PSoC 提高实验　亮度可调 LED

实验报告

姓名：赵文亮

学号：2016011452

班级：自 64

日期：2018 年 7 月 9 日

目录

1 实验内容

1.1 必做功能

用电位器或者 Capsense 按键或滑条，控制 LED 灯的亮度，并将亮度信息通过 UART 或者 USB 发送给上位机。

1.2 附加功能

为了实现更多的功能，我设计了一个可以同时使用电位器和 Capsense 滑条来控制 LED 灯的系统。Capsense 的优先级比电位器要高，当用户使用 Capsense 滑条时，LED 的亮度会跟随滑条触摸位置而变化；用户不使用 Capsense 滑条时，LED 的亮度会随着电位器的阻值而变化。此外，该系统并可以实时将亮度信息打印在 LCD 屏幕和串口上。

2 设计方案

2.1 CapSense 控制亮度

CapSense 模块有直接获取其状态以及滑条位置的接口，我获取位置后将其写入 DMA 的源地址中，便可以实现使用 Capsense 滑条控制 LED 的亮度。

2.2 电位器控制亮度

我使用 ADC 模块采集电位器的输入电压值，并通过 DMA 传送到 VDAC。为了保证输出电压的稳定性，我在 VDAC 的输出连接一个运放之后再接入 LED。

2.3 LCD 显示

使用 Character LCD 模块即可方便地实现 LCD 上的显示。我将 LED 的亮度值归一化为 0-100 之间的数值，调用相关的函数将其写入到 LCD 屏幕上。

2.4 串口显示

我使用 USB 串口模块进行串口通信。当检测到亮度发生变化时，就通过串口向电脑发送一条消息，实时地把亮度信息显示在电脑上。

3 电路图

系统的顶层原理图如图 1 所示。电位器控制 LED 部分包括 ADC、DMA、VDAC、Status Reg、Opamp 模块；CapSense 控制 LED 部分包括 CapSense 模块。输出部分为 LED 的输出、LCD、UART 模块。

