0.LPCB.EC0.BSEL, FieldUnitObj)
    External (_SB.PCI0.LPCB.EC0.BST, FieldUnitObj)
    External (_SB.PCI0.LPCB.EC0.BSTA, MethodObj)    // 1 Arguments
    External (_SB.PCI0.LPCB.EC0.BTDR, MethodObj)    // 1 Arguments
```

```
    External (_SB.PCI0.LPCB.EC0.BTIF, MethodObj)    // 1 Arguments
    External (_SB.PCI0.LPCB.EC0.BTMX, MutexObj)
    External (_SB.PCI0.LPCB.EC0.ECMX, MutexObj)
    External (_SB.PCI0.LPCB.EC0.ECRG, IntObj)
    External (_SB.PCI0.LPCB.EC0.GACS, MethodObj)    // 0 Arguments
    External (_SB.PCI0.LPCB.EC0.GBMF, MethodObj)    // 0 Arguments
    External (_SB.PCI0.LPCB.EC0.GBSS, MethodObj)    // 2 Arguments
    External (_SB.PCI0.LPCB.EC0.GCTL, MethodObj)    // 1 Arguments
    External (_SB.PCI0.LPCB.EC0.GDCH, MethodObj)    // 1 Arguments
    External (_SB.PCI0.LPCB.EC0.GDNM, MethodObj)    // 1 Arguments
    External (_SB.PCI0.LPCB.EC0.IDIS, FieldUnitObj)
    External (_SB.PCI0.LPCB.EC0.INAC, FieldUnitObj)
    External (_SB.PCI0.LPCB.EC0.INCH, FieldUnitObj)
    External (_SB.PCI0.LPCB.EC0.LB1, FieldUnitObj)
    External (_SB.PCI0.LPCB.EC0.LB2, FieldUnitObj)
    External (_SB.NBST, PkgObj)
    External (_SB.NBTI, PkgObj)
    External (_SB.NDBS, PkgObj)
    External (_SB.PCI0.LPCB.EC0.NDCB, IntObj)
    External (_SB.PCI0.LPCB.EC0.NGBF, IntObj)
    External (_SB.PCI0.LPCB.EC0.NGBT, IntObj)
    External (_SB.PCI0.LPCB.EC0.NLB1, IntObj)
    External (_SB.PCI0.LPCB.EC0.NLB2, IntObj)
    External (_SB.PCI0.LPCB.EC0.NLO2, IntObj)
    External (_SB.PCI0.LPCB.EC0.PSSB, FieldUnitObj)
    //
    External (_SB.PCI0.LPCB.EC0.XGAC, MethodObj)
    External (_SB.PCI0.LPCB.EC0.XGAB, MethodObj)
    External (_SB.PCI0.LPCB.EC0.XTIF, MethodObj)
    External (_SB.PCI0.LPCB.EC0.XTST, MethodObj)
    External (_SB.PCI0.LPCB.EC0.XITL, MethodObj)
    External (_SB.PCI0.LPCB.EC0.XBTI, MethodObj)
    External (_SB.PCI0.LPCB.EC0.XGBU, MethodObj)
    External (_SB.PCI0.LPCB.EC0.XSBT, MethodObj)
    External (_TZ.XGCG, MethodObj)

    Method (B1B2, 2, NotSerialized)
    {
        Return ((Arg0 | (Arg1 << 0x08)))
    }

    Scope (_SB.PCI0.LPCB.EC0)
    {
            OperationRegion (ERM2, EmbeddedControl, Zero, 0xFF)
            Field (ERM2, ByteAcc, NoLock, Preserve)
            {
                Offset (0x89),
                BDC1,   8,
                BDC2,   8,
                Offset (0x8D),
                BFC1,   8,
```

```
                BFC2,   8,
                BRT1,   8,
                BRT2,   8,
                Offset (0x92),
                BME1,   8,
                BME2,   8,
                Offset (0x95),
                BDV1,   8,
                BDV2,   8,
                BCV1,   8,
                BCV2,   8,
                Offset (0x9B),
                BA01,   8,
                BA02,   8,
                BPR1,   8,
                BPR2,   8,
                BCR1,   8,
                BCR2,   8,
                BRC1,   8,
                BRC2,   8,
                BCC1,   8,
                BCC2,   8,
                BPV1,   8,
                BPV2,   8,
                BC01,   8,
                BC02,   8,
                BC03,   8,
                BC04,   8,
                BC05,   8,
                BC06,   8,
                Offset (0xAF),
                BA03,   8,
                BA04,   8,
                Offset (0xB3),
                MAX1,   8,
                MAX2,   8,
                Offset (0xB7),
                BST1,   8,
                BST2,   8,
                Offset (0xC9),
                BSN1,   8,
                BSN2,   8,
                BDA1,   8,
                BDA2,   8,
                Offset (0xDE),
                CCB1,   8,
                CCB2,   8,
                CBT1,   8,
                CBT2,   8,
                Offset (0xF9),
                ACP1,   8,
```

```
            ACP2,   8
        }

        Method (GACW, 0, NotSerialized)
        {
          If (_OSI ("Darwin"))
          {
            Local0 = Zero
            Acquire (ECMX, 0xFFFF)
            If (\_SB.PCI0.LPCB.EC0.ECRG)
            {
                Local0 = B1B2 (ACP1, ACP2)
            }

            Release (ECMX)
            Return (Local0)
          }
          Else
          {
            Return (\_SB.PCI0.LPCB.EC0.XGAC())
          }
        }

        Method (GBAW, 0, NotSerialized)
        {
          If (_OSI ("Darwin"))
          {
            Local0 = Zero
            Acquire (ECMX, 0xFFFF)
            If (\_SB.PCI0.LPCB.EC0.ECRG)
            {
                Local1 = B1B2 (BDV1, BDV2)
                Local2 = B1B2 (BDC1, BDC2)
                Local0 = (Local1 * Local2)
                Divide (Local0, 0x000F4240, Local3, Local0)
                If ((Local3 >= 0x0007A120))
                {
                    Local0++
                }
            }

            Release (ECMX)
            Return (Local0)
          }
          Else
          {
            Return (\_SB.PCI0.LPCB.EC0.XGAB())
          }
        }

        Method (BTIF, 1, Serialized)
```

```
            {
              If (_OSI ("Darwin"))
              {
                Local7 = (One << Arg0)
                BTDR (One)
                If ((BSTA (Local7) == 0x0F))
                {
                    Return (0xFF)
                }

                Acquire (BTMX, 0xFFFF)
                Local0 = \_SB.PCI0.LPCB.EC0_.NGBF /* External reference */
                Release (BTMX)
                If (((Local0 & Local7) == Zero))
                {
                    Return (Zero)
                }

                NBST [Arg0] = \_SB.NDBS /* External reference */
                Acquire (BTMX, 0xFFFF)
                NGBT |= Local7
                Release (BTMX)
                Acquire (ECMX, 0xFFFF)
                If (\_SB.PCI0.LPCB.EC0.ECRG)
                {
                    BSEL = Arg0
                    Local0 = B1B2 (BFC1, BFC2)
                    DerefOf (NBTI [Arg0]) [One] = Local0
                    DerefOf (NBTI [Arg0]) [0x02] = Local0
                    DerefOf (NBTI [Arg0]) [0x04] = B1B2 (BDV1, BDV2)
                    Local0 = (B1B2 (BFC1, BFC2) * NLB1) /* External reference */
                    Local4 = (Local0 / 0x64)
                    DerefOf (NBTI [Arg0]) [0x05] = Local4
                    Local0 = (B1B2 (BFC1, BFC2) * NLO2) /* External reference */
                    Local4 = (Local0 / 0x64)
                    DerefOf (NBTI [Arg0]) [0x06] = Local4
                    Local0 = B1B2 (BSN1, BSN2)
                    Local1 = B1B2 (BDA1, BDA2)
                }

                Release (ECMX)
                Local2 = GBSS (Local0, Local1)
                DerefOf (NBTI [Arg0]) [0x0A] = Local2
                Acquire (BTMX, 0xFFFF)
                NGBF &= ~Local7
                Release (BTMX)
                Return (Zero)
              }
              Else
              {
                Return (\_SB.PCI0.LPCB.EC0.XTIF(Arg0))
```

```
            }
        }

        Method (BTST, 2, Serialized)
        {
            If (_OSI ("Darwin"))
            {
                Local7 = (One << Arg0)
                BTDR (One)
                If ((BSTA (Local7) == 0x0F))
                {
                    NBST [Arg0] = Package (0x04)
                        {
                            Zero,
                            0xFFFFFFFF,
                            0xFFFFFFFF,
                            0xFFFFFFFF
                        }
                    Return (0xFF)
                }

                Acquire (BTMX, 0xFFFF)
                If (Arg1)
                {
                    NGBT = 0xFF
                }

                Local0 = \_SB.PCI0.LPCB.EC0.NGBT
                Release (BTMX)
                If (((Local0 & Local7) == Zero))
                {
                    Return (Zero)
                }

                Acquire (ECMX, 0xFFFF)
                If (\_SB.PCI0.LPCB.EC0.ECRG)
                {
                    BSEL = Arg0
                    Local0 = \_SB.PCI0.LPCB.EC0.BST
                    Local3 = B1B2 (BPR1, BPR2)
                    DerefOf (NBST [Arg0]) [0x02] = B1B2 (BRC1, BRC2)
                    DerefOf (NBST [Arg0]) [0x03] = B1B2 (BPV1, BPV2)
                }

                Release (ECMX)
                If ((GACS () == One))
                {
                    Local0 &= 0xFFFFFFFFFFFFFFFE
                }
                Else
                {
```

```
                Local0 &= 0xFFFFFFFFFFFFFFFD
            }

            If ((Local0 & One))
            {
                Acquire (BTMX, 0xFFFF)
                NDCB = Local7
                Release (BTMX)
            }

            DerefOf (NBST [Arg0]) [Zero] = Local0
            If ((Local0 & One))
            {
                If (((Local3 < 0x0190) || (Local3 > 0x1964)))
                {
                    Local5 = DerefOf (DerefOf (NBST [Arg0]) [One])
                    If (((Local5 < 0x0190) || (Local5 > 0x1964)))
                    {
                        Local3 = 0x0D7A
                    }
                    Else
                    {
                        Local3 = Local5
                    }
                }

                Local3 = 0xFFFFFFFF
            }
            ElseIf (((Local0 & 0x02) == Zero))
            {
                Local3 = Zero
            }

            DerefOf (NBST [Arg0]) [One] = Local3
            Acquire (BTMX, 0xFFFF)
            NGBT &= ~Local7
            Release (BTMX)
            Return (Zero)
          }
          Else
          {
            Return (\_SB.PCI0.LPCB.EC0.XTST(Arg0, Arg1))
          }
        }

        Method (ITLB, 0, NotSerialized)
        {
          If (_OSI ("Darwin"))
          {
            Local0 = (B1B2 (BFC1, BFC2) * NLB1) /* External reference */
            Local4 = (Local0 / 0x64)
```

```
                Divide ((Local4 + 0x09), 0x0A, Local0, Local1)
                Local0 = (B1B2 (BFC1, BFC2) * NLB2) /* External reference */
                Local4 = (Local0 / 0x64)
                Divide ((Local4 + 0x09), 0x0A, Local0, Local2)
                If (\_SB.PCI0.LPCB.EC0.ECRG)
                {
                    LB1 = Local1
                    LB2 = Local2
                }
            }
            Else
            {
              \_SB.PCI0.LPCB.EC0.XITL()
            }
        }

        Method (GBTI, 1, NotSerialized)
        {
          If (_OSI ("Darwin"))
          {
            Debug = "Enter getbattinfo"
            Acquire (ECMX, 0xFFFF)
            If (\_SB.PCI0.LPCB.EC0.ECRG)
            {
                If ((BATP & (One << Arg0)))
                {
                    BSEL = Arg0
                    Local0 = Package (0x02)
                        {
                            Zero,
                            Buffer (0x6B){}
                        }
                    DerefOf (Local0 [One]) [Zero] = B1B2 (BDC1, BDC2)
                    DerefOf (Local0 [One]) [One] = (B1B2 (BDC1, BDC2) >>
                        0x08)
                    DerefOf (Local0 [One]) [0x02] = B1B2 (BFC1, BFC2)
                    DerefOf (Local0 [One]) [0x03] = (B1B2 (BFC1, BFC2) >>
                        0x08)
                    DerefOf (Local0 [One]) [0x04] = B1B2 (BRC1, BRC2)
                    DerefOf (Local0 [One]) [0x05] = (B1B2 (BRC1, BRC2) >>
                        0x08)
                    DerefOf (Local0 [One]) [0x06] = B1B2 (BME1, BME2)
                    DerefOf (Local0 [One]) [0x07] = (B1B2 (BME1, BME2) >>
                        0x08)
                    DerefOf (Local0 [One]) [0x08] = B1B2 (BCC1, BCC2)
                    DerefOf (Local0 [One]) [0x09] = (B1B2 (BCC1, BCC2) >>
                        0x08)
                    Local1 = B1B2 (CBT1, CBT2)
                    Local1 -= 0x0AAC
                    Divide (Local1, 0x0A, Local2, Local3)
                    DerefOf (Local0 [One]) [0x0A] = Local3
```

```
                        DerefOf (Local0 [One]) [0x0B] = (Local3 >> 0x08)
                        DerefOf (Local0 [One]) [0x0C] = B1B2 (BPV1, BPV2)
                        DerefOf (Local0 [One]) [0x0D] = (B1B2 (BPV1, BPV2) >>
                            0x08)
                        Local1 = B1B2 (BPR1, BPR2)
                        If (Local1)
                        {
                            If ((B1B2 (BST1, BST2) & 0x40))
                            {
                                Local1 = (~Local1 + One)
                                Local1 &= 0xFFFF
                            }
                        }

                        DerefOf (Local0 [One]) [0x0E] = Local1
                        DerefOf (Local0 [One]) [0x0F] = (Local1 >> 0x08)
                        DerefOf (Local0 [One]) [0x10] = B1B2 (BDV1, BDV2)
                        DerefOf (Local0 [One]) [0x11] = (B1B2 (BDV1, BDV2) >>
                            0x08)
                        DerefOf (Local0 [One]) [0x12] = B1B2 (BST1, BST2)
                        DerefOf (Local0 [One]) [0x13] = (B1B2 (BST1, BST2) >>
                            0x08)
                        DerefOf (Local0 [One]) [0x14] = B1B2 (BCV1, BCV2)
                        DerefOf (Local0 [One]) [0x15] = (B1B2 (BCV1, BCV2) >>
                            0x08)
                        DerefOf (Local0 [One]) [0x16] = B1B2 (BC01, BC02)
                        DerefOf (Local0 [One]) [0x17] = (B1B2 (BC01, BC02) >>
                            0x08)
                        DerefOf (Local0 [One]) [0x18] = B1B2 (BC03, BC04)
                        DerefOf (Local0 [One]) [0x19] = (B1B2 (BC03, BC04) >>
                            0x08)
                        DerefOf (Local0 [One]) [0x1A] = B1B2 (BC05, BC06)
                        DerefOf (Local0 [One]) [0x1B] = (B1B2 (BC05, BC06) >>
                            0x08)
                        CreateField (DerefOf (Local0 [One]), 0xE0, 0x80, BTSN)
                        BTSN = GBSS (B1B2 (BSN1, BSN2), B1B2 (BDA1, BDA2))
                        Local1 = GBMF ()
                        Local2 = SizeOf (Local1)
                        CreateField (DerefOf (Local0 [One]), 0x0160, (Local2 * 0x08
), BMAN)
                        BMAN = Local1
                        Local2 += 0x2C
                        CreateField (DerefOf (Local0 [One]), (Local2 * 0x08), 0x80,
 CLBL)
                        CLBL = GCTL (Zero)
                        Local2 += 0x11
                        CreateField (DerefOf (Local0 [One]), (Local2 * 0x08), 0x38,
 DNAM)
                        DNAM = GDNM (Zero)
                        Local2 += 0x07
                        CreateField (DerefOf (Local0 [One]), (Local2 * 0x08), 0x20,
```

```
    DCHE)
                        DCHE = GDCH (Zero)
                        Local2 += 0x04
                        CreateField (DerefOf (Local0 [One]), (Local2 * 0x08), 0x10,
    BMAC)
                        BMAC = Zero
                        Local2 += 0x02
                        CreateField (DerefOf (Local0 [One]), (Local2 * 0x08), 0x10,
    BMAD)
                        BMAD = B1B2 (BDA1, BDA2)
                        Local2 += 0x02
                        CreateField (DerefOf (Local0 [One]), (Local2 * 0x08), 0x10,
    BCCU)
                        BCCU = \_SB.PCI0.LPCB.EC0.BRCC
                        Local2 += 0x02
                        CreateField (DerefOf (Local0 [One]), (Local2 * 0x08), 0x10,
    BCVO)
                        BCVO = \_SB.PCI0.LPCB.EC0.BRCV
                        Local2 += 0x02
                        CreateField (DerefOf (Local0 [One]), (Local2 * 0x08), 0x10,
    BAVC)
                        Local1 = B1B2 (BCR1, BCR2)
                        If (Local1)
                        {
                            If ((B1B2 (BST1, BST2) & 0x40))
                            {
                                Local1 = (~Local1 + One)
                                Local1 &= 0xFFFF
                            }
                        }

                        BAVC = Local1
                        Local2 += 0x02
                        CreateField (DerefOf (Local0 [One]), (Local2 * 0x08), 0x10,
    RTTE)
                        RTTE = B1B2 (BRT1, BRT2)
                        Local2 += 0x02
                        CreateField (DerefOf (Local0 [One]), (Local2 * 0x08), 0x10,
    ATTE)
                        ATTE = B1B2 (BA01, BA02)
                        Local2 += 0x02
                        CreateField (DerefOf (Local0 [One]), (Local2 * 0x08), 0x10,
    ATTF)
                        ATTF = B1B2 (BA03, BA04)
                        Local2 += 0x02
                        CreateField (DerefOf (Local0 [One]), (Local2 * 0x08), 0x08,
    NOBS)
                        NOBS = \_SB.PCI0.LPCB.EC0.BATN
                    }
                    Else
                    {
```

```
                    Local0 = Package (0x01)
                        {
                            0x34
                        }
                }
            }
            Else
            {
                Local0 = Package (0x01)
                    {
                        0x0D
                    }
            }

            Release (ECMX)
            Return (Local0)
        }
        Else
        {
            Return (\_SB.PCI0.LPCB.EC0.XBTI(Arg0))
        }
    }

    Method (GBTC, 0, NotSerialized)
    {
        If (_OSI ("Darwin"))
        {
            Debug = "Enter GetBatteryControl"
            Acquire (ECMX, 0xFFFF)
            If (\_SB.PCI0.LPCB.EC0.ECRG)
            {
                Local0 = Package (0x02)
                    {
                        Zero,
                        Buffer (0x04){}
                    }
                If ((BATP & One))
                {
                    BSEL = Zero
                    DerefOf (Local0 [One]) [Zero] = Zero
                    If ((((INAC == Zero) && (INCH == Zero)) && (IDIS == Zero)))
                    {
                        DerefOf (Local0 [One]) [Zero] = Zero
                    }
                    ElseIf (((((INAC == Zero) && (INCH == 0x02)) && (
                        IDIS == One)) && (B1B2 (MAX1, MAX2) == Zero)))
                    {
                        DerefOf (Local0 [One]) [Zero] = One
                    }
                    ElseIf (((INAC == One) && (IDIS == 0x02)))
                    {
```

```
                            DerefOf (Local0 [One]) [Zero] = 0x02
                        }
                        ElseIf (((((INAC == Zero) && (INCH == 0x02)) && (
                            IDIS == One)) && (B1B2 (MAX1, MAX2) == 0xFA)))
                        {
                            DerefOf (Local0 [One]) [Zero] = 0x03
                        }
                        ElseIf (((INAC == Zero) && (INCH == 0x03)))
                        {
                            DerefOf (Local0 [One]) [Zero] = 0x04
                        }
                    }
                    Else
                    {
                        DerefOf (Local0 [One]) [Zero] = 0xFF
                    }

                    If ((BATP & 0x02))
                    {
                        BSEL = One
                        DerefOf (Local0 [One]) [One] = Zero
                        If ((((INAC == Zero) && (INCH == Zero)) && (IDIS == Zero)))
                        {
                            DerefOf (Local0 [One]) [One] = Zero
                        }
                        ElseIf (((((INAC == Zero) && (INCH == One)) && (
                            IDIS == 0x02)) && (B1B2 (MAX1, MAX2) == Zero)))
                        {
                            DerefOf (Local0 [One]) [One] = One
                        }
                        ElseIf (((INAC == One) && (IDIS == One)))
                        {
                            DerefOf (Local0 [One]) [One] = 0x02
                        }
                        ElseIf (((((INAC == Zero) && (INCH == One)) && (
                            IDIS == 0x02)) && (B1B2 (MAX1, MAX2) == 0xFA)))
                        {
                            DerefOf (Local0 [One]) [One] = 0x03
                        }
                        ElseIf (((INAC == Zero) && (INCH == 0x03)))
                        {
                            DerefOf (Local0 [One]) [One] = 0x04
                        }
                    }
                    Else
                    {
                        DerefOf (Local0 [One]) [One] = 0xFF
                    }
                }
                Else
                {
```

```
                    Local0 = Package (0x02)
                        {
                            0x35,
                            Zero
                        }
                }

                Release (ECMX)
                Return (Local0)
            }
            Else
            {
                Return (\_SB.PCI0.LPCB.EC0.XGBU())
            }
        }

        Method (SBTC, 3, NotSerialized)
        {
            If (_OSI ("Darwin"))
            {
                Debug = "Enter SetBatteryControl"
                Debug = Arg0
                Debug = Arg1
                Debug = Arg2
                Acquire (ECMX, 0xFFFF)
                If (\_SB.PCI0.LPCB.EC0.ECRG)
                {
                    Local0 = Arg2
                    Debug = Local0
                    Local4 = Package (0x01)
                        {
                            0x06
                        }
                    Local1 = Zero
                    Local2 = Zero
                    Local1 = DerefOf (Local0 [Zero])
                    If ((Local1 == Zero))
                    {
                        Debug = "battery 0"
                        If ((BATP & One))
                        {
                            Local2 = DerefOf (Local0 [One])
                            If ((Local2 == Zero))
                            {
                                INCH = Zero
                                IDIS = Zero
                                INAC = Zero
                                B1B2 (MAX1, MAX2) = Zero
                                PSSB = One
                                Local4 = Package (0x01)
                                    {
```

```
                Zero
            }
        }

        If ((Local2 == One))
        {
            INAC = Zero
            INCH = 0x02
            IDIS = One
            B1B2 (MAX1, MAX2) = Zero
            PSSB = Zero
            Local4 = Package (0x01)
                {
                    Zero
                }
        }

        If ((Local2 == 0x02))
        {
            INAC = One
            INCH = One
            IDIS = 0x02
            PSSB = Zero
            Local4 = Package (0x01)
                {
                    Zero
                }
        }

        If ((Local2 == 0x03))
        {
            INCH = 0x02
            IDIS = One
            INAC = Zero
            B1B2 (MAX1, MAX2) = 0xFA
            PSSB = Zero
            Local4 = Package (0x01)
                {
                    Zero
                }
        }

        If ((Local2 == 0x04))
        {
            B1B2 (MAX1, MAX2) = 0xFA
            Local4 = Package (0x01)
                {
                    Zero
                }
        }
```

```
                            If ((Local2 == 0x05))
                            {
                                INAC = Zero
                                INCH = 0x03
                                Local4 = Package (0x01)
                                    {
                                        Zero
                                    }
                            }
                        }
                        Else
                        {
                            Local4 = Package (0x01)
                                {
                                    0x34
                                }
                        }
                    }

                    If ((Local1 == One))
                    {
                        If ((BATP & 0x02))
                        {
                            Debug = "battery 1"
                            Local2 = DerefOf (Local0 [One])
                            If ((Local2 == Zero))
                            {
                                INCH = Zero
                                IDIS = Zero
                                INAC = Zero
                                B1B2 (MAX1, MAX2) = Zero
                                PSSB = One
                                Local4 = Package (0x01)
                                    {
                                        Zero
                                    }
                            }

                            If ((Local2 == One))
                            {
                                INAC = Zero
                                INCH = One
                                IDIS = 0x02
                                B1B2 (MAX1, MAX2) = Zero
                                PSSB = Zero
                                Local4 = Package (0x01)
                                    {
                                        Zero
                                    }
                            }
```

```
                If ((Local2 == 0x02))
                {
                    INAC = One
                    INCH = 0x02
                    IDIS = One
                    PSSB = Zero
                    Local4 = Package (0x01)
                        {
                            Zero
                        }
                }

                If ((Local2 == 0x03))
                {
                    INCH = One
                    IDIS = 0x02
                    INAC = Zero
                    B1B2 (MAX1, MAX2) = 0xFA
                    PSSB = Zero
                    Local4 = Package (0x01)
                        {
                            Zero
                        }
                }

                If ((Local2 == 0x04))
                {
                    INCH = Zero
                    IDIS = Zero
                    INAC = Zero
                    Local4 = Package (0x01)
                        {
                            Zero
                        }
                }

                If ((Local2 == 0x05))
                {
                    INAC = Zero
                    INCH = 0x03
                    Local4 = Package (0x01)
                        {
                            Zero
                        }
                }
            }
            Else
            {
                Local4 = Package (0x01)
                    {
                        0x34
```

```
                                }
                            }
                        }
                    }

                    Release (ECMX)
                    Return (Local4)
                }
                Else
                {
                    Return (\_SB.PCI0.LPCB.EC0.XSBT(Arg0, Arg1, Arg2))
                }
            }
    }

    Scope (_TZ)
    {
        Method (GCGC, 0, Serialized)
        {
          If (_OSI ("Darwin"))
          {
            Name (LTMP, Buffer (0x02){})
            If (\_SB.PCI0.LPCB.EC0.ECRG)
            {
                Acquire (\_SB.PCI0.LPCB.EC0.ECMX, 0xFFFF)
                LTMP = B1B2 (\_SB.PCI0.LPCB.EC0.BPR1, \_SB.PCI0.LPCB.EC0.BPR2)
                Release (\_SB.PCI0.LPCB.EC0.ECMX)
            }

            Return (LTMP) /* \_TZ_.GCGC.LTMP */
          }
          Else
          {
            Return (\_TZ.XGCG())
          }
        }
    }
}
```

```
//// battery
// In config ACPI, UPBI to XPBI
// Find:     5550424900
// Replace:  5850424900
//
// In config ACPI, UPBS  to XPBS
// Find:     5550425300
// Replace:  5850425300
//
// In config ACPI, IVBI to XVBI
// Find:     4956424900
// Replace:  5856424900
//
DefinitionBlock ("", "SSDT", 2, "OCLT", "BAT0", 0x00000000)
{
    External (_SB.BAT0, DeviceObj)
    External (_SB.PCI0.LPCB.EC0, DeviceObj)
    External (_SB.BAT0.XPBI, MethodObj)
    External (_SB.BAT0.XPBS, MethodObj)
    External (_SB.BAT0.XVBI, MethodObj)
    External (_SB.PCI0.LPCB.EC0.ECWT, MethodObj)
    External (_SB.PCI0.LPCB.EC0.MBST, FieldUnitObj)
    External (_SB.PCI0.LPCB.EC0.BACR, FieldUnitObj)

    External (_SB.BAT0.PBIF, PkgObj)
    External (_SB.BAT0.PBST, PkgObj)
    External (_SB.BATM, MutexObj)
    External (_SB.BAT0.BTUR, IntObj)
    External (POSW, MethodObj)

    Method (B1B2, 2, NotSerialized)
    {
        Return ((Arg0 | (Arg1 << 0x08)))
    }

    Scope (_SB.PCI0.LPCB.EC0)
    {
        Method (RE1B, 1, NotSerialized)
        {
            OperationRegion(ERAM, EmbeddedControl, Arg0, 1)
            Field(ERAM, ByteAcc, NoLock, Preserve) { BYTE, 8 }
            Return(BYTE)
        }
        Method (RECB, 2, Serialized)
        {
            ShiftRight(Arg1, 3, Arg1)
            Name(TEMP, Buffer(Arg1) { })
            Add(Arg0, Arg1, Arg1)
            Store(0, Local0)
            While (LLess(Arg0, Arg1))
```

```
            {
                Store(RE1B(Arg0), Index(TEMP, Local0))
                Increment(Arg0)
                Increment(Local0)
            }
            Return(TEMP)
        }
        OperationRegion (ERM2, EmbeddedControl, Zero, 0xFF)
        Field (ERM2, ByteAcc, NoLock, Preserve)
        {
                Offset (0x70),
                SCP0,8,SCP1,8, //DSCP,   16,
                ACP0,8,ACP1,8, //LACP,   16,
                SVG0,8,SVG1,8, //DSVG,   16,
                Offset (0x77),
                //BANA,   64, Offset (0x77) ,RECB(0x77,64)

                Offset (0x82),
                    ,8,
                CUR0,8,CUR1,8, //MCUR,   16,
                BRM0,8,BRM1,8, //MBRM,   16,
                BCV0,8,BCV1,8, //MBCV,   16,
        }
    }

    Scope (_SB.BAT0)
    {
        Method (UPBI, 0, NotSerialized)
        {
            If (_OSI ("Darwin"))
            {
                Acquire (BATM, 0xFFFF)
                PBIF [One] = B1B2 (^^PCI0.LPCB.EC0.SCP0, ^^PCI0.LPCB.EC0.SCP1)
                PBIF [0x02] = B1B2 (^^PCI0.LPCB.EC0.ACP0, ^^PCI0.LPCB.EC0.ACP1)
                PBIF [0x04] = B1B2 (^^PCI0.LPCB.EC0.SVG0, ^^PCI0.LPCB.EC0.SVG1)
                PBIF [0x05] = (B1B2 (^^PCI0.LPCB.EC0.SCP0, ^^PCI0.LPCB.EC0.SCP1)/0x
0A)
                PBIF [0x06] = (B1B2 (^^PCI0.LPCB.EC0.SCP0, ^^PCI0.LPCB.EC0.SCP1)/0x
64)
                PBIF [0x09] = "MWL32b"
                If ((B1B2 (^^PCI0.LPCB.EC0.SCP0, ^^PCI0.LPCB.EC0.SCP1) < 0x1770))
                {
                    PBIF [0x09] = "MWL32b"
                }

                If ((B1B2 (^^PCI0.LPCB.EC0.SCP0, ^^PCI0.LPCB.EC0.SCP1) < 0x0BB8))
                {
                    PBIF [0x09] = "MWL31b"
                }

                Release (BATM)
```

```
            }
            Else
            {
                \_SB.BAT0.XPBI()
            }
        }

        Method (UPBS, 0, NotSerialized)
        {
            If (_OSI ("Darwin"))
            {
                If ((B1B2 (^^PCI0.LPCB.EC0.BRM0, ^^PCI0.LPCB.EC0.BRM1) == Zero))
                {
                    BTUR = One
                }
                ElseIf ((BTUR == One))
                {
                    Notify (BAT0, 0x81) // Information Change
                    Notify (BAT0, 0x80) // Status Change
                    BTUR = Zero
                }

                Local5 = B1B2 (^^PCI0.LPCB.EC0.CUR0, ^^PCI0.LPCB.EC0.CUR1)
                PBST [One] = POSW (Local5)
                Local5 = B1B2 (^^PCI0.LPCB.EC0.BRM0, ^^PCI0.LPCB.EC0.BRM1)
                If ((^^PCI0.LPCB.EC0.BACR == One))
                {
                    Local5 = ((B1B2 (^^PCI0.LPCB.EC0.SCP0, ^^PCI0.LPCB.EC0.SCP1) /
0x32) +
                        B1B2 (^^PCI0.LPCB.EC0.BRM0, ^^PCI0.LPCB.EC0.BRM1))
                }

                Local5 = B1B2 (^^PCI0.LPCB.EC0.BRM0, ^^PCI0.LPCB.EC0.BRM1)
                If (!(Local5 & 0x8000))
                {
                    If ((Local5 != DerefOf (PBST [0x02])))
                    {
                        PBST [0x02] = Local5
                    }
                }

                PBST [0x03] = B1B2 (^^PCI0.LPCB.EC0.BCV0, ^^PCI0.LPCB.EC0.BCV1)
                PBST [Zero] = ^^PCI0.LPCB.EC0.MBST
            }
            Else
            {
                \_SB.BAT0.XPBS()
            }
        }

        Method (IVBI, 0, NotSerialized)
```

```
        {
            If (_OSI ("Darwin"))
            {
                PBIF [One] = 0xFFFFFFFF
                PBIF [0x02] = 0xFFFFFFFF
                PBIF [0x04] = 0xFFFFFFFF
                PBIF [0x09] = "Bad"
                PBIF [0x0A] = "Bad"
                PBIF [0x0B] = "Bad"
                PBIF [0x0C] = "Bad"
                ^^PCI0.LPCB.EC0.ECWT (Zero, ^^PCI0.LPCB.EC0.RECB (0x77, 64))
            }
            Else
            {
                \_SB.BAT0.XVBI()
            }
        }
    }
}
```

