屏蔽双显卡笔记本的独显

本贴是对Tonymacx86的RehabMan大神贴子的授权翻译。在远景发的时候，总出现贴子内容丢失。在Austere.J的帮助下，发到他的博客。感谢。:-)

原贴地址，Credit to RehabMan：http://www.tonymacx86.com/yosemite-laptop-support/163772-guide-disabling-discrete-graphics-dual-gpu-laptops.html

翻译者：daxuexinsheng

博客版地址（感谢Austere.J提供平台）：http://www.firewolf.science/2015/05/屏蔽双显卡笔记本的独显/

概述

这个教程的目的，是向大家展示，怎样通过修改DSDT和SSDT，来屏蔽双显卡笔记本的独立显卡。（例如：Intel集成显卡 + 英伟达独立显卡[Optimus技术]，还有Intel集成显卡 + Radeon独立显卡）。

因为在黑苹果下，双显卡笔记本只能驱动英特尔的集成显卡，而独显如果不做任何处理，虽然它不会工作，但是，一般情况下，独显还是会处于激活状态，并且消耗电力，产生热量，造成风扇噪音，和电池电量的快速消耗。虽然我们可以在BIOS里关闭独显，但是，通过修改ACPI文件来屏蔽是更好的选择，因为这样屏蔽的独显，是不会影响到Windows的。（如果用BIOS屏蔽，那么当你想进Windows玩游戏的时候，就要先进BIOS再开启独显）。

虽然看起来，屏蔽独显的补丁很简单（RehabMan提供的补丁）（有时只需要一行代码），但是，由于我们需要修改1个或多个SSDT，所以问题实际上会比较复杂，也因此会存在着许多陷阱。还有，对于某些电脑和ACPI文件，也有着不同的处理方法，就使得这个问题更为复杂了。本贴提供的DSDT/SSDT，就是一种比较复杂的情况。所以，本贴关于对这些文件的处理，也将会涵盖大部分你将来会遇到的情况。

你应该下载例子文件，按照介绍自己做一遍，以充分理解贴子。再来改你自己的文件。

作为例子的文件，是 华硕 G53SW（Intel HD4600+Nvidia），通过在Clover的启动界面按 F4键，提取的文件（在ACPI/origin目录）。

基本概念

我们的目标非常简单。通常，在SSDT里，笔记本给我们提供了一个 _OFF 方法，我们可以通过调用这个方法，来切段独显的供电。最最简单的方法，就是在 相应的_INI方法里，调用_OFF方法。需要注意，这个_OFF方法，还可能会在DSDT里，或者可能会有不同的名字（如：GPOF、OPOF、_PS3，等等）。

某些_OFF方法的实现，会由于它包含了对EC(Embedded Controller)的依赖，而使得它不能在_INI方法里被调用。对于这样的情况，整个_OFF方法或者它的一部分代码，需要被移动到_REG方法里，以延迟执行（当_REG方法接收的参数Arg0==3 且 Arg1==1时，它会在_INI方法之后被执行）（详见ACPI规范）。对于一些情况，在_REG方法里调用_OFF的时机太迟了，从而导致要么屏蔽独显失败，要么系统五国。对于这样的情况，修改_OFF方法，移除它对于EC的依赖，将变得必要。之后，我们就可以在_INI里调用它（移除了对EC的依赖的_OFF）。同时，在_OFF里移除的代码，需要加到_REG里去。这样，虽然EC关联的代码在后（_INI后）执行（因为代码加到了_REG里，所以后执行），但却能达到更好的效果。贴子提供的例子，就是这种情况。

基本的打补丁操作

明白怎么提取ACPI文件，并反编译它们，再给它们打补丁，修改好后放到哪里，是非常重要的。关于这些，请到这个贴子学习：http://bbs.pcbeta.com/viewthread-1571455-1-1.html

你需要在实际操作之前，先熟悉好这些。

还可以看一下楼主的视频教程（由于视频教程录的较早，所以操作与本贴稍有不同。以本贴内容为准，视频作为对如何修改ACPI文件的一个展示）：http://bbs.pcbeta.com/viewthread-1569867-1-1.html