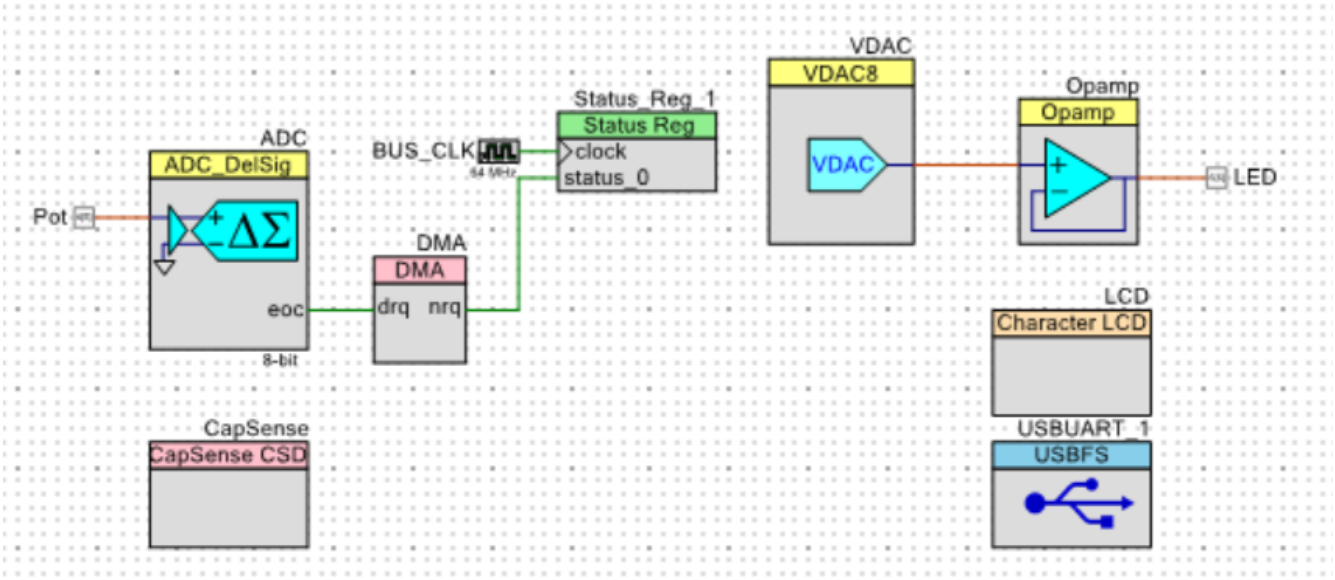


图 1: 顶层模块图

4 模块选择与参数设置

本次实验中主要模块及参数设置如表 1 所示。

表 1: 主要模块参数设置

模块	参数设置
ADC	Resolution=8; Conversion rate = 10000
DMA	Hardware Request=Level
VDAC	Range=0-4.080V; Data Source=CPU or DMA; Strobe Mode=Register Write
LCD	LCD Custom Character Set=None
USBUART	Report: 3-Button Mouse
CapSense	Tuning method=Auto(SmartSense)

5 流程图及实验代码

实验的流程图如图 2 所示。下面将分析每一步骤的代码实现[1]。

5.1 CapSense 滑条控制 LED 亮度

由于 CapSense 滑条的优先级比电位器要高，我先对它进行判断。首先读取 CapSense 是否有输入，如果有输入则获取滑条的位置。若滑条位置变化且有手指按下，则更新一下滑条的位置信息，并写入 ADDA_BUFFER 中。

[1]完整代码见https://github.com/thu-jw/PSoC-LED_Controller

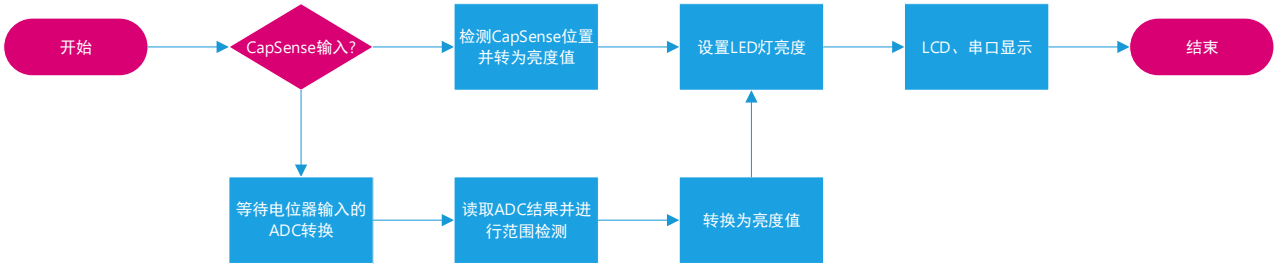


图 2: 实验流程图

```
        if(!CapSense_IsBusy())
        {
            /* Update baseline for all the sensors */
            CapSense_UpdateEnabledBaselines();
            CapSense_ScanEnabledWidgets();
            sliderPosition =(uint8)CapSense_GetCentroidPos(CapSense_LINEARSLIDER0__LS);
            /* Finger detected on the slider */
            if (sliderPosition != NO_FINGER)
            {
                /* If finger position on the slider is changed then update the LCD */
                if(sliderPosition != lastPosition)
                {
                    lastPosition = sliderPosition;
                }
            }
        }
        if (sliderPosition != NO_FINGER)
        {
            ADDA_BUFFER[0] = (uint8)(sliderPosition / 100.0 * 255);
```

5.2 电位器控制亮度

我参考了 PSoC Creator 中的 ADC_DMA_VDAC 例程来实现电位器控制 LED 的亮度。该例程中使用电位器来控制 LED 灯闪烁的频率。我基于 DMA 的原理，修改了 DMA 的源地址配置，实现了将 ADC 的数据通过 DMA 传送到 DAC 的输入。

```
#define DMA_SRC_BASE (ADDA_BUFFER)
#define DMA_DST_BASE (CYDEV_PERIPH_BASE)
uint8 ADDA_BUFFER[1];
```