```
// battery
// In config ACPI, _BIF renamed XBIF
// Find:     5F 42 49 46
// Replace:  58 42 49 46
//
// In config ACPI, _BST renamed XBST
// Find:     5F 42 53 54
// Replace:  58 42 53 54
//
DefinitionBlock ("", "SSDT", 2, "OCLT", "BAT1", 0)
{
    External(_SB.PCI0.LPCB.H_EC, DeviceObj)
    External(_SB.PCI0.LPCB.H_EC.BAT1, DeviceObj)
    External(_SB.PCI0.LPCB.H_EC.BAT1.XBIF, MethodObj)
    External(_SB.PCI0.LPCB.H_EC.BAT1.XBST, MethodObj)
    External(_SB.PCI0.LPCB.H_EC.BAT1.POSW, MethodObj)
    External(_SB.PCI0.LPCB.H_EC.ECAV, IntObj)
    External(_SB.PCI0.LPCB.H_EC.B1IC, FieldUnitObj)
    External(_SB.PCI0.LPCB.H_EC.B1DI, FieldUnitObj)
    External(_SB.PCI0.LPCB.H_EC.ECRD, MethodObj)

    Method (B1B2, 2, NotSerialized)
    {
        ShiftLeft (Arg1, 8, Local0)
        Or (Arg0, Local0, Local0)
        Return (Local0)
    }
    Scope(_SB.PCI0.LPCB.H_EC)
    {
        Method (RE1B, 1, NotSerialized)
        {
            OperationRegion(ERAM, EmbeddedControl, Arg0, 1)
            Field(ERAM, ByteAcc, NoLock, Preserve) { BYTE, 8 }
            Return(BYTE)
        }
        Method (RECB, 2, Serialized)
        {
            ShiftRight(Arg1, 3, Arg1)
            Name(TEMP, Buffer(Arg1) { })
            Add(Arg0, Arg1, Arg1)
            Store(0, Local0)
            While (LLess(Arg0, Arg1))
            {
                Store(RE1B(Arg0), Index(TEMP, Local0))
                Increment(Arg0)
                Increment(Local0)
            }
            Return(TEMP)
        }
        OperationRegion (BAM0, EmbeddedControl, 0x00, 0xFF)
```

```
        Field (BAM0, ByteAcc, Lock, Preserve)
        {
                Offset (0x62),
                    ,   16, //B1TM,   16,
                BVT0,8,BVT1,8,//B1VT,   16,
                BCR0,8,BCR1,8,//B1CR,   16,
                    ,   16,
                BRC0,8,BRC1,8,//B1RC,   16,
                BFC0,8,BFC1,8,//B1FC,   16,

                Offset (0x76),
                BDC0,8,BDC1,8,//B1DC,   16,
                BDV0,8,BDV1,8,//B1DV,   16,
                    ,   16, //BDCW,   16,
                    ,   16, //BDCL,   16,
                    ,   16, //B1AR,   16,
                //B1MA,   64, //Offset (0x80) ,RECB(0x80,64)
        }
    }

    Scope(_SB.PCI0.LPCB.H_EC.BAT1)
    {
        Method (_BIF, 0, Serialized)
        {
                If (_OSI ("Darwin"))
                {
                    Name (BPK1, Package (0x0D)
                    {
                        Zero,
                        0xFFFFFFFF,
                        0xFFFFFFFF,
                        One,
                        0xFFFFFFFF,
                        Zero,
                        Zero,
                        0x0100,
                        0x40,
                        "BASE-BAT",
                        "123456789",
                        "LiP",
                        "Simplo"
                    })
                    If (\_SB.PCI0.LPCB.H_EC.ECAV)
                    {
                        Local0 = B1B2 (BFC0, BFC1) * 0x0A
                        If (Local0)
                        {
                            BPK1 [One] = B1B2 (BDC0, BDC1) * 0x0A
                            BPK1 [0x02] = Local0
                            BPK1 [0x04] = B1B2 (BDV0, BDV1)
                            Divide (Local0, 0x0A, Local1, Local2)
```

```
                        BPK1 [0x05] = Local2
                        Divide (Local0, 0x32, Local1, Local2)
                        BPK1 [0x06] = Local2
                        If ((RECB(0x80,64) == 0x0000313100504D53))
                        {
                            BPK1 [0x0C] = "Simplo"
                        }

                        If ((RECB(0x80,64) == 0x20534F432D545043))
                        {
                            BPK1 [0x0C] = "Celxpert"
                        }

                        If ((RECB(0x80,64) == 0x74726570786C6543))
                        {
                            BPK1 [0x0C] = "Celxpert"
                        }

                        If ((RECB(0x80,64) == 0x3831303200504D53))
                        {
                            BPK1 [0x0C] = "Simplo"
                        }

                        If ((RECB(0x80,64) == 0x393130320043474C))
                        {
                            BPK1 [0x0C] = "LGC"
                        }

                        If ((RECB(0x80,64) == 0x0061646F776E7553))
                        {
                            BPK1 [0x0C] = "Sunwoda"
                        }
                    }
                }

                Return (BPK1)
            }
            Else
            {
                Return (\_SB.PCI0.LPCB.H_EC.BAT1.XBIF())
            }
        }

        Method (_BST, 0, Serialized)
        {
            If (_OSI ("Darwin"))
            {
                Name (PKG1, Package (0x04)
                {
                    0xFFFFFFFF,
                    0xFFFFFFFF,
```

```
                        0xFFFFFFFF,
                        0xFFFFFFFF
                    })
                    If (\_SB.PCI0.LPCB.H_EC.ECAV)
                    {
                        Local0 = (ECRD (RefOf (B1IC)) << One)
                        Local1 = (ECRD (RefOf (B1DI)) | Local0)
                        PKG1 [Zero] = Local1
                        Local2 = B1B2 (BCR0, BCR1)
                        Local2 = POSW (Local2)
                        Local3 = B1B2 (BVT0, BVT1)
                        Divide (Local3, 0x03E8, Local4, Local3)
                        Local2 *= Local3
                        PKG1 [One] = Local2
                        PKG1 [0x02] = B1B2 (BRC0, BRC1) * 0x0A
                        PKG1 [0x03] = B1B2 (BVT0, BVT1)
                    }

                    Return (PKG1)
                }
                Else
                {
                    Return (\_SB.PCI0.LPCB.H_EC.BAT1.XBST())
                }
            }
        }
    }
}
//EOF
```

禁用EHCx

描述

下列情况之一需要禁用 EHC1 和 EHC2 总线：

- ACPI 包含 EHC1 或者 EHC2，而机器本身不存在相关硬件。
- ACPI 包含 EHC1 或者 EHC2，机器存在相关硬件但并没有实际输出端口（外置和内置）。

补丁

- ***SSDT-EHC1_OFF***：禁用 `EHC1` 。
- ***SSDT-EHC2_OFF***：禁用 `EHC2` 。
- ***SSDT-EHCx_OFF***：是 ***SSDT-EHC1_OFF*** 和 ***SSDT-EHC2_OFF*** 的合并补丁。

使用方法

- 优先 BIOS 设置： `XHCI Mode` = `Enabled` 。
- 如果 BIOS 没有 `XHCI Mode` 选项，同时符合 描述 的情况之一，使用上述补丁。

注意事项

- 适用于 7, 8, 9 系机器，且 macOS 是 10.11 以上版本。
- 对于 7 系机器，***SSDT-EHC1_OFF*** 和 ***SSDT-EHC2_OFF*** 二者不可同时使用。
- 补丁在 `Scope (\)` 下添加了 `_INI` 方法，如果和其他补丁的 `_INI` 发生重复，应合并 `_INI` 里的内容。

```
//
DefinitionBlock ("", "SSDT", 2, "OCLT", "EHCx_OFF", 0x00001000)
{
    Scope (\)
    {
        OperationRegion (RCRG, SystemMemory, 0xFED1F418, One)
        Field (RCRG, DWordAcc, Lock, Preserve)
        {
                ,   13,
            EH2D,   1,
                ,   1,
            EH1D,   1
        }

        Method (_INI, 0, NotSerialized)
        {
            If (_OSI ("Darwin"))
            {
                EH1D = One  // Disable EHC1
            }
        }
    }
}
```

```
//
DefinitionBlock ("", "SSDT", 2, "OCLT", "EHCx_OFF", 0x00001000)
{
    Scope (\)
    {
        OperationRegion (RCRG, SystemMemory, 0xFED1F418, One)
        Field (RCRG, DWordAcc, Lock, Preserve)
        {
                ,   13,
            EH2D,   1,
                ,   1,
            EH1D,   1
        }

        Method (_INI, 0, NotSerialized)
        {
            If (_OSI ("Darwin"))
            {
                EH2D = One  // Disable EHC2
            }
        }
    }
}
```

```
/*
 * USB compatibility table.
 *
 * Attention!
 * Only for 7,8,9-series chipsets and 10.11 and newer!
 *
 * To disable EHC1 and EHC2 - set an option "XHCI Mode" to "Enabled" in yours BIOS.
 * If the "XHCI Mode" option is not available in yours BIOS or works incorrectly, t
hen use this ACPI table.
 * Disabling through BIOS is preferable whenever possible.
 *
 * WARN: for 7-series you need to use either "EH1D = One" or "EH2D = One" but not b
oth!
 * This is because only one of the devices (EHC1 or EHC2) is used by macOS. Check t
he IOReg.
 */
DefinitionBlock ("", "SSDT", 2, "ACDT", "EHCx_OFF", 0x00001000)
{
    Scope (\)
    {
        OperationRegion (RCRG, SystemMemory, 0xFED1F418, One)
        Field (RCRG, DWordAcc, Lock, Preserve)
        {
                ,   13,
            EH2D,   1,
                ,   1,
            EH1D,   1
        }

        Method (_INI, 0, NotSerialized)
        {
            If (_OSI ("Darwin"))
            {
                EH1D = One  // Disable EHC1
                EH2D = One  // Disable EHC2
            }
        }
    }
}
```

综合补丁

描述

- 通过对 `_PTS` 和 `_WAK` 更名，加上综合补丁和它的扩展补丁，解决某些机器睡眠或唤醒过程中出现一些问题。
- 综合补丁是一个框架，它包括：
  - 屏蔽独显接口 `_ON`，`_OFF`。
  - 6 个扩展补丁接口 `EXT1`，`EXT2`，`EXT3`，`EXT4`，`EXT5` 和 `EXT6`。
  - 定义强制睡眠传递参数 `FNOK` 和 `MODE`，详见《PNP0C0E睡眠修正方法》。
  - 定义调试参数 `TPTS` 和 `TWAK`，用于在睡眠和唤醒过程中，侦测、跟踪 `Arg0` 变化。比如，在亮度快捷键补丁里添加以下代码：

    ```
    ...
    /* 某按键： */
    \RMDT.P2 ("ABCD-_PTS-Arg0=", \_SB.PCI9.TPTS)
    \RMDT.P2 ("ABCD-_WAK-Arg0=", \_SB.PCI9.TWAK)
    ...
    ```

    当按下亮度快捷键后，能够在控制台上看到前一次睡眠、唤醒后 `Arg0` 的值。

    注：调试ACPI需要安装驱动 ACPIDebug.kext，添加补丁 SSDT-RMDT，以及自定义的调试补丁。具体方法参见《ACPIDebug》。

更名

使用综合补丁必须对 `_PTS` 和 `_WAK` 更名。依据原始 DSDT 内容选择正确的更名，如：

- `_PTS` to `ZPTS(1,N)`：

  ```
  Method (_PTS, 1, NotSerialized)  /* _PTS: Prepare To Sleep */
  {
  ```

- `_WAK` to `ZWAK(1,N)`：

  ```
  Method (_WAK, 1, NotSerialized)  /* _WAK: Wake */
  {
  ```

- `_PTS` to `ZPTS(1,S)`：

  ```
  Method (_PTS, 1, Serialized)  /* _PTS: Prepare To Sleep */
  {
  ```

- `_WAK` to `ZWAK(1,S)`：

```
Method (_WAK, 1, Serialized)  /* _WAK: Wake */
{
```

如果 DSDT 中存在 `_TTS` 也需要对其更名；如果不存在，则无需更名。依据原始 DSDT 内容选择正确的更名，如：

- `_TTS` to `ZTTS(1,N)`：

```
Method (_TTS, 1, NotSerialized)  /* _WAK: Wake */
{
```

- `_TTS` to `ZTTS(1,S)`：

```
Method (_TTS, 1, Serialized)  /* _WAK: Wake */
{
```

补丁

- ***SSDT-PTSWAKTTS*** —— 综合补丁。
- ***SSDT-EXT1-FixShutdown*** —— `EXT1` 扩展补丁。 修复因 XHC 控制器导致的关机变重启的问题，原理是当 `_PTS` 中传入的参数为 `5` 时将 `XHC.PMEE` 置 0。该补丁与 Clover 的 `FixShutdown` 效果等同。部分 XPS / ThinkPad 机器会需要这个补丁。
- ***SSDT-EXT4-WakeScreen*** —— `EXT4` 扩展补丁。解决某些机器唤醒后需按任意键亮屏的问题。使用时应查询 `PNP0C0D` 设备名称和路径是否已存在补丁文件中，如 `_SB.PCI0.LPCB.LID0` 。如果不存在自行添加。
- ***SSDT-EXT5-TP-LED*** —— `EXT5` 扩展补丁。 解决 ThinkPad 机器唤醒后 A 面呼吸灯和电源键呼吸灯未恢复正常的问题；修复在 ThinkPad 老机型上唤醒后 F4 麦克风指示灯状态不正常的问题。

注意

具有相同扩展名称的补丁不可同时使用。如有同时使用的要求必须合并后使用。

```
//
DefinitionBlock("", "SSDT", 2, "OCLT", "EXT4", 0)
{
    External(_SB.LID, DeviceObj)
    External(_SB.LID0, DeviceObj)
    External(_SB.PCI0.LPCB.LID, DeviceObj)
    External(_SB.PCI0.LPCB.LID0, DeviceObj)

    Method (EXT4, 1, NotSerialized)
    {
        If (3 == Arg0)
        {
            If (CondRefOf (\_SB.LID))
            {
                Notify (\_SB.LID, 0x80)
            }
            If (CondRefOf (\_SB.LID0))
            {
                Notify (\_SB.LID0, 0x80)
            }
            //
            If (CondRefOf (\_SB.PCI0.LPCB.LID))
            {
                Notify (\_SB.PCI0.LPCB.LID, 0x80)
            }
            If (CondRefOf (\_SB.PCI0.LPCB.LID0))
            {
                Notify (\_SB.PCI0.LPCB.LID0, 0x80)
            }
        }
    }
}
//EOF
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/Pro
pertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>ACPI</key>
    <dict>
        <key>Patch</key>
        <array>
            <dict>
                <key>Comment</key>
                <string>_PTS to ZPTS(1,N)</string>
                <key>Count</key>
                <integer>0</integer>
                <key>Enabled</key>
                <true/>
                <key>Find</key>
                <data>
                X1BUUwE=
                </data>
                <key>Limit</key>
                <integer>0</integer>
                <key>Mask</key>
                <data>
                </data>
                <key>OemTableId</key>
                <data>
                </data>
                <key>Replace</key>
                <data>
                WlBUUwE=
                </data>
                <key>ReplaceMask</key>
                <data>
                </data>
                <key>Skip</key>
                <integer>0</integer>
                <key>TableLength</key>
                <integer>0</integer>
                <key>TableSignature</key>
                <data>
                </data>
            </dict>
            <dict>
                <key>Comment</key>
                <string>_PTS to ZPTS(1,S)</string>
                <key>Count</key>
                <integer>0</integer>
                <key>Enabled</key>
                <true/>
```

```
                <key>Find</key>
                <data>
                X1BUUwk=
                </data>
                <key>Limit</key>
                <integer>0</integer>
                <key>Mask</key>
                <data>
                </data>
                <key>OemTableId</key>
                <data>
                </data>
                <key>Replace</key>
                <data>
                WlBUUwk=
                </data>
                <key>ReplaceMask</key>
                <data>
                </data>
                <key>Skip</key>
                <integer>0</integer>
                <key>TableLength</key>
                <integer>0</integer>
                <key>TableSignature</key>
                <data>
                </data>
            </dict>
            <dict>
                <key>Comment</key>
                <string>_WAK to ZWAK(1,N)</string>
                <key>Count</key>
                <integer>0</integer>
                <key>Enabled</key>
                <true/>
                <key>Find</key>
                <data>
                X1dBSwE=
                </data>
                <key>Limit</key>
                <integer>0</integer>
                <key>Mask</key>
                <data>
                </data>
                <key>OemTableId</key>
                <data>
                </data>
                <key>Replace</key>
                <data>
                WldBSwE=
                </data>
                <key>ReplaceMask</key>
```

```
                <data>
                </data>
                <key>Skip</key>
                <integer>0</integer>
                <key>TableLength</key>
                <integer>0</integer>
                <key>TableSignature</key>
                <data>
                </data>
            </dict>
            <dict>
                <key>Comment</key>
                <string>_WAK to ZWAK(1,S)</string>
                <key>Count</key>
                <integer>0</integer>
                <key>Enabled</key>
                <true/>
                <key>Find</key>
                <data>
                X1dBSwk=
                </data>
                <key>Limit</key>
                <integer>0</integer>
                <key>Mask</key>
                <data>
                </data>
                <key>OemTableId</key>
                <data>
                </data>
                <key>Replace</key>
                <data>
                WldBSwk=
                </data>
                <key>ReplaceMask</key>
                <data>
                </data>
                <key>Skip</key>
                <integer>0</integer>
                <key>TableLength</key>
                <integer>0</integer>
                <key>TableSignature</key>
                <data>
                </data>
            </dict>
            <dict>
                <key>Comment</key>
                <string>_TTS to ZTTS(1,N)</string>
                <key>Count</key>
                <integer>0</integer>
                <key>Enabled</key>
                <true/>
```

```
                <key>Find</key>
                <data>
                X1RUUwE=
                </data>
                <key>Limit</key>
                <integer>0</integer>
                <key>Mask</key>
                <data>
                </data>
                <key>OemTableId</key>
                <data>
                </data>
                <key>Replace</key>
                <data>
                WlRUUwE=
                </data>
                <key>ReplaceMask</key>
                <data>
                </data>
                <key>Skip</key>
                <integer>0</integer>
                <key>TableLength</key>
                <integer>0</integer>
                <key>TableSignature</key>
                <data>
                </data>
            </dict>
            <dict>
                <key>Comment</key>
                <string>_TTS to ZTTS(1,S)</string>
                <key>Count</key>
                <integer>0</integer>
                <key>Enabled</key>
                <true/>
                <key>Find</key>
                <data>
                X1RUUwk=
                </data>
                <key>Limit</key>
                <integer>0</integer>
                <key>Mask</key>
                <data>
                </data>
                <key>OemTableId</key>
                <data>
                </data>
                <key>Replace</key>
                <data>
                WlRUUwk=
                </data>
                <key>ReplaceMask</key>
```

```
                <data>
                </data>
                <key>Skip</key>
                <integer>0</integer>
                <key>TableLength</key>
                <integer>0</integer>
                <key>TableSignature</key>
                <data>
                </data>
            </dict>
        </array>
    </dict>
</dict>
</plist>
```

PNP0C0E 睡眠修正方法

`PNP0C0E` 和 `PNP0C0D` 睡眠方式

- ACPI 规范

  `PNP0C0E` — Sleep Button Device

  `PNP0C0D` — Lid Device

  有关 `PNP0C0E` 和 `PNP0C0D` 详细内容请查阅 ACPI 规范。

- `PNP0C0E` 睡眠条件
  - 执行 `Notify(***.SLPB, 0x80)` 。 `SLPB` 是 `PNP0C0E` 设备名称。
- `PNP0C0D` 睡眠条件
  - `_LID` 返回 `Zero` 。 `_LID` 是 `PNP0C0D` 设备当前状态。
  - 执行 `Notify(***.LID0, 0x80)` 。 `LID0` 是 `PNP0C0D` 设备名称。

问题描述

部分机器提供了睡眠按键（小月亮按键），如：部分 ThinkPad 的 Fn+F4，Dell 的 Fn+Insert 等。当按下这个按键后，系统执行了 `PNP0C0E` 睡眠。可是，ACPI 错误地向系统传递了关机参数而非睡眠参数，从而导致系统崩溃。即使能够睡眠也能正常唤醒，系统工作状态也被破坏。

下列方法之一可以修复这个问题：

- 截取ACPI 传递的参数并纠正它。
- 将 `PNP0C0E` 睡眠转换为 `PNP0C0D` 睡眠。

解决方案

关联的3个补丁

- ***SSDT-PTSWAK*** ：定义变量 `FNOK` 和 `MODE` ，捕捉 `FNOK` 的变化。见《PTSWAK综合扩展补丁》。
  - `FNOK` 表示按键状态
    - `FNOK` =1：按下睡眠按键
    - `FNOK` =0：再次按下睡眠按键或者机器被唤醒后
  - `MODE` 设定睡眠模式
    - `MODE` =1： `PNP0C0E` 睡眠
    - `MODE` =0： `PNP0C0D` 睡眠

  注意：根据自己的需要设置 `MODE` ，但不可以更改 `FNOK` 。

- ***SSDT-LIDpatch*** ：捕捉 `FNOK` 变化
  - 如果 `FNOK` =1，盖子设备当前状态返回 `Zero`
  - 如果 `FNOK` =0，盖子设备当前状态返回原始值

  注意： `PNP0C0D` 设备名称、路径要和ACPI一致。
- 睡眠按键补丁 ：按键按下后，令 `FNOK` = `1` ，并根据不同的睡眠模式执行相应的操作

  注意： `PNP0C0D` 设备名称、路径要和ACPI一致。

两种睡眠方式描述

- `MODE` =1模式：当按下睡眠按键时， 睡眠按键补丁 令 `FNOK=1` 。 ***SSDT-PTSWAK*** 捕捉到 `FNOK` 为 `1` ，强制令 `Arg0=3` （否则 `Arg0=5` ）。待唤醒后恢复 `FNOK=0` 。一次完整的 `PNP0C0E` 睡眠和唤醒过程结束。
- `MODE` =0模式：当按下睡眠按键时，除了完成上述过程外， ***SSDT-LIDpatch*** 同时扑捉到 `FNOK=1` ，使 `_LID` 返回 `Zero` 并执行 `PNP0C0D` 睡眠。待唤醒后恢复 `FNOK=0` 。一次完整的 `PNP0C0D` 睡眠和唤醒过程结束。

以下是 ***SSDT-LIDpatch*** 主要内容：

```
Method (_LID, 0, NotSerialized)
{
    if(\_SB.PCI9.FNOK==1)
    {
        Return (0) /* 返回 Zero, 满足 PNP0C0D 睡眠条件之一 */
    }
    Else
    {
        Return (\_SB.LID0.XLID()) /* 返回原始值 */
    }
}
```

以下是 睡眠按键补丁 主要内容：

```
If (\_SB.PCI9.MODE == 1) /* PNP0C0E 睡眠 */
{
    \_SB.PCI9.FNOK =1 /* 按下睡眠按键 */
    \_SB.PCI0.LPCB.EC.XQ13() /* 原始睡眠按键位置，示例是 TP 机器 */
}
Else /* PNP0C0D 睡眠 */
{
    If (\_SB.PCI9.FNOK!=1)
    {
            \_SB.PCI9.FNOK =1 /* 按下睡眠按键 */
    }
    Else
    {
            \_SB.PCI9.FNOK =0 /* 再次按下睡眠按键 */
```

```
    }
    Notify (\_SB.LID, 0x80) /* 执行 PNP0C0D 睡眠 */
}
```

更名和补丁组合示例:（Dell Latitude 5480 和 ThinkPad X1C5th）

- **Dell Latitude 5480**

  PTSWAK更名：`_PTS` to `ZPTS`、`_WAK` to `ZWAK`。

  盖子状态更名：`_LID` to `XLID`

  按键更名：`BTNV` to `XTNV` (Dell-Fn+Insert)

  补丁组合：

  - ***SSDT-PTSWAK***：综合补丁。根据自己的需要设置 `MODE`。
  - ***SSDT-LIDpatch***：盖子状态补丁。
  - ***SSDT-FnInsert_BTNV-dell***：睡眠按键补丁。
- **ThinkPad X1C5th**

  PTSWAK更名：`_PTS` to `ZPTS`、`_WAK` to `ZWAK`。

  盖子状态更名：`_LID` to `XLID`

  按键更名：`_Q13 to XQ13` (TP-Fn+F4)

  补丁组合：

  - ***SSDT-PTSWAK***：综合补丁。根据自己的需要设置 `MODE`。
  - ***SSDT-LIDpatch***：盖子状态补丁。修改补丁内 `LID0` 为 `LID`。
  - ***SSDT-FnF4_Q13-X1C5th***：睡眠按键补丁。

  注意**1**：X1C5th 的睡眠按键是 Fn+4，有的TP的睡眠按键是 Fn+F4。

  注意**2**：TP 机器 `LPC` 控制器名称可能是 `LPC`、也可能是 `LPCB`。

其他机器修复 `PNP0C0E` 睡眠

- 使用补丁：***SSDT-PTSWAK*** ；更名：`_PTS` to `ZPTS`、`_WAK` to `ZWAK`。见《PTSWAK综合扩展补丁》。

  根据自己的需要修改 `MODE`。
- 使用补丁：***SSDT-LIDpatch*** ；更名：`_LID` to `XLID`。

  注意：`PNP0C0D` 设备名称、路径要和ACPI一致。
- 查找睡眠按键位置、制作 睡眠按键补丁
  - 一般情况下，睡眠按键是 `EC` 下的 `_Qxx`，这个 `_Qxx` 里包涵 `Notify(***.SLPB,0x80)` 指令。如果查找不到，DSDT 全文搜索 `Notify(***.SLPB,0x80)` ，找到其所在位置，逐步向上查找最初位置。

- 参考示例制作睡眠按键补丁以及必要的更名。

注意1：SLPB是 `PNP0C0E` 设备名称。如果确认没有 `PNP0C0E` 设备，添加补丁：SSDT-SLPB（位于《添加缺失的部件》）。

注意2： `PNP0C0D` 设备名称、路径要和ACPI一致。

`PNP0C0E` 睡眠特点

- 睡眠过程稍快。
- 睡眠过程无法被终止。

`PNP0C0D` 睡眠特点

- 睡眠过程中，再次按下睡眠按键立即终止睡眠。
- 接入外显时，按下睡眠按键后，工作屏幕为外显（内屏灭）；再次按下睡眠按键，内屏、外显正常。

注意事项

- `PNP0C0E` 和 `PNP0C0D` 设备名称、路径要和ACPI一致。

```
// In config ACPI, _Q13 to XQ13(TP-Fn+F4)
// Find:     5F 51 31 33
// Replace:  58 51 31 33
//
DefinitionBlock("", "SSDT", 2, "OCLT", "FnF4", 0)
{
    External(_SB.PCI9.FNOK, IntObj)
    External(_SB.PCI9.MODE, IntObj)
    External(_SB.LID, DeviceObj)
    External(_SB.PCI0.LPCB.EC, DeviceObj)
    External(_SB.PCI0.LPCB.EC.XQ13, MethodObj)

    Scope (_SB.PCI0.LPCB.EC)
    {
        Method (_Q13, 0, NotSerialized)
        {
            If (_OSI ("Darwin"))
            {
                If (\_SB.PCI9.MODE == 1) //PNP0C0E
                {
                    \_SB.PCI9.FNOK =1
                    \_SB.PCI0.LPCB.EC.XQ13()
                }
                Else //PNP0C0D
                {
                    If (\_SB.PCI9.FNOK!=1)
                    {
                        \_SB.PCI9.FNOK =1
                    }
                    Else
                    {
                        \_SB.PCI9.FNOK =0
                    }
                    Notify (\_SB.LID, 0x80)
                }
            }
            Else
            {
                \_SB.PCI0.LPCB.EC.XQ13()
            }
        }
    }
}
//EOF
```

```
// Overriding BTNV
// In config ACPI, BTNV to XTNV(dell-Fn+Insert)
// Find:     42 54 4E 56 02
// Replace:  58 54 4E 56 02
//
DefinitionBlock("", "SSDT", 2, "OCLT", "FnInsert", 0)
{
    External(_SB.PCI9.FNOK, IntObj)
    External(_SB.PCI9.MODE, IntObj)
    External(_SB.XTNV, MethodObj)

    Scope (_SB)
    {
        Method (BTNV, 2, NotSerialized)
        {
            If (_OSI ("Darwin") && (Arg0 == 2))
            {
                If (\_SB.PCI9.MODE == 1) //PNP0C0E
                {
                    \_SB.PCI9.FNOK =1
                    \_SB.XTNV(Arg0, Arg1)
                }
                Else //PNP0C0D
                {
                    If (\_SB.PCI9.FNOK!=1)
                    {
                        \_SB.PCI9.FNOK =1
                    }
                    Else
                    {
                        \_SB.PCI9.FNOK =0
                    }
                    \_SB.XTNV(0x03, Arg1)
                }
            }
            Else
            {
                \_SB.XTNV(Arg0, Arg1)
            }
        }
    }
}
//EOF
```

```
//
// In config ACPI, _LID to XLID
// Find:     5F4C4944 00
// Replace:  584C4944 00
//
DefinitionBlock("", "SSDT", 2, "OCLT", "LIDpatch", 0)
{
    //note:_LID 's path
    //path:_SB.LID0._LID
    External(_SB.LID0, DeviceObj)
    External(_SB.LID0.XLID, MethodObj)
    External(_SB.PCI9.FNOK, IntObj)
    Scope (_SB.LID0)
    {
        Method (_LID, 0, NotSerialized)
        {
            If (_OSI ("Darwin"))
            {
                if(\_SB.PCI9.FNOK==1)
                {
                    Return (0)
                }
                Else
                {
                    Return (\_SB.LID0.XLID())
                }
            }
            Else
            {
                Return (\_SB.LID0.XLID())
            }
        }
    }
}
//EOF
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/Pro
pertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>ACPI</key>
    <dict>
        <key>Patch</key>
        <array>
            <dict>
                <key>Comment</key>
                <string>_LID to XLID</string>
                <key>Count</key>
                <integer>0</integer>
                <key>Enabled</key>
                <true/>
                <key>Find</key>
                <data>
                X0xJRAA=
                </data>
                <key>Limit</key>
                <integer>0</integer>
                <key>Mask</key>
                <data>
                </data>
                <key>OemTableId</key>
                <data>
                </data>
                <key>Replace</key>
                <data>
                WExJRAA=
                </data>
                <key>ReplaceMask</key>
                <data>
                </data>
                <key>Skip</key>
                <integer>0</integer>
                <key>TableLength</key>
                <integer>0</integer>
                <key>TableSignature</key>
                <data>
                </data>
            </dict>
            <dict>
                <key>Comment</key>
                <string>BTNV to XTNV(dell-Fn+Insert)</string>
                <key>Count</key>
                <integer>0</integer>
                <key>Enabled</key>
                <true/>
```

```
                <key>Find</key>
                <data>
                QlROVgI=
                </data>
                <key>Limit</key>
                <integer>0</integer>
                <key>Mask</key>
                <data>
                </data>
                <key>OemTableId</key>
                <data>
                </data>
                <key>Replace</key>
                <data>
                WFROVgI=
                </data>
                <key>ReplaceMask</key>
                <data>
                </data>
                <key>Skip</key>
                <integer>0</integer>
                <key>TableLength</key>
                <integer>0</integer>
                <key>TableSignature</key>
                <data>
                </data>
            </dict>
            <dict>
                <key>Comment</key>
                <string>_Q13 to XQ13(TP-Fn+F4)</string>
                <key>Count</key>
                <integer>0</integer>
                <key>Enabled</key>
                <true/>
                <key>Find</key>
                <data>
                X1ExMw==
                </data>
                <key>Limit</key>
                <integer>0</integer>
                <key>Mask</key>
                <data>
                </data>
                <key>OemTableId</key>
                <data>
                </data>
                <key>Replace</key>
                <data>
                WFExMw==
                </data>
                <key>ReplaceMask</key>
```

```
                <data>
                </data>
                <key>Skip</key>
                <integer>0</integer>
                <key>TableLength</key>
                <integer>0</integer>
                <key>TableSignature</key>
                <data>
                </data>
            </dict>
        </array>
    </dict>
</dict>
</plist>
```

0D/6D 补丁

概述

- `_PRW` 定义了一个部件的唤醒方法。其 `Return` 2 个或者 2 个以上字节组成的数据包。有关 `_PRW` 详细的内容参见 ACPI 规范。
- 有这么一些部件，由于他们的 `_PRW` 和 macOS 发生了冲突从而导致机器刚刚睡眠成功就被立即唤醒。为了解决问题，必须对这些部件实施补丁。这些部件 `_PRW` 数据包的第 1 个字节是 `0D` 或者 `6D` 。因此，这种补丁被称为 `0D/6D补丁` ，也叫 `秒醒补丁` ，也叫 `睡了即醒补丁` 。为了描述方便，以下统一称之为 `0D/6D补丁` 。
- `_PRW` 数据包的第 2 个字节多为 `03` 或者 `04` ，将这个字节修正为 `0` 即完成了 `0D/6D补丁` 。
- 不同的机器对 `_PRW` 定义的方法可能不同，其数据包的内容、形式也可能多样化。实际的 `0D/6D补丁` 应视具体情况而定。见后文的描述。
- 我们期待 OpenCore 后续版本能够解决 `0D/6D` 问题。

可能需要 `0D/6D补丁` 的部件

- USB 类设备
  - `ADR` 地址： `0x001D0000` ，部件名称： `EHC1` 【6代之前】
  - `ADR` 地址： `0x001A0000` ，部件名称： `EHC2` 【6代之前】
  - `ADR` 地址： `0x00140000` ，部件名称： `XHC` ， `XHCI` ， `XHC1` 等
  - `ADR` 地址： `0x00140001` ，部件名称： `XDCI`
  - `ADR` 地址： `0x00140003` ，部件名称： `CNVW`
- 以太网
  - 6 代以前， `ADR` 地址： `0x00190000` ，部件名称： `GLAN` ， `IGBE` 等。
  - 6 代及 6 代以后， `ADR` 地址： `0x001F0006` ，部件名称： `GLAN` ， `IGBE` 等。
- 声卡
  - 6 代以前， `ADR` 地址： `0x001B0000` ，部件名称： `HDEF` ， `AZAL` 等。
  - 6 代及 6 代以后， `ADR` 地址： `0x001F0003` ，部件名称： `HDAS` ， `AZAL` 等。

  注意**1**：通过查找名称确认上述部件的方法并不可靠。可靠的方法是搜索 `ADR 地址` ， `_PRW` 。

  注意**2**：新发布的机器可能会有新的部件需要 `0D/6D补丁` 。

`_PRW` 的多样性和对应的补丁方法

- `Name 类型`

