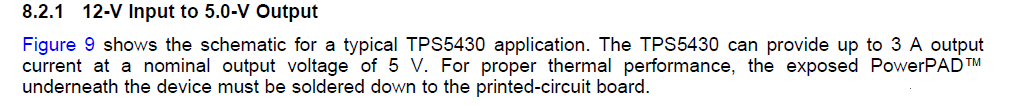
# 开发问题记录与解决

## 2016/11/17

1：焊接好TPS5430后，发现输出一直在跳动，不能稳定

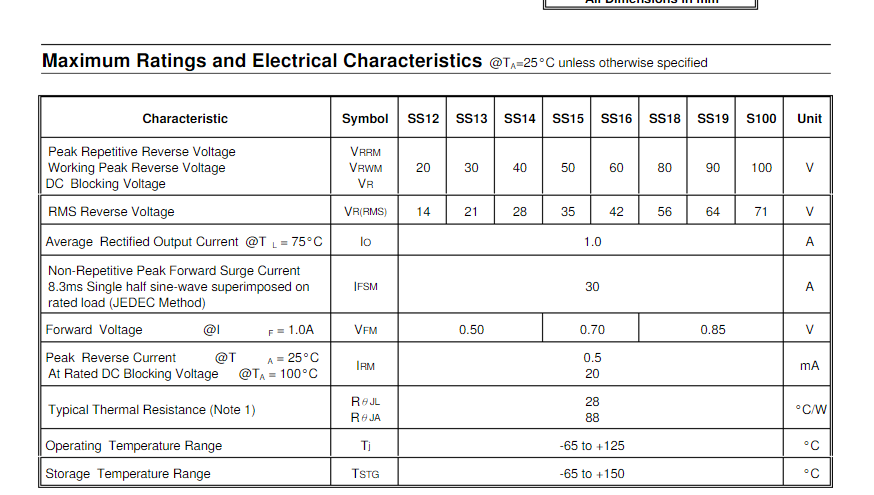
问题解决：原来是TPS的器间底部的散热板类似的焊盘需要焊接到底上。

在数据手册中有：下图中最后一句话， must be soldered down to the printed-circuit board



## 2016/11/18

1：肖特基二极管的各种型号说明：



## 2016/12/5

1:利用 OM13056 下载和调试OBD simulator 和 OBD-II 板子时，注意ISP1 和 ISP2 都不需要接跳帽就可以。

2：

## 2016/12/14

1:发现建立在LPCXpresso 中建立工程时，需要建立**LPCOpen – C++ Project,**并且项目一定要**包含chip library** 和 **board\_library** 。**否则**就会出现**调试不了**的情况。但是可以建立一个C++ Project，可以不用包含任何库文件，但是这样的话就需要我们自己去写库文件。

问题解决：发**现这些问题是由于在另外一个项目中设了断点**，然后**导致了gdb 调试器崩溃**的问题，而不是其他的问题，但是**对于LPC-C++ Project** 却**没有该问题**。也就是说**不能够设置多个断点**，不能**在其他项目中**设置**断点过多**，否则会导致调试本项目失败。gdb 崩溃。

## 2016/12/22

1:在建立 led.h 和 led.cpp 时，利用单例模式，发现在返回一旦添加该函数后，就会在链接阶段出现错误，

c:/nxp/lpcxpresso\_8.2.2\_650/lpcxpresso/tools/bin/../lib/gcc/arm-none-eabi/5.4.1/../../../../arm-none-eabi/lib/armv7-m\libc.a(lib\_a-sbrkr.o): In function `\_sbrk\_r':

sbrkr.c:(.text.\_sbrk\_r+0xc): undefined reference to `\_sbrk'

c:/nxp/lpcxpresso\_8.2.2\_650/lpcxpresso/tools/bin/../lib/gcc/arm-none-eabi/5.4.1/../../../../arm-none-eabi/lib/armv7-m\libc.a(lib\_a-abort.o): In function `abort':