在主循环里面，通过 ADC 读取电位器当前输出的电压。注意实际情况有可能会出现负值，这是因为 AD 转换时电位器输出电压不在转换范围内，这时就要做相应的处理。

```
        /* Wait for end of conversion */
        ADC_IsEndConversion(ADC_WAIT_FOR_RESULT);
```

```
        voltageRawCount = ADC_GetResult16();
        if (voltageRawCount < MIN_COUNT)
        {
            voltageRawCount = MIN_COUNT;
        }
        /* Used to remove the count beyond 8 bit value, see resolution section
        * in DelSig ADC datasheet */
        else if(voltageRawCount > MAX_COUNT)
        {
            voltageRawCount = MAX_COUNT;
        }
        ADDA_BUFFER[0] = voltageRawCount & 0xFF;
```

5.3 LCD 与 UART 输出

判断当前的亮度值与上一次的是否相同，如果亮度不同则刷新 LCD 的显示值，并向 UART 中发送最新的 LED 亮度信息。

```
            if (last != vdac_data)
            {
                sprintf(buffer, "LED Intensity: %3d\r\n", vdac_data);
                USBUART_1_PutData((uint8 *)buffer, strlen(buffer)); /* Send data back to PC */
                while(USBUART_1_CDCIsReady() == 0u);
                last = vdac_data;
            }
```

6 实验结果

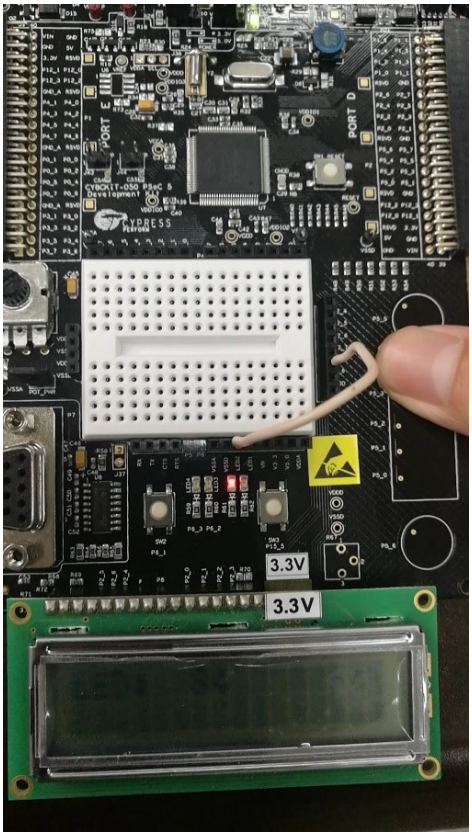
实验的部分截图如图 3 和图 4 所示。其中图 3a 使用 CapSense 滑条控制 LED 灯亮度，图 3b 为使用电位器控制 LED 灯亮度。由于 Capsense 滑条的优先级较高，当检测到有触摸后，LED 灯的亮度不再由当前电位器的位置确定，而是完全由滑条的位置决定。验收时，助教还特意尝试了同时使用电位器和 Capsense 滑条来控制，且一个将亮度调高，一个将亮度调低。结果 LED 的亮度确实随着滑条的位置变化而变化，这也验证了我对优先级的设计。

LED 灯的亮度信息可以通过串口实时传送到电脑端。当检测到亮度发生变化时，串口就会将最新的亮度信息通过串口发送出来，如图 4 所示。

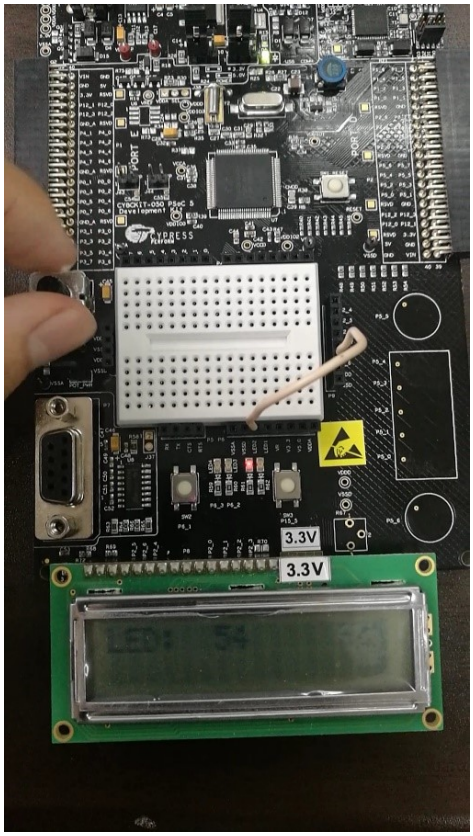
7 实验中遇到的问题及解决方法

一开始，我的 Capsense 采用了例程中的参数，表现得不是很灵敏。我最初以为是 Capsense 硬件的问题。验收时，助教提示我可以通过修改参数的方式改变其灵敏度。于是我查阅了 datasheet，如图 5 所示。