```
Name (_PRW, Package (0x02)
{
    0x0D, /* 可能是0x6D */
    0x03 ,/* 可能是0x04 */
```

```
        ...
    })
```

这种类型的 `0D/6D补丁` 适合用二进制更名方法修正 `0x03` （或 `0x04` ）为 `0x00` 。文件包提供了：

- Name-0D 更名 .plist
  - `Name0D-03` to `00`
  - `Name0D-04` to `00`
- Name-6D 更名 .plist
  - `Name6D-03` to `00`
  - `Name6D-04` to `00`

- `Method 类型` 之一： `GPRW(UPRW)`

```
    Method (_PRW, 0, NotSerialized)
    {
      Return (GPRW (0x6D, 0x04)) /* 或者Return (UPRW (0x6D, 0x04)) */
    }
```

较新的机器大多数属于这种情况。按常规方法（更名-补丁）即可。文件包提供了：

- ***SSDT-GPRW***（补丁文件内有二进制更名数据）
- ***SSDT-UPRW***（补丁文件内有二进制更名数据）

- `Method 类型` 之二： `Scope`

```
    Scope (_SB.PCI0.XHC)
    {
        Method (_PRW, 0, NotSerialized)
        {
            ...
            If ((Local0 == 0x03))
            {
                Return (Package (0x02)
                {
                    0x6D,
                    0x03
                })
            }
            If ((Local0 == One))
            {
                Return (Package (0x02)
                {
                    0x6D,
                    One
                })
            }
            Return (Package (0x02)
            {
```

```
                0x6D,
                Zero
            })
        }
    }
```

这种情况并不常见。对于示例的情况，使用二进制更名 ***Name6D-03 to 00*** 即可。其他形式内容自行尝试。

- `Name 类型`，`Method 类型` 混合方式

  对于多数 TP 机器，涉及 `0D/6D补丁` 的部件既有 `Name 类型` ，也有 `Method 类型` 。采用各自类型的补丁即可。需要注意的是二进制更名补丁不可滥用，有些不需要 `0D/6D补丁` 的部件 `_PRW` 也可能是 `0D` 或 `6D` 。为防止这种错误，应提取 `System DSDT` 文件加以验证、核实。

注意事项

- 本文描述的方法适用于Hotpatch。
- 凡是用到了二进制更名，应提取 `System DSDT` 文件加以验证。
- 惠普机器 `06/0D` 补丁比较特殊，详见《12-2-惠普特殊的060D补丁》

0D/6D 补丁

概述

- `_PRW` 定义了一个部件的唤醒方法。其 `Return` 2 个或者 2 个以上字节组成的数据包。有关 `_PRW` 详细的内容参见 ACPI 规范。
- 有这么一些部件，由于他们的 `_PRW` 和 macOS 发生了冲突从而导致机器刚刚睡眠成功就被立即唤醒。为了解决问题，必须对这些部件实施补丁。这些部件 `_PRW` 数据包的第 1 个字节是 `0D` 或者 `6D` 。因此，这种补丁被称为 `0D/6D补丁` ，也叫 `秒醒补丁` ，也叫 `睡了即醒补丁` 。为了描述方便，以下统一称之为 `0D/6D补丁` 。
- `_PRW` 数据包的第 2 个字节多为 `03` 或者 `04` ，将这个字节修正为 `0` 即完成了 `0D/6D补丁` 。
- 不同的机器对 `_PRW` 定义的方法可能不同，其数据包的内容、形式也可能多样化。实际的 `0D/6D补丁` 应视具体情况而定。见后文的描述。
- 我们期待 OpenCore 后续版本能够解决 `0D/6D` 问题。

可能需要 `0D/6D补丁` 的部件

- USB 类设备
  - `ADR` 地址： `0x001D0000` ，部件名称： `EHC1` 。
  - `ADR` 地址： `0x001A0000` ，部件名称： `EHC2` 。
  - `ADR` 地址： `0x00140000` ，部件名称： `XHC` ， `XHCI` ， `XHC1` 等。
  - `ADR` 地址： `0x00140001` ，部件名称： `XDCI` 。
  - `ADR` 地址： `0x00140003` ，部件名称： `CNVW` 。
- 以太网
  - 6 代以前， `ADR` 地址： `0x00190000` ，部件名称： `GLAN` ， `IGBE` 等。
  - 6 代及 6 代以后， `ADR` 地址： `0x001F0006` ，部件名称： `GLAN` ， `IGBE` 等。
- 声卡
  - 6 代以前， `ADR` 地址： `0x001B0000` ，部件名称： `HDEF` ， `AZAL` 等。
  - 6 代及 6 代以后， `ADR` 地址： `0x001F0003` ，部件名称： `HDAS` ， `AZAL` 等。

  注意**1**：通过查找名称确认上述部件的方法并不可靠。可靠的方法是搜索 `ADR 地址` ， `_PRW` 。

  注意**2**：新发布的机器可能会有新的部件需要 `0D/6D补丁` 。

`_PRW` 的多样性和对应的补丁方法

- `Name 类型`

```
Name (_PRW, Package (0x02)
{
    0x0D, /* 可能是0x6D */
    0x03 ,/* 可能是0x04 */
```

```
        ...
    })
```

这种类型的 `0D/6D补丁` 适合用二进制更名方法修正 `0x03`（或 `0x04`）为 `0x00`。文件包提供了：

- Name-0D 更名 .plist
  - `Name0D-03` to `00`
  - `Name0D-04` to `00`
- Name-6D 更名 .plist
  - `Name6D-03` to `00`
  - `Name6D-04` to `00`

- `Method 类型` 之一：`GPRW(UPRW)`

```
    Method (_PRW, 0, NotSerialized)
    {
      Return (GPRW (0x6D, 0x04)) /* 或者Return (UPRW (0x6D, 0x04)) */
    }
```

较新的机器大多数属于这种情况。按常规方法（更名-补丁）即可。文件包提供了：

- ***SSDT-GPRW***（补丁文件内有二进制更名数据）
- ***SSDT-UPRW***（补丁文件内有二进制更名数据）

- `Method 类型` 之二：`Scope`

```
    Scope (_SB.PCI0.XHC)
    {
        Method (_PRW, 0, NotSerialized)
        {
            ...
            If ((Local0 == 0x03))
            {
                Return (Package (0x02)
                {
                    0x6D,
                    0x03
                })
            }
            If ((Local0 == One))
            {
                Return (Package (0x02)
                {
                    0x6D,
                    One
                })
            }
            Return (Package (0x02)
            {
```

```
                0x6D,
                Zero
            })
        }
    }
```

这种情况并不常见。对于示例的情况，使用二进制更名 ***Name6D-03 to 00*** 即可。其他形式内容自行尝试。

- `Name 类型`，`Method 类型` 混合方式

  对于多数 TP 机器，涉及 `0D/6D补丁` 的部件既有 `Name 类型` ，也有 `Method 类型` 。采用各自类型的补丁即可。需要注意的是二进制更名补丁不可滥用，有些不需要 `0D/6D补丁` 的部件 `_PRW` 也可能是 `0D` 或 `6D` 。为防止这种错误，应提取 `System DSDT` 文件加以验证、核实。

注意事项

- 本文描述的方法适用于Hotpatch。
- 凡是用到了二进制更名，应提取 `System DSDT` 文件加以验证。

```
//
// In config ACPI, GPRW to XPRW
// Find:     47505257 02
// Replace:  58505257 02
//
DefinitionBlock ("", "SSDT", 2, "OCLT", "GPRW", 0)
{
    External(XPRW, MethodObj)
    Method (GPRW, 2, NotSerialized)
    {
        If (_OSI ("Darwin"))
        {
            If ((0x6D == Arg0))
            {
                Return (Package ()
                {
                    0x6D,
                    Zero
                })
            }

            If ((0x0D == Arg0))
            {
                Return (Package ()
                {
                    0x0D,
                    Zero
                })
            }
        }
        Return (XPRW (Arg0, Arg1))
    }
}
```

```
//
// In config ACPI, UPRW to XPRW
// Find:     55505257 02
// Replace:  58505257 02
//
DefinitionBlock ("", "SSDT", 2, "OCLT", "UPRW", 0)
{
    External(XPRW, MethodObj)
    Method (UPRW, 2, NotSerialized)
    {
        If (_OSI ("Darwin"))
        {
            If ((0x6D == Arg0))
            {
                Return (Package ()
                {
                    0x6D,
                    Zero
                })
            }

            If ((0x0D == Arg0))
            {
                Return (Package ()
                {
                    0x0D,
                    Zero
                })
            }
        }
        Return (XPRW (Arg0, Arg1))
    }
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/Pro
pertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>ACPI</key>
    <dict>
        <key>Patch</key>
        <array>
            <dict>
                <key>Comment</key>
                <string>Name0D-03 to 00</string>
                <key>Count</key>
                <integer>0</integer>
                <key>Enabled</key>
                <true/>
                <key>Find</key>
                <data>
                Cg0KAw==
                </data>
                <key>Limit</key>
                <integer>0</integer>
                <key>Mask</key>
                <data>
                </data>
                <key>OemTableId</key>
                <data>
                </data>
                <key>Replace</key>
                <data>
                Cg0KAA==
                </data>
                <key>ReplaceMask</key>
                <data>
                </data>
                <key>Skip</key>
                <integer>0</integer>
                <key>TableLength</key>
                <integer>0</integer>
                <key>TableSignature</key>
                <data>
                </data>
            </dict>
            <dict>
                <key>Comment</key>
                <string>Name0D-04 to 00</string>
                <key>Count</key>
                <integer>0</integer>
                <key>Enabled</key>
                <true/>
```

```
                <key>Find</key>
                <data>
                Cg0KBA==
                </data>
                <key>Limit</key>
                <integer>0</integer>
                <key>Mask</key>
                <data>
                </data>
                <key>OemTableId</key>
                <data>
                </data>
                <key>Replace</key>
                <data>
                Cg0KAA==
                </data>
                <key>ReplaceMask</key>
                <data>
                </data>
                <key>Skip</key>
                <integer>0</integer>
                <key>TableLength</key>
                <integer>0</integer>
                <key>TableSignature</key>
                <data>
                </data>
            </dict>
        </array>
    </dict>
</dict>
</plist>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/Pro
pertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>ACPI</key>
    <dict>
        <key>Patch</key>
        <array>
            <dict>
                <key>Comment</key>
                <string>Name6D-03 to 00</string>
                <key>Count</key>
                <integer>0</integer>
                <key>Enabled</key>
                <true/>
                <key>Find</key>
                <data>
                Cm0KAw==
                </data>
                <key>Limit</key>
                <integer>0</integer>
                <key>Mask</key>
                <data>
                </data>
                <key>OemTableId</key>
                <data>
                </data>
                <key>Replace</key>
                <data>
                Cm0KAA==
                </data>
                <key>ReplaceMask</key>
                <data>
                </data>
                <key>Skip</key>
                <integer>0</integer>
                <key>TableLength</key>
                <integer>0</integer>
                <key>TableSignature</key>
                <data>
                </data>
            </dict>
            <dict>
                <key>Comment</key>
                <string>Name6D-04 to 00</string>
                <key>Count</key>
                <integer>0</integer>
                <key>Enabled</key>
                <true/>
```

```
                <key>Find</key>
                <data>
                Cm0KBA==
                </data>
                <key>Limit</key>
                <integer>0</integer>
                <key>Mask</key>
                <data>
                </data>
                <key>OemTableId</key>
                <data>
                </data>
                <key>Replace</key>
                <data>
                Cm0KAA==
                </data>
                <key>ReplaceMask</key>
                <data>
                </data>
                <key>Skip</key>
                <integer>0</integer>
                <key>TableLength</key>
                <integer>0</integer>
                <key>TableSignature</key>
                <data>
                </data>
            </dict>
        </array>
    </dict>
</dict>
</plist>
```

惠普特殊的 `0D6D` 补丁

- 关于 `0D/6D补丁` 的相关内容请参阅《0D6D补丁》
- 某些惠普机器，其 `ACPI` 的一些部件（和 `0D6D` 有关的）的 `_PRW` 方法如下：

```
Method (_PRW, 0, NotSerialized)
{
    Local0 = Package (0x02)
    {
        Zero,
        Zero
    }
    Local0 [Zero] = 0x6D
    If ((USWE == One))    /* 注意USWE */
    {
        Local0 [One] = 0x03
    }
    Return (Local0)
}
```

这种情况可以使用《预置变量法》完成 `0D/6D补丁` ，如：

```
Scope (\)
{
    If (_OSI ("Darwin"))
    {
        USWE = 0
    }
}
```

- 示例： ***SSDT-0D6D-HP***

  SSDT-0D6D-HP 适用于 `HP 840 G3` ，补丁修补了 `XHC` 、 `GLAN` 的 `_PRW` 返回值。

- 其他有类似情况的机器参考示例文件。

```
// 0D6D patch
DefinitionBlock("", "SSDT", 2, "OCLT", "0D6D", 0)
{
    External(USWE, FieldUnitObj)
    External(WOLE, FieldUnitObj)

    //0D6D patch:HP-XHC
    Scope (\)
    {
        If (_OSI ("Darwin"))
        {
            USWE = 0
        }
    }

    //0D6D patch:HP-GLAN
    Scope (\)
    {
        If (_OSI ("Darwin"))
        {
            WOLE = 0
        }
    }
}
//EOF
```

仿冒以太网

描述

一些机器无以太网，仿冒以太网实现APP正常下载功能。

使用

- 添加 ***SSDT-LAN*** 至 `OC\ACPI` 。
- 添加 **NullEthernet.kext** 至 `OC\Kexts` 。
- 在config中添加相关列表。

附：重置以太网 `BSD Name` 为 `en0`

- 打开 系统偏好设置 的 网络 。
- 删除所有网络，如图《清除所有网络》。
- 删除 `资源库\Preferences\SystemConfiguration\NetworkInterfaces.plist` 文件。
- 重启。
- 再次进入系统后，再次打开 系统偏好设置 的 网络 。
- 依次按顺序添加 以太网 和其他需要的网络，点击 应用。


网络
搜索
位置：
自动
以太网
自分配的 IP
Wi-Fi
已连接
蓝牙 PAN
未连接
全部删除
状态：
已连接
“以太网”有一个自分配的 IP 地址，将无法接入互联网。
配置 IPv4：
使用 DHCP
IP 地址：
169.254.167.83
子网掩码：
255.255.0.0
路由器：
DNS 服务器：
搜索域：
点击删除
删除后点击应用
高级...
?
复原
应用
blog.tlhub.cn

```
//Fake LAN
DefinitionBlock ("", "SSDT", 2, "OCLT", "FakeLAN", 0x00001000)
{
    Device (RMNE)
    {
        Name (_ADR, Zero)
        Name (_HID, "NULE0000")
        Name (MAC, Buffer (0x06)
        {
             0x11, 0x22, 0x33, 0x44, 0x55, 0x66
        })
        Method (_DSM, 4, NotSerialized)
        {
            If (LEqual (Arg2, Zero)) { Return (Buffer() { 0x03 } ) }
            Return (Package()
            {
                "built-in", Buffer() { 0x00 },
                "IOName", "ethernet",
                "name", Buffer() { "ethernet" },
                "model", Buffer() { "RM-NullEthernet-1001" },
                "device_type", Buffer() { "ethernet" },
            })
        }

        Method (_STA, 0, NotSerialized)
        {
            If (_OSI ("Darwin"))
            {
                Return (0x0F)
            }
            Else
            {
                Return (Zero)
            }
        }
    }
}
//EOF
```

CMOS相关

CMOS重置补丁

描述

- 某些机器在关机或者重启时会出现 **“开机自检错误”**，这是由于 CMOS 被重置所导致的。
- 当使用 Clover 时，勾选 `ACPI\FixRTC` 可以解决上述问题。
- 当使用 OpenCore 时，官方提供了以下解决方法，见 ***Sample.plist*** ：
  - 安装 **RTCMemoryFixup.kext**
  - `Kernel\Patch` 补丁：**__ZN11BCM5701Enet14getAdapterInfoEv**
- 本章提供一种 SSDT 补丁方法来解决上述问题。这个 SSDT 补丁本质是仿冒 RTC，见《预置变量法》和《仿冒设备》。

解决方案

详见《15-1-CMOS重置补丁》。

CMOS 内存和 RTCMemoryFixup

- 当 **AppleRTC** 与 **BIOS** 发生冲突而时，可以尝试使用 **RTCMemoryFixup** 模拟 **CMOS** 内存规避冲突。
- **RTCMemoryFixup** 下载地址：https://github.com/acidanthera/RTCMemoryFixup

CMOS 内存

- **CMOS** 内存保存着日期、时间、硬件配置信息、辅助设置信息、启动设置、休眠信息等重要数据。
- 一些 **CMOS** 内存空间定义：
  - 日期、时间：`00-0D`
  - 休眠信息存放区间：`80-AB`
  - 电源管理：`B0-B4`
  - 其他

模拟 CMOS 内存方法

详见《15-2-CMOS内存和RTCMemoryFixup》。

CMOS 内存和 RTCMemoryFixup

描述

- 当 **AppleRTC** 与 **BIOS** 发生冲突而时，可以尝试使用 **RTCMemoryFixup** 模拟 **CMOS** 内存规避冲突。
- **RTCMemoryFixup** 下载地址：https://github.com/acidanthera/RTCMemoryFixup

CMOS 内存

- **CMOS** 内存保存着日期、时间、硬件配置信息、辅助设置信息、启动设置、休眠信息等重要数据。
- 一些 **CMOS** 内存空间定义：
  - 日期、时间、硬件配置：`00-0D`
  - 休眠信息存放区间：`80-AB`
  - 电源管理：`B0-B4`
  - 其他

查看COMS内存

- `EFI\OC\Tools` 安装 ***RtcRw.efi***
- config添加 ***RtcRw.efi*** 的 `items`
- 引导界面进入 `Shell` 【确保已经安装 ***OpenShell.efi*** 】，进入目录tools，键入rtcrw read XX并回车。其中XX是CMOS内存地址。如：rtcrw read 08 可以查看当前月份。如本月是5月份，查看结果为 0x05（BCD码）。

模拟 CMOS 内存

- 安装 **RTCMemoryFixup** 至 `OC\Kexts`，并添加驱动列表。
- 引导 **`boot-args`** 添加 `rtcfx_exclude=...`

  格式：`rtcfx_exclude=offset1,offset2,start_offset-end_offset,...`
  如：`rtcfx_exclude=0D` ，`rtcfx_exclude=40-AF` ，`rtcfx_exclude=2A,2D,80-AB` 等。

注意事项

- 模拟 **CMOS** 内存会抹掉原来定义的功能，请 谨慎使用 。如: `rtcfx_exclude=00-0D` 将导致睡眠期间机器的日期、时间不再更新。

附录：CMOS 内存 **`00-3F`** 定义

地址	说明
0	秒
1	秒报警
2	分
3	分报警
4	时
5	时报警
6	星期
7	日
8	月
9	年
A	状态寄存器 A
B	状态寄存器 B
C	状态寄存器 C
D	状态寄存器 D (0:电池失效；80:电池有效)
E	诊断状态字节
F	关机状态字节 (上电诊断定义)
10	软盘驱动器类型 (位 7-4: A 驱，位 3-0: B 驱 1-360KB; 2-1.2MB; 6-1.44MB; 7-720KB)
11	保留
12	硬盘驱动器类型 (位 7-4: C驱，位 3-0: D 驱)
13	保留
14	设备字节 (软驱数目，显示器类型，协处理器)
15	基本存储器低字节
16	基本存储器高字节
17	扩展存储器低字节
18	扩展存储器高字节
19	第一主硬盘类型
1A	第一从硬盘类型
1B–1C	保留
1D–24	第一主硬盘的柱面、磁头、扇区等
25–2C	第一从硬盘的柱面、磁头、扇区等
2D	保留

2E–2F	CMOS 校验和 (10-2D 各字节和)
30	扩充存储器低字节
31	扩充存储器高字节
32	日期世纪字节(19H:19 世纪)
33	信息标志
34–3F	保留

ACPI定制USB端口

描述

- 本方法通过修改 ACPI 文件实现USB端口的定制。
- 本方法操作过程需要 drop 某个 ACPI 文件。通常情况下，OpenCore 不建议 这样做，定制 USB 端口一般使用 ***Hackintool.app*** 工具。
- 本方法献给爱好者们。

适用范围

- XHC 以及它的 `_UPC` 存在于单独的 ACPI 文件中
- 此方法不适用于 `_UPC` 存在于 DSDT 中的设备 (e.g. 华硕)

`_UPC` 规范

```
_UPC, Package ()
{
    xxxx,
    yyyy,
    0x00,
    0x00
}
```

解释

1. **`xxxx`**
   - `0x00` 代表这个端口不存在
   - 其他值 (通常为 `0x0F`) 代表这个端口存在
2. **`yyyy`**

   `yyyy` 处定义的是端口的类型, 参考下表

   | **`yyyy`** | 端口类型 | | :------: | ---------- | | `0x00` | USB Type `A` | | `0x01` | USB `Mini-AB` | | `0x02` | USB 智能卡 | | `0x03` | USB 3 标准 Type `A` | | `0x04` | USB 3 标准 Type `B` | | `0x05` | USB 3 `Micro-B` | | `0x06` | USB 3 `Micro-AB` | | `0x07` | USB 3 `Power-B` | | `0x08` | USB Type `C` **(只有 USB 2)** | | `0x09` | USB Type `C` **(带有转向器)** | | `0x0A` | USB Type `C` **(不带转向器)** | | `0xFF` | 内置 |

   > 如果 USB-C 正反两面插入在 Hackintool 都显示为同一个端口就说明这个端口有转向器
   >
   > 反之，如果正反两面占用了两个端口就说明没有转向器

USB定制过程

- 清除其他定制方法的补丁、驱动等。
- drop ACPI文件
  - 确认 XHC 并包括 `_UPC` 的 ACPI 文件

    > 如 dell5480 的 ***SSDT-2-xh_OEMBD.aml***
    >
    > 如 小新 PRO13（i5）的 ***SSDT-8-CB-01.aml***（无独显机器是 ***SSDT-6-CB-01.aml***）

  - `config\ACPI\Block\` 以 `TableLength`（十进制）和 `TableSignature` 方式 drop ACPI 文件。如：

    dell5480：`TableLength` = `2001`，`TableSignature` = `53534454`（SSDT）

    小新**PRO13（i5）**：`TableLength` = `12565`，`TableSignature` = `53534454`（SSDT）
- 定制 SSDT 补丁文件
  - 将需要 drop 的原始 ACPI 文件拖到桌面，建议：
    - 另存为 `.asl / .dsl` 格式
    - 修改文件名。如：***SSDT-xh_OEMBD_XHC.dsl***, ***SSDT-CB-01_XHC.dsl***
    - 修改文件内的 `OEM Table ID` 为自己喜欢的名字。
    - 排除错误。
  - SSDT 文件中所有端口的 `_UPC` 最前端添加以下代码：

```
Method (_UPC, 0, NotSerialized)
{
    If (_OSI ("Darwin"))
    {
        Return (Package ()
        {
            xxxx,
            yyyy,
            0x00,
            0x00
        })
    }
    /* 以下是原内容 */
    ...
}
```

  - 根据 `_UPC` 规范定制 USB 端口。即，修正 xxxx、yyyy 的值。
    - 如果端口不存在
      - **`xxxx`** = `0x00`
      - **`yyyy`** = `0x00`
    - 如果端口存在

        - **`xxxx`** = `0xFF`
        - **`yyyy`**

        > 参考上文的表格

    - 排错、编译、补丁文件放至 `ACPI` 、添加补丁列表。

参考示例

- ***SSDT-xh_OEMBD_XHC***
- ***SSDT-CB-01_XHC***

```
//
DefinitionBlock ("", "SSDT", 2, "OCLT", "MY-CB-01", 0x00000001)
{
    External (_SB_.PCI0.LPCB.CRID, IntObj)
    External (_SB_.PCI0.RP01.PXSX.WIST, MethodObj)    // 0 Arguments
    External (_SB_.PCI0.RP02.PXSX.WIST, MethodObj)    // 0 Arguments
    External (_SB_.PCI0.RP03.PXSX.WIST, MethodObj)    // 0 Arguments
    External (_SB_.PCI0.RP04.PXSX.WIST, MethodObj)    // 0 Arguments
    External (_SB_.PCI0.RP05.PXSX.WIST, MethodObj)    // 0 Arguments
    External (_SB_.PCI0.RP06.PXSX.WIST, MethodObj)    // 0 Arguments
    External (_SB_.PCI0.RP07.PXSX.WIST, MethodObj)    // 0 Arguments
    External (_SB_.PCI0.RP08.PXSX.WIST, MethodObj)    // 0 Arguments
    External (_SB_.PCI0.RP09.PXSX.WIST, MethodObj)    // 0 Arguments
    External (_SB_.PCI0.RP10.PXSX.WIST, MethodObj)    // 0 Arguments
    External (_SB_.PCI0.RP11.PXSX.WIST, MethodObj)    // 0 Arguments
    External (_SB_.PCI0.RP12.PXSX.WIST, MethodObj)    // 0 Arguments
    External (_SB_.PCI0.RP13.PXSX.WIST, MethodObj)    // 0 Arguments
    External (_SB_.PCI0.RP14.PXSX.WIST, MethodObj)    // 0 Arguments
    External (_SB_.PCI0.RP15.PXSX.WIST, MethodObj)    // 0 Arguments
    External (_SB_.PCI0.RP16.PXSX.WIST, MethodObj)    // 0 Arguments
    External (_SB_.PCI0.RP17.PXSX.WIST, MethodObj)    // 0 Arguments
    External (_SB_.PCI0.RP18.PXSX.WIST, MethodObj)    // 0 Arguments
    External (_SB_.PCI0.RP19.PXSX.WIST, MethodObj)    // 0 Arguments
    External (_SB_.PCI0.RP20.PXSX.WIST, MethodObj)    // 0 Arguments
    External (_SB_.PCI0.XDCI, DeviceObj)
    External (_SB_.PCI0.XHC_.RHUB, DeviceObj)
    External (_SB_.PCI0.XHC_.RHUB.HS01, DeviceObj)
    External (_SB_.PCI0.XHC_.RHUB.HS02, DeviceObj)
    External (_SB_.PCI0.XHC_.RHUB.HS03, DeviceObj)
    External (_SB_.PCI0.XHC_.RHUB.HS04, DeviceObj)
    External (_SB_.PCI0.XHC_.RHUB.HS05, DeviceObj)
    External (_SB_.PCI0.XHC_.RHUB.HS06, DeviceObj)
    External (_SB_.PCI0.XHC_.RHUB.HS07, DeviceObj)
    External (_SB_.PCI0.XHC_.RHUB.HS08, DeviceObj)
    External (_SB_.PCI0.XHC_.RHUB.HS09, DeviceObj)
    External (_SB_.PCI0.XHC_.RHUB.HS10, DeviceObj)
    External (_SB_.PCI0.XHC_.RHUB.HS11, DeviceObj)
    External (_SB_.PCI0.XHC_.RHUB.HS12, DeviceObj)
    External (_SB_.PCI0.XHC_.RHUB.HS13, DeviceObj)
    External (_SB_.PCI0.XHC_.RHUB.HS14, DeviceObj)
    External (_SB_.PCI0.XHC_.RHUB.SS01, DeviceObj)
    External (_SB_.PCI0.XHC_.RHUB.SS02, DeviceObj)
    External (_SB_.PCI0.XHC_.RHUB.SS03, DeviceObj)
    External (_SB_.PCI0.XHC_.RHUB.SS04, DeviceObj)
    External (_SB_.PCI0.XHC_.RHUB.SS05, DeviceObj)
    External (_SB_.PCI0.XHC_.RHUB.SS06, DeviceObj)
    External (_SB_.PCI0.XHC_.RHUB.SS07, DeviceObj)
    External (_SB_.PCI0.XHC_.RHUB.SS08, DeviceObj)
    External (_SB_.PCI0.XHC_.RHUB.SS09, DeviceObj)
    External (_SB_.PCI0.XHC_.RHUB.SS10, DeviceObj)
```

```
    External (_SB_.PCI0.XHC_.RHUB.USR1, DeviceObj)
    External (_SB_.PCI0.XHC_.RHUB.USR2, DeviceObj)
    External (_SB_.UBTC.RUCC, MethodObj)    // 2 Arguments
    External (ATDV, UnknownObj)
    External (BED2, UnknownObj)
    External (BED3, UnknownObj)
    External (BTBR, UnknownObj)
    External (BTL2, UnknownObj)
    External (BTLE, UnknownObj)
    External (BTLL, UnknownObj)
    External (BTSE, UnknownObj)
    External (CECV, UnknownObj)
    External (SXI1, IntObj)
    External (SXI2, IntObj)
    External (SXP1, IntObj)
    External (SXP2, IntObj)

    Name (NHSP, 0x0A)
    Name (NSSP, 0x06)
    Name (DBPN, 0x0A)
    Scope (\_SB.PCI0.XHC.RHUB)
    {
        Name (H1CN, One)
        Name (H1VS, Zero)
        Name (H1TC, Zero)
        Name (H1CR, Zero)
        Name (H2CN, Zero)
        Name (H2VS, Zero)
        Name (H2TC, Zero)
        Name (H2CR, Zero)
        Name (H3CN, One)
        Name (H3VS, One)
        Name (H3TC, One)
        Name (H3CR, One)
        Name (H4CN, One)
        Name (H4VS, One)
        Name (H4TC, One)
        Name (H4CR, 0x02)
        Name (H5CN, Zero)
        Name (H5VS, Zero)
        Name (H5TC, Zero)
        Name (H5CR, Zero)
        Name (H6CN, Zero)
        Name (H6VS, Zero)
        Name (H6TC, Zero)
        Name (H6CR, Zero)
        Name (H7CN, One)
        Name (H7VS, One)
        Name (H7TC, Zero)
        Name (H7CR, Zero)
        Name (H8CN, Zero)
```

```
        Name (H8VS, Zero)
        Name (H8TC, Zero)
        Name (H8CR, Zero)
        Name (H9CN, Zero)
        Name (H9VS, Zero)
        Name (H9TC, Zero)
        Name (H9CR, Zero)
        Name (HACN, One)
        Name (HAVS, Zero)
        Name (HATC, Zero)
        Name (HACR, Zero)
        Name (HBCN, Zero)
        Name (HBVS, Zero)
        Name (HBTC, Zero)
        Name (HBCR, Zero)
        Name (HCCN, Zero)
        Name (HCVS, Zero)
        Name (HCTC, Zero)
        Name (HCCR, Zero)
        Name (HDCN, Zero)
        Name (HDVS, Zero)
        Name (HDTC, Zero)
        Name (HDCR, Zero)
        Name (HECN, Zero)
        Name (HEVS, Zero)
        Name (HETC, Zero)
        Name (HECR, Zero)
        Name (S1CN, One)
        Name (S1VS, One)
        Name (S1CP, 0x07)
        Name (S1TC, Zero)
        Name (S1CR, Zero)
        Name (S2CN, One)
        Name (S2VS, One)
        Name (S2CP, 0x03)
        Name (S2TC, One)
        Name (S2CR, One)
        Name (S3CN, One)
        Name (S3VS, Zero)
        Name (S3CP, 0x06)
        Name (S3TC, Zero)
        Name (S3CR, Zero)
        Name (S4CN, One)
        Name (S4VS, One)
        Name (S4CP, 0x04)
        Name (S4TC, One)
        Name (S4CR, 0x02)
        Name (S5CN, One)
        Name (S5VS, Zero)
        Name (S5CP, 0x05)
        Name (S5TC, Zero)
```