因为基本的知识非常重要，所以我们需要在实际操作屏蔽独显之前，先了解下这些。

首先，对于提取好的文件（你可以下载贴子提供的例子进行练习）。用iasl同时反编译所有文件。
iasl -da -dl *.aml
反编译好之后，我们会得到所有dsl文件，接下来，开始对dsl文件打补丁，对于我们的例子文件，需要打的补丁的情况：

DSDT.dsl:
"Fix PARSEOP_ZERO Error" (先删除"more agressive" 的注释（就是删除补丁最后两行前的井号）)
"Fix ADBG Error"
"Remove _DSM methods"
"IRQ Fix"
"SMBUS Fix"
"OS Check Fix (Windows 8)"
"Add IMEI"
"RTC Fix"
"Fix _WAK Arg0 v2"
"Fix _WAK IAOE" (RehabMan新做的补丁)
"Rename GFX0 to IGPU"
"Rename B0D3 to HDAU" (新补丁)
"ASUS N55SL/VivoBook"
"USB3 _PRW(0x6D) and Rename XHC to XHC1" (新补丁)
"Audio Layout 12"

SSDT-0.dsl:
"Remove _PSS placeholders" (新补丁)

不需要修改SSDT-2x, SSDT-3x, SSDT-4x，因为这几个是动态加载的（也就是每次提取都会不同）
而需要对SSDT-1, SSDT-5, SSDT-6, SSDT-8打补丁

SSDT-7.dsl:
"Rename GFX0 to IGPU"

SSDT-9.dsl
"Rename GFX0 to IGPU"
"Brightness Fix (Haswell)"
"Rename B0D3 to HDAU" (新补丁)

SSDT-10.dsl:
"Rename GFX0 to IGPU"

SSDT-11.dsl
"Remove _DSM methods"
"Rename GFX0 to IGPU"
"Cleanup/Fix Errors (SSDT)"

SSDT-12.dsl:
"Rename GFX0 to IGPU”

如果顺利，修改好后，所有文件都可以被顺利编译。

你可以用iasl编译试试
iasl *.dsl
注意：要先备份好所有的原始aml文件，因为编译会产生新的aml文件，而覆盖原始文件。

关于上面选择补丁的注意事项：
- B0D3 (名字可能不一样) 可以用这个命令查找包含它的文件 'grep -B3 _ADR.*0x00030000 *.dsl'
- GFX0 (名字可能不一样) 可以用这个命令查找包含它的文件 'grep -B3 _ADR.*0x00020000 *.dsl'
- "Rename GFX0 to IGPU" 需要给所用引用到GFX0的文件打。需要"balancing renames”（也就是有GFX0的文件都要打），可以用这个命令查找包含它的文件 'grep -l GFX0 *.dsl'
- 和前面的说明一样，”Rename B0D3 to HDAU". 用这个命令查找包含它的文件 'grep -l B0D3 *.dsl'
- "Remove _DSM methods" 需要在一开始就打，（对包含_DSM methods的文件） (可以用这个命令查找包含它的文件 'grep -l Method.*_DSM *.dsl'
- 以前的贴子，我们提倡删除所有 CPU 相关的 SSDT，但现在，我们需要保留它们(如果有错误，就打相应补丁)。使用原始的CPU相关SSDT，就可以不再给DSDT打"Fix PNOT/PPNT”补丁。
- 电量显示补丁是跟具体机型相关的，这里的例子是华硕的G53SW。

打补丁屏蔽独显
还记得我们的目标吗？在_INI方法里调用_OFF方法。为了这个目的，我们需要对一系列之前得到的DSDT和SSDT打补丁，以修复编译错误、完整地对一些名字进行改名、移除所有_DSM方法。