由此可见，将 Tuning method 修改为 Auto(SmartSense)，便可以获得很高的灵敏度。于是我在模块设置中修改了这一参数，如图 6 所示。之后再重新进行实验，果然发现滑条的灵敏度变得非常高。



(a) Capsense 滑条控制 LED 亮度



(b) 电位器控制 LED 亮度

图 3: 两种方法控制 LED 亮度

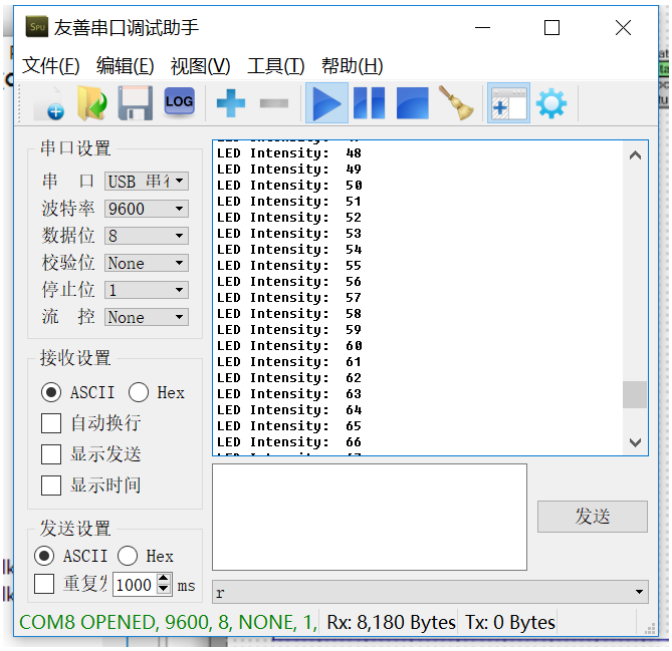


图 4: 串口显示 LED 亮度

Tuning method

This parameter specifies the tuning method. There are three options:

- **Auto (SmartSense)** – Provides automatic tuning of the CapSense CSD component.

This is the recommended tuning method for all designs. Firmware algorithms determine the best tuning parameters continuously at run time. Additional RAM and CPU resources are required in this mode.

图 5: Datasheet 关于 Tuning method 部分截图

图 6: CapSense 修改灵敏度

8 体会、收获与建议

本次实验总体来说较为顺利。PSoC 的设计方式是原理图与代码相结合的方式，在原理图中可以可视化地设置一些参数，这些设置会由 PSoC Ceator 自动生成对应的初始化代码。在用户的代码中，我们只需要一句 Start 就可以完成某个模块的初始化步骤。

PsoC Creator 有一个很大的优点就是例程和文档比较丰富。任何一个模块都可以右键打开 datasheet 查看，同时在 Code Example 中有很多值得学习的样例，这些极大地方便了我们的开发。

值得一提的是，模块中的参数设置很大程度上决定了模块的性能。我通过调节 CapSense 灵敏度的过程中对这一点深有体会，我也再次明白了阅读 datasheet 的重要性。今后进行开发时，只要是用到现有的模块或元件，一定要先阅读 datasheet 来了解其参数，才能更好地将它们的功能发挥出来。感谢老师和助教的指导！

一个小建议：由于下载程序需要一根 USB 线，PSoC 板与计算机进行串口通信时也需要一根 USB 线，而盒子里只有一根 USB 线。目前我的做法是，先通过 USB 把程序烧进去，再拔下来接在通信的 USB 上完成串口通信，这一过程稍有麻烦。所以建议在盒子里面准备两根 USB 线，这样会比较方便。