```
        Name (S5CR, Zero)
        Name (S6CN, Zero)
        Name (S6VS, Zero)
        Name (S6CP, Zero)
        Name (S6TC, Zero)
        Name (S6CR, Zero)
        Name (S7CN, Zero)
        Name (S7VS, Zero)
        Name (S7CP, Zero)
        Name (S7TC, Zero)
        Name (S7CR, Zero)
        Name (S8CN, Zero)
        Name (S8VS, Zero)
        Name (S8CP, Zero)
        Name (S8TC, Zero)
        Name (S8CR, Zero)
        Name (S9CN, Zero)
        Name (S9VS, Zero)
        Name (S9CP, Zero)
        Name (S9TC, Zero)
        Name (S9CR, Zero)
        Name (SACN, Zero)
        Name (SAVS, Zero)
        Name (SACP, Zero)
        Name (SATC, Zero)
        Name (SACR, Zero)
        Method (GPLD, 2, Serialized)
        {
            Name (PCKG, Package (0x01)
            {
                Buffer (0x10){}
            })
            CreateField (DerefOf (PCKG [Zero]), Zero, 0x07, REV)
            REV = 0x02
            CreateField (DerefOf (PCKG [Zero]), 0x40, One, VISI)
            VISI = Arg0
            CreateField (DerefOf (PCKG [Zero]), 0x57, 0x08, GPOS)
            GPOS = Arg1
            Return (PCKG) /* \_SB_.PCI0.XHC_.RHUB.GPLD.PCKG */
        }

        Method (TPLD, 2, Serialized)
        {
            Name (PCKG, Package (0x01)
            {
                Buffer (0x10){}
            })
            CreateField (DerefOf (PCKG [Zero]), Zero, 0x07, REV)
            REV = One
            CreateField (DerefOf (PCKG [Zero]), 0x40, One, VISI)
            VISI = Arg0
```

```
            CreateField (DerefOf (PCKG [Zero]), 0x57, 0x08, GPOS)
            GPOS = Arg1
            CreateField (DerefOf (PCKG [Zero]), 0x4A, 0x04, SHAP)
            SHAP = One
            CreateField (DerefOf (PCKG [Zero]), 0x20, 0x10, WID)
            WID = 0x08
            CreateField (DerefOf (PCKG [Zero]), 0x30, 0x10, HGT)
            HGT = 0x03
            Return (PCKG) /* \_SB_.PCI0.XHC_.RHUB.TPLD.PCKG */
        }

        Method (GUPC, 1, Serialized)
        {
            Name (PCKG, Package (0x04)
            {
                Zero,
                0xFF,
                Zero,
                Zero
            })
            PCKG [Zero] = Arg0
            Return (PCKG) /* \_SB_.PCI0.XHC_.RHUB.GUPC.PCKG */
        }

        Method (TUPC, 1, Serialized)
        {
            Name (PCKG, Package (0x04)
            {
                One,
                Zero,
                Zero,
                Zero
            })
            PCKG [One] = Arg0
            Return (PCKG) /* \_SB_.PCI0.XHC_.RHUB.TUPC.PCKG */
        }
    }

    Method (CNDP, 0, NotSerialized)
    {
        If (\_SB.PCI0.RP01.PXSX.WIST ())
        {
            Return (One)
        }

        If (\_SB.PCI0.RP02.PXSX.WIST ())
        {
            Return (One)
        }

        If (\_SB.PCI0.RP03.PXSX.WIST ())
```

```
        {
            Return (One)
        }

        If (\_SB.PCI0.RP04.PXSX.WIST ())
        {
            Return (One)
        }

        If (\_SB.PCI0.RP05.PXSX.WIST ())
        {
            Return (One)
        }

        If (\_SB.PCI0.RP06.PXSX.WIST ())
        {
            Return (One)
        }

        If (\_SB.PCI0.RP07.PXSX.WIST ())
        {
            Return (One)
        }

        If (\_SB.PCI0.RP08.PXSX.WIST ())
        {
            Return (One)
        }

        If (\_SB.PCI0.RP09.PXSX.WIST ())
        {
            Return (One)
        }

        If (\_SB.PCI0.RP10.PXSX.WIST ())
        {
            Return (One)
        }

        If (\_SB.PCI0.RP11.PXSX.WIST ())
        {
            Return (One)
        }

        If (\_SB.PCI0.RP12.PXSX.WIST ())
        {
            Return (One)
        }

        If (\_SB.PCI0.RP13.PXSX.WIST ())
        {
```

```
            Return (One)
        }

        If (\_SB.PCI0.RP14.PXSX.WIST ())
        {
            Return (One)
        }

        If (\_SB.PCI0.RP15.PXSX.WIST ())
        {
            Return (One)
        }

        If (\_SB.PCI0.RP16.PXSX.WIST ())
        {
            Return (One)
        }

        If (\_SB.PCI0.RP17.PXSX.WIST ())
        {
            Return (One)
        }

        If (\_SB.PCI0.RP18.PXSX.WIST ())
        {
            Return (One)
        }

        If (\_SB.PCI0.RP19.PXSX.WIST ())
        {
            Return (One)
        }

        If (\_SB.PCI0.RP20.PXSX.WIST ())
        {
            Return (One)
        }

        Return (Zero)
    }

    If ((NHSP >= One))
    {
        Scope (\_SB.PCI0.XHC.RHUB.HS01)
        {
            Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
            {
                If (_OSI ("Darwin"))
                {
                    Return (Package ()
                    {
```

```
                        0xFF,
                        0xFF,
                        0x00,
                        0x00
                    })
                }

                If ((H1TC == Zero))
                {
                    Return (GUPC (H1CN))
                }
                Else
                {
                    Return (\_SB.UBTC.RUCC (H1CR, One))
                }
            }

            Method (_PLD, 0, Serialized)  // _PLD: Physical Location of Device
            {
                Name (PLDP, Package (0x01)
                {
                    Buffer (0x14)
                    {
                        /* 0000 */  0x82, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
// ........
                        /* 0008 */  0x24, 0x41, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00,
// $A......
                        /* 0010 */  0xC3, 0x00, 0xC3, 0x00
// ....
                    }
                })
                Return (PLDP) /* \_SB_.PCI0.XHC_.RHUB.HS01._PLD.PLDP */
            }

            Device (CAM0)
            {
                Name (_ADR, One)  // _ADR: Address
                Name (_UID, "MI_00")  // _UID: Unique ID
                Name (_PLD, Package (0x01)  // _PLD: Physical Location of Device
                {
                    ToPLD (
                        PLD_Revision          = 0x2,
                        PLD_IgnoreColor       = 0x1,
                        PLD_Red               = 0x0,
                        PLD_Green             = 0x0,
                        PLD_Blue              = 0x0,
                        PLD_Width             = 0x0,
                        PLD_Height            = 0x0,
                        PLD_UserVisible       = 0x0,
                        PLD_Dock              = 0x0,
                        PLD_Lid               = 0x1,
```

```
                    PLD_Panel               = "FRONT",
                    PLD_VerticalPosition    = "UPPER",
                    PLD_HorizontalPosition  = "CENTER",
                    PLD_Shape               = "ROUND",
                    PLD_GroupOrientation    = 0x1,
                    PLD_GroupToken          = 0x0,
                    PLD_GroupPosition       = 0x1,
                    PLD_Bay                 = 0x0,
                    PLD_Ejectable           = 0x0,
                    PLD_EjectRequired       = 0x0,
                    PLD_CabinetNumber       = 0x0,
                    PLD_CardCageNumber      = 0x0,
                    PLD_Reference           = 0x0,
                    PLD_Rotation            = 0x0,
                    PLD_Order               = 0x0,
                    PLD_VerticalOffset      = 0xC3,
                    PLD_HorizontalOffset    = 0xC3)

            })
        }

        Device (CAM1)
        {
            Name (_ADR, 0x03)  // _ADR: Address
            Name (_UID, "MI_02")  // _UID: Unique ID
            Name (_PLD, Package (0x01)  // _PLD: Physical Location of Device
            {
                ToPLD (
                    PLD_Revision            = 0x2,
                    PLD_IgnoreColor         = 0x1,
                    PLD_Red                 = 0x0,
                    PLD_Green               = 0x0,
                    PLD_Blue                = 0x0,
                    PLD_Width               = 0x0,
                    PLD_Height              = 0x0,
                    PLD_UserVisible         = 0x0,
                    PLD_Dock                = 0x0,
                    PLD_Lid                 = 0x1,
                    PLD_Panel               = "FRONT",
                    PLD_VerticalPosition    = "UPPER",
                    PLD_HorizontalPosition  = "CENTER",
                    PLD_Shape               = "ROUND",
                    PLD_GroupOrientation    = 0x1,
                    PLD_GroupToken          = 0x0,
                    PLD_GroupPosition       = 0x1,
                    PLD_Bay                 = 0x0,
                    PLD_Ejectable           = 0x0,
                    PLD_EjectRequired       = 0x0,
                    PLD_CabinetNumber       = 0x0,
                    PLD_CardCageNumber      = 0x0,
                    PLD_Reference           = 0x0,
```

```
                        PLD_Rotation            = 0x0,
                        PLD_Order               = 0x0,
                        PLD_VerticalOffset      = 0xC3,
                        PLD_HorizontalOffset    = 0xC3)

                })
            }
        }
    }

    If ((NHSP >= 0x02))
    {
        Scope (\_SB.PCI0.XHC.RHUB.HS02)
        {
            Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
            {
                If (_OSI ("Darwin"))
                {
                    Return (Package ()
                    {
                        0x00,
                        0x00,
                        0x00,
                        0x00
                    })
                }

                If ((H2TC == Zero))
                {
                    Return (GUPC (H2CN))
                }
                Else
                {
                    Return (\_SB.UBTC.RUCC (H2CR, One))
                }
            }

            Method (_PLD, 0, NotSerialized)  // _PLD: Physical Location of Device
            {
                If ((H2TC == Zero))
                {
                    Return (GPLD (H2VS, 0x02))
                }
                Else
                {
                    Return (\_SB.UBTC.RUCC (H2CR, 0x02))
                }
            }

            If (CondRefOf (DBPN))
            {
```

```
                If (((DBPN == 0x02) && CNDP ()))
                {
                    Name (SADX, Package (0x03)
                    {
                        Zero,
                        Package (0x02)
                        {
                            0x07,
                            0x80000000
                        },

                        Package (0x02)
                        {
                            0x12,
                            0x80000000
                        }
                    })
                    Method (SADS, 0, Serialized)
                    {
                        DerefOf (SADX [One]) [One] = \ATDV /* External reference */
                        DerefOf (SADX [0x02]) [One] = \ATDV /* External reference *
/
                        Return (SADX) /* \_SB_.PCI0.XHC_.RHUB.HS02.SADX */
                    }

                    Name (BRDY, Package (0x02)
                    {
                        Zero,
                        Package (0x08)
                        {
                            0x12,
                            0x80,
                            0x80,
                            0x80,
                            0x80,
                            0x80,
                            0x80,
                            0x80
                        }
                    })
                    Method (BRDS, 0, Serialized)
                    {
                        DerefOf (BRDY [One]) [One] = \BTSE /* External reference */
                        DerefOf (BRDY [One]) [0x02] = \BTBR /* External reference *
/
                        DerefOf (BRDY [One]) [0x03] = \BED2 /* External reference *
/
                        DerefOf (BRDY [One]) [0x04] = \BED3 /* External reference *
/
                        DerefOf (BRDY [One]) [0x05] = \BTLE /* External reference *
/
```

```
                        DerefOf (BRDY [One]) [0x06] = \BTL2 /* External reference *
/
                        DerefOf (BRDY [One]) [0x07] = \BTLL /* External reference *
/
                        Return (BRDY) /* \_SB_.PCI0.XHC_.RHUB.HS02.BRDY */
                    }

                    Name (ECKY, Package (0x02)
                    {
                        Zero,
                        Package (0x02)
                        {
                            0x12,
                            Zero
                        }
                    })
                    Method (ECKV, 0, Serialized)
                    {
                        DerefOf (ECKY [One]) [One] = \CECV /* External reference */
                        Return (ECKY) /* \_SB_.PCI0.XHC_.RHUB.HS02.ECKY */
                    }

                    Name (GPCX, Package (0x03)
                    {
                        Zero,
                        Package (0x02)
                        {
                            0x07,
                            Package (0x03)
                            {
                                Zero,
                                Zero,
                                Zero
                            }
                        },

                        Package (0x02)
                        {
                            0x12,
                            Package (0x03)
                            {
                                Zero,
                                Zero,
                                Zero
                            }
                        }
                    })
                    Method (GPC, 0, Serialized)
                    {
                        Return (GPCX) /* \_SB_.PCI0.XHC_.RHUB.HS02.GPCX */
                    }
```

```
                }
            }

            If ((CondRefOf (SXI1) && CondRefOf (SXP1)))
            {
                If (((SXI1 > Zero) && (SXP1 == 0x02)))
                {
                    Device (CIR)
                    {
                        Method (_ADR, 0, NotSerialized)  // _ADR: Address
                        {
                            Return ((0x02 + SXI1))
                        }
                    }
                }
            }

            If ((CondRefOf (SXI2) && CondRefOf (SXP2)))
            {
                If (((SXI2 > Zero) && (SXP2 == 0x02)))
                {
                    Device (CIR2)
                    {
                        Method (_ADR, 0, NotSerialized)  // _ADR: Address
                        {
                            Return ((0x02 + SXI2))
                        }
                    }
                }
            }
        }
    }

    If ((NHSP >= 0x03))
    {
        Scope (\_SB.PCI0.XHC.RHUB.HS03)
        {
            Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
            {
                If (_OSI ("Darwin"))
                {
                    Return (Package ()
                    {
                        0xFF,
                        0x09,
                        0x00,
                        0x00
                    })
                }
                If ((H3TC == Zero))
                {
```

```
                    Return (GUPC (H3CN))
                }
                Else
                {
                    Return (\_SB.UBTC.RUCC (H3CR, One))
                }
            }

            Method (_PLD, 0, NotSerialized)  // _PLD: Physical Location of Device
            {
                If ((H3TC == Zero))
                {
                    Return (GPLD (H3VS, 0x03))
                }
                Else
                {
                    Return (\_SB.UBTC.RUCC (H3CR, 0x02))
                }
            }

            If (CondRefOf (DBPN))
            {
                If (((DBPN == 0x03) && CNDP ()))
                {
                    Name (SADX, Package (0x03)
                    {
                        Zero,
                        Package (0x02)
                        {
                            0x07,
                            0x80000000
                        },

                        Package (0x02)
                        {
                            0x12,
                            0x80000000
                        }
                    })
                    Method (SADS, 0, Serialized)
                    {
                        DerefOf (SADX [One]) [One] = \ATDV /* External reference */
                        DerefOf (SADX [0x02]) [One] = \ATDV /* External reference *
/

                        Return (SADX) /* \_SB_.PCI0.XHC_.RHUB.HS03.SADX */
                    }

                    Name (BRDY, Package (0x02)
                    {
                        Zero,
                        Package (0x08)
```

```
                        {
                            0x12,
                            0x80,
                            0x80,
                            0x80,
                            0x80,
                            0x80,
                            0x80,
                            0x80
                        }
                    })
                    Method (BRDS, 0, Serialized)
                    {
                        DerefOf (BRDY [One]) [One] = \BTSE /* External reference */
                        DerefOf (BRDY [One]) [0x02] = \BTBR /* External reference *
/
                        DerefOf (BRDY [One]) [0x03] = \BED2 /* External reference *
/
                        DerefOf (BRDY [One]) [0x04] = \BED3 /* External reference *
/
                        DerefOf (BRDY [One]) [0x05] = \BTLE /* External reference *
/
                        DerefOf (BRDY [One]) [0x06] = \BTL2 /* External reference *
/
                        DerefOf (BRDY [One]) [0x07] = \BTLL /* External reference *
/
                        Return (BRDY) /* \_SB_.PCI0.XHC_.RHUB.HS03.BRDY */
                    }

                    Name (ECKY, Package (0x02)
                    {
                        Zero,
                        Package (0x02)
                        {
                            0x12,
                            Zero
                        }
                    })
                    Method (ECKV, 0, Serialized)
                    {
                        DerefOf (ECKY [One]) [One] = \CECV /* External reference */
                        Return (ECKY) /* \_SB_.PCI0.XHC_.RHUB.HS03.ECKY */
                    }

                    Name (GPCX, Package (0x03)
                    {
                        Zero,
                        Package (0x02)
                        {
                            0x07,
                            Package (0x03)
```

```
                        {
                            Zero,
                            Zero,
                            Zero
                        }
                    },

                    Package (0x02)
                    {
                        0x12,
                        Package (0x03)
                        {
                            Zero,
                            Zero,
                            Zero
                        }
                    }
                })
                Method (GPC, 0, Serialized)
                {
                    Return (GPCX) /* \_SB_.PCI0.XHC_.RHUB.HS03.GPCX */
                }
            }
        }

        If ((CondRefOf (SXI1) && CondRefOf (SXP1)))
        {
            If (((SXI1 > Zero) && (SXP1 == 0x03)))
            {
                Device (CIR)
                {
                    Method (_ADR, 0, NotSerialized)  // _ADR: Address
                    {
                        Return ((0x03 + SXI1))
                    }
                }
            }
        }

        If ((CondRefOf (SXI2) && CondRefOf (SXP2)))
        {
            If (((SXI2 > Zero) && (SXP2 == 0x03)))
            {
                Device (CIR2)
                {
                    Method (_ADR, 0, NotSerialized)  // _ADR: Address
                    {
                        Return ((0x03 + SXI2))
                    }
                }
            }
```

```
            }
        }
    }

    If ((NHSP >= 0x04))
    {
        Scope (\_SB.PCI0.XHC.RHUB.HS04)
        {
            Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
            {
                If (_OSI ("Darwin"))
                {
                    Return (Package ()
                    {
                        0xFF,
                        0x09,
                        0x00,
                        0x00
                    })
                }
                If ((H4TC == Zero))
                {
                    Return (GUPC (H4CN))
                }
                Else
                {
                    Return (\_SB.UBTC.RUCC (H4CR, One))
                }
            }

            Method (_PLD, 0, NotSerialized)  // _PLD: Physical Location of Device
            {
                If ((H4TC == Zero))
                {
                    Return (GPLD (H4VS, 0x04))
                }
                Else
                {
                    Return (\_SB.UBTC.RUCC (H4CR, 0x02))
                }
            }

            If (CondRefOf (DBPN))
            {
                If (((DBPN == 0x04) && CNDP ()))
                {
                    Name (SADX, Package (0x03)
                    {
                        Zero,
                        Package (0x02)
                        {
```

```
                            0x07,
                            0x80000000
                        },

                        Package (0x02)
                        {
                            0x12,
                            0x80000000
                        }
                    })
                    Method (SADS, 0, Serialized)
                    {
                        DerefOf (SADX [One]) [One] = \ATDV /* External reference */
                        DerefOf (SADX [0x02]) [One] = \ATDV /* External reference *
/
                        Return (SADX) /* \_SB_.PCI0.XHC_.RHUB.HS04.SADX */
                    }

                    Name (BRDY, Package (0x02)
                    {
                        Zero,
                        Package (0x08)
                        {
                            0x12,
                            0x80,
                            0x80,
                            0x80,
                            0x80,
                            0x80,
                            0x80,
                            0x80
                        }
                    })
                    Method (BRDS, 0, Serialized)
                    {
                        DerefOf (BRDY [One]) [One] = \BTSE /* External reference */
                        DerefOf (BRDY [One]) [0x02] = \BTBR /* External reference *
/
                        DerefOf (BRDY [One]) [0x03] = \BED2 /* External reference *
/
                        DerefOf (BRDY [One]) [0x04] = \BED3 /* External reference *
/
                        DerefOf (BRDY [One]) [0x05] = \BTLE /* External reference *
/
                        DerefOf (BRDY [One]) [0x06] = \BTL2 /* External reference *
/
                        DerefOf (BRDY [One]) [0x07] = \BTLL /* External reference *
/
                        Return (BRDY) /* \_SB_.PCI0.XHC_.RHUB.HS04.BRDY */
                    }
```

```
                    Name (ECKY, Package (0x02)
                    {
                        Zero,
                        Package (0x02)
                        {
                            0x12,
                            Zero
                        }
                    })
                    Method (ECKV, 0, Serialized)
                    {
                        DerefOf (ECKY [One]) [One] = \CECV /* External reference */
                        Return (ECKY) /* \_SB_.PCI0.XHC_.RHUB.HS04.ECKY */
                    }

                    Name (GPCX, Package (0x03)
                    {
                        Zero,
                        Package (0x02)
                        {
                            0x07,
                            Package (0x03)
                            {
                                Zero,
                                Zero,
                                Zero
                            }
                        },

                        Package (0x02)
                        {
                            0x12,
                            Package (0x03)
                            {
                                Zero,
                                Zero,
                                Zero
                            }
                        }
                    })
                    Method (GPC, 0, Serialized)
                    {
                        Return (GPCX) /* \_SB_.PCI0.XHC_.RHUB.HS04.GPCX */
                    }
                }
            }

            If ((CondRefOf (SXI1) && CondRefOf (SXP1)))
            {
                If (((SXI1 > Zero) && (SXP1 == 0x04)))
                {
```

```
                    Device (CIR)
                    {
                        Method (_ADR, 0, NotSerialized)  // _ADR: Address
                        {
                            Return ((0x04 + SXI1))
                        }
                    }
                }
            }

            If ((CondRefOf (SXI2) && CondRefOf (SXP2)))
            {
                If (((SXI2 > Zero) && (SXP2 == 0x04)))
                {
                    Device (CIR2)
                    {
                        Method (_ADR, 0, NotSerialized)  // _ADR: Address
                        {
                            Return ((0x04 + SXI2))
                        }
                    }
                }
            }
        }
    }

    If ((NHSP >= 0x05))
    {
        Scope (\_SB.PCI0.XHC.RHUB.HS05)
        {
            Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
            {
                If (_OSI ("Darwin"))
                {
                    Return (Package ()
                    {
                        0x00,
                        0x00,
                        0x00,
                        0x00
                    })
                }

                If ((H5TC == Zero))
                {
                    Return (GUPC (H5CN))
                }
                Else
                {
                    Return (\_SB.UBTC.RUCC (H5CR, One))
                }
```

```
            }

            Method (_PLD, 0, NotSerialized)  // _PLD: Physical Location of Device
            {
                If ((H5TC == Zero))
                {
                    Return (GPLD (H5VS, 0x05))
                }
                Else
                {
                    Return (\_SB.UBTC.RUCC (H5CR, 0x02))
                }
            }

            If (CondRefOf (DBPN))
            {
                If (((DBPN == 0x05) && CNDP ()))
                {
                    Name (SADX, Package (0x03)
                    {
                        Zero,
                        Package (0x02)
                        {
                            0x07,
                            0x80000000
                        },

                        Package (0x02)
                        {
                            0x12,
                            0x80000000
                        }
                    })
                    Method (SADS, 0, Serialized)
                    {
                        DerefOf (SADX [One]) [One] = \ATDV /* External reference */
                        DerefOf (SADX [0x02]) [One] = \ATDV /* External reference *
/
                        Return (SADX) /* \_SB_.PCI0.XHC_.RHUB.HS05.SADX */
                    }

                    Name (BRDY, Package (0x02)
                    {
                        Zero,
                        Package (0x08)
                        {
                            0x12,
                            0x80,
                            0x80,
                            0x80,
                            0x80,
```

```
                            0x80,
                            0x80,
                            0x80
                        }
                    })
                    Method (BRDS, 0, Serialized)
                    {
                        DerefOf (BRDY [One]) [One] = \BTSE /* External reference */
                        DerefOf (BRDY [One]) [0x02] = \BTBR /* External reference *
/
                        DerefOf (BRDY [One]) [0x03] = \BED2 /* External reference *
/
                        DerefOf (BRDY [One]) [0x04] = \BED3 /* External reference *
/
                        DerefOf (BRDY [One]) [0x05] = \BTLE /* External reference *
/
                        DerefOf (BRDY [One]) [0x06] = \BTL2 /* External reference *
/
                        DerefOf (BRDY [One]) [0x07] = \BTLL /* External reference *
/
                        Return (BRDY) /* \_SB_.PCI0.XHC_.RHUB.HS05.BRDY */
                    }

                    Name (ECKY, Package (0x02)
                    {
                        Zero,
                        Package (0x02)
                        {
                            0x12,
                            Zero
                        }
                    })
                    Method (ECKV, 0, Serialized)
                    {
                        DerefOf (ECKY [One]) [One] = \CECV /* External reference */
                        Return (ECKY) /* \_SB_.PCI0.XHC_.RHUB.HS05.ECKY */
                    }

                    Name (GPCX, Package (0x03)
                    {
                        Zero,
                        Package (0x02)
                        {
                            0x07,
                            Package (0x03)
                            {
                                Zero,
                                Zero,
                                Zero
                            }
                        },
```

```
                    Package (0x02)
                    {
                        0x12,
                        Package (0x03)
                        {
                            Zero,
                            Zero,
                            Zero
                        }
                    }
                })
                Method (GPC, 0, Serialized)
                {
                    Return (GPCX) /* \_SB_.PCI0.XHC_.RHUB.HS05.GPCX */
                }
            }
        }

        If ((CondRefOf (SXI1) && CondRefOf (SXP1)))
        {
            If (((SXI1 > Zero) && (SXP1 == 0x05)))
            {
                Device (CIR)
                {
                    Method (_ADR, 0, NotSerialized)  // _ADR: Address
                    {
                        Return ((0x05 + SXI1))
                    }
                }
            }
        }

        If ((CondRefOf (SXI2) && CondRefOf (SXP2)))
        {
            If (((SXI2 > Zero) && (SXP2 == 0x05)))
            {
                Device (CIR2)
                {
                    Method (_ADR, 0, NotSerialized)  // _ADR: Address
                    {
                        Return ((0x05 + SXI2))
                    }
                }
            }
        }
    }
}

If ((NHSP >= 0x06))
{
```

```
        Scope (\_SB.PCI0.XHC.RHUB.HS06)
        {
            Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
            {
                If (_OSI ("Darwin"))
                {
                    Return (Package ()
                    {
                        0x00,
                        0x00,
                        0x00,
                        0x00
                    })
                }

                If ((H6TC == Zero))
                {
                    Return (GUPC (H6CN))
                }
                Else
                {
                    Return (\_SB.UBTC.RUCC (H6CR, One))
                }
            }

            Method (_PLD, 0, NotSerialized)  // _PLD: Physical Location of Device
            {
                If ((H6TC == Zero))
                {
                    Return (GPLD (H6VS, 0x06))
                }
                Else
                {
                    Return (\_SB.UBTC.RUCC (H6CR, 0x02))
                }
            }

            If (CondRefOf (DBPN))
            {
                If (((DBPN == 0x06) && CNDP ()))
                {
                    Name (SADX, Package (0x03)
                    {
                        Zero,
                        Package (0x02)
                        {
                            0x07,
                            0x80000000
                        },

                        Package (0x02)
```

```
                        {
                            0x12,
                            0x80000000
                        }
                    })
                    Method (SADS, 0, Serialized)
                    {
                        DerefOf (SADX [One]) [One] = \ATDV /* External reference */
                        DerefOf (SADX [0x02]) [One] = \ATDV /* External reference *
/
                        Return (SADX) /* \_SB_.PCI0.XHC_.RHUB.HS06.SADX */
                    }

                    Name (BRDY, Package (0x02)
                    {
                        Zero,
                        Package (0x08)
                        {
                            0x12,
                            0x80,
                            0x80,
                            0x80,
                            0x80,
                            0x80,
                            0x80,
                            0x80
                        }
                    })
                    Method (BRDS, 0, Serialized)
                    {
                        DerefOf (BRDY [One]) [One] = \BTSE /* External reference */
                        DerefOf (BRDY [One]) [0x02] = \BTBR /* External reference *
/
                        DerefOf (BRDY [One]) [0x03] = \BED2 /* External reference *
/
                        DerefOf (BRDY [One]) [0x04] = \BED3 /* External reference *
/
                        DerefOf (BRDY [One]) [0x05] = \BTLE /* External reference *
/
                        DerefOf (BRDY [One]) [0x06] = \BTL2 /* External reference *
/
                        DerefOf (BRDY [One]) [0x07] = \BTLL /* External reference *
/
                        Return (BRDY) /* \_SB_.PCI0.XHC_.RHUB.HS06.BRDY */
                    }

                    Name (ECKY, Package (0x02)
                    {
                        Zero,
                        Package (0x02)
                        {
```

```
                        0x12,
                        Zero
                    }
                })
                Method (ECKV, 0, Serialized)
                {
                    DerefOf (ECKY [One]) [One] = \CECV /* External reference */
                    Return (ECKY) /* \_SB_.PCI0.XHC_.RHUB.HS06.ECKY */
                }

                Name (GPCX, Package (0x03)
                {
                    Zero,
                    Package (0x02)
                    {
                        0x07,
                        Package (0x03)
                        {
                            Zero,
                            Zero,
                            Zero
                        }
                    },

                    Package (0x02)
                    {
                        0x12,
                        Package (0x03)
                        {
                            Zero,
                            Zero,
                            Zero
                        }
                    }
                })
                Method (GPC, 0, Serialized)
                {
                    Return (GPCX) /* \_SB_.PCI0.XHC_.RHUB.HS06.GPCX */
                }
            }
        }

        If ((CondRefOf (SXI1) && CondRefOf (SXP1)))
        {
            If (((SXI1 > Zero) && (SXP1 == 0x06)))
            {
                Device (CIR)
                {
                    Method (_ADR, 0, NotSerialized)  // _ADR: Address
                    {
                        Return ((0x06 + SXI1))
```

```
                    }
                }
            }
        }

        If ((CondRefOf (SXI2) && CondRefOf (SXP2)))
        {
            If (((SXI2 > Zero) && (SXP2 == 0x06)))
            {
                Device (CIR2)
                {
                    Method (_ADR, 0, NotSerialized)  // _ADR: Address
                    {
                        Return ((0x06 + SXI2))
                    }
                }
            }
        }
    }
}

If ((NHSP >= 0x07))
{
    Scope (\_SB.PCI0.XHC.RHUB.HS07)
    {
        Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
        {
            If (_OSI ("Darwin"))
            {
                Return (Package ()
                {
                    0xFF,
                    0x03,
                    0x00,
                    0x00
                })
            }

            If ((H7TC == Zero))
            {
                Return (GUPC (H7CN))
            }
            Else
            {
                Return (\_SB.UBTC.RUCC (H7CR, One))
            }
        }

        Method (_PLD, 0, NotSerialized)  // _PLD: Physical Location of Device
        {
            If ((H7TC == Zero))
```

```
                {
                    Return (GPLD (H7VS, 0x07))
                }
                Else
                {
                    Return (\_SB.UBTC.RUCC (H7CR, 0x02))
                }
            }

            If (CondRefOf (DBPN))
            {
                If (((DBPN == 0x07) && CNDP ()))
                {
                    Name (SADX, Package (0x03)
                    {
                        Zero,
                        Package (0x02)
                        {
                            0x07,
                            0x80000000
                        },

                        Package (0x02)
                        {
                            0x12,
                            0x80000000
                        }
                    })
                    Method (SADS, 0, Serialized)
                    {
                        DerefOf (SADX [One]) [One] = \ATDV /* External reference */
                        DerefOf (SADX [0x02]) [One] = \ATDV /* External reference *
/
                        Return (SADX) /* \_SB_.PCI0.XHC_.RHUB.HS07.SADX */
                    }

                    Name (BRDY, Package (0x02)
                    {
                        Zero,
                        Package (0x08)
                        {
                            0x12,
                            0x80,
                            0x80,
                            0x80,
                            0x80,
                            0x80,
                            0x80,
                            0x80
                        }
                    })
```

```
                    Method (BRDS, 0, Serialized)
                    {
                        DerefOf (BRDY [One]) [One] = \BTSE /* External reference */
                        DerefOf (BRDY [One]) [0x02] = \BTBR /* External reference *
/
                        DerefOf (BRDY [One]) [0x03] = \BED2 /* External reference *
/
                        DerefOf (BRDY [One]) [0x04] = \BED3 /* External reference *
/
                        DerefOf (BRDY [One]) [0x05] = \BTLE /* External reference *
/
                        DerefOf (BRDY [One]) [0x06] = \BTL2 /* External reference *
/
                        DerefOf (BRDY [One]) [0x07] = \BTLL /* External reference *
/
                        Return (BRDY) /* \_SB_.PCI0.XHC_.RHUB.HS07.BRDY */
                    }

                    Name (ECKY, Package (0x02)
                    {
                        Zero,
                        Package (0x02)
                        {
                            0x12,
                            Zero
                        }
                    })
                    Method (ECKV, 0, Serialized)
                    {
                        DerefOf (ECKY [One]) [One] = \CECV /* External reference */
                        Return (ECKY) /* \_SB_.PCI0.XHC_.RHUB.HS07.ECKY */
                    }

                    Name (GPCX, Package (0x03)
                    {
                        Zero,
                        Package (0x02)
                        {
                            0x07,
                            Package (0x03)
                            {
                                Zero,
                                Zero,
                                Zero
                            }
                        },

                        Package (0x02)
                        {
                            0x12,
                            Package (0x03)
```

```
                            {
                                Zero,
                                Zero,
                                Zero
                            }
                        }
                    })
                    Method (GPC, 0, Serialized)
                    {
                        Return (GPCX) /* \_SB_.PCI0.XHC_.RHUB.HS07.GPCX */
                    }
                }
            }

            If ((CondRefOf (SXI1) && CondRefOf (SXP1)))
            {
                If (((SXI1 > Zero) && (SXP1 == 0x07)))
                {
                    Device (CIR)
                    {
                        Method (_ADR, 0, NotSerialized)  // _ADR: Address
                        {
                            Return ((0x07 + SXI1))
                        }
                    }
                }
            }

            If ((CondRefOf (SXI2) && CondRefOf (SXP2)))
            {
                If (((SXI2 > Zero) && (SXP2 == 0x07)))
                {
                    Device (CIR2)
                    {
                        Method (_ADR, 0, NotSerialized)  // _ADR: Address
                        {
                            Return ((0x07 + SXI2))
                        }
                    }
                }
            }
        }
    }

