**使用EnergyTrace™来测量功耗**

关于电路板功耗的测量有多种方式，这里主要介绍一下EnergyTrace小工具的使用。

传统上是通过放大目标信号并在离散时间测量分流电阻上的电流消耗和电压降来测量功耗的。EnergyTrace技术实现了一种测量功率的新方法。在支持EnergyTrace技术的调试器中，软件控制的DC-DC转换器会生成目标电源（1.2 V-3.6 V）。DC-DC转换器充电脉冲的时间密度等于目标微控制器的能耗。调试工具中的内置校准电路定义了单个充电脉冲的等效能量。每个充电脉冲的宽度保持恒定。调试工具对每个充电脉冲进行计数，并且将充电脉冲的总和与经过的时间结合使用以计算平均电流。

由于这种测量技术会不断地采样提供给微控制器的能量，因此即使是消耗能量的较短设备活动也有助于记录总能量。相对于无法检测极短持续时间的能耗的基于分流的测量系统而言，这是一个明显的优势。

SimpleLink MSP432 LaunchPad支持EnergyTrace和EnergyTrace ++。EnergyTrace ++具有与EnergyTrace相同的方式来测量微控制器能量的功能，而且还使开发人员能够访问有关微控制器内部状态的信息。这些状态包括外设和所有系统时钟（与时钟源无关）的ON / OFF状态以及当前正在使用的低功耗模式（LPM）。有关EnergyTrace的更多信息，请参见以下应用报告：<http://www.ti.com/lit/an/slaa603/slaa603.pdf>

**硬件**

* [MSP432P401R LaunchPad](http://www.ti.com/tool/msp-exp432p401r)

**开发软件**

* [Code Composer Studio v9.2](http://www.ti.com/tool/CCSTUDIO)
* [SimpleLink MSP432P4 SDK v3.40](http://www.ti.com/tool/simplelink-msp432-sdk)

**代码概述**

该项目使用SDK随附的powerdeepsleep TI驱动程序示例以及TI驱动程序电源管理API中提供的少量功能。示例代码路径为：simplelink\_msp432p4\_sdk\_3\_40\_00\_05\examples\rtos\MSP\_EXP432P401R\drivers\powerdeepsleep

该项目仅使用一个线程，该线程用于演示如何在SimpleLink MSP432上实现LPM3（PowerMSP432\_DEEPSLEEP\_0）和LPM4（PowerMSP432\_DEEPSLEEP\_1）。

* mainThread()：在源文件中：powerdeepsleep.c
* 初始化GPIO
* 设置GPIO回调函数以处理按钮输入，以指示设备进入睡眠状态。
* main()：在源文件中：main\_tirtos.c
* 初始化外围设备
* 定义全局变量
* 创建线程
* TI Drivers 电源策略

PowerMSP432\_deepSleepPolicy是一项积极的电源策略，可检查活动约束并自动将设备转换为可能的最深睡眠状态。

首先第一个目标是进入PowerMSP432\_DEEPSLEEP\_1（LPM4）；如果不允许，则次要目标是PowerMSP432\_DEEPSLEEP\_0（LPM3）；如果不允许，则下一个目标是PowerMSP432\_SLEEP（LPM0）; 最后，如果不允许这样做，将选择CPU等待中断（WFI）。

可以通过使用Power\_setConstraint()函数来进行应用约束。例如，为防止设备进入PowerMSP432\_DEEPSLEEP\_1（LPM4），应传递参数PowerMSP432\_DISALLOW\_DEEPSLEEP\_1。如下面的代码，之后会在实验中添加

 /\* Select PowerMSP432\_deepSleepPolicy to enable transition

 \* into the PowerMSP432\_DEEPSLEEP\_1 (LPM4) state

 \*

 \* To ensure the device doesn't move to LPM4 state, we set

 \* a constraint to disallow DEEPSLEEP\_1. Now the device will

 \* only go to DEEPSLEEP\_0, which is LPM3.