那么，怎么找出包含了_OFF方法的SSDT呢？我们可以用grep命令做到：
grep -l Method.*_OFF *.dsl
对于我们给的文件，会显示如下结果：
SSDT-10.dsl
SSDT-11.dsl
当然也能用这个命令找_INI方法：
grep -l Method.*_INI *.dsl
结果：
DSDT.dsl
SSDT-10.dsl
SSDT-11.dsl
SSDT-9.dsl
注意：SSDT-10 和 SSDT-11出现了两次。我们要找的_OFF方法和与之相关的_INI方法，很可能就在这两个文件里。

当然我们可以用MaciASL一个一个打开文件搜索来找_OFF和_INI，但用grep命令显然更方便、更快。

打开SSDT-10.dsl，搜索"Method (_INI”，可以找到：
```
Method (_INI, 0, NotSerialized)  // _INI: Initialize
    {
        Store (Zero, \_SB.PCI0.RP05.PEGP._ADR)
    }
```
这是一个典型的独显_INI方法（INI是initial的缩写，即为初始化方法），也是我们要在里面调用_OFF的方法。

如果我们把光标点到方法体里（编程术语，也就是这个方法的大括号范围里。可以简单地把光标点到方法的名字那里），我们就可以看到这个方法所处的ACPI路径。MaciASL会在左下角显示这个路径，我们的例子是：SSDT -> _SB.PCI0.RP05.PEGP -> _INI。这样，可以推测出_OFF方法的路径，应该是_SB.PCI0.RP05.PEGP._OFF。

到现在，我们可以推知，_OFF方法就在SSDT-10.dsl 或者 SSDT-11.dsl里。如果你打开SSDT-10.dsl搜索_OFF，可以找到一个定义在PowerResource宏里的方法。这样就不是我们要找的_OFF方法

了。接着打开SSDT-11.dsl搜索，你可以找到一个常规的（比较标准的）_OFF方法，它正是我们想找的方法。现在，我们找到_OFF在哪儿了。我们需要检查_OFF的代码，看看有没有访问到EC。

SSDT-11.dsl里的_OFF定义：

```
Method (_OFF, 0, Serialized)  // _OFF: Power Off
    {
        If (LEqual (CTXT, Zero))
        {
            \_SB.PCI0.LPCB.EC0.SPIN (0x96, Zero)
            If (LNotEqual (GPRF, One))
            {
                Store (VGAR, VGAB)
            }

            Store (One, CTXT)
        }

        SGOF ()
    }
```

通过检查，我们可以看到里面访问到了EC（_SB.PCI0.LPCB.EC0.SPIN (0x96, Zero)。这里访问EC的代码，将会造成一些问题，它（们）将在_INI调用_OFF时，阻止代码的完全执行。好好注意这点，有时候这样的代码可能不好看出来。有的情况的代码，不会直接访问EC，而是调用EC里定义的方法（属于间接访问））。所以，对于有的电脑的情况，你需要深入检查代码。而我们的例子，是直接给出了EC0这样的字眼。

最好的处理EC访问的方法，是移除_OFF里那些令人讨厌的代码（哈哈，英语说的真直接）。

我们可以手工删除，也可以用补丁自动删除：

```
into method label _OFF parent_label \_SB.PCI0.RP05.PEGP code_regex .*EC.*
removeall_matched;
```

The modified _OFF method is this:

Code:

```
    Method (_OFF, 0, Serialized)  // _OFF: Power Off
    {
        If (LEqual (CTXT, Zero))
        {
            If (LNotEqual (GPRF, One))
            {
                Store (VGAR, VGAB)
            }

            Store (One, CTXT)
        }

        SGOF ()
    }
```

（楼主提醒：这个补丁和型号有很强相关性，可以说是针对这个型号的。其它电脑需要重新做补丁，或者还是手工删除吧。）

我们需要在其它地方保存好删除了的代码，因为我们还要在_REG用到这些删除的代码。（我们的例子是：_SB.PCI0.LPCB.EC0.SPIN (0x96, Zero)）。