    If ((NHSP >= 0x08))
    {
        Scope (\_SB.PCI0.XHC.RHUB.HS08)
        {
            Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
            {
                If (_OSI ("Darwin"))
```

```
                {
                    Return (Package ()
                    {
                        0x00,
                        0x00,
                        0x00,
                        0x00
                    })
                }
                If ((H8TC == Zero))
                {
                    Return (GUPC (H8CN))
                }
                Else
                {
                    Return (\_SB.UBTC.RUCC (H8CR, One))
                }
            }

            Method (_PLD, 0, NotSerialized)  // _PLD: Physical Location of Device
            {
                If ((H8TC == Zero))
                {
                    Return (GPLD (H8VS, 0x08))
                }
                Else
                {
                    Return (\_SB.UBTC.RUCC (H8CR, 0x02))
                }
            }

            If (CondRefOf (DBPN))
            {
                If (((DBPN == 0x08) && CNDP ()))
                {
                    Name (SADX, Package (0x03)
                    {
                        Zero,
                        Package (0x02)
                        {
                            0x07,
                            0x80000000
                        },

                        Package (0x02)
                        {
                            0x12,
                            0x80000000
                        }
                    })
                    Method (SADS, 0, Serialized)
```

```
                    {
                        DerefOf (SADX [One]) [One] = \ATDV /* External reference */
                        DerefOf (SADX [0x02]) [One] = \ATDV /* External reference *
/
                        Return (SADX) /* \_SB_.PCI0.XHC_.RHUB.HS08.SADX */
                    }

                    Name (BRDY, Package (0x02)
                    {
                        Zero,
                        Package (0x08)
                        {
                            0x12,
                            0x80,
                            0x80,
                            0x80,
                            0x80,
                            0x80,
                            0x80,
                            0x80
                        }
                    })
                    Method (BRDS, 0, Serialized)
                    {
                        DerefOf (BRDY [One]) [One] = \BTSE /* External reference */
                        DerefOf (BRDY [One]) [0x02] = \BTBR /* External reference *
/
                        DerefOf (BRDY [One]) [0x03] = \BED2 /* External reference *
/
                        DerefOf (BRDY [One]) [0x04] = \BED3 /* External reference *
/
                        DerefOf (BRDY [One]) [0x05] = \BTLE /* External reference *
/
                        DerefOf (BRDY [One]) [0x06] = \BTL2 /* External reference *
/
                        DerefOf (BRDY [One]) [0x07] = \BTLL /* External reference *
/
                        Return (BRDY) /* \_SB_.PCI0.XHC_.RHUB.HS08.BRDY */
                    }

                    Name (ECKY, Package (0x02)
                    {
                        Zero,
                        Package (0x02)
                        {
                            0x12,
                            Zero
                        }
                    })
                    Method (ECKV, 0, Serialized)
                    {
```

```
                    DerefOf (ECKY [One]) [One] = \CECV /* External reference */
                    Return (ECKY) /* \_SB_.PCI0.XHC_.RHUB.HS08.ECKY */
                }

                Name (GPCX, Package (0x03)
                {
                    Zero,
                    Package (0x02)
                    {
                        0x07,
                        Package (0x03)
                        {
                            Zero,
                            Zero,
                            Zero
                        }
                    },

                    Package (0x02)
                    {
                        0x12,
                        Package (0x03)
                        {
                            Zero,
                            Zero,
                            Zero
                        }
                    }
                })
                Method (GPC, 0, Serialized)
                {
                    Return (GPCX) /* \_SB_.PCI0.XHC_.RHUB.HS08.GPCX */
                }
            }
        }

        If ((CondRefOf (SXI1) && CondRefOf (SXP1)))
        {
            If (((SXI1 > Zero) && (SXP1 == 0x08)))
            {
                Device (CIR)
                {
                    Method (_ADR, 0, NotSerialized)  // _ADR: Address
                    {
                        Return ((0x08 + SXI1))
                    }
                }
            }
        }

        If ((CondRefOf (SXI2) && CondRefOf (SXP2)))
```

```
            {
                If (((SXI2 > Zero) && (SXP2 == 0x08)))
                {
                    Device (CIR2)
                    {
                        Method (_ADR, 0, NotSerialized)  // _ADR: Address
                        {
                            Return ((0x08 + SXI2))
                        }
                    }
                }
            }
        }

        If ((NHSP >= 0x09))
        {
            Scope (\_SB.PCI0.XHC.RHUB.HS09)
            {
                Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
                {
                    If (_OSI ("Darwin"))
                    {
                        Return (Package ()
                        {
                            0x00,
                            0x00,
                            0x00,
                            0x00
                        })
                    }
                    If ((H9TC == Zero))
                    {
                        Return (GUPC (H9CN))
                    }
                    Else
                    {
                        Return (\_SB.UBTC.RUCC (H9CR, One))
                    }
                }

                Method (_PLD, 0, NotSerialized)  // _PLD: Physical Location of Device
                {
                    If ((H9TC == Zero))
                    {
                        Return (GPLD (H9VS, 0x09))
                    }
                    Else
                    {
                        Return (\_SB.UBTC.RUCC (H9CR, 0x02))
                    }
```

```
            }

            If (CondRefOf (DBPN))
            {
                If (((DBPN == 0x09) && CNDP ()))
                {
                    Name (SADX, Package (0x03)
                    {
                        Zero,
                        Package (0x02)
                        {
                            0x07,
                            0x80000000
                        },

                        Package (0x02)
                        {
                            0x12,
                            0x80000000
                        }
                    })
                    Method (SADS, 0, Serialized)
                    {
                        DerefOf (SADX [One]) [One] = \ATDV /* External reference */
                        DerefOf (SADX [0x02]) [One] = \ATDV /* External reference *
/
                        Return (SADX) /* \_SB_.PCI0.XHC_.RHUB.HS09.SADX */
                    }

                    Name (BRDY, Package (0x02)
                    {
                        Zero,
                        Package (0x08)
                        {
                            0x12,
                            0x80,
                            0x80,
                            0x80,
                            0x80,
                            0x80,
                            0x80,
                            0x80
                        }
                    })
                    Method (BRDS, 0, Serialized)
                    {
                        DerefOf (BRDY [One]) [One] = \BTSE /* External reference */
                        DerefOf (BRDY [One]) [0x02] = \BTBR /* External reference *
/
                        DerefOf (BRDY [One]) [0x03] = \BED2 /* External reference *
/
```

```
                        DerefOf (BRDY [One]) [0x04] = \BED3 /* External reference *
/
                        DerefOf (BRDY [One]) [0x05] = \BTLE /* External reference *
/
                        DerefOf (BRDY [One]) [0x06] = \BTL2 /* External reference *
/
                        DerefOf (BRDY [One]) [0x07] = \BTLL /* External reference *
/
                        Return (BRDY) /* \_SB_.PCI0.XHC_.RHUB.HS09.BRDY */
                    }

                    Name (ECKY, Package (0x02)
                    {
                        Zero,
                        Package (0x02)
                        {
                            0x12,
                            Zero
                        }
                    })
                    Method (ECKV, 0, Serialized)
                    {
                        DerefOf (ECKY [One]) [One] = \CECV /* External reference */
                        Return (ECKY) /* \_SB_.PCI0.XHC_.RHUB.HS09.ECKY */
                    }

                    Name (GPCX, Package (0x03)
                    {
                        Zero,
                        Package (0x02)
                        {
                            0x07,
                            Package (0x03)
                            {
                                Zero,
                                Zero,
                                Zero
                            }
                        },

                        Package (0x02)
                        {
                            0x12,
                            Package (0x03)
                            {
                                Zero,
                                Zero,
                                Zero
                            }
                        }
                    })
```

```
                    Method (GPC, 0, Serialized)
                    {
                        Return (GPCX) /* \_SB_.PCI0.XHC_.RHUB.HS09.GPCX */
                    }
                }
            }

            If ((CondRefOf (SXI1) && CondRefOf (SXP1)))
            {
                If (((SXI1 > Zero) && (SXP1 == 0x09)))
                {
                    Device (CIR)
                    {
                        Method (_ADR, 0, NotSerialized)  // _ADR: Address
                        {
                            Return ((0x09 + SXI1))
                        }
                    }
                }
            }

            If ((CondRefOf (SXI2) && CondRefOf (SXP2)))
            {
                If (((SXI2 > Zero) && (SXP2 == 0x09)))
                {
                    Device (CIR2)
                    {
                        Method (_ADR, 0, NotSerialized)  // _ADR: Address
                        {
                            Return ((0x09 + SXI2))
                        }
                    }
                }
            }
        }
    }

    If ((NHSP >= 0x0A))
    {
        Scope (\_SB.PCI0.XHC.RHUB.HS10)
        {
            Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
            {
                If (_OSI ("Darwin"))
                {
                    Return (Package ()
                    {
                        0xFF,
                        0xFF,
                        0x00,
                        0x00
```

```
                    })
                }
                If ((HATC == Zero))
                {
                    Return (GUPC (HACN))
                }
                Else
                {
                    Return (\_SB.UBTC.RUCC (HACR, One))
                }
            }

            Method (_PLD, 0, NotSerialized)  // _PLD: Physical Location of Device
            {
                If ((HATC == Zero))
                {
                    Return (GPLD (HAVS, 0x0A))
                }
                Else
                {
                    Return (\_SB.UBTC.RUCC (HACR, 0x02))
                }
            }

            If (CondRefOf (DBPN))
            {
                If (((DBPN == 0x0A) && CNDP ()))
                {
                    Name (SADX, Package (0x03)
                    {
                        Zero,
                        Package (0x02)
                        {
                            0x07,
                            0x80000000
                        },

                        Package (0x02)
                        {
                            0x12,
                            0x80000000
                        }
                    })
                    Method (SADS, 0, Serialized)
                    {
                        DerefOf (SADX [One]) [One] = \ATDV /* External reference */
                        DerefOf (SADX [0x02]) [One] = \ATDV /* External reference *
/
                        Return (SADX) /* \_SB_.PCI0.XHC_.RHUB.HS10.SADX */
                    }
```

```
                    Name (BRDY, Package (0x02)
                    {
                        Zero,
                        Package (0x08)
                        {
                            0x12,
                            0x80,
                            0x80,
                            0x80,
                            0x80,
                            0x80,
                            0x80,
                            0x80
                        }
                    })
                    Method (BRDS, 0, Serialized)
                    {
                        DerefOf (BRDY [One]) [One] = \BTSE /* External reference */
                        DerefOf (BRDY [One]) [0x02] = \BTBR /* External reference *
/
                        DerefOf (BRDY [One]) [0x03] = \BED2 /* External reference *
/
                        DerefOf (BRDY [One]) [0x04] = \BED3 /* External reference *
/
                        DerefOf (BRDY [One]) [0x05] = \BTLE /* External reference *
/
                        DerefOf (BRDY [One]) [0x06] = \BTL2 /* External reference *
/
                        DerefOf (BRDY [One]) [0x07] = \BTLL /* External reference *
/
                        Return (BRDY) /* \_SB_.PCI0.XHC_.RHUB.HS10.BRDY */
                    }

                    Name (ECKY, Package (0x02)
                    {
                        Zero,
                        Package (0x02)
                        {
                            0x12,
                            Zero
                        }
                    })
                    Method (ECKV, 0, Serialized)
                    {
                        DerefOf (ECKY [One]) [One] = \CECV /* External reference */
                        Return (ECKY) /* \_SB_.PCI0.XHC_.RHUB.HS10.ECKY */
                    }

                    Name (GPCX, Package (0x03)
                    {
                        Zero,
```

```
                    Package (0x02)
                    {
                        0x07,
                        Package (0x03)
                        {
                            Zero,
                            Zero,
                            Zero
                        }
                    },

                    Package (0x02)
                    {
                        0x12,
                        Package (0x03)
                        {
                            Zero,
                            Zero,
                            Zero
                        }
                    }
                })
                Method (GPC, 0, Serialized)
                {
                    Return (GPCX) /* \_SB_.PCI0.XHC_.RHUB.HS10.GPCX */
                }
            }
        }

        If ((CondRefOf (SXI1) && CondRefOf (SXP1)))
        {
            If (((SXI1 > Zero) && (SXP1 == 0x0A)))
            {
                Device (CIR)
                {
                    Method (_ADR, 0, NotSerialized)  // _ADR: Address
                    {
                        Return ((0x0A + SXI1))
                    }
                }
            }
        }

        If ((CondRefOf (SXI2) && CondRefOf (SXP2)))
        {
            If (((SXI2 > Zero) && (SXP2 == 0x0A)))
            {
                Device (CIR2)
                {
                    Method (_ADR, 0, NotSerialized)  // _ADR: Address
                    {
```

```
                            Return ((0x0A + SXI2))
                        }
                    }
                }
            }
        }

        If ((NHSP >= 0x0B))
        {
            Scope (\_SB.PCI0.XHC.RHUB.HS11)
            {
                Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
                {
                    If (_OSI ("Darwin"))
                    {
                        Return (Package ()
                        {
                            0x00,
                            0x00,
                            0x00,
                            0x00
                        })
                    }

                    If ((HBTC == Zero))
                    {
                        Return (GUPC (HBCN))
                    }
                    Else
                    {
                        Return (\_SB.UBTC.RUCC (HBCR, One))
                    }
                }

                Method (_PLD, 0, NotSerialized)  // _PLD: Physical Location of Device
                {
                    If ((HBTC == Zero))
                    {
                        Return (GPLD (HBVS, 0x0B))
                    }
                    Else
                    {
                        Return (\_SB.UBTC.RUCC (HBCR, 0x02))
                    }
                }

                If (CondRefOf (DBPN))
                {
                    If (((DBPN == 0x0B) && CNDP ()))
                    {
```

```
                    Name (SADX, Package (0x03)
                    {
                        Zero,
                        Package (0x02)
                        {
                            0x07,
                            0x80000000
                        },

                        Package (0x02)
                        {
                            0x12,
                            0x80000000
                        }
                    })
                    Method (SADS, 0, Serialized)
                    {
                        DerefOf (SADX [One]) [One] = \ATDV /* External reference */
                        DerefOf (SADX [0x02]) [One] = \ATDV /* External reference *
/
                        Return (SADX) /* \_SB_.PCI0.XHC_.RHUB.HS11.SADX */
                    }

                    Name (BRDY, Package (0x02)
                    {
                        Zero,
                        Package (0x08)
                        {
                            0x12,
                            0x80,
                            0x80,
                            0x80,
                            0x80,
                            0x80,
                            0x80,
                            0x80
                        }
                    })
                    Method (BRDS, 0, Serialized)
                    {
                        DerefOf (BRDY [One]) [One] = \BTSE /* External reference */
                        DerefOf (BRDY [One]) [0x02] = \BTBR /* External reference *
/
                        DerefOf (BRDY [One]) [0x03] = \BED2 /* External reference *
/
                        DerefOf (BRDY [One]) [0x04] = \BED3 /* External reference *
/
                        DerefOf (BRDY [One]) [0x05] = \BTLE /* External reference *
/
                        DerefOf (BRDY [One]) [0x06] = \BTL2 /* External reference *
/
```

```
                        DerefOf (BRDY [One]) [0x07] = \BTLL /* External reference *
/
                        Return (BRDY) /* \_SB_.PCI0.XHC_.RHUB.HS11.BRDY */
                    }

                    Name (ECKY, Package (0x02)
                    {
                        Zero,
                        Package (0x02)
                        {
                            0x12,
                            Zero
                        }
                    })
                    Method (ECKV, 0, Serialized)
                    {
                        DerefOf (ECKY [One]) [One] = \CECV /* External reference */
                        Return (ECKY) /* \_SB_.PCI0.XHC_.RHUB.HS11.ECKY */
                    }

                    Name (GPCX, Package (0x03)
                    {
                        Zero,
                        Package (0x02)
                        {
                            0x07,
                            Package (0x03)
                            {
                                Zero,
                                Zero,
                                Zero
                            }
                        },

                        Package (0x02)
                        {
                            0x12,
                            Package (0x03)
                            {
                                Zero,
                                Zero,
                                Zero
                            }
                        }
                    })
                    Method (GPC, 0, Serialized)
                    {
                        Return (GPCX) /* \_SB_.PCI0.XHC_.RHUB.HS11.GPCX */
                    }
                }
            }
```

```
            If ((CondRefOf (SXI1) && CondRefOf (SXP1)))
            {
                If (((SXI1 > Zero) && (SXP1 == 0x0B)))
                {
                    Device (CIR)
                    {
                        Method (_ADR, 0, NotSerialized)  // _ADR: Address
                        {
                            Return ((0x0B + SXI1))
                        }
                    }
                }
            }

            If ((CondRefOf (SXI2) && CondRefOf (SXP2)))
            {
                If (((SXI2 > Zero) && (SXP2 == 0x0B)))
                {
                    Device (CIR2)
                    {
                        Method (_ADR, 0, NotSerialized)  // _ADR: Address
                        {
                            Return ((0x0B + SXI2))
                        }
                    }
                }
            }
        }
    }

    If ((NHSP >= 0x0C))
    {
        Scope (\_SB.PCI0.XHC.RHUB.HS12)
        {
            Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
            {
                If (_OSI ("Darwin"))
                {
                    Return (Package ()
                    {
                        0x00,
                        0x00,
                        0x00,
                        0x00
                    })
                }

                If ((HCTC == Zero))
                {
                    Return (GUPC (HCCN))
```

```
                }
                Else
                {
                    Return (\_SB.UBTC.RUCC (HCCR, One))
                }
            }

            Method (_PLD, 0, NotSerialized)  // _PLD: Physical Location of Device
            {
                If ((HCTC == Zero))
                {
                    Return (GPLD (HCVS, 0x0C))
                }
                Else
                {
                    Return (\_SB.UBTC.RUCC (HCCR, 0x02))
                }
            }

            If (CondRefOf (DBPN))
            {
                If (((DBPN == 0x0C) && CNDP ()))
                {
                    Name (SADX, Package (0x03)
                    {
                        Zero,
                        Package (0x02)
                        {
                            0x07,
                            0x80000000
                        },

                        Package (0x02)
                        {
                            0x12,
                            0x80000000
                        }
                    })
                    Method (SADS, 0, Serialized)
                    {
                        DerefOf (SADX [One]) [One] = \ATDV /* External reference */
                        DerefOf (SADX [0x02]) [One] = \ATDV /* External reference *
/
                        Return (SADX) /* \_SB_.PCI0.XHC_.RHUB.HS12.SADX */
                    }

                    Name (BRDY, Package (0x02)
                    {
                        Zero,
                        Package (0x08)
                        {
```

```
                            0x12,
                            0x80,
                            0x80,
                            0x80,
                            0x80,
                            0x80,
                            0x80,
                            0x80
                        }
                    })
                    Method (BRDS, 0, Serialized)
                    {
                        DerefOf (BRDY [One]) [One] = \BTSE /* External reference */
                        DerefOf (BRDY [One]) [0x02] = \BTBR /* External reference *
/
                        DerefOf (BRDY [One]) [0x03] = \BED2 /* External reference *
/
                        DerefOf (BRDY [One]) [0x04] = \BED3 /* External reference *
/
                        DerefOf (BRDY [One]) [0x05] = \BTLE /* External reference *
/
                        DerefOf (BRDY [One]) [0x06] = \BTL2 /* External reference *
/
                        DerefOf (BRDY [One]) [0x07] = \BTLL /* External reference *
/
                        Return (BRDY) /* \_SB_.PCI0.XHC_.RHUB.HS12.BRDY */
                    }

                    Name (ECKY, Package (0x02)
                    {
                        Zero,
                        Package (0x02)
                        {
                            0x12,
                            Zero
                        }
                    })
                    Method (ECKV, 0, Serialized)
                    {
                        DerefOf (ECKY [One]) [One] = \CECV /* External reference */
                        Return (ECKY) /* \_SB_.PCI0.XHC_.RHUB.HS12.ECKY */
                    }

                    Name (GPCX, Package (0x03)
                    {
                        Zero,
                        Package (0x02)
                        {
                            0x07,
                            Package (0x03)
                            {
```

```
                            Zero,
                            Zero,
                            Zero
                        }
                    },

                    Package (0x02)
                    {
                        0x12,
                        Package (0x03)
                        {
                            Zero,
                            Zero,
                            Zero
                        }
                    }
                })
                Method (GPC, 0, Serialized)
                {
                    Return (GPCX) /* \_SB_.PCI0.XHC_.RHUB.HS12.GPCX */
                }
            }
        }

        If ((CondRefOf (SXI1) && CondRefOf (SXP1)))
        {
            If (((SXI1 > Zero) && (SXP1 == 0x0C)))
            {
                Device (CIR)
                {
                    Method (_ADR, 0, NotSerialized)  // _ADR: Address
                    {
                        Return ((0x0C + SXI1))
                    }
                }
            }
        }

        If ((CondRefOf (SXI2) && CondRefOf (SXP2)))
        {
            If (((SXI2 > Zero) && (SXP2 == 0x0C)))
            {
                Device (CIR2)
                {
                    Method (_ADR, 0, NotSerialized)  // _ADR: Address
                    {
                        Return ((0x0C + SXI2))
                    }
                }
            }
        }
```

```
        }
    }

    If ((NHSP >= 0x0D))
    {
        Scope (\_SB.PCI0.XHC.RHUB.HS13)
        {
            Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
            {
                If (_OSI ("Darwin"))
                {
                    Return (Package ()
                    {
                        0x00,
                        0x00,
                        0x00,
                        0x00
                    })
                }

                If ((HDTC == Zero))
                {
                    Return (GUPC (HDCN))
                }
                Else
                {
                    Return (\_SB.UBTC.RUCC (HDCR, One))
                }
            }

            Method (_PLD, 0, NotSerialized)  // _PLD: Physical Location of Device
            {
                If ((HDTC == Zero))
                {
                    Return (GPLD (HDVS, 0x0D))
                }
                Else
                {
                    Return (\_SB.UBTC.RUCC (HDCR, 0x02))
                }
            }

            If (CondRefOf (DBPN))
            {
                If (((DBPN == 0x0D) && CNDP ()))
                {
                    Name (SADX, Package (0x03)
                    {
                        Zero,
                        Package (0x02)
                        {
```

```
                            0x07,
                            0x80000000
                        },

                        Package (0x02)
                        {
                            0x12,
                            0x80000000
                        }
                    })
                    Method (SADS, 0, Serialized)
                    {
                        DerefOf (SADX [One]) [One] = \ATDV /* External reference */
                        DerefOf (SADX [0x02]) [One] = \ATDV /* External reference *
/
                        Return (SADX) /* \_SB_.PCI0.XHC_.RHUB.HS13.SADX */
                    }

                    Name (BRDY, Package (0x02)
                    {
                        Zero,
                        Package (0x08)
                        {
                            0x12,
                            0x80,
                            0x80,
                            0x80,
                            0x80,
                            0x80,
                            0x80
                        }
                    })
                    Method (BRDS, 0, Serialized)
                    {
                        DerefOf (BRDY [One]) [One] = \BTSE /* External reference */
                        DerefOf (BRDY [One]) [0x02] = \BTBR /* External reference *
/
                        DerefOf (BRDY [One]) [0x03] = \BED2 /* External reference *
/
                        DerefOf (BRDY [One]) [0x04] = \BED3 /* External reference *
/
                        DerefOf (BRDY [One]) [0x05] = \BTLE /* External reference *
/
                        DerefOf (BRDY [One]) [0x06] = \BTL2 /* External reference *
/
                        DerefOf (BRDY [One]) [0x07] = \BTLL /* External reference *
/
                        Return (BRDY) /* \_SB_.PCI0.XHC_.RHUB.HS13.BRDY */
                    }
```

```
                    Name (ECKY, Package (0x02)
                    {
                        Zero,
                        Package (0x02)
                        {
                            0x12,
                            Zero
                        }
                    })
                    Method (ECKV, 0, Serialized)
                    {
                        DerefOf (ECKY [One]) [One] = \CECV /* External reference */
                        Return (ECKY) /* \_SB_.PCI0.XHC_.RHUB.HS13.ECKY */
                    }

                    Name (GPCX, Package (0x03)
                    {
                        Zero,
                        Package (0x02)
                        {
                            0x07,
                            Package (0x03)
                            {
                                Zero,
                                Zero,
                                Zero
                            }
                        },

                        Package (0x02)
                        {
                            0x12,
                            Package (0x03)
                            {
                                Zero,
                                Zero,
                                Zero
                            }
                        }
                    })
                    Method (GPC, 0, Serialized)
                    {
                        Return (GPCX) /* \_SB_.PCI0.XHC_.RHUB.HS13.GPCX */
                    }
                }
            }

            If ((CondRefOf (SXI1) && CondRefOf (SXP1)))
            {
                If (((SXI1 > Zero) && (SXP1 == 0x0D)))
                {
```

```
                    Device (CIR)
                    {
                        Method (_ADR, 0, NotSerialized)  // _ADR: Address
                        {
                            Return ((0x0D + SXI1))
                        }
                    }
                }
            }

            If ((CondRefOf (SXI2) && CondRefOf (SXP2)))
            {
                If (((SXI2 > Zero) && (SXP2 == 0x0D)))
                {
                    Device (CIR2)
                    {
                        Method (_ADR, 0, NotSerialized)  // _ADR: Address
                        {
                            Return ((0x0D + SXI2))
                        }
                    }
                }
            }
        }
    }

    If ((NHSP >= 0x0E))
    {
        Scope (\_SB.PCI0.XHC.RHUB.HS14)
        {
            Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
            {
                If (_OSI ("Darwin"))
                {
                    Return (Package ()
                    {
                        0x00,
                        0x00,
                        0x00,
                        0x00
                    })
                }
                If ((HETC == Zero))
                {
                    Return (GUPC (HECN))
                }
                Else
                {
                    Return (\_SB.UBTC.RUCC (HECR, One))
                }
            }
```

```
            Method (_PLD, 0, NotSerialized)  // _PLD: Physical Location of Device
            {
                If ((HETC == Zero))
                {
                    Return (GPLD (HEVS, 0x0E))
                }
                Else
                {
                    Return (\_SB.UBTC.RUCC (HECR, 0x02))
                }
            }

            If (CondRefOf (DBPN))
            {
                If (((DBPN == 0x0E) && CNDP ()))
                {
                    Name (SADX, Package (0x03)
                    {
                        Zero,
                        Package (0x02)
                        {
                            0x07,
                            0x80000000
                        },

                        Package (0x02)
                        {
                            0x12,
                            0x80000000
                        }
                    })
                    Method (SADS, 0, Serialized)
                    {
                        DerefOf (SADX [One]) [One] = \ATDV /* External reference */
                        DerefOf (SADX [0x02]) [One] = \ATDV /* External reference *
/

                        Return (SADX) /* \_SB_.PCI0.XHC_.RHUB.HS14.SADX */
                    }

                    Name (BRDY, Package (0x02)
                    {
                        Zero,
                        Package (0x08)
                        {
                            0x12,
                            0x80,
                            0x80,
                            0x80,
                            0x80,
                            0x80,
```

```
                            0x80,
                            0x80
                        }
                    })
                    Method (BRDS, 0, Serialized)
                    {
                        DerefOf (BRDY [One]) [One] = \BTSE /* External reference */
                        DerefOf (BRDY [One]) [0x02] = \BTBR /* External reference *
/
                        DerefOf (BRDY [One]) [0x03] = \BED2 /* External reference *
/
                        DerefOf (BRDY [One]) [0x04] = \BED3 /* External reference *
/
                        DerefOf (BRDY [One]) [0x05] = \BTLE /* External reference *
/
                        DerefOf (BRDY [One]) [0x06] = \BTL2 /* External reference *
/
                        DerefOf (BRDY [One]) [0x07] = \BTLL /* External reference *
/
                        Return (BRDY) /* \_SB_.PCI0.XHC_.RHUB.HS14.BRDY */
                    }

                    Name (ECKY, Package (0x02)
                    {
                        Zero,
                        Package (0x02)
                        {
                            0x12,
                            Zero
                        }
                    })
                    Method (ECKV, 0, Serialized)
                    {
                        DerefOf (ECKY [One]) [One] = \CECV /* External reference */
                        Return (ECKY) /* \_SB_.PCI0.XHC_.RHUB.HS14.ECKY */
                    }

                    Name (GPCX, Package (0x03)
                    {
                        Zero,
                        Package (0x02)
                        {
                            0x07,
                            Package (0x03)
                            {
                                Zero,
                                Zero,
                                Zero
                            }
                        },
```

```
                        Package (0x02)
                        {
                            0x12,
                            Package (0x03)
                            {
                                Zero,
                                Zero,
                                Zero
                            }
                        }
                    })
                    Method (GPC, 0, Serialized)
                    {
                        Return (GPCX) /* \_SB_.PCI0.XHC_.RHUB.HS14.GPCX */
                    }
                }
            }

            If ((CondRefOf (SXI1) && CondRefOf (SXP1)))
            {
                If (((SXI1 > Zero) && (SXP1 == 0x0E)))
                {
                    Device (CIR)
                    {
                        Method (_ADR, 0, NotSerialized)  // _ADR: Address
                        {
                            Return ((0x0E + SXI1))
                        }
                    }
                }
            }

            If ((CondRefOf (SXI2) && CondRefOf (SXP2)))
            {
                If (((SXI2 > Zero) && (SXP2 == 0x0E)))
                {
                    Device (CIR2)
                    {
                        Method (_ADR, 0, NotSerialized)  // _ADR: Address
                        {
                            Return ((0x0E + SXI2))
                        }
                    }
                }
            }
        }
    }

    Scope (\_SB.PCI0.XHC.RHUB.USR1)
    {
        Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
```

```
        {
                If (_OSI ("Darwin"))
                {
                    Return (Package ()
                    {
                        0x00,
                        0x00,
                        0x00,
                        0x00
                    })
                }

            Return (GUPC (Zero))
        }

        Method (_PLD, 0, NotSerialized)  // _PLD: Physical Location of Device
        {
            Return (GPLD (Zero, Zero))
        }
    }

    Scope (\_SB.PCI0.XHC.RHUB.USR2)
    {
        Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
        {
                If (_OSI ("Darwin"))
                {
                    Return (Package ()
                    {
                        0x00,
                        0x00,
                        0x00,
                        0x00
                    })
                }
            Return (GUPC (Zero))
        }

        Method (_PLD, 0, NotSerialized)  // _PLD: Physical Location of Device
        {
            Return (GPLD (Zero, Zero))
        }
    }

    If ((NSSP >= One))
    {
        Scope (\_SB.PCI0.XHC.RHUB.SS01)
        {
            Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
            {
                If (_OSI ("Darwin"))
```

```
                {
                    Return (Package ()
                    {
                        0xFF,
                        0x03,
                        0x00,
                        0x00
                    })
                }

                If ((S1TC == Zero))
                {
                    Return (GUPC (S1CN))
                }
                Else
                {
                    Return (\_SB.UBTC.RUCC (S1CR, One))
                }
            }

            Method (_PLD, 0, NotSerialized)  // _PLD: Physical Location of Device
            {
                If ((S1TC == Zero))
                {
                    Return (GPLD (S1VS, S1CP))
                }
                Else
                {
                    Return (\_SB.UBTC.RUCC (S1CR, 0x02))
                }
            }
        }
    }

    If ((NSSP >= 0x02))
    {
        Scope (\_SB.PCI0.XHC.RHUB.SS02)
        {
            Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
            {
                If (_OSI ("Darwin"))
                {
                    Return (Package ()
                    {
                        0xFF,
                        0x09,
                        0x00,
                        0x00
                    })
                }
                If ((S2TC == Zero))
```

```
                {
                    Return (GUPC (S2CN))
                }
                Else
                {
                    Return (\_SB.UBTC.RUCC (S2CR, One))
                }
            }

            Method (_PLD, 0, NotSerialized)  // _PLD: Physical Location of Device
            {
                If ((S2TC == Zero))
                {
                    Return (GPLD (S2VS, S2CP))
                }
                Else
                {
                    Return (\_SB.UBTC.RUCC (S2CR, 0x02))
                }
            }
        }
    }

    If ((NSSP >= 0x03))
    {
        Scope (\_SB.PCI0.XHC.RHUB.SS03)
        {
            Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
            {
                If (_OSI ("Darwin"))
                {
                    Return (Package ()
                    {
                        0x00,
                        0x00,
                        0x00,
                        0x00
                    })
                }

                If ((S3TC == Zero))
                {
                    Return (GUPC (S3CN))
                }
                Else
                {
                    Return (\_SB.UBTC.RUCC (S3CR, One))
                }
            }

            Method (_PLD, 0, NotSerialized)  // _PLD: Physical Location of Device
```

```
            {
                If ((S3TC == Zero))
                {
                    Return (GPLD (S3VS, S3CP))
                }
                Else
                {
                    Return (\_SB.UBTC.RUCC (S3CR, 0x02))
                }
            }
        }
    }

    If ((NSSP >= 0x04))
    {
        Scope (\_SB.PCI0.XHC.RHUB.SS04)
        {
            Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
            {
                If (_OSI ("Darwin"))
                {
                    Return (Package ()
                    {
                        0xFF,
                        0x09,
                        0x00,
                        0x00
                    })
                }
                If ((S4TC == Zero))
                {
                    Return (GUPC (S4CN))
                }
                Else
                {
                    Return (\_SB.UBTC.RUCC (S4CR, One))
                }
            }