 \*/

**Power\_setConstraint**(PowerMSP432\_DISALLOW\_DEEPSLEEP\_1);//added by susan

为了执行此政策：

必须选择它作为电源策略（通过在PowerMSP432\_config结构中的policyFxn指定，或在运行时使用来指定Power\_setPolicy()），

必须启用电源策略（通过PowerMSP432\_config结构中的“ enablePolicy” 或在运行时调用Power\_enablePolicy()）。

**基础实验**

导入两个项目powerdeepsleep 以及  tirtos\_builds\_MSP\_EXP432P401R\_release\_ccs

通过USB将MSP432 LaunchPad插入计算机并检查开发板上的跳线设置（实验中会进行跳线重新设置）如图1所示



图1 跳线设置

* EnergyTrace设置

为了在Code Composer Studio中正确设置EnergyTrace，我们应该首先导航到以下位置：

*Window -> Preferences -> Code Composer Studio -> Advaanced Tools -> EnergyTrace Technology，*然后进行如图2设置

****

图2 EnergyTrace设置

* LPM4下的电流

通过USB电缆将LaunchPad连接到计算机。然后，将该程序下载到LaunchPad并运行示例。

验证红色LED在您的LaunchPad上是否点亮。按下S1按钮后，红色LED应熄灭，并且设备现在处于PowerMSP432\_DEEPSLEEP\_1（LPM4）。

当程序运行时，在Code Composer Studio的“调试”视图中，我们应该导航到Tools -> EnergyTrace，这将在“调试”视图中提示我们一个新窗口，点击绿色的三角图标默认测量10s内的功耗情况，如图3所示



图3 LPM4下的电流（带跳线）

从结果看电流消耗远远高于PowerMSP432\_DEEPSLEEP\_1电源策略中（LPM4）的预期〜1uA电流消耗。主要原因如下：

* 当设备处于活动的调试会话中并连接到Code Compuser Studio时，该设备仅模拟LPM4以维持调试连接。为了使调试连接保持活动状态，该设备实际上保留在LPM0中，消耗了更多功率以使设备的调试外围设备保持活动状态。
* 有许多跳线连接到我们的LaunchPad上的其他I / O，这将导致更高的功耗测量。

为了优化电流消耗，以确保仅测量MSP432消耗的电流，我们首先必须进行一些更改。

* 首先，断开以下跳线的连接：TDI，TDO，TCK，TMS，RST，TXD >>，RXD <<，5V，P1.0，P2.0，P2.1和P2.2。
* 断开USB电缆的连接，然后重新连接，以重新启动MSP432。
* 单击CCS中的“停止调试”按钮。这将使我们回到CCS Edit视图。
* 单击“构建”和“下载并调试”按钮旁边的EnergyTrace按钮。

单击LaunchPad上的S1按钮使设备进入低功耗模式，因为在重启设备电源后我们已经重新启动了设备。再次运行EnergyTrace之后，现在设备实际上处于所需的低功耗模式，我们看到了一个新的，更准确的电流消耗结果，如下图4所示



图4 LPM4下的电流（移除跳线）

* 修改电源策略到LPM3

我们还可以添加代码来重现LPM3功率测量值。为此，我们需要将以下代码添加gpioButtonFxn0（）函数中，如图5所示。



图5修改电源策略到LPM3

重复（LPM4下的电流）下的步骤来测量LPM3下的电流，下图6是连接跳线时的结果



图6 LPM3下的电流（带跳线）

下图7时移除跳线后的实验结果



图7 LPM3下的电流（移除跳线）

由于LPM3和LPM4在MSP432上的电流消耗很接近，两者在EnergyTrace基本是没有差别的。

**总结**

以上介绍了EnergyTrace的使用方法。值得注意的是，要获得比EnergyTrace提供的功率测量更精确的功率，请在考虑硬件和软件的相同因素的情况下，使用专业的功率分析仪。欢迎大家讨论或指正相关的问题，谢谢！