现在，我们修复好了_OFF方法。那么让我们去SSDT-10.dsl里的_INI调用_OFF吧。

我们可以用现成的补丁："Disable from _INI (SSDT)"。但是，补丁里的访问路径是一般的情况。而我们的例子比较特殊，所以我们需要改一下补丁。另外，我们要调用的_OFF定义在SSDT-11.dsl里（也就是在SSDT-10.dsl的外面），所以需要用External声明，来告诉编译器这个方法在外面。

修改了的补丁：（注意自己的路径，需要根据实际，改parent_label和External的路径）

```
into method label _INI parent_label \_SB.PCI0.RP05.PEGP insert
begin
//added to turn nvidia/radeon off\n
External(\_SB.PCI0.RP05.PEGP._OFF, MethodObj)\n
_OFF()\n
end;
```

打好补丁后，现在的_INI方法变成了这样：

```
Method (_INI, 0, NotSerialized)  // _INI: Initialize
    {
        Store (Zero, \_SB.PCI0.RP05.PEGP._ADR)
        //added to turn nvidia/radeon off
        External(\_SB.PCI0.RP05.PEGP._OFF, MethodObj)
        _OFF()
    }
```

现在，我们需要把目光转向DSDT里的_REG了。_REG方法需要代替之前的_OFF所做的EC的工作。

下面是原始的_REG方法：

```
Method (_REG, 2, NotSerialized)  // _REG: Region Availability
        {
            If (LEqual (Arg0, 0x03))
            {
                Store (Arg1, ECFL)
            }
        }
```

RehabMan的补丁源里，有用来在_REG调用_OFF的补丁。我们可以基于这个补丁做下修改：

```
into method label _REG parent_hid PNP0C09 insert
begin
//added to turn nvidia/radeon off\n
If (LAnd(LEqual(Arg0,3),LEqual(Arg1,1)))\n
{\n
   External(\_SB.PCI0.PEG0.PEGP._OFF, MethodObj)\n
   \_SB.PCI0.PEG0.PEGP._OFF()\n
}\n
end;
```

我们需要把补丁里的_OFF，改成SPIN。（根据实际情况，换成之前找到的EC访问代码）

修改后的补丁：

```
into method label _REG parent_hid PNP0C09 insert
begin
//added to turn nvidia/radeon off\n
If (LAnd(LEqual(Arg0,3),LEqual(Arg1,1)))\n
{\n
   \_SB.PCI0.LPCB.EC0.SPIN (0x96, Zero)\n
}\n
end;
```

打过补丁后的_REG方法变成了这样：

```
Method (_REG, 2, NotSerialized)  // _REG: Region Availability
```

```
    {
        If (LEqual (Arg0, 0x03))
        {
            Store (Arg1, ECFL)
        }
        //added to turn nvidia/radeon off
        If (LAnd(LEqual(Arg0,3),LEqual(Arg1,1)))
        {
            \_SB.PCI0.LPCB.EC0.SPIN (0x96, Zero)
        }
    }
```
（我们要根据情况改的，是If (LAnd(LEqual(Arg0,3),LEqual(Arg1,1)))里的代码）

做好全部步骤后，如果顺利，我们就可以一起编译全部改好的文件，再把它们放到引导读取的文件夹，让引导加载这些文件，来查看效果了。

注意：我们的例子文件是属于比较复杂的，并不是所有笔记本的情况都一样。大部分情况下，和独显相关的_INI方法和_OFF方法，是在同一个SSDT里的。对于那样的情况，我们就不需要使用External声明了（用于告诉编译器，某个对象在本文件的外面）。你可以直接打这个补丁："Call _OFF from _INI (SSDT)"。另外还有的情况是，独显相关的_INI和_OFF在DSDT里（这种情况补丁也可以直接打）。还有就是，不是所有电脑的_OFF都会访问EC，这样的情况，就不存在要把访问EC的代码移到_REG了。

请注意：所有的文件，需要根据使用的引导（变色龙或者Clover），放到正确的地方，引导才能读取和加载它们。在另一个贴子，有对这个问题的详细介绍：http://bbs.pcbeta.com/viewthread-1571455-1-1.html