            Method (_PLD, 0, NotSerialized)  // _PLD: Physical Location of Device
            {
                If ((S4TC == Zero))
                {
                    Return (GPLD (S4VS, S4CP))
                }
                Else
                {
                    Return (\_SB.UBTC.RUCC (S4CR, 0x02))
                }
            }
        }
```

```
    }

    If ((NSSP >= 0x05))
    {
        Scope (\_SB.PCI0.XHC.RHUB.SS05)
        {
            Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
            {
                If (_OSI ("Darwin"))
                {
                    Return (Package ()
                    {
                        0x00,
                        0x00,
                        0x00,
                        0x00
                    })
                }

                If ((S5TC == Zero))
                {
                    Return (GUPC (S5CN))
                }
                Else
                {
                    Return (\_SB.UBTC.RUCC (S5CR, One))
                }
            }

            Method (_PLD, 0, NotSerialized)  // _PLD: Physical Location of Device
            {
                If ((S5TC == Zero))
                {
                    Return (GPLD (S5VS, S5CP))
                }
                Else
                {
                    Return (\_SB.UBTC.RUCC (S5CR, 0x02))
                }
            }
        }
    }

    If ((NSSP >= 0x06))
    {
        Scope (\_SB.PCI0.XHC.RHUB.SS06)
        {
            Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
            {
                If (_OSI ("Darwin"))
                {
```

```
                    Return (Package ()
                    {
                        0x00,
                        0x00,
                        0x00,
                        0x00
                    })
                }

                If ((S6TC == Zero))
                {
                    Return (GUPC (S6CN))
                }
                Else
                {
                    Return (\_SB.UBTC.RUCC (S6CR, One))
                }
            }

            Method (_PLD, 0, NotSerialized)  // _PLD: Physical Location of Device
            {
                If ((S6TC == Zero))
                {
                    Return (GPLD (S6VS, S6CP))
                }
                Else
                {
                    Return (\_SB.UBTC.RUCC (S6CR, 0x02))
                }
            }
        }
    }

    If ((NSSP >= 0x07))
    {
        Scope (\_SB.PCI0.XHC.RHUB.SS07)
        {
            Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
            {
                If (_OSI ("Darwin"))
                {
                    Return (Package ()
                    {
                        0x00,
                        0x00,
                        0x00,
                        0x00
                    })
                }

                If ((S7TC == Zero))
```

```
                {
                    Return (GUPC (S7CN))
                }
                Else
                {
                    Return (\_SB.UBTC.RUCC (S7CR, One))
                }
            }

            Method (_PLD, 0, NotSerialized)  // _PLD: Physical Location of Device
            {
                If ((S7TC == Zero))
                {
                    Return (GPLD (S7VS, S7CP))
                }
                Else
                {
                    Return (\_SB.UBTC.RUCC (S7CR, 0x02))
                }
            }
        }
    }

    If ((NSSP >= 0x08))
    {
        Scope (\_SB.PCI0.XHC.RHUB.SS08)
        {
            Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
            {
                If (_OSI ("Darwin"))
                {
                    Return (Package ()
                    {
                        0x00,
                        0x00,
                        0x00,
                        0x00
                    })
                }

                If ((S8TC == Zero))
                {
                    Return (GUPC (S8CN))
                }
                Else
                {
                    Return (\_SB.UBTC.RUCC (S8CR, One))
                }
            }

            Method (_PLD, 0, NotSerialized)  // _PLD: Physical Location of Device
```

```
            {
                If ((S8TC == Zero))
                {
                    Return (GPLD (S8VS, S8CP))
                }
                Else
                {
                    Return (\_SB.UBTC.RUCC (S8CR, 0x02))
                }
            }
        }
    }

    If ((NSSP >= 0x09))
    {
        Scope (\_SB.PCI0.XHC.RHUB.SS09)
        {
            Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
            {
                If (_OSI ("Darwin"))
                {
                    Return (Package ()
                    {
                        0x00,
                        0x00,
                        0x00,
                        0x00
                    })
                }

                If ((S9TC == Zero))
                {
                    Return (GUPC (S9CN))
                }
                Else
                {
                    Return (\_SB.UBTC.RUCC (S9CR, One))
                }
            }

            Method (_PLD, 0, NotSerialized)  // _PLD: Physical Location of Device
            {
                If ((S9TC == Zero))
                {
                    Return (GPLD (S9VS, S9CP))
                }
                Else
                {
                    Return (\_SB.UBTC.RUCC (S9CR, 0x02))
                }
            }
```

```
        }
    }

    If ((NSSP >= 0x0A))
    {
        Scope (\_SB.PCI0.XHC.RHUB.SS10)
        {
            Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
            {
                If (_OSI ("Darwin"))
                {
                    Return (Package ()
                    {
                        0x00,
                        0x00,
                        0x00,
                        0x00
                    })
                }

                If ((SATC == Zero))
                {
                    Return (GUPC (SACN))
                }
                Else
                {
                    Return (\_SB.UBTC.RUCC (SACR, One))
                }
            }

            Method (_PLD, 0, NotSerialized)  // _PLD: Physical Location of Device
            {
                If ((SATC == Zero))
                {
                    Return (GPLD (SAVS, SACP))
                }
                Else
                {
                    Return (\_SB.UBTC.RUCC (SACR, 0x02))
                }
            }
        }
    }

    Scope (\_SB.PCI0.XDCI)
    {
        Method (_PLD, 0, NotSerialized)  // _PLD: Physical Location of Device
        {
            Return (\_SB.PCI0.XHC.RHUB.TPLD (One, 0x0E))
        }
```

```
        Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
        {
            Return (\_SB.PCI0.XHC.RHUB.TUPC (0x09))
        }
    }
}
```

```
//
DefinitionBlock ("", "SSDT", 2, "OCLT", "MY_OEMBD", 0x00000000)
{
    External (_SB_.CBID, IntObj)
    External (_SB_.PCI0.XHC_.RHUB, DeviceObj)
    External (_SB_.PCI0.XHC_.RHUB.HS01, DeviceObj)
    External (_SB_.PCI0.XHC_.RHUB.HS02, DeviceObj)
    External (_SB_.PCI0.XHC_.RHUB.HS03, DeviceObj)
    External (_SB_.PCI0.XHC_.RHUB.HS04, DeviceObj)
    External (_SB_.PCI0.XHC_.RHUB.HS05, DeviceObj)
    External (_SB_.PCI0.XHC_.RHUB.HS06, DeviceObj)
    External (_SB_.PCI0.XHC_.RHUB.HS07, DeviceObj)
    External (_SB_.PCI0.XHC_.RHUB.HS08, DeviceObj)
    External (_SB_.PCI0.XHC_.RHUB.HS09, DeviceObj)
    External (_SB_.PCI0.XHC_.RHUB.HS10, DeviceObj)
    External (_SB_.PCI0.XHC_.RHUB.SS01, DeviceObj)
    External (_SB_.PCI0.XHC_.RHUB.SS02, DeviceObj)
    External (_SB_.PCI0.XHC_.RHUB.SS03, DeviceObj)
    External (_SB_.PCI0.XHC_.RHUB.SS04, DeviceObj)
    External (_SB_.PCI0.XHC_.RHUB.SS05, DeviceObj)
    External (_SB_.PCI0.XHC_.RHUB.SS06, DeviceObj)
    External (_SB_.PCI0.XHC_.RHUB.USR1, DeviceObj)
    External (_SB_.PCI0.XHC_.RHUB.USR2, DeviceObj)
    //External (CBID, UnknownObj)

    Scope (\_SB.PCI0.XHC.RHUB)
    {
        Method (GPLD, 2, Serialized)
        {
            Name (PCKG, Package (0x01)
            {
                Buffer (0x10){}
            })
            CreateField (DerefOf (PCKG [Zero]), Zero, 0x07, REV)
            REV = One
            CreateField (DerefOf (PCKG [Zero]), 0x40, One, VISI)
            VISI = Arg0
            CreateField (DerefOf (PCKG [Zero]), 0x57, 0x08, GPOS)
            GPOS = Arg1
            Return (PCKG) /* \_SB_.PCI0.XHC_.RHUB.GPLD.PCKG */
        }

        Method (GUPC, 1, Serialized)
        {
            Name (PCKG, Package (0x04)
            {
                Zero,
                0xFF,
                Zero,
                Zero
```

```
            })
            PCKG [Zero] = Arg0
            Return (PCKG) /* \_SB_.PCI0.XHC_.RHUB.GUPC.PCKG */
        }

        Method (TPLD, 2, Serialized)
        {
            Name (PCKG, Package (0x01)
            {
                Buffer (0x10){}
            })
            CreateField (DerefOf (PCKG [Zero]), Zero, 0x07, REV)
            REV = One
            CreateField (DerefOf (PCKG [Zero]), 0x40, One, VISI)
            VISI = Arg0
            CreateField (DerefOf (PCKG [Zero]), 0x57, 0x08, GPOS)
            GPOS = Arg1
            CreateField (DerefOf (PCKG [Zero]), 0x4A, 0x04, SHAP)
            SHAP = One
            CreateField (DerefOf (PCKG [Zero]), 0x20, 0x10, WID)
            WID = 0x08
            CreateField (DerefOf (PCKG [Zero]), 0x30, 0x10, HGT)
            HGT = 0x03
            Return (PCKG) /* \_SB_.PCI0.XHC_.RHUB.TPLD.PCKG */
        }

        Method (TUPC, 1, Serialized)
        {
            Name (PCKG, Package (0x04)
            {
                One,
                Zero,
                Zero,
                Zero
            })
            PCKG [One] = Arg0
            Return (PCKG) /* \_SB_.PCI0.XHC_.RHUB.TUPC.PCKG */
        }
    }

    Scope (\_SB.PCI0.XHC.RHUB.HS01)
    {
        Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
        {
            If (_OSI ("Darwin"))
            {
                Return (Package ()
                {
                    0xFF,
                    0x03,
                    0x00,
```

```
                    0x00
                })
            }

            Return (GUPC (One))
        }

        Method (_PLD, 0, NotSerialized)  // _PLD: Physical Location of Device
        {
            Return (GPLD (One, One))
        }
    }

    Scope (\_SB.PCI0.XHC.RHUB.HS02)
    {
        Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
        {
            If (_OSI ("Darwin"))
            {
                Return (Package ()
                {
                    0xFF,
                    0x03,
                    0x00,
                    0x00
                })
            }

            Return (GUPC (One))
        }

        Method (_PLD, 0, NotSerialized)  // _PLD: Physical Location of Device
        {
            Return (GPLD (One, 0x02))
        }
    }

    Scope (\_SB.PCI0.XHC.RHUB.HS03)
    {
        Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
        {
            If (_OSI ("Darwin"))
            {
                Return (Package ()
                {
                    0xFF,
                    0x03,
                    0x00,
                    0x00
                })
            }
```

```
            If ((CBID == 0x07A6))
            {
                Return (GUPC (Zero))
            }
            Else
            {
                Return (GUPC (One))
            }
        }

        Method (_PLD, 0, NotSerialized)  // _PLD: Physical Location of Device
        {
            If ((CBID == 0x07A6))
            {
                Return (GPLD (Zero, Zero))
            }
            Else
            {
                Return (GPLD (One, 0x03))
            }
        }
    }

    Scope (\_SB.PCI0.XHC.RHUB.HS04)
    {
        Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
        {
            If (_OSI ("Darwin"))
            {
                Return (Package ()
                {
                    0x00,
                    0x00,
                    0x00,
                    0x00
                })
            }
            Return (GUPC (One))
        }

        Method (_PLD, 0, NotSerialized)  // _PLD: Physical Location of Device
        {
            Return (GPLD (Zero, 0x04))
        }
    }

    Scope (\_SB.PCI0.XHC.RHUB.HS05)
    {
        Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
        {
```

```
            If (_OSI ("Darwin"))
            {
                Return (Package ()
                {
                    0xFF,
                    0xFF,
                    0x00,
                    0x00
                })
            }

            Return (GUPC (One))
        }

        Method (_PLD, 0, NotSerialized)  // _PLD: Physical Location of Device
        {
            Return (GPLD (Zero, 0x05))
        }

        Device (CAM5)
        {
            Name (_ADR, 0x05)  // _ADR: Address
            Name (_PLD, Package (0x01)  // _PLD: Physical Location of Device
            {
                ToPLD (
                    PLD_Revision           = 0x1,
                    PLD_IgnoreColor        = 0x1,
                    PLD_Red                = 0x0,
                    PLD_Green              = 0x0,
                    PLD_Blue               = 0x0,
                    PLD_Width              = 0x0,
                    PLD_Height             = 0x0,
                    PLD_UserVisible        = 0x1,
                    PLD_Dock               = 0x0,
                    PLD_Lid                = 0x1,
                    PLD_Panel              = "FRONT",
                    PLD_VerticalPosition   = "UPPER",
                    PLD_HorizontalPosition = "CENTER",
                    PLD_Shape              = "UNKNOWN",
                    PLD_GroupOrientation   = 0x0,
                    PLD_GroupToken         = 0x0,
                    PLD_GroupPosition      = 0x0,
                    PLD_Bay                = 0x0,
                    PLD_Ejectable          = 0x0,
                    PLD_EjectRequired      = 0x0,
                    PLD_CabinetNumber      = 0x0,
                    PLD_CardCageNumber     = 0x0,
                    PLD_Reference          = 0x0,
                    PLD_Rotation           = 0x0,
                    PLD_Order              = 0x0)
```

```
            })
        }

        Device (CAMC)
        {
            Name (_ADR, 0x0D)  // _ADR: Address
            Name (_PLD, Package (0x01)  // _PLD: Physical Location of Device
            {
                ToPLD (
                    PLD_Revision          = 0x1,
                    PLD_IgnoreColor       = 0x1,
                    PLD_Red               = 0x0,
                    PLD_Green             = 0x0,
                    PLD_Blue              = 0x0,
                    PLD_Width             = 0x0,
                    PLD_Height            = 0x0,
                    PLD_UserVisible       = 0x1,
                    PLD_Dock              = 0x0,
                    PLD_Lid               = 0x1,
                    PLD_Panel             = "FRONT",
                    PLD_VerticalPosition  = "UPPER",
                    PLD_HorizontalPosition = "CENTER",
                    PLD_Shape             = "UNKNOWN",
                    PLD_GroupOrientation  = 0x0,
                    PLD_GroupToken        = 0x0,
                    PLD_GroupPosition     = 0x0,
                    PLD_Bay               = 0x0,
                    PLD_Ejectable         = 0x0,
                    PLD_EjectRequired     = 0x0,
                    PLD_CabinetNumber     = 0x0,
                    PLD_CardCageNumber    = 0x0,
                    PLD_Reference         = 0x0,
                    PLD_Rotation          = 0x0,
                    PLD_Order             = 0x0)

            })
        }
    }

    Scope (\_SB.PCI0.XHC.RHUB.HS06)
    {
        Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
        {
            If (_OSI ("Darwin"))
            {
                Return (Package ()
                {
                    0x00,
                    0x00,
                    0x00,
                    0x00
```

```
                })
            }

            Return (GUPC (Zero))
        }

        Method (_PLD, 0, NotSerialized)  // _PLD: Physical Location of Device
        {
            Return (GPLD (Zero, Zero))
        }
    }

    Scope (\_SB.PCI0.XHC.RHUB.HS07)
    {
        Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
        {
            If (_OSI ("Darwin"))
            {
                Return (Package ()
                {
                    0xFF,
                    0xFF,
                    0x00,
                    0x00
                })
            }

            Return (GUPC (One))
        }

        Method (_PLD, 0, NotSerialized)  // _PLD: Physical Location of Device
        {
            Return (GPLD (Zero, 0x07))
        }
    }

    Scope (\_SB.PCI0.XHC.RHUB.HS08)
    {
        Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
        {
            If (_OSI ("Darwin"))
            {
                Return (Package ()
                {
                    0x00,
                    0x00,
                    0x00,
                    0x00
                })
            }
            Return (GUPC (One))
```

```
        }

        Method (_PLD, 0, NotSerialized)  // _PLD: Physical Location of Device
        {
            Return (GPLD (Zero, 0x08))
        }
    }

    Scope (\_SB.PCI0.XHC.RHUB.HS09)
    {
        Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
        {
            If (_OSI ("Darwin"))
            {
                Return (Package ()
                {
                    0xFF,
                    0x09,
                    0x00,
                    0x00
                })
            }
            Return (TUPC (0x09))
        }

        Method (_PLD, 0, NotSerialized)  // _PLD: Physical Location of Device
        {
            Return (TPLD (One, 0x09))
        }
    }

    Scope (\_SB.PCI0.XHC.RHUB.HS10)
    {
        Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
        {
            If (_OSI ("Darwin"))
            {
                Return (Package ()
                {
                    0x00,
                    0x00,
                    0x00,
                    0x00
                })
            }
            Return (GUPC (One))
        }

        Method (_PLD, 0, NotSerialized)  // _PLD: Physical Location of Device
        {
            Return (GPLD (Zero, 0x0A))
```

```
        }
    }

    Scope (\_SB.PCI0.XHC.RHUB.USR1)
    {
        Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
        {
            If (_OSI ("Darwin"))
            {
                Return (Package ()
                {
                    0x00,
                    0x00,
                    0x00,
                    0x00
                })
            }
            Return (GUPC (Zero))
        }

        Method (_PLD, 0, NotSerialized)  // _PLD: Physical Location of Device
        {
            Return (GPLD (Zero, Zero))
        }
    }

    Scope (\_SB.PCI0.XHC.RHUB.USR2)
    {
        Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
        {
            If (_OSI ("Darwin"))
            {
                Return (Package ()
                {
                    0x00,
                    0x00,
                    0x00,
                    0x00
                })
            }
            Return (GUPC (Zero))
        }

        Method (_PLD, 0, NotSerialized)  // _PLD: Physical Location of Device
        {
            Return (GPLD (Zero, Zero))
        }
    }

    Scope (\_SB.PCI0.XHC.RHUB.SS01)
    {
```

```
        Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
        {
            If (_OSI ("Darwin"))
            {
                Return (Package ()
                {
                    0xFF,
                    0x03,
                    0x00,
                    0x00
                })
            }
            Return (GUPC (One))
        }

        Method (_PLD, 0, NotSerialized)  // _PLD: Physical Location of Device
        {
            Return (GPLD (One, One))
        }
    }

    Scope (\_SB.PCI0.XHC.RHUB.SS02)
    {
        Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
        {
            If (_OSI ("Darwin"))
            {
                Return (Package ()
                {
                    0x00,
                    0x00,
                    0x00,
                    0x00
                })
            }
            Return (GUPC (One))
        }

        Method (_PLD, 0, NotSerialized)  // _PLD: Physical Location of Device
        {
            Return (GPLD (Zero, 0x04))
        }
    }

    Scope (\_SB.PCI0.XHC.RHUB.SS03)
    {
        Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
        {
            If (_OSI ("Darwin"))
            {
                Return (Package ()
```

```
                {
                    0xFF,
                    0x03,
                    0x00,
                    0x00
                })
            }
            Return (GUPC (One))
        }

        Method (_PLD, 0, NotSerialized)  // _PLD: Physical Location of Device
        {
            Return (GPLD (One, 0x02))
        }
    }

    Scope (\_SB.PCI0.XHC.RHUB.SS04)
    {
        Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
        {
            If (_OSI ("Darwin"))
            {
                Return (Package ()
                {
                    0xFF,
                    0x03,
                    0x00,
                    0x00
                })
            }
            If ((CBID == 0x07A6))
            {
                Return (GUPC (Zero))
            }
            Else
            {
                Return (GUPC (One))
            }
        }

        Method (_PLD, 0, NotSerialized)  // _PLD: Physical Location of Device
        {
            If ((CBID == 0x07A6))
            {
                Return (GPLD (Zero, Zero))
            }
            Else
            {
                Return (GPLD (One, 0x03))
            }
        }
```

```
    }

    Scope (\_SB.PCI0.XHC.RHUB.SS05)
    {
        Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
        {
            If (_OSI ("Darwin"))
            {
                Return (Package ()
                {
                    0x00,
                    0x00,
                    0x00,
                    0x00
                })
            }
            Return (GUPC (Zero))
        }

        Method (_PLD, 0, NotSerialized)  // _PLD: Physical Location of Device
        {
            Return (GPLD (Zero, Zero))
        }
    }

    Scope (\_SB.PCI0.XHC.RHUB.SS06)
    {
        Method (_UPC, 0, NotSerialized)  // _UPC: USB Port Capabilities
        {
            If (_OSI ("Darwin"))
            {
                Return (Package ()
                {
                    0xFF,
                    0x09,
                    0x00,
                    0x00
                })
            }
            Return (TUPC (0x09))
        }

        Method (_PLD, 0, NotSerialized)  // _PLD: Physical Location of Device
        {
            Return (TPLD (One, 0x09))
        }
    }
}
```

禁止 PCI 设备

描述

- 某些情况下，我们希望禁用某个 PCI 设备。比如，PCI 总线的 SD 卡通常无法驱动，即使驱动了也几乎不能正常工作。这种情况下，我们可以通过自定义 SSDT 补丁禁用这个设备。
- 这些设备有以下特征：
  - 它是某个PCI 父设备 的 子设备
  - 父设备 定义了一些 `PCI_Config` 或者 **`SystemMemory`** 类型的变量，其中偏移量0x55数据的第 `D4` 位为设备运行属性
  - 子设备 地址： `Name (_ADR, Zero)`

设备名称

- 较新机器的 子设备 名称为 **`PXSX`** ；父设备 名称为 **`RP01`** ， **`RP02`** ， **`RP03`** ...等。
- 早期 Thinkpad 机器 子设备 名称为 **`SLOT`** 或者 无；父设备 名称为 **`EXP1`** ， **`EXP2`** ， **`EXP3`** ... 等。
- 其他机器可能是其他名称。
- 笔记本的内置无线网卡属于这样的设备。

SSDT 禁用补丁示例

- dell Latitude5480 的 SD 卡属于 PCI 设备，设备路径： `_SB.PCI0.RP01.PXSX`
- 补丁文件：***SSDT-RP01.PXSX-disbale***

```
External (_SB.PCI0.RP01, DeviceObj)
Scope (_SB.PCI0.RP01)
{
    OperationRegion (DE01, PCI_Config, 0x50, 0x01)
    Field (DE01, AnyAcc, NoLock, Preserve)
    {
            ,   4,
        DDDD,   1
    }
        //possible start
        Method (_STA, 0, Serialized)
        {
                If (_OSI ("Darwin"))
                {
                    Return (Zero)
                }
        }
        //possible end
}
```

```
Scope (\)
{
    If (_OSI ("Darwin"))
    {
        \_SB.PCI0.RP01.DDDD = One
    }
}
```

注意

- 如果 父设备 存在多个 子设备 ，请 谨慎使用 本方法。
- 使用时请将示例中的 `RP01` 替换为被禁用设备所属 父设备 名称，参考示例。
- 如果被禁用设备已经包括了 `_STA` 方法，忽略 *possible start* 至 *possible end* 内容，见示例注释部分。

感谢

- @哞

```
// disable RP01.PXSX (ReadCARD)
DefinitionBlock ("", "SSDT", 2, "OCLT", "noRPxx", 0)
{
    External (_SB.PCI0.RP01, DeviceObj)

    Scope (_SB.PCI0.RP01)
    {
        OperationRegion (DE01, PCI_Config, 0x50, 1)
        Field (DE01, AnyAcc, NoLock, Preserve)
        {
                ,   4,
            DDDD,   1
        }
        //possible start
        Method (_STA, 0, Serialized)
        {
            If (_OSI ("Darwin"))
            {
                Return (Zero)
            }
        }
        //possible end
    }

    Scope (\)
    {
        If (_OSI ("Darwin"))
        {
            \_SB.PCI0.RP01.DDDD = One
        }
    }
}
//EOF
```

ACPIDebug说明

描述

通过在 ***SSDT-xxxx*** 补丁里添加调试代码，能够在控制台上看到补丁或者 ACPI的工作过程，用于调试补丁。

要求

- 驱动
  - 安装 ***ACPIDebug.kext*** 至 `OC\Kexts` ，并添加驱动列表。
- 补丁
  - 添加主补丁 ***SSDT-RMDT*** 至 `OC\ACPI` ，并添加补丁列表。
  - 添加 自定义补丁 至 `OC\ACPI` ，并添加补丁列表。

调试

- 打开 控制台 ，搜索 `关键字` （ `关键字` 来自 自定义补丁 ）
- 观察控制台输出结果

示例

- 目的
  - 观察机器睡眠、唤醒后， `ACPI` 的 `_PTS` 、 `_WAK` 接收 `Arg0` 的情况。
- 驱动和补丁
  - ***ACPIDebug.kext*** ——见前文
  - ***SSDT-RMDT*** ——见前文
  - ***SSDT-PTSWAK*** ——补丁内置了参数传递变量 `\_SB.PCI9.TPTS` 、 `\_SB.PCI9.TWAK` ，方便其他补丁使用。参见《PTSWAK综合扩展补丁》
  - ***SSDT-BKeyQxx-Debug*** ——本补丁只是范例。补丁内添加了调试代码，能够在按键响应后执行调试代码。实际使用时可以指定亮度快捷键，或者其他按键。

    注：以上补丁所要求的更名在对应补丁文件的注释里。
- 观察控制台输出结果
  - 打开控制台，搜索 `ABCD-`
  - 完成一次睡眠、唤醒过程
  - 按下 ***SSDT-BKeyQxx-Debug*** 指定的键，观察控制台输出结果。一般情况下，显示结果如下：

```
13:19:50.542733+0800  kernel  ACPIDebug: { "ABCD-_PTS-Arg0=", 0x3, }
13:19:55.541826+0800  kernel  ACPIDebug: { "ABCD-_WAK-Arg0=", 0x3, }
```

以上显示结果是最近一次睡眠、唤醒后 `Arg0` 的值。

备注

- Debug调试代码可以多样化，如：`\RMDT.P1`，`\RMDT.P2`，`\RMDT.P3` 等等，详见 ***SSDT-RMDT.dsl***
- 以上驱动、主要补丁来自 @RehabMan

```
//Debug
DefinitionBlock("", "SSDT", 2, "OCLT", "BrightFN", 0)
{
    External(_SB.PCI0.LPCB.EC0, DeviceObj)
    External(_SB.PCI0.LPCB.PS2K, DeviceObj)
    External(_SB.PCI0.LPCB.EC0.XQXX, MethodObj)
    //
    External(RMDT.P2, MethodObj)
    External(_SB.PCI9.TPTS, IntObj)
    External(_SB.PCI9.TWAK, IntObj)
    //
    Scope (_SB.PCI0.LPCB.EC0)
    {
        Method (_QXX, 0, NotSerialized)
        {
            If (_OSI ("Darwin"))
            {
                //Debug...
                \RMDT.P2 ("ABCD-_PTS-Arg0=", \_SB.PCI9.TPTS)
                \RMDT.P2 ("ABCD-_WAK-Arg0=", \_SB.PCI9.TWAK)
                //Debug...end

                Notify(\_SB.PCI0.LPCB.PS2K, 0x0405)
                Notify(\_SB.PCI0.LPCB.PS2K, 0x20)
            }
            Else
            {
                \_SB.PCI0.LPCB.EC0.XQXX()
            }
        }
    }
}
//EOF
```

```
// for use with ACPIDebug
// This file was created by applying "Add DSDT Debug Methods" to an empty SSDT
//
// Use "Add SSDT Debug Extern Declarations" to access these methods from other
// hotpatch SSDTs or even patched OEM ACPI files.
//
DefinitionBlock("", "SSDT", 2, "OCLT", "RMDT", 0)
{
    If (_OSI ("Darwin"))
    {
        Device (RMDT)
        {
            Name (_HID, "RMD0000")  // _HID: Hardware ID
            Name (RING, Package (0x0100){})
            Mutex (RTMX, 0x00)
            Name (HEAD, Zero)
            Name (TAIL, Zero)
            Method (PUSH, 1, NotSerialized)
            {
                Acquire (RTMX, 0xFFFF)
                Local0 = (HEAD + One)
                If ((Local0 >= SizeOf (RING)))
                {
                    Local0 = Zero
                }

                If ((Local0 != TAIL))
                {
                    RING [HEAD] = Arg0
                    HEAD = Local0
                }

                Release (RTMX)
                Notify (RMDT, 0x80) // Status Change
            }

            Method (FTCH, 0, NotSerialized)
            {
                Acquire (RTMX, 0xFFFF)
                Local0 = Zero
                If ((HEAD != TAIL))
                {
                    Local0 = DerefOf (RING [TAIL])
                    TAIL++
                    If ((TAIL >= SizeOf (RING)))
                    {
                        TAIL = Zero
                    }
                }
```

```
            Release (RTMX)
            Return (Local0)
        }

        Method (COUN, 0, NotSerialized)
        {
            Acquire (RTMX, 0xFFFF)
            Local0 = (HEAD - TAIL) /* \RMDT.TAIL */
            If ((Local0 < Zero))
            {
                Local0 += SizeOf (RING)
            }

            Release (RTMX)
            Return (Local0)
        }

        Method (P1, 1, NotSerialized)
        {
            PUSH (Arg0)
        }

        Method (P2, 2, NotSerialized)
        {
            Local0 = Package (0x02){}
            Local0 [Zero] = Arg0
            Local0 [One] = Arg1
            PUSH (Local0)
        }

        Method (P3, 3, NotSerialized)
        {
            Local0 = Package (0x03){}
            Local0 [Zero] = Arg0
            Local0 [One] = Arg1
            Local0 [0x02] = Arg2
            PUSH (Local0)
        }

        Method (P4, 4, NotSerialized)
        {
            Local0 = Package (0x04){}
            Local0 [Zero] = Arg0
            Local0 [One] = Arg1
            Local0 [0x02] = Arg2
            Local0 [0x03] = Arg3
            PUSH (Local0)
        }

        Method (P5, 5, NotSerialized)
        {
```

```
                Local0 = Package (0x05){}
                Local0 [Zero] = Arg0
                Local0 [One] = Arg1
                Local0 [0x02] = Arg2
                Local0 [0x03] = Arg3
                Local0 [0x04] = Arg4
                PUSH (Local0)
            }

            Method (P6, 6, NotSerialized)
            {
                Local0 = Package (0x06){}
                Local0 [Zero] = Arg0
                Local0 [One] = Arg1
                Local0 [0x02] = Arg2
                Local0 [0x03] = Arg3
                Local0 [0x04] = Arg4
                Local0 [0x05] = Arg5
                PUSH (Local0)
            }

            Method (P7, 7, NotSerialized)
            {
                Local0 = Package (0x07){}
                Local0 [Zero] = Arg0
                Local0 [One] = Arg1
                Local0 [0x02] = Arg2
                Local0 [0x03] = Arg3
                Local0 [0x04] = Arg4
                Local0 [0x05] = Arg5
                Local0 [0x06] = Arg6
                PUSH (Local0)
            }
        }
    }
}
//EOF
```

品牌机器特殊补丁

- Dell 机器特殊补丁
- 小新 PRO13 特殊补丁
- ThinkPad 机器专用补丁

Dell机器特殊补丁

要求

- 检查ACPI是否存在下列 `Device` 和 `Method` ，并且名称相符，否则忽略本章内容。
  - `Device` ：ECDV【PNP0C09】
  - `Device` ：LID0【PNP0C0D】
  - `Method` ：OSID
  - `Method` ：BTNV

特殊补丁

- ***SSDT-OCWork-dell***
  - 补丁里的 `ACSE` 可以设置工作模式。
    - `\_SB.ACSE` = 0为 win7 模式，在此模式下，机器睡眠期间呼吸灯闪烁。
    - `\_SB.ACSE` = 1为 win8 模式，在此模式下，机器睡眠期间呼吸灯灭。
- 修复 Fn+Insert 功能键的补丁组合
  - ***SSDT-PTSWAK*** 参见《PTSWAK综合扩展补丁》
  - ***SSDT-EXT4-WakeScreen*** 参见《PTSWAK综合扩展补丁》
  - ***SSDT-LIDpatch*** 参见《PNP0C0E睡眠修正方法》
  - ***SSDT-FnInsert_BTNV-dell*** 参见《PNP0C0E睡眠修正方法》

其他补丁(参考)

- ***SSDT-EC*** ——参见《仿冒EC》
- ***SSDT-PLUG-xxx*** ——参见《注入X86》
- ***SSDT-PNLF-xxx*** ——参见《PNLF注入方法》
- ***SSDT-GPRW*** ——参见《0D6D补丁》
- ***SSDT-ALS0*** ——参见《仿冒环境光传感器》
- ***SSDT-MCHC*** ——参见《添加缺失的部件》
- ***SSDT-SBUS*** ——参见《SBUS/SMBU补丁》
- ***SSDT-OCI2C-TPXX-dell5480*** ——参见《I2C专用部件》，适用于 `dell_Latitude5480`
- ***SSDT-RMCF-MouseAsTrackpad*** ——参见《PS2键盘映射》
- ***SSDT-RP01.PXSX-disbale*** ——参见《禁止PCI设备》，用于禁止 `dell_Latitude5480` 的SD卡

注：以上补丁所要求的更名在对应补丁文件的注释里。

```
//
DefinitionBlock("", "SSDT", 2, "OCLT", "OCWork", 0)
{
    External (_SB.ACOS, IntObj)
    External (_SB.ACSE, IntObj)

    Scope (\)
    {
        If (_OSI ("Darwin"))
        {
            \_SB.ACOS = 0x80
            \_SB.ACSE = 0 //ACSE=0:win7;;ACSE=1:win8
        }
    }
}
//EOF
```

小新PRO13特殊补丁

特殊更名

PNLF renamed XNLF