abort.c:(.text.abort+0xa): undefined reference to `\_exit'

c:/nxp/lpcxpresso\_8.2.2\_650/lpcxpresso/tools/bin/../lib/gcc/arm-none-eabi/5.4.1/../../../../arm-none-eabi/lib/armv7-m\libc.a(lib\_a-signalr.o): In function `\_kill\_r':

signalr.c:(.text.\_kill\_r+0x10): undefined reference to `\_kill'

c:/nxp/lpcxpresso\_8.2.2\_650/lpcxpresso/tools/bin/../lib/gcc/arm-none-eabi/5.4.1/../../../../arm-none-eabi/lib/armv7-m\libc.a(lib\_a-signalr.o): In function `\_getpid\_r':

signalr.c:(.text.\_getpid\_r+0x0): undefined reference to `\_getpid'

collect2.exe: error: ld returned 1 exit status

链接器输出的信息如上。

问题解决：经过查找对比发现是在项目属性 **C/C++ build -> Settings -> MCU C++ Linker -> Managed Linker Scriipt -> Library** : 把 **Newlib(none) 改成 Newlib（nohost）**。可能是之前不小心改动的。

## 2016/12/26

1：对于RITINT的写 1 清零的问题发现

在对寄存器进行读写时，一般如果要设置某些位，会用一个只有对应位为 1 的数或上寄存器本来的数值，然后写入寄存器，要清除某些位，则是把对应位设为 0 ，与上该数，写入寄存器。**可是对于写 1 置位的某些位来说**，不可以！因为该位本来为 1，对出来也为 1，最后或上该数确实该位还为 1，可是写入的时候，由于该位写入的还是原来的值，但是却被清零，导致状态改变了，所以对于写 1 清零的某些位，需要特殊处理，在置位其他位时，写入的时候始终要把该位变为零，最后写进去。比如：

**0xf1 是寄存器的值**，**或上**了 **0x01**,值还是 **0xf1** ，写入的也是 0xf1 ，貌似值不变，但是由于最后一位是写 1 清零的，导致了寄存器值变成了 0xf0；所以出现问题。这里就需要对最后一位进行特别处理，把或之后得到的值 0xf1 再与上 0xfe ,最后得到 0xf0 在写入。

总结：对于寄存器中存在写1清零的位：如果该特殊位为1 ，无论是或上零（置位其他位）还是与上1（清除其他位），看似保证了特殊位为1，可是写入后，确改变了该位的值为0，如果该位本来就为0，却无所谓，因为写0 对该位不影响。

1）如果要置位其他位，**先把其他非特殊**位**按照正常程序处理**，**最后把特殊位清零**，再写入寄存器，以保证特殊位不变。

2）如果要清零其他位，写入的时候，也是在最后要把该些特殊为清零，再写入。

3）如果要清零特殊位：或上1，写入，但是要记得清零其他特殊位的写入值。

总之，不可以按常规逻辑来。

## 2016/12/30

1:发现 adapter 板子不可以接受串口数据，结果发现是因为DX-B05的蓝牙不行，不可以发送数据到 HM-13，换成手机就可以了。

2: 把关于 **led ， timer ，uart 的一些硬件变量**的指针定义和初始化都放在 main 函数之外，这些初始化都不会被执行，所以如果**把初始化放在外**面，又由于是**单例模式**，那么**构造函数**将**在其他地方**被**调用**，所以为了确保**每个资源的对象**都在进main函数之后能够**被正确的初始化和构造**，都需要在 main 函数中初始化，可以**在main 之外声明**。

另外也需要注意，**配置这些硬件变量**的工作时，我们需要**确保硬件的初始化**工作要已经**完成**了，比如**寄存器的配置**，**时钟的配置**和开启，只有**硬件在合适**的**状态**，我们才能正确的初始化封装好的硬件资源类。