在本贴只是简单地提一下，你必须屏蔽所有的OEM SSDT，以加载自己修改的SSDT。（变色龙：DropSSDT=Yes,，Clover: DropOem=true）。

睡眠/唤醒问题

在按本贴的方法屏蔽独显后，有的笔记本会出现睡眠/唤醒问题，甚至是关机/重启问题。例如惠普的ProBook（Radeon的机型）。解决办法是：在睡眠前开启独显，并且在唤醒的时候再次屏蔽独显。

RehabMan的补丁源里有实现这样操作的补丁。补丁名字是"Disable/Enable on _WAK/_PTS (DSDT)"。

但是，由于我们的例子的_OFF/_ON的路径比较特殊（补丁的是一般情况），所以我们需要改一下补丁。

补丁源的原始补丁：
```
into method label _PTS code_regex ([\s\S]*) replace_matched
begin
External(\\_SB.PCI0.PEG0.PEGP._ON, MethodObj)\n
If (CondRefOf(\\_SB.PCI0.PEG0.PEGP._ON)) { \\_SB.PCI0.PEG0.PEGP._ON() }\n
%1
end;

into method label _WAK code_regex (Return\s+\(.*) replace_matched
begin
External(\\_SB.PCI0.PEG0.PEGP._OFF, MethodObj)\n
If (CondRefOf(\\_SB.PCI0.PEG0.PEGP._OFF)) { \\_SB.PCI0.PEG0.PEGP._OFF() }\n
%1
end;
```

修改了的补丁：（把PEG0改成了RP05，PEG0是一般情况）
into method label _PTS code_regex ([\s\S]*) replace_matched
begin
External(_SB.PCI0.RP05.PEGP._ON, MethodObj)\n
If (CondRefOf(_SB.PCI0.RP05.PEGP._ON)) { _SB.PCI0.RP05.PEGP._ON() }\n
%1
end;

into method label _WAK code_regex (Return\s+\(.*) replace_matched
begin
External(_SB.PCI0.RP05.PEGP._OFF, MethodObj)\n
If (CondRefOf(_SB.PCI0.RP05.PEGP._OFF)) { _SB.PCI0.RP05.PEGP._OFF() }\n
%1
end;

如果你的笔记本屏蔽独显后，没有睡眠/唤醒/重启/关机的问题，那么你就不需要上面的补丁。你需要先测试下有没有这些问题，再决定做不做上面的修改。

问题反馈

如果你遇到了关于DSDT/SSDT打补丁的问题，或者是修复错误的问题，请提供所有你的电脑的原始文件。
如果你按本贴方法进行屏蔽独显，但是没有成功，请提供下面提到的用于拍错的数据：

下载patchmatic：https://bitbucket.org/RehabMan/os-x-...-2015-0107.zip
解压patchmatic命令行工具，把它拷贝到 /usr/bin 目录。

打开终端，依次输入：
rm -R ~/Downloads/RehabMan
mkdir ~/Downloads/RehabMan
cd ~/Downloads/RehabMan
patchmatic -extract
把得到的文件压缩好（Zip）上传。

另外，还需要上传ioreg：[Guide] How to Make a Copy of IOReg http://www.tonymacx86.com/audio/58368-guide-how-make-copy-ioreg.html 。请用贴子里提供的 IORegistryExplorer v2.1 版本。千万不要用其它版本的IORegistryExplorer。

在终端依次输入：
kextstat|grep -y acpiplat
kextstat|grep -y appleintelcpu

如果你用的是Clover引导的，请再上传 EFI/Clover 文件夹。如果你用的是变色龙或者Chimera引导的，请再上传 /Extra 文件夹。（你可以删除里面的主题文件，来减小需要上传的文件体积）。

所有数据都需要 压缩。（Zip格式）

楼主现在还没有查看这些数据的能力，所以如果遇到比较难解决的问题的话，请到原贴向RehabMan大神求助。

本贴例子文件：