```
Find:     504E4C46
Replace:  584E4C46
```

小新PRO的 DSDT 存在变量 `PNLF` ， `PNLF` 和亮度补丁名称有可能冲突，固使用以上更名规避之。

特殊的AOAC补丁

- ***SSDT-NameS3-disable*** ——参见《AOAC方法》之《禁止S3睡眠》
- ***SSDT-NDGP_OFF-AOAC*** ——参见《AOAC方法》之《AOAC禁止独显》
- ***SSDT-DeepIdle*** ——参见《AOAC方法》之《电源空闲管理》
- ***SSDT-PCI0.LPCB-Wake-AOAC*** ——参见《AOAC方法》之《AOAC唤醒方法》

其他补丁(参考)

- ***SSDT-PLUG-_SB.PR00*** ——参见《注入X86》
- ***SSDT-EC*** ——参见《仿冒设备—仿冒EC》
- ***SSDT-PNLF-CFL*** ——参见《PNLF注入方法》
- ***SSDT-PMCR*** ——参见《添加缺失的部件》
- ***SSDT-SBUS*** ——参见《SBUS/SMBU补丁》
- ***SSDT-OCBAT1-lenovoPRO13*** ——参见《电池补丁》
- ***SSDT-I2CxConf*** ——参见《I2C专用部件》
- ***SSDT-OCI2C-TPXX-lenovoPRO13*** ——参见《I2C专用部件》
- ***SSDT-CB-01_XHC*** ——参见《ACPI定制USB端口》
- ***SSDT-GPRW*** ——参见《060D补丁》
- ***SSDT-RTC_Y-AWAC_N*** ——参见《二进制更名与预置变量》
- ***SSDT-RMCF-PS2Map-LenovoPRO13*** ——本章补丁，参见《PS2键盘映射》
- ***SSDT-OCPublic-Merge*** ——本章补丁，参见 附件 说明
- ***SSDT-BATS-PRO13*** ——参见《电池补丁》

注：以上补丁所要求的更名在对应补丁文件的注释里。

备注

- 请阅读《AOAC方法》

附件：合并共用补丁

- 为了简化操作以及减少补丁数量，将某些公用补丁合并为：***SSDT-OCPublic-Merge*** 。

合并的补丁

- ***SSDT-EC-USBX*** ——来自OC官方补丁示例的 **USBX** 部分
- ***SSDT-ALS0*** ——原补丁位于《仿冒设备—仿冒环境光传感器》
- ***SSDT-MCHC*** ——原补丁位于《添加缺失的部件》

注意事项

- ***SSDT-OCPublic-Merge*** 适用于所有机器。
- 使用 ***SSDT-OCPublic-Merge*** 后，上述 合并的补丁 所列补丁不再适用。

OCI2C-TPXX补丁方法

说明

本方法提供一种对 I2C 设备实施 Hotpatch 补丁的解决方案。本方法不涉及 I2C 补丁具体过程和细节。有关更多的 I2C 方面内容详见：

- @penghubingzhou：https://www.penghubingzhou.cn
- @神楽小白(GZ小白):https://blog.gzxiaobai.cn/
- @神楽小白(GZ小白)的触摸板热补丁范例库:https://github.com/GZXiaoBai/Hackintosh-TouchPad-Hotpatch
- VoodooI2C 官方文档：https://voodooi2c.github.io/#GPIO%20Pinning/GPIO%20Pinning
- VoodooI2C 官方支持帖 https://www.tonymacx86.com/threads/voodooi2c-help-and-support.243378/
- Q群：`837538729` (1 群已满)，`921143329` (2 群)

补丁原理和过程

- 禁止原 I2C 设备。详见《二进制更名与预置变量》。

```
/*
 * GPI0 enable
 */
DefinitionBlock("", "SSDT", 2, "OCLT", "GPI0", 0)
{
    External(GPEN, FieldUnitObj)
    // External(GPHD, FieldUnitObj)
    Scope (\)
    {
        If (_OSI ("Darwin"))
        {
            GPEN = 1
            // GPHD = 2
        }
    }
}
```

- 新建一个 I2C 设备 `TPXX`，将原设备所有内容移植到 `TPXX` 中。
- 修正 `TPXX` 有关内容：
  - 原 I2C 设备 `名称` 全部替换为 `TPXX`
  - 修正 `_STA` 部分为：

```
Method (_STA, 0, NotSerialized)
{
```

```
    If (_OSI ("Darwin"))
    {
        Return (0x0F)
    }
    Else
    {
        Return (Zero)
    }
}
```

  - 修正禁止原 I2C 设备时用到的变量的 `有关内容` ，使其符合逻辑关系。
  - 修正涉及到操作系统变量 OSYS 的 `有关内容` ，使其符合逻辑关系。
- 排除错误。
- I2C 补丁。

示例 (Dell Latitude 5480，设备路径： `\_SB.PCI0.I2C1.TPD1`)

- 使用《预置变量法》禁止 `TPD1` 。

```
Scope (\)
{
    If (_OSI ("Darwin"))
    {
        SDS1 = 0
    }
}
```

- 新建设备 `TPXX` ，将原 `TPD1` 所有内容移植到 `TPXX` 中。

```
External(_SB.PCI0.I2C1, DeviceObj)
Scope (_SB.PCI0.I2C1)
{
    Device (TPXX)
    {
       原TPD1内容
    }
}
```

- 修正 `TPXX` 内容
  - 所有 `TPD1` 替换为 `TPXX` 。
  - 补丁中 `_STA` 部分替换为：

```
Method (_STA, 0, NotSerialized)
{
```

```
    If (_OSI ("Darwin"))
    {
        Return (0x0F)
    }
    Else
    {
        Return (Zero)
    }
}
```

- 查找 `SDS1` (禁止 `TPD1` 时用到的变量)，将原 `If (SDS1...)` 修改为 `If (one)` 。
- 查找 `OSYS` ，删除（注释掉）以下内容：

```
//If (LLess (OSYS, 0x07DC))
//{
//    SRXO (GPDI, One)
//}
```

注： `OSYS` 小于 `0x07DC` 时，I2C 设备不工作（ `0x07DC` 代表 Windows8）。

- 添加外部引用 `External...` 修补所有错误。
- I2C 补丁（略）

```
// TPxx is my new's device
DefinitionBlock("", "SSDT", 2, "OCLT", "I2C-TPXX", 0)
{
    External(_SB.PCI0.I2C0, DeviceObj)
    External(TPDD, FieldUnitObj)
    External(TPDF, FieldUnitObj)


    Scope (\)
    {
        If (_OSI ("Darwin"))
        {
            TPDD = 1
        }
    }

    //path:_SB.PCI0.I2C0.TPAD
    Scope (_SB.PCI0.I2C0)
    {
        Device (TPXX)
        {
                Name (_ADR, One)  // _ADR: Address
                Name (_UID, One)  // _UID: Unique ID
                Name (_S0W, 0x03)  // _S0W: S0 Device Wake State
                Name (HID2, Zero)
                Name (TPID, Package (0x04)
                {
                    Package (0x05)
                    {
                        0x04,
                        0x2C,
                        0x20,
                        "SYNA2B2C",
                        "PNP0C50"
                    },

                    Package (0x05)
                    {
                        0x08,
                        0x15,
                        One,
                        "ELAN0650",
                        "PNP0C50"
                    },

                    Package (0x05)
                    {
                        0xFE,
                        0x2C,
                        0x20,
```

```
                        "MSFT0001",
                        "PNP0C50"
                    },

                    Package (0x05)
                    {
                        0xFF,
                        0xFF,
                        0xFF,
                        "MSFT0003",
                        0x130FD041
                    }
                })
                Name (SBFB, ResourceTemplate ()
                {
                    I2cSerialBusV2 (0x0000, ControllerInitiated, 0x00061A80,
                        AddressingMode7Bit, "\\_SB.PCI0.I2C0",
                        0x00, ResourceConsumer, _Y2A, Exclusive,
                        )
                })
                Name (SBFI, ResourceTemplate ()
                {
                    Interrupt (ResourceConsumer, Level, ActiveLow, ExclusiveAndWake
, ,, )
                    {
                        0x00000050,
                    }
                })
                Name (SBFG, ResourceTemplate ()
                {
                    GpioInt (Level, ActiveLow, Exclusive, PullUp, 0x0000,
                        "\\_SB.PCI0.GPI0", 0x00, ResourceConsumer, ,
                        )
                        {   // Pin list
                            0x0038
                        }
                })
                CreateWordField (SBFB, \_SB.PCI0.I2C0.TPXX._Y2A._ADR, ADR0)
                Method (_HID, 0, Serialized)
                {
                    If (~CondRefOf (TPDF))
                    {
                        Name (TPDF, 0xFE)
                    }

                    Switch (One)
                    {
                        Case (Zero)
                        {
                            TPDF = 0xFE
                        }
```

```
                        Case (One)
                        {
                        }
                        Default
                        {
                            TPDF = 0xFE
                        }

                    }

                    Return (TPDS (0x03, 0xFE, "MSFT0001"))
                }

                Method (_CID, 0, Serialized)  // _CID: Compatible ID
                {
                    If (~CondRefOf (TPDF))
                    {
                        Name (TPDF, 0xFE)
                    }

                    Switch (One)
                    {
                        Case (Zero)
                        {
                            TPDF = 0xFE
                        }
                        Case (One)
                        {
                        }
                        Default
                        {
                            TPDF = 0xFE
                        }

                    }

                    Return (TPDS (0x04, 0xFE, "PNP0C50"))
                }

                Method (TPDS, 3, NotSerialized)
                {
                    Local0 = Zero
                    Local1 = Zero
                    Local1 = DerefOf (DerefOf (TPID [Local0]) [Zero])
                    While (((Local1 != Arg1) && (Local1 != TPDF)))
                    {
                        Local0++
                        If ((Local0 >= SizeOf (TPID)))
                        {
                            Return (Arg2)
                        }
```

```
                Local1 = DerefOf (DerefOf (TPID [Local0]) [Zero])
            }

            Return (DerefOf (DerefOf (TPID [Local0]) [Arg0]))
        }

        Method (_DSM, 4, NotSerialized)
        {
            If ((Arg0 == ToUUID ("3cdff6f7-4267-4555-ad05-b30a3d8938de")))
            {
                If ((Arg2 == Zero))
                {
                    If ((Arg1 == One))
                    {
                        Return (Buffer (One)
                        {
                             0x03
                        })
                    }
                    Else
                    {
                        Return (Buffer (One)
                        {
                             0x00
                        })
                    }
                }

                If ((Arg2 == One))
                {
                    Return (HID2)
                }
            }
            Else
            {
                Return (Buffer (One)
                {
                     0x00
                })
            }

            If ((Arg0 == ToUUID ("ef87eb82-f951-46da-84ec-14871ac6f84b")))
            {
                If ((Arg2 == Zero))
                {
                    If ((Arg1 == One))
                    {
                        Return (Buffer (One)
                        {
                             0x03
```

```
                        })
                    }
                    Else
                    {
                        Return (Buffer (One)
                        {
                             0x00
                        })
                    }
                }

                If ((Arg2 == One))
                {
                    Return (ConcatenateResTemplate (SBFB, SBFG))
                }
            }
            Else
            {
                Return (Buffer (One)
                {
                     0x00
                })
            }

            Return (Buffer (One)
            {
                 0x00
            })
        }

        Method (_STA, 0, Serialized)
        {
            If (_OSI ("Darwin"))
            {
                Return (0x0F)
            }
            Else
            {
                Return (Zero)
            }
        }

        Method (_CRS, 0, Serialized)
        {
            Local0 = Zero
            Local1 = Zero
            Local1 = DerefOf (DerefOf (TPID [Local0]) [Zero])
            While (((Local1 != 0xFE) && (Local1 != TPDF)))
            {
                Local0++
                If ((Local0 >= SizeOf (TPID)))
```

```
                        {
                            Break
                        }

                        Local1 = DerefOf (DerefOf (TPID [Local0]) [Zero])
                    }

                    ADR0 = DerefOf (DerefOf (TPID [Local0]) [One])
                    HID2 = DerefOf (DerefOf (TPID [Local0]) [0x02])

                    Return (ConcatenateResTemplate (SBFB, SBFG))
                    //Return (ConcatenateResTemplate (SBFB, SBFI))
                }
            }
        }

}
//EOF
```

```
// TPxx is my new's device
DefinitionBlock("", "SSDT", 2, "OCLT", "I2C-TPXX", 0)
{
    External(_SB.PCI0.I2C1, DeviceObj)
    //External(OSYS, FieldUnitObj)
    External(GPDI, FieldUnitObj)
    External(SDM1, FieldUnitObj)
    External(SDS1, FieldUnitObj)
    External(_SB.SRXO, MethodObj)
    External(_SB.SHPO, MethodObj)
    External(_SB.CBID, MethodObj)
    External(_SB.GNUM, MethodObj)
    External(_SB.INUM, MethodObj)
    External(_SB.PCI0.HIDD, MethodObj)
    External(_SB.PCI0.TP7D, MethodObj)
    External(_SB.PCI0.HIDG, IntObj)
    External(_SB.PCI0.TP7G, IntObj)


    Scope (\)
    {
        If (_OSI ("Darwin"))
        {
            SDS1 = 0
        }
    }

    //path:_SB.PCI0.I2C1
    Scope (_SB.PCI0.I2C1)
    {
        Device (TPXX)
        {
            Name (HID2, Zero)
            Name (SBFB, ResourceTemplate ()
            {
                I2cSerialBusV2 (0x002C, ControllerInitiated, 0x00061A80,
                    AddressingMode7Bit, "\\_SB.PCI0.I2C1",
                    0x00, ResourceConsumer, , Exclusive,
                    )
            })
            Name (SBFI, ResourceTemplate ()
            {
                Interrupt (ResourceConsumer, Level, ActiveLow, ExclusiveAndWake, ,,
 _Y2A)
                {
                    0x00000000,
                }
            })
            Name (SBFG, ResourceTemplate ()
            {
```

```
                GpioInt (Level, ActiveLow, ExclusiveAndWake, PullDefault, 0x0000,
                    "\\_SB.PCI0.GPI0", 0x00, ResourceConsumer, ,
                    )
                    {   // Pin list
                        0x0000
                    }
            })
            CreateWordField (SBFG, 0x17, INT1)
            CreateDWordField (SBFI, \_SB.PCI0.I2C1.TPXX._Y2A._INT, INT2)  // _INT:
Interrupts
            Method (_INI, 0, NotSerialized)  // _INI: Initialize
            {
                //If (LLess (OSYS, 0x07DC))
                //{
                //    SRXO (GPDI, One)
                //}

                Store (GNUM (GPDI), INT1)
                Store (INUM (GPDI), INT2)
                If (LEqual (SDM1, Zero))
                {
                    SHPO (GPDI, One)
                }

                //If (LEqual (SDS1, 0x07))
                If (One)
                {
                    If (LEqual (CBID, 0x07A6))
                    {
                        Store ("DLL07A6", _HID)
                    }

                    If (LEqual (CBID, 0x07A7))
                    {
                        Store ("DLL07A7", _HID)
                    }

                    If (LEqual (CBID, 0x07A8))
                    {
                        Store ("DLL07A8", _HID)
                    }

                    If (LEqual (CBID, 0x07A9))
                    {
                        Store ("DLL07A9", _HID)
                    }

                    If (LEqual (CBID, 0x07D0))
                    {
                        Store ("DLL07D0", _HID)
                    }
```

```
                If (LEqual (CBID, 0x07D1))
                {
                    Store ("DLL07D1", _HID)
                }

                Store (0x20, HID2)
                //Return (Zero)
            }
        }

        Name (_HID, "XXXX0000")  // _HID: Hardware ID
        Name (_CID, "PNP0C50")  // _CID: Compatible ID
        Name (_S0W, 0x03)  // _S0W: S0 Device Wake State
        Method (_DSM, 4, Serialized)  // _DSM: Device-Specific Method
        {
            If (LEqual (Arg0, HIDG))
            {
                Return (HIDD (Arg0, Arg1, Arg2, Arg3, HID2))
            }

            If (LEqual (Arg0, TP7G))
            {
                Return (TP7D (Arg0, Arg1, Arg2, Arg3, SBFB, SBFG))
            }

            Return (Buffer (One)
            {
                 0x00
            })
        }

        Method (_STA, 0, NotSerialized)
        {
            If (_OSI ("Darwin"))
            {
                Return (0x0F)
            }
            Else
            {
                Return (Zero)
            }
        }

        Method (_CRS, 0, NotSerialized)  // _CRS: Current Resource Settings
        {
            Return (ConcatenateResTemplate (SBFB, SBFG))
            //Return (ConcatenateResTemplate (SBFB, SBFI))
        }
    }
}
```

```
}
//EOF
```

```
// TPxx is my new's device
DefinitionBlock("", "SSDT", 2, "OCLT", "I2C-TPXX", 0)
{
    External(_SB.PCI0.I2C0, DeviceObj)
    External(TPDD, FieldUnitObj)
    External(TPDF, FieldUnitObj)


    Scope (\)
    {
        If (_OSI ("Darwin"))
        {
            TPDD = 3
        }
    }

    //path:_SB.PCI0.I2C0.TPAD
    Scope (_SB.PCI0.I2C0)
    {
        Device (TPXX)
        {
                Name (_ADR, One)  // _ADR: Address
                Name (_UID, One)  // _UID: Unique ID
                Name (_S0W, 0x03)  // _S0W: S0 Device Wake State
                Name (HID2, Zero)
                Name (TPID, Package (0x04)
                {
                    Package (0x05)
                    {
                        0x10,
                        0x15,
                        One,
                        "MSFT0002",
                        "PNP0C50"
                    },

                    Package (0x05)
                    {
                        0x11,
                        0x2C,
                        0x20,
                        "MSFT0004",
                        "PNP0C50"
                    },

                    Package (0x05)
                    {
                        0xFE,
                        0x2C,
                        0x20,
```

```
                        "MSFT0001",
                        "PNP0C50"
                    },

                    Package (0x05)
                    {
                        0xFF,
                        0xFF,
                        0xFF,
                        "MSFT0003",
                        0x030FD041
                    }
                })
                Name (SBFB, ResourceTemplate ()
                {
                    I2cSerialBusV2 (0x0000, ControllerInitiated, 0x00061A80,
                        AddressingMode7Bit, "\\_SB.PCI0.I2C0",
                        0x00, ResourceConsumer, _Y4B, Exclusive,
                        )
                })
                Name (SBFI, ResourceTemplate ()
                {
                    Interrupt (ResourceConsumer, Level, ActiveLow, ExclusiveAndWake
, ,, )
                    {
                        0x00000050,
                    }
                })
                Name (SBFG, ResourceTemplate ()
                {
                    GpioInt (Level, ActiveLow, Exclusive, PullUp, 0x0000,
                        "\\_SB.PCI0.GPI0", 0x00, ResourceConsumer, ,
                        )
                        {   // Pin list
                            0x0038
                        }
                })
                CreateWordField (SBFB, \_SB.PCI0.I2C0.TPXX._Y4B._ADR, ADR0)
                Method (_HID, 0, Serialized)
                {
                    If (~CondRefOf (TPDF))
                    {
                        Name (TPDF, 0xFE)
                    }

                    Switch (One)
                    {
                        Case (Zero)
                        {
                            TPDF = 0xFE
                        }
```

```
                        Case (One)
                        {
                        }
                        Default
                        {
                            TPDF = 0xFE
                        }

                    }

                    Return (TPDS (0x03, 0xFE, "MSFT0001"))
                }

                Method (_CID, 0, Serialized)  // _CID: Compatible ID
                {
                    If (~CondRefOf (TPDF))
                    {
                        Name (TPDF, 0xFE)
                    }

                    Switch (One)
                    {
                        Case (Zero)
                        {
                            TPDF = 0xFE
                        }
                        Case (One)
                        {
                        }
                        Default
                        {
                            TPDF = 0xFE
                        }

                    }

                    Return (TPDS (0x04, 0xFE, "PNP0C50"))
                }

                Method (TPDS, 3, NotSerialized)
                {
                    Local0 = Zero
                    Local1 = Zero
                    Local1 = DerefOf (DerefOf (TPID [Local0]) [Zero])
                    While (((Local1 != Arg1) && (Local1 != TPDF)))
                    {
                        Local0++
                        If ((Local0 >= SizeOf (TPID)))
                        {
                            Return (Arg2)
                        }
```

```
                    Local1 = DerefOf (DerefOf (TPID [Local0]) [Zero])
                }

                Return (DerefOf (DerefOf (TPID [Local0]) [Arg0]))
            }

            Method (_DSM, 4, NotSerialized)
            {
                If ((Arg0 == ToUUID ("3cdff6f7-4267-4555-ad05-b30a3d8938de")))
                {
                    If ((Arg2 == Zero))
                    {
                        If ((Arg1 == One))
                        {
                            Return (Buffer (One)
                            {
                                 0x03
                            })
                        }
                        Else
                        {
                            Return (Buffer (One)
                            {
                                 0x00
                            })
                        }
                    }

                    If ((Arg2 == One))
                    {
                        Return (HID2)
                    }
                }
                Else
                {
                    Return (Buffer (One)
                    {
                         0x00
                    })
                }

                If ((Arg0 == ToUUID ("ef87eb82-f951-46da-84ec-14871ac6f84b")))
                {
                    If ((Arg2 == Zero))
                    {
                        If ((Arg1 == One))
                        {
                            Return (Buffer (One)
                            {
                                 0x03
```

```
                        })
                    }
                    Else
                    {
                        Return (Buffer (One)
                        {
                             0x00
                        })
                    }
                }

                If ((Arg2 == One))
                {
                    Return (ConcatenateResTemplate (SBFB, SBFG))
                }
            }
            Else
            {
                Return (Buffer (One)
                {
                     0x00
                })
            }

            Return (Buffer (One)
            {
                 0x00
            })
        }

        Method (_STA, 0, Serialized)
        {
            If (_OSI ("Darwin"))
            {
                Return (0x0F)
            }
            Else
            {
                Return (Zero)
            }
        }

        Method (_CRS, 0, Serialized)
        {
            Local0 = Zero
            Local1 = Zero
            Local1 = DerefOf (DerefOf (TPID [Local0]) [Zero])
            While (((Local1 != 0xFE) && (Local1 != TPDF)))
            {
                Local0++
                If ((Local0 >= SizeOf (TPID)))
```

```
                        {
                            Break
                        }

                        Local1 = DerefOf (DerefOf (TPID [Local0]) [Zero])
                    }

                    ADR0 = DerefOf (DerefOf (TPID [Local0]) [One])
                    HID2 = DerefOf (DerefOf (TPID [Local0]) [0x02])

                    //Return (ConcatenateResTemplate (SBFB, SBFG))
                    Return (ConcatenateResTemplate (SBFB, SBFI))
                }
            }
        }

}
//EOF
```

```
// TPxx is my new's device
DefinitionBlock("", "SSDT", 2, "ACDT", "I2C-TPXX", 0)
{
    External(_SB.PCI0.I2C1, DeviceObj)
    External(_SB.GNUM, MethodObj)
    External(_SB.INUM, MethodObj)
    External(_SB.SHPO, MethodObj)
    External(GPDI, FieldUnitObj)
    External(TPDM, FieldUnitObj)
    External(TPDT, FieldUnitObj)
    External(TPTY, FieldUnitObj)
    External(TPDS, FieldUnitObj)
    External(TPTP, FieldUnitObj)
    External(_SB.PCI0.HIDG, IntObj)
    External(_SB.PCI0.TP7G, IntObj)
    External(_SB.PCI0.HIDD, MethodObj)
    External(_SB.PCI0.TP7D, MethodObj)
    External(_SB.PCI0.I2CM, MethodObj)
    External(_SB.PCI0.I2C1.I2CX, IntObj)

    Scope (\)
    {
        If (_OSI ("Darwin"))
        {
            TPDT = 0
        }
    }

    Scope (_SB.PCI0.I2C1)
    {
        Device (TPXX)
        {
            Name (HID2, Zero)
            Name (SBFB, ResourceTemplate ()
            {
                I2cSerialBusV2 (0x002C, ControllerInitiated, 0x00061A80,
                    AddressingMode7Bit, "\\_SB.PCI0.I2C1",
                    0x00, ResourceConsumer, _Y3B, Exclusive,
                    )
            })
            Name (SBFG, ResourceTemplate ()
            {
                GpioInt (Level, ActiveLow, ExclusiveAndWake, PullDefault, 0x0000,
                    "\\_SB.PCI0.GPI0", 0x00, ResourceConsumer, ,
                    )
                    {   // Pin list
                        0x0000
                    }
            })
            Name (SBFI, ResourceTemplate ()
```

```
            {
                Interrupt (ResourceConsumer, Level, ActiveLow, ExclusiveAndWake, ,,
_Y3C)
                {
                    0x00000000,
                }
            })
            CreateWordField (SBFB, \_SB.PCI0.I2C1.TPXX._Y3B._ADR, BADR)
            CreateDWordField (SBFB, \_SB.PCI0.I2C1.TPXX._Y3B._SPE, SPED)
            CreateWordField (SBFG, 0x17, INT1)
            CreateDWordField (SBFI, \_SB.PCI0.I2C1.TPXX._Y3C._INT, INT2)
            Method (_INI, 0, NotSerialized)
            {
                //If ((OSYS < 0x07DC))
                //{
                //    SRXO (GPDI, One)
                //}

                INT1 = GNUM (GPDI)
                INT2 = INUM (GPDI)
                //If ((TPDM == Zero))
                //{
                SHPO (GPDI, One)
                //}

                /*
                If ((TPDT == One))
                {
                    _HID = "SYNA2393"
                    HID2 = 0x20
                    Return (Zero)
                }

                If ((TPDT == 0x02))
                {
                    _HID = "06CB2846"
                    HID2 = 0x20
                    Return (Zero)
                }

                If ((TPDT == 0x06))
                {
                    _HID = "ALPS0000"
                    HID2 = 0x20
                    BADR = 0x2C
                    Return (Zero)
                }
                */

                //If ((TPDT == 0x05))
                //{
```

```
                    If ((TPTY == One))
                    {
                        //ADBG ("TPTY=1")
                        _HID = "MSFT0001"
                        _SUB = "ELAN0001"
                        BADR = 0x15
                        HID2 = One
                    }

                    If ((TPTY == 0x02))
                    {
                        //ADBG ("TPTY=2")
                        If ((TPTP == 0xCDF4))
                        {
                            _HID = "SYNACDF4"
                        }
                        Else
                        {
                            _HID = "SYNACD3E"
                        }

                        _SUB = "SYNA0001"
                        BADR = 0x2C
                        HID2 = 0x20
                    }

                    If ((TPDS == Zero))
                    {
                        SPED = 0x000186A0
                    }

                    If ((TPDS == One))
                    {
                        SPED = 0x00061A80
                    }

                    If ((TPDS == 0x02))
                    {
                        SPED = 0x000F4240
                    }

                    Return (Zero)
                //}
            }
            Name (_HID, "XXXX0000")
            Name (_CID, "PNP0C50" )
            Name (_SUB, "XXXX0000")
            Name (_S0W, 0x03)
            Method (_DSM, 4, Serialized)
            {
                If ((Arg0 == HIDG))
```

```
                {
                    Return (HIDD (Arg0, Arg1, Arg2, Arg3, HID2))
                }

                If ((Arg0 == TP7G))
                {
                    Return (TP7D (Arg0, Arg1, Arg2, Arg3, SBFB, SBFG))
                }

                Return (Buffer (One)
                {
                     0x00
                })
            }

            Method (_CRS, 0, NotSerialized)
            {
                //Return (ConcatenateResTemplate (SBFB, SBFG))
                Return (ConcatenateResTemplate (I2CM (I2CX, BADR, SPED), SBFG))
            }

            Method (_STA, 0, Serialized)
            {
                If (_OSI ("Darwin"))
                {
                    Return (0x0F)
                }
                Else
                {
                    Return (Zero)
                }
            }
        }
    }
}
//EOF
```

```
// GPI0 enable
DefinitionBlock("", "SSDT", 2, "OCLT", "GPI0", 0)
{
    External(GPEN, FieldUnitObj)

    Scope (\)
    {
        If (_OSI ("Darwin"))
        {
            GPEN = 1
        }
    }
}
//EOF
```

```
/*
 * Attention! Before you use this hotpatch!
 * Please follow the instructions below!
 * Otherwise you may receive ACPI Errors!
 *
 * ============================ *
 *                              *
 * 1. Rename SSCN to XSCN       *
 * ---- Find:     5353434e ---- *
 * ---- Replace:  5853434e ---- *
 *                              *
 *============================= *
 *                              *
 * 1. Rename FMCN to XMCN       *
 * ---- Find:     464d434e ---- *
 * ---- Replace:  584d434e ---- *
 *                              *
 *============================= *
 *
 */
DefinitionBlock ("", "SSDT", 2, "OCLT", "I2CxConf", 0x00000000)
{
    External (_SB.PCI0.GPI0._HID, MethodObj)
    External (_SB.PCI0.I2C0, DeviceObj)
    External (_SB.PCI0.I2C0.XMCN, MethodObj)
    External (_SB.PCI0.I2C0.XSCN, MethodObj)
    External (_SB.PCI0.I2C1, DeviceObj)
    External (_SB.PCI0.I2C1.XMCN, MethodObj)
    External (_SB.PCI0.I2C1.XSCN, MethodObj)
    External (USTP, FieldUnitObj)
    External (FMD0, IntObj)
    External (FMD1, IntObj)
    External (FMH0, IntObj)
    External (FMH1, IntObj)
    External (FML0, IntObj)
    External (FML1, IntObj)
    External (SSD0, IntObj)
    External (SSD1, IntObj)
    External (SSH0, IntObj)
    External (SSH1, IntObj)
    External (SSL0, IntObj)
    External (SSL1, IntObj)

    Scope (\)
    {
        If (_OSI ("Darwin"))
        {
            Method (XXEN, 0, NotSerialized)
            {
                If ((\_SB.PCI0.GPI0._HID() == "INT344B") || (\_SB.PCI0.GPI0._HID()
```

```
 == "INT345D"))
                {
                    Return (One)
                }

                Return (Zero)
            }

            Method (PKGX, 3, Serialized)
            {
                Name (PKG, Package (0x03)
                {
                    Zero,
                    Zero,
                    Zero
                })
                PKG [Zero] = Arg0
                PKG [One] = Arg1
                PKG [0x02] = Arg2
                Return (PKG)
            }
        }
    }

    Scope (_SB.PCI0.I2C0)
    {
        Method (SSCN, 0, NotSerialized)
        {
            If (_OSI ("Darwin"))
            {
                If (((CondRefOf (SSH0) && CondRefOf (SSL0)) && CondRefOf (SSD0)))
                {
                    Return (PKGX (SSH0, SSL0, SSD0))
                }

                If (\XXEN ())
                {
                    Return (PKGX (0x01B0, 0x01FB, 0x1E))
                }

                Return (PKGX (0x03F2, 0x043D, 0x62))
            }
            ElseIf (CondRefOf (\_SB.PCI0.I2C0.XSCN))
            {
                If (USTP)
                {
                    Return (\_SB.PCI0.I2C0.XSCN ())
                }
            }

            Return (Zero)
```

```
        }

        Method (FMCN, 0, NotSerialized)
        {
            If (_OSI ("Darwin"))
            {
                If (((CondRefOf (FMH0) && CondRefOf (FML0)) && CondRefOf (FMD0)))
                {
                    Return (PKGX (FMH0, FML0, FMD0))
                }

                If (\XXEN ())
                {
                    Return (PKGX (0x48, 0xA0, 0x1E))
                }

                Return (PKGX (0x0101, 0x012C, 0x62))
            }
            ElseIf (CondRefOf (\_SB.PCI0.I2C0.XMCN))
            {
                If (USTP)
                {
                    Return (\_SB.PCI0.I2C0.XMCN ())
                }
            }

            Return (Zero)
        }
    }

    Scope (_SB.PCI0.I2C1)
    {
        Method (SSCN, 0, NotSerialized)
        {
            If (_OSI ("Darwin"))
            {
                If (((CondRefOf (SSH1) && CondRefOf (SSL1)) && CondRefOf (SSD1)))
                {
                    Return (PKGX (SSH1, SSL1, SSD1))
                }

                If (\XXEN ())
                {
                    Return (PKGX (0x0210, 0x0280, 0x1E))
                }

                Return (PKGX (0x03F2, 0x043D, 0x62))
            }
            ElseIf (CondRefOf (\_SB.PCI0.I2C1.XSCN))
            {
                If (USTP)
```

```
                {
                    Return (\_SB.PCI0.I2C1.XSCN ())
                }
            }

            Return (Zero)
        }

        Method (FMCN, 0, NotSerialized)
        {
            If (_OSI ("Darwin"))
            {
                If (((CondRefOf (FMH1) && CondRefOf (FML1)) && CondRefOf (FMD1)))
                {
                    Return (PKGX (FMH1, FML1, FMD1))
                }

                If (\XXEN ())
                {
                    Return (PKGX (0x80, 0xA0, 0x1E))
                }

                Return (PKGX (0x0101, 0x012C, 0x62))
            }
            ElseIf (CondRefOf (\_SB.PCI0.I2C1.XMCN))
            {
                If (USTP)
                {
                    Return (\_SB.PCI0.I2C1.XMCN ())
                }
            }

            Return (Zero)
        }
    }
}
//EOF
```

```
// GPI0 enable
DefinitionBlock("", "SSDT", 2, "OCLT", "GPI0", 0)
{
    External(GPHD, FieldUnitObj)


    Scope (\)
    {
        If (_OSI ("Darwin"))
        {
            GPHD = 2
        }
    }

}
//EOF
```

SSDT 屏蔽独显方法

二种屏蔽独显方法

- `config` 方法
  - `DeviceProperties\Add\PciRoot(0x0)/Pci(0x2,0x0)` 添加

    ```
    disable-external-gpu  01000000
    ```

  - 添加引导参数

    ```
    boot-args             -wegnoegpu
    ```

- 本方法 —— SSDT 屏蔽独显方法

SSDT屏蔽独显过程

- 初始化阶段禁用独显。
- 机器睡眠期间启用独显，防止独显在被禁用状态下进入 `S3` 而可能导致的系统崩溃。
- 机器唤醒后再次禁用独显。

补丁组合

- 综合补丁—— ***SSDT-PTSWAK***
- 屏蔽独显补丁—— ***SSDT-NDGP_OFF*** 【或者 ***SSDT-NDGP_PS3*** 】

示例

- ***SSDT-PTSWAK***

  略，详见《PTSWAK综合扩展补丁》。

- ***SSDT-NDGP_OFF***
  - 查询独显的名称和路径，确认其存在 `_ON` 和 `_OFF` 方法
  - 参考示例，修改其名称和路径同查询结果一致
- ***SSDT-NDGP_PS3***
  - 查询独显的名称和路径，确认其存在 `_PS0` 、 `_PS3` 和 `_DSM` 方法
  - 参考示例，修改其名称和路径同查询结果一致
- 注意
  - 查询独显名称和路径以及 `_ON` 、 `_OFF` 、 `_PS0` 、 `_PS3` 和 `_DSM` 时，应对全部

`ACPI` 文件进行搜索，它可能存在于 `DSDT` 文件中，也可能存在于 `ACPI` 的其他 `SSDT` 文件中。
    - 示例中的独显名称和路径是： `_SB.PCI0.RP13.PXSX` 。

注意事项

- 按照 补丁组合 要求，须同时使用 ***SSDT-PTSWAK*** 和 ***SSDT-NDGP_OFF*** 【或者 ***SSDT-NDGP_PS3*** 】
- 如果 ***SSDT-NDGP_OFF*** 和 ***SSDT-NDGP_PS3*** 均满足使用要求，优先使用 ***SSDT-NDGP_OFF***

注 ：以上主要内容来自 @RehabMan

```
//
DefinitionBlock("", "SSDT", 2, "OCLT", "NDGP", 0)
{
    External(_SB.PCI0.RP13.PXSX._ON, MethodObj)
    External(_SB.PCI0.RP13.PXSX._OFF, MethodObj)

    If (_OSI ("Darwin"))
    {
        Device(DGPU)
        {
            Name(_HID, "DGPU1000")
            Method (_INI, 0, NotSerialized)
            {
                _OFF()
            }

            Method (_ON, 0, NotSerialized)
            {
                //path:
                If (CondRefOf (\_SB.PCI0.RP13.PXSX._ON))
                {
                    \_SB.PCI0.RP13.PXSX._ON()
                }
            }

            Method (_OFF, 0, NotSerialized)
            {
                //path:
                If (CondRefOf (\_SB.PCI0.RP13.PXSX._OFF))
                {
                    \_SB.PCI0.RP13.PXSX._OFF()
                }
            }
        }
    }
}
//EOF
```

```
//
DefinitionBlock("", "SSDT", 2, "OCLT", "NDGP", 0)
{
    External(_SB.PCI0.RP13.PXSX._PS0, MethodObj)
    External(_SB.PCI0.RP13.PXSX._PS3, MethodObj)
    External(_SB.PCI0.RP13.PXSX._DSM, MethodObj)

    If (_OSI ("Darwin"))
    {
        Device(DGPU)
        {
            Name(_HID, "DGPU1000")
            Method (_INI, 0, NotSerialized)
            {
                _OFF()
            }

            Method (_ON, 0, NotSerialized)
            {
                //path:
                If (CondRefOf (\_SB.PCI0.RP13.PXSX._PS0))
                {
                    \_SB.PCI0.RP13.PXSX._PS0()
                }
            }

            Method (_OFF, 0, NotSerialized)
            {
                //path:
                If (CondRefOf (\_SB.PCI0.RP13.PXSX._PS3))
                {
                    \_SB.PCI0.RP13.PXSX._DSM (Buffer ()
                    {
                        /* 0000 */ 0xF8, 0xD8, 0x86, 0xA4, 0xDA, 0x0B, 0x1B, 0x47,
                        /* 0008 */ 0xA7, 0x2B, 0x60, 0x42, 0xA6, 0xB5, 0xBE, 0xE0
                    }, 0x0100, 0x1A,Buffer ()
                    { 0x01, 0x00, 0x00, 0x03 })
                    \_SB.PCI0.RP13.PXSX._PS3()
                }
            }
        }
    }
}
//EOF
```

保留部件

- 新的 PS2 驱动已经支持亮度快捷键，不再需要《亮度快捷键》部件
- 《CMOS重置补丁》和《声卡IRQ补丁》不再适用

声卡 IRQ 补丁

描述

- 早期机器的声卡要求部件 **HPET** `PNP0103` 提供中断号 `0` & `8` ，否则声卡不能正常工作。实际情况几乎所有机器的 **HPET** 未提供任何中断号。通常情况下，中断号 `0` & `8` 分别被 **RTC** `PNP0B00` 、 **TIMR** `PNP0100` 占用。
- 解决上述问题需同步修复 **HPET**、**RTC**、**TIMR**。

补丁原理

- 禁用 **HPET**、**RTC**、**TIMR** 三部件。
- 仿冒三部件，即：**HPE0**、**RTC0**、**TIM0**。
- 将 **RTC0** 的 `IRQNoFlags (){8}` 和 **TIM0** 的 `IRQNoFlags (){0}` 移除并添加到 **HPE0**。

补丁方法

- 禁用 **HPET**、**RTC**、**TIMR**
  - **HPET**

    通常HPET存在 `_STA` ，因此，禁用HPET需使用《预置变量法》。如：

    ```
    External (HPAE, IntObj) /* 或者 External (HPTE, IntObj) */
    Scope (\)
    {
        If (_OSI ("Darwin"))
        {
            HPAE =0 /* 或者 HPTE =0 */
        }
    }
    ```

    注意： `_STA` 内 `HPAE` 变量随机器不同可能不同。

  - **RTC**

    早期机器的 RTC 无 `_STA` ，按 `Method (_STA,` 方法禁用 RTC。如：

    ```
    Method (_STA, 0, NotSerialized)
    {
        If (_OSI ("Darwin"))
        {
            Return (0)
        }
        Else
    ```

```
    {
        Return (0x0F)
    }
}
```

- **TIMR**

  同 **RTC**

- 补丁文件：***SSDT-HPET_RTC_TIMR-fix***

  见上述 补丁原理 ，参考示例。

注意事项

- 本补丁不可以和下列补丁同时使用：
  - 《二进制更名与预置变量》的 ***SSDT-RTC_Y-AWAC_N***
  - OC 官方的 ***SSDT-AWAC***
  - 《仿冒设备》或者 OC 官方的 ***SSDT-RTC0***
  - 《CMOS重置补丁》的 ***SSDT-RTC0-NoFlags***
- `LPCB` 名称、 三部件 名称以及 `IPIC` 名称应和原始 `ACPI` 部件名称一致。
- 若三合一补丁无法解决，在使用三合一补丁的前提下尝试 ***SSDT-IPIC***。按照上文禁用 ***HPET***、***RTC*** 和 ***TIMR*** 的方法禁用 ***IPIC*** 设备，然后仿冒一个 ***IPI0*** 设备，设备内容为原始 `DSDT` 中的 ***IPIC*** 或 ***PIC*** 设备内容，最后将 `IRQNoFlags{2}` 删除即可，参考示例。

```
//Fix HPET,RTC,TIMR
DefinitionBlock ("", "SSDT", 2, "OCLT", "HRTfix", 0)
{
    External (_SB.PCI0.LPCB, DeviceObj)
    External (_SB.PCI0.LPCB.RTC, DeviceObj)
    External (_SB.PCI0.LPCB.TIMR, DeviceObj)
    External (HPAE, IntObj)
    //External (HPTE, IntObj)


    //disable HPET
    Scope (\)
    {
        If (_OSI ("Darwin"))
        {
            HPAE =0
            //HPTE =0
        }
    }

    //disable RTC
    Scope (_SB.PCI0.LPCB.RTC)
    {
        Method (_STA, 0, NotSerialized)
        {
            If (_OSI ("Darwin"))
            {
                Return (0)
            }
            Else
            {
                Return (0x0F)
            }
        }
    }

    //disable TIMR
    Scope (_SB.PCI0.LPCB.TIMR)
    {
        Method (_STA, 0, NotSerialized)
        {
            If (_OSI ("Darwin"))
            {
                Return (0)
            }
            Else
            {
                Return (0x0F)
            }
        }
```

```
    }

    Scope (_SB.PCI0.LPCB)
    {
        //Fake HPE0
        Device (HPE0)
        {
            Name (_HID, EisaId ("PNP0103"))
            Name (_UID, Zero)
            Name (BUF0, ResourceTemplate ()
            {
                IRQNoFlags() { 0, 8 }
                Memory32Fixed (ReadWrite,
                    0xFED00000,
                    0x00000400,
                    )
            })
            Method (_STA, 0, NotSerialized)
            {
                If (_OSI ("Darwin"))
                {
                    Return (0x0F)
                }
                Else
                {
                    Return (0)
                }
            }
            Method (_CRS, 0, Serialized)
            {
                Return (BUF0)
            }
        }

        //Fake RTC0
        Device (RTC0)
        {
            Name (_HID, EisaId ("PNP0B00"))
            Name (_CRS, ResourceTemplate ()
            {
                IO (Decode16,
                    0x0070,
                    0x0070,
                    0x01,
                    0x02,
                    //0x08
                    )
            })
            Method (_STA, 0, NotSerialized)
            {
                If (_OSI ("Darwin"))
```

```
                {
                    Return (0x0F)
                }
                Else
                {
                    Return (0)
                }
            }
        }

        //Fake TIM0
        Device (TIM0)
        {
            Name (_HID, EisaId ("PNP0100"))
            Name (_CRS, ResourceTemplate ()
            {
                IO (Decode16,
                    0x0040,
                    0x0040,
                    0x01,
                    0x04,
                    )
                IO (Decode16,
                    0x0050,
                    0x0050,
                    0x10,
                    0x04,
                    )
            })
            Method (_STA, 0, NotSerialized)
            {
                If (_OSI ("Darwin"))
                {
                    Return (0x0F)
                }
                Else
                {
                    Return (0)
                }
            }
        }
    }
}
```

```
//Fix IPIC
DefinitionBlock ("", "SSDT", 2, "OCLT", "IPIC", 0)
{
    External (_SB.PCI0.LPCB, DeviceObj)
    External (_SB.PCI0.LPCB.PIC, DeviceObj)
    //External (_SB.PCI0.LPCB.IPIC, DeviceObj)

    //disable IPIC
    Scope (_SB.PCI0.LPCB.PIC)    // Scope (_SB.PCI0.LPCB.IPIC)
    {
        Method (_STA, 0, NotSerialized)
        {
            If (_OSI ("Darwin"))
            {
                Return (0)
            }
            Else
            {
                Return (0x0F)
            }
        }
    }

    Scope (_SB.PCI0.LPCB)
    {
        //Fake IPIC
        Device (IPI0)
        {
            Name (_HID, EisaId ("PNP0000"))
            Name (_CRS, ResourceTemplate ()
            {
                IO (Decode16,
                    0x0020,
                    0x0020,
                    0x01,
                    0x02,
                    )
                IO (Decode16,
                    0x00A0,
                    0x00A0,
                    0x01,
                    0x02,
                    )
                IO (Decode16,
                    0x04D0,
                    0x04D0,
                    0x01,
                    0x02,
                    )
                // Remove IRQNoFlags
```

```
                /*
                 * IRQNoFlags ()
                 * {2}
                 */
            })
            Method (_STA, 0, NotSerialized)
            {
                If (_OSI ("Darwin"))
                {
                    Return (0x0F)
                }
                Else
                {
                    Return (0)
                }
            }
        }
    }
}
```

CMOS重置补丁

描述

- 某些机器在关机或者重启时会出现 “开机自检错误”，这是由于 CMOS 被重置所导致的。
- 当使用 Clover 时，勾选 `ACPI\FixRTC` 可以解决上述问题。
- 当使用 OpenCore 时，官方提供了以下解决方法，见 ***Sample.plist*** ：
  - 安装 **RTCMemoryFixup.kext**
  - `Kernel\Patch` 补丁：**__ZN11BCM5701Enet14getAdapterInfoEv**
- 本章提供一种 SSDT 补丁方法来解决上述问题。这个 SSDT 补丁本质是仿冒 RTC，见《预置变量法》和《仿冒设备》。

解决方案

- 删除 **RTC** `PNP0B00` 部件 `_CRS` 的 中断号。

```
Device (RTC)
{
    Name (_HID, EisaId ("PNP0B00"))
    Name (_CRS, ResourceTemplate ()
    {
        IO (Decode16,
            0x0070,
            0x0070,
            0x01,
            0x08,      /* 或者0x02,试验确定 */
            )
        IRQNoFlags () /* 删除此行 */
            {8}       /* 删除此行 */
    })
}
```

补丁：SSDT-RTC0-NoFlags

- 禁用原始部件：**RTC**
  - 如果 **RTC** 不存在 `_STA` ，使用下列方法禁用 **RTC**：

```
External(_SB.PCI0.LPCB.RTC, DeviceObj)
Scope (_SB.PCI0.LPCB.RTC)
{
    Method (_STA, 0, NotSerialized)
    {
        If (_OSI ("Darwin"))
```

```
        {
            Return (Zero)
        }
        Else
        {
            Return (0x0F)
        }
    }
}
```

- 如果 **RTC** 存在 `_STA` ，使用预置变量法禁用 **RTC**。示例中的变量是 `STAS` ，使用时应注意 `STAS` 对其他设备、部件的影响。

```
External (STAS, FieldUnitObj)
Scope (\)
{
    If (_OSI ("Darwin"))
    {
        STAS = 2
    }
}
```

- 仿冒 **RTC0** ，参见样本。

注意

- 补丁里的设备名称、路径应和原始 ACPI 一致。
- 如果机器本身因某种原因禁用了 RTC，需仿冒 RTC 才能正常工作。在这种情况下出现了 “开机自检错误”，删除仿冒补丁的中断号即可：

```
IRQNoFlags () /* 删除此行 */
    {8}       /* 删除此行 */
```

感谢 @Chic Cheung、@Noctis 辛苦付出！

```
//
DefinitionBlock ("", "SSDT", 2, "OCLT", "RTCfix", 0)
{
    External(_SB.PCI0.LPCB, DeviceObj)

    /*
    //Example 1;;disable RTC
    External(_SB.PCI0.LPCB.RTC, DeviceObj)
    Scope (_SB.PCI0.LPCB.RTC)
    {
        Method (_STA, 0, NotSerialized)
        {
            If (_OSI ("Darwin"))
            {
                Return (Zero)
            }
            Else
            {
                Return (0x0F)
            }
        }
    }
    */


    //Example 2;;disable RTC
    External (STAS, FieldUnitObj)
    Scope (\)
    {
        If (_OSI ("Darwin"))
        {
            STAS = 2
        }
    }

    //Fake RTC0
    Scope (_SB.PCI0.LPCB)
    {
        Device (RTC0)
        {
            Name (_HID, EisaId ("PNP0B00"))
            Name (_CRS, ResourceTemplate ()
            {
                IO (Decode16,
                    0x0070,
                    0x0070,
                    0x01,
                    0x02,//0x08,
                    )
            })
```

```
            Method (_STA, 0, NotSerialized)
            {
                If (_OSI ("Darwin"))
                {
                    Return (0x0F)
                }
                Else
                {
                    Return (Zero)
                }
            }
        }
    }
}
//EOF
```

常见驱动加载顺序

需要注意事项

- `config-5-PS2Smart键盘驱动列表` 与 `config-2-PS2键盘驱动列表` 二选一，不可同时使用，建议优先选用 config-2-PS2 键盘驱动列表

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/Pro
pertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>Kernel</key>
    <dict>
        <key>Add</key>
        <array>
            <dict>
                <key>Arch</key>
                <string>x86_64</string>
                <key>BundlePath</key>
                <string>Lilu.kext</string>
                <key>Comment</key>
                <string>Lilu</string>
                <key>Enabled</key>
                <true/>
                <key>ExecutablePath</key>
                <string>Contents/MacOS/Lilu</string>
                <key>MaxKernel</key>
                <string></string>
                <key>MinKernel</key>
                <string></string>
                <key>PlistPath</key>
                <string>Contents/Info.plist</string>
            </dict>
            <dict>
                <key>Arch</key>
                <string>x86_64</string>
                <key>BundlePath</key>
                <string>VirtualSMC.kext</string>
                <key>Comment</key>
                <string>SMC</string>
                <key>Enabled</key>
                <true/>
                <key>ExecutablePath</key>
                <string>Contents/MacOS/VirtualSMC</string>
                <key>MaxKernel</key>
                <string></string>
                <key>MinKernel</key>
                <string></string>
                <key>PlistPath</key>
                <string>Contents/Info.plist</string>
            </dict>
            <dict>
                <key>Arch</key>
                <string>x86_64</string>
                <key>BundlePath</key>
                <string>WhateverGreen.kext</string>
```

```
                <key>Comment</key>
                <string>Video card</string>
                <key>Enabled</key>
                <true/>
                <key>ExecutablePath</key>
                <string>Contents/MacOS/WhateverGreen</string>
                <key>MaxKernel</key>
                <string></string>
                <key>MinKernel</key>
                <string></string>
                <key>PlistPath</key>
                <string>Contents/Info.plist</string>
            </dict>
            <dict>
                <key>Arch</key>
                <string>x86_64</string>
                <key>BundlePath</key>
                <string>SMCBatteryManager.kext</string>
                <key>Comment</key>
                <string>Battery</string>
                <key>Enabled</key>
                <true/>
                <key>ExecutablePath</key>
                <string>Contents/MacOS/SMCBatteryManager</string>
                <key>MaxKernel</key>
                <string></string>
                <key>MinKernel</key>
                <string></string>
                <key>PlistPath</key>
                <string>Contents/Info.plist</string>
            </dict>
            <dict>
                <key>Arch</key>
                <string>x86_64</string>
                <key>BundlePath</key>
                <string>SMCLightSensor.kext</string>
                <key>Comment</key>
                <string>Light Sensor</string>
                <key>Enabled</key>
                <true/>
                <key>ExecutablePath</key>
                <string>Contents/MacOS/SMCLightSensor</string>
                <key>MaxKernel</key>
                <string></string>
                <key>MinKernel</key>
                <string></string>
                <key>PlistPath</key>
                <string>Contents/Info.plist</string>
            </dict>
            <dict>
                <key>Arch</key>
```

```
                <string>x86_64</string>
                <key>BundlePath</key>
                <string>SMCProcessor.kext</string>
                <key>Comment</key>
                <string>Prosessor</string>
                <key>Enabled</key>
                <true/>
                <key>ExecutablePath</key>
                <string>Contents/MacOS/SMCProcessor</string>
                <key>MaxKernel</key>
                <string></string>
                <key>MinKernel</key>
                <string></string>
                <key>PlistPath</key>
                <string>Contents/Info.plist</string>
            </dict>
            <dict>
                <key>Arch</key>
                <string>x86_64</string>
                <key>BundlePath</key>
                <string>SMCSuperIO.kext</string>
                <key>Comment</key>
                <string>SuperIO</string>
                <key>Enabled</key>
                <true/>
                <key>ExecutablePath</key>
                <string>Contents/MacOS/SMCSuperIO</string>
                <key>MaxKernel</key>
                <string></string>
                <key>MinKernel</key>
                <string></string>
                <key>PlistPath</key>
                <string>Contents/Info.plist</string>
            </dict>
            <dict>
                <key>Arch</key>
                <string>x86_64</string>
                <key>BundlePath</key>
                <string>AppleALC.kext</string>
                <key>Comment</key>
                <string>Sound</string>
                <key>Enabled</key>
                <true/>
                <key>ExecutablePath</key>
                <string>Contents/MacOS/AppleALC</string>
                <key>MaxKernel</key>
                <string></string>
                <key>MinKernel</key>
                <string></string>
                <key>PlistPath</key>
                <string>Contents/Info.plist</string>
```

```
            </dict>
        </array>
    </dict>
</dict>
</plist>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/Pro
pertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>Kernel</key>
    <dict>
        <key>Add</key>
        <array>
            <dict>
                <key>Arch</key>
                <string>x86_64</string>
                <key>BundlePath</key>
                <string>VoodooPS2Controller.kext</string>
                <key>Comment</key>
                <string>PS2Controller</string>
                <key>Enabled</key>
                <true/>
                <key>ExecutablePath</key>
                <string>Contents/MacOS/VoodooPS2Controller</string>
                <key>MaxKernel</key>
                <string></string>
                <key>MinKernel</key>
                <string></string>
                <key>PlistPath</key>
                <string>Contents/Info.plist</string>
            </dict>
            <dict>
                <key>Arch</key>
                <string>x86_64</string>
                <key>BundlePath</key>
                <string>VoodooPS2Controller.kext/Contents/PlugIns/VoodooInput.kext<
/string>
                <key>Comment</key>
                <string>VoodooInput</string>
                <key>Enabled</key>
                <true/>
                <key>ExecutablePath</key>
                <string>Contents/MacOS/VoodooInput</string>
                <key>MaxKernel</key>
                <string></string>
                <key>MinKernel</key>
                <string></string>
                <key>PlistPath</key>
                <string>Contents/Info.plist</string>
            </dict>
            <dict>
                <key>Arch</key>
                <string>x86_64</string>
                <key>BundlePath</key>
```

```
                <string>VoodooPS2Controller.kext/Contents/PlugIns/VoodooPS2Keyboard
.kext</string>
                <key>Comment</key>
                <string>Keyboard</string>
                <key>Enabled</key>
                <true/>
                <key>ExecutablePath</key>
                <string>Contents/MacOS/VoodooPS2Keyboard</string>
                <key>MaxKernel</key>
                <string></string>
                <key>MinKernel</key>
                <string></string>
                <key>PlistPath</key>
                <string>Contents/Info.plist</string>
            </dict>
            <dict>
                <key>Arch</key>
                <string>x86_64</string>
                <key>BundlePath</key>
                <string>VoodooPS2Controller.kext/Contents/PlugIns/VoodooPS2Mouse.ke
xt</string>
                <key>Comment</key>
                <string>Mouse</string>
                <key>Enabled</key>
                <true/>
                <key>ExecutablePath</key>
                <string>Contents/MacOS/VoodooPS2Mouse</string>
                <key>MaxKernel</key>
                <string></string>
                <key>MinKernel</key>
                <string></string>
                <key>PlistPath</key>
                <string>Contents/Info.plist</string>
            </dict>
            <dict>
                <key>Arch</key>
                <string>x86_64</string>
                <key>BundlePath</key>
                <string>VoodooPS2Controller.kext/Contents/PlugIns/VoodooPS2Trackpad
.kext</string>
                <key>Comment</key>
                <string>Trackpad</string>
                <key>Enabled</key>
                <true/>
                <key>ExecutablePath</key>
                <string>Contents/MacOS/VoodooPS2Trackpad</string>
                <key>MaxKernel</key>
                <string></string>
                <key>MinKernel</key>
                <string></string>
                <key>PlistPath</key>
```

```
                <string>Contents/Info.plist</string>
            </dict>
            <dict>
                <key>Arch</key>
                <string>x86_64</string>
                <key>BundlePath</key>
                <string>BrightnessKeys.kext</string>
                <key>Comment</key>
                <string></string>
                <key>Enabled</key>
                <true/>
                <key>ExecutablePath</key>
                <string>Contents/MacOS/BrightnessKeys</string>
                <key>MaxKernel</key>
                <string></string>
                <key>MinKernel</key>
                <string></string>
                <key>PlistPath</key>
                <string>Contents/Info.plist</string>
            </dict>
        </array>
    </dict>
</dict>
</plist>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/Pro
pertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>Kernel</key>
    <dict>
        <key>Add</key>
        <array>
            <dict>
                <key>Arch</key>
                <string>x86_64</string>
                <key>BundlePath</key>
                <string>AirportBrcmFixup.kext</string>
                <key>Comment</key>
                <string>WIFI</string>
                <key>Enabled</key>
                <true/>
                <key>ExecutablePath</key>
                <string>Contents/MacOS/AirportBrcmFixup</string>
                <key>MaxKernel</key>
                <string></string>
                <key>MinKernel</key>
                <string></string>
                <key>PlistPath</key>
                <string>Contents/Info.plist</string>
            </dict>
            <dict>
                <key>Arch</key>
                <string>x86_64</string>
                <key>BundlePath</key>
                <string>BrcmBluetoothInjector.kext</string>
                <key>Comment</key>
                <string>BrcmBluetooth</string>
                <key>Enabled</key>
                <true/>
                <key>ExecutablePath</key>
                <string></string>
                <key>MaxKernel</key>
                <string></string>
                <key>MinKernel</key>
                <string></string>
                <key>PlistPath</key>
                <string>Contents/Info.plist</string>
            </dict>
            <dict>
                <key>Arch</key>
                <string>x86_64</string>
                <key>BundlePath</key>
                <string>BrcmFirmwareData.kext</string>
```

```
                <key>Comment</key>
                <string>BrcmBluetooth</string>
                <key>Enabled</key>
                <true/>
                <key>ExecutablePath</key>
                <string>Contents/MacOS/BrcmFirmwareData</string>
                <key>MaxKernel</key>
                <string></string>
                <key>MinKernel</key>
                <string></string>
                <key>PlistPath</key>
                <string>Contents/Info.plist</string>
            </dict>
            <dict>
                <key>Arch</key>
                <string>x86_64</string>
                <key>BundlePath</key>
                <string>BrcmPatchRAM3.kext</string>
                <key>Comment</key>
                <string>BrcmBluetooth</string>
                <key>Enabled</key>
                <true/>
                <key>ExecutablePath</key>
                <string>Contents/MacOS/BrcmPatchRAM3</string>
                <key>MaxKernel</key>
                <string></string>
                <key>MinKernel</key>
                <string></string>
                <key>PlistPath</key>
                <string>Contents/Info.plist</string>
            </dict>
        </array>
    </dict>
</dict>
</plist>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/Pro
pertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>Kernel</key>
    <dict>
        <key>Add</key>
        <array>
            <dict>
                <key>Arch</key>
                <string>x86_64</string>
                <key>BundlePath</key>
                <string>VoodooI2C.kext/Contents/PlugIns/VoodooI2CServices.kext</str
ing>
                <key>Comment</key>
                <string>I2CServices</string>
                <key>Enabled</key>
                <true/>
                <key>ExecutablePath</key>
                <string>Contents/MacOS/VoodooI2CServices</string>
                <key>MaxKernel</key>
                <string></string>
                <key>MinKernel</key>
                <string></string>
                <key>PlistPath</key>
                <string>Contents/Info.plist</string>
            </dict>
            <dict>
                <key>Arch</key>
                <string>x86_64</string>
                <key>BundlePath</key>
                <string>VoodooI2C.kext/Contents/PlugIns/VoodooGPIO.kext</string>
                <key>Comment</key>
                <string>GPIO</string>
                <key>Enabled</key>
                <true/>
                <key>ExecutablePath</key>
                <string>Contents/MacOS/VoodooGPIO</string>
                <key>MaxKernel</key>
                <string></string>
                <key>MinKernel</key>
                <string></string>
                <key>PlistPath</key>
                <string>Contents/Info.plist</string>
            </dict>
            <dict>
                <key>Arch</key>
                <string>x86_64</string>
                <key>BundlePath</key>
```

```
                <string>VoodooI2C.kext/Contents/PlugIns/VoodooInput.kext</string>
                <key>Comment</key>
                <string>MT2</string>
                <key>Enabled</key>
                <true/>
                <key>ExecutablePath</key>
                <string>Contents/MacOS/VoodooInput</string>
                <key>MaxKernel</key>
                <string></string>
                <key>MinKernel</key>
                <string></string>
                <key>PlistPath</key>
                <string>Contents/Info.plist</string>
            </dict>
            <dict>
                <key>Arch</key>
                <string>x86_64</string>
                <key>BundlePath</key>
                <string>VoodooI2C.kext</string>
                <key>Comment</key>
                <string>I2C</string>
                <key>Enabled</key>
                <true/>
                <key>ExecutablePath</key>
                <string>Contents/MacOS/VoodooI2C</string>
                <key>MaxKernel</key>
                <string></string>
                <key>MinKernel</key>
                <string></string>
                <key>PlistPath</key>
                <string>Contents/Info.plist</string>
            </dict>
            <dict>
                <key>Arch</key>
                <string>x86_64</string>
                <key>BundlePath</key>
                <string>VoodooI2CHID.kext</string>
                <key>Comment</key>
                <string>I2CHID</string>
                <key>Enabled</key>
                <true/>
                <key>ExecutablePath</key>
                <string>Contents/MacOS/VoodooI2CHID</string>
                <key>MaxKernel</key>
                <string></string>
                <key>MinKernel</key>
                <string></string>
                <key>PlistPath</key>
                <string>Contents/Info.plist</string>
            </dict>
        </array>
```

```
    </dict>
</dict>
</plist>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/Pro
pertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>Kernel</key>
    <dict>
        <key>Add</key>
        <array>
            <dict>
                <key>Arch</key>
                <string>x86_64</string>
                <key>BundlePath</key>
                <string>ApplePS2SmartTouchPad.kext/Contents/PlugIns/ApplePS2Control
ler.kext</string>
                <key>Comment</key>
                <string>PS2Controller</string>
                <key>Enabled</key>
                <true/>
                <key>ExecutablePath</key>
                <string>Contents/MacOS/ApplePS2Controller</string>
                <key>MaxKernel</key>
                <string></string>
                <key>MinKernel</key>
                <string></string>
                <key>PlistPath</key>
                <string>Contents/Info.plist</string>
            </dict>
            <dict>
                <key>Arch</key>
                <string>x86_64</string>
                <key>BundlePath</key>
                <string>ApplePS2SmartTouchPad.kext/Contents/PlugIns/ApplePS2Keyboar
d.kext</string>
                <key>Comment</key>
                <string>PS2Keyboard</string>
                <key>Enabled</key>
                <true/>
                <key>ExecutablePath</key>
                <string>Contents/MacOS/ApplePS2Keyboard</string>
                <key>MaxKernel</key>
                <string></string>
                <key>MinKernel</key>
                <string></string>
                <key>PlistPath</key>
                <string>Contents/Info.plist</string>
            </dict>
            <dict>
                <key>Arch</key>
                <string>x86_64</string>
```

```
                <key>BundlePath</key>
                <string>ApplePS2SmartTouchPad.kext</string>
                <key>Comment</key>
                <string>PS2SmartTouchPad</string>
                <key>Enabled</key>
                <true/>
                <key>ExecutablePath</key>
                <string>Contents/MacOS/ApplePS2SmartTouchPad</string>
                <key>MaxKernel</key>
                <string></string>
                <key>MinKernel</key>
                <string></string>
                <key>PlistPath</key>
                <string>Contents/Info.plist</string>
            </dict>
        </array>
    </dict>
</dict>
</plist>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/Pro
pertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>Kernel</key>
    <dict>
        <key>Add</key>
        <array>
            <dict>
                <key>Arch</key>
                <string>x86_64</string>
                <key>BundlePath</key>
                <string>AirportItlwm.kext</string>
                <key>Comment</key>
                <string>intelWIFI</string>
                <key>Enabled</key>
                <true/>
                <key>ExecutablePath</key>
                <string>Contents/MacOS/AirportItlwm</string>
                <key>MaxKernel</key>
                <string></string>
                <key>MinKernel</key>
                <string></string>
                <key>PlistPath</key>
                <string>Contents/Info.plist</string>
            </dict>
            <dict>
                <key>Arch</key>
                <string>x86_64</string>
                <key>BundlePath</key>
                <string>IntelBluetoothInjector.kext</string>
                <key>Comment</key>
                <string>IntelBluetoothInjector</string>
                <key>Enabled</key>
                <true/>
                <key>ExecutablePath</key>
                <string>Contents/MacOS/IntelBluetoothInjector</string>
                <key>MaxKernel</key>
                <string></string>
                <key>MinKernel</key>
                <string></string>
                <key>PlistPath</key>
                <string>Contents/Info.plist</string>
            </dict>
            <dict>
                <key>Arch</key>
                <string>x86_64</string>
                <key>BundlePath</key>
                <string>IntelBluetoothFirmware.kext</string>
```

```
                <key>Comment</key>
                <string>IntelBluetoothFirmware</string>
                <key>Enabled</key>
                <true/>
                <key>ExecutablePath</key>
                <string>Contents/MacOS/IntelBluetoothFirmware</string>
                <key>MaxKernel</key>
                <string></string>
                <key>MinKernel</key>
                <string></string>
                <key>PlistPath</key>
                <string>Contents/Info.plist</string>
            </dict>
        </array>
    </dict>
</dict>
</plist>
```