GUI Composer 1

难度级别

- 入门级，从零开始
- Step by Step 操作

阅读对象

- 使用过 CCS，对 GUI Composer 感兴趣，但从来没有接触过的用户

简单来说，GUI Composer 可以看做是一个接口工具。对于用户来讲，可以对这个接口的可视化界面进行自定义。利用 GUI Composer 可以帮助用户将在 MCU 中运行的各项参数系统整合后显示在 PC 端的应用中，此外 GUI Composer 也会非专业用户进行调试提供了一种新的方式。

GUI Composer 分为两大块，一个是集成在 CCS 中的 GUI Composer 开发应用选项，用户可以利用 CCS 中的所有调试选项对 GUI Composer 进行调试；另一个为 GUI Composer Runtime，需单独下载并安装，其可以脱离 CCS 进行程序的下载和可视化界面的调试和查看。

GUI Composer wiki：http://processors.wiki.ti.com/index.php/Category:GUI_Composer 页面中提供一个视频演示教程和一个 PPT 文档对 GUI Composer 的入门应用进行介绍。在这里将结合一个简单的实验了解 GUI Composer 的基本应用。

实验内容

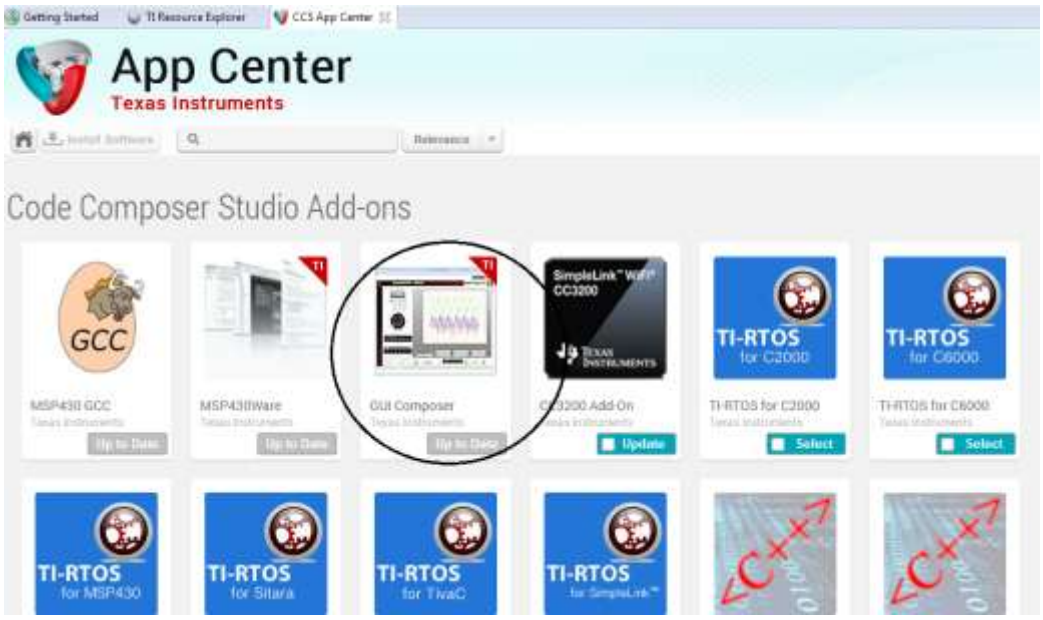
- 新建 GUI Composer 工程
- 生成并导出 GUI Composer 应用
- 在 CCS 中运行 GUI Composer 应用
- GUI Composer Runtime 应用

实现

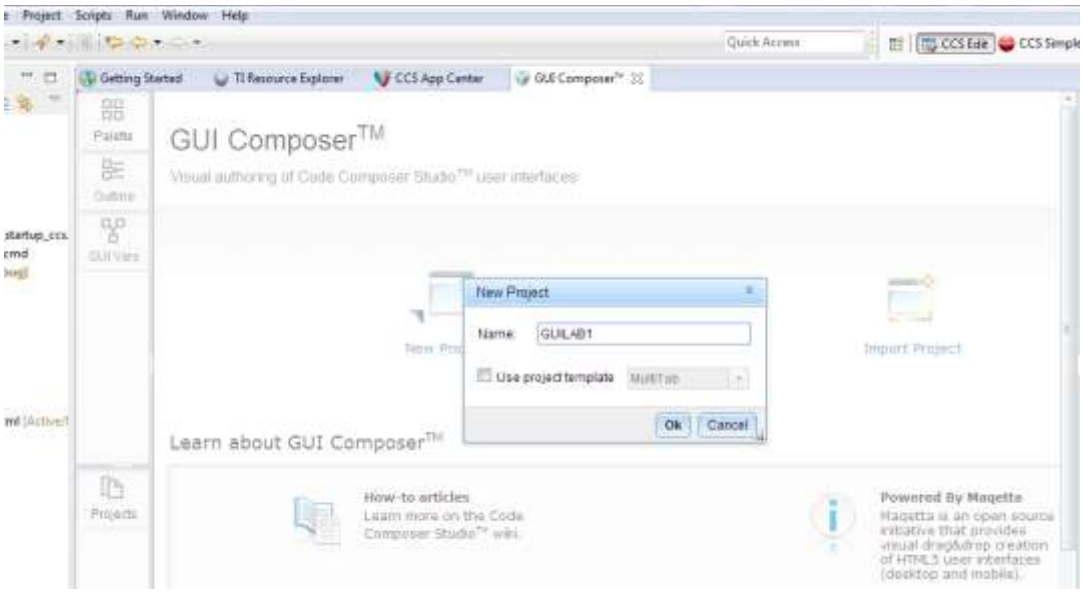
- LED 闪烁控制
- 电压表显示

实验平台为 TM4C123G Launchpad（不定期发布新的参考例程和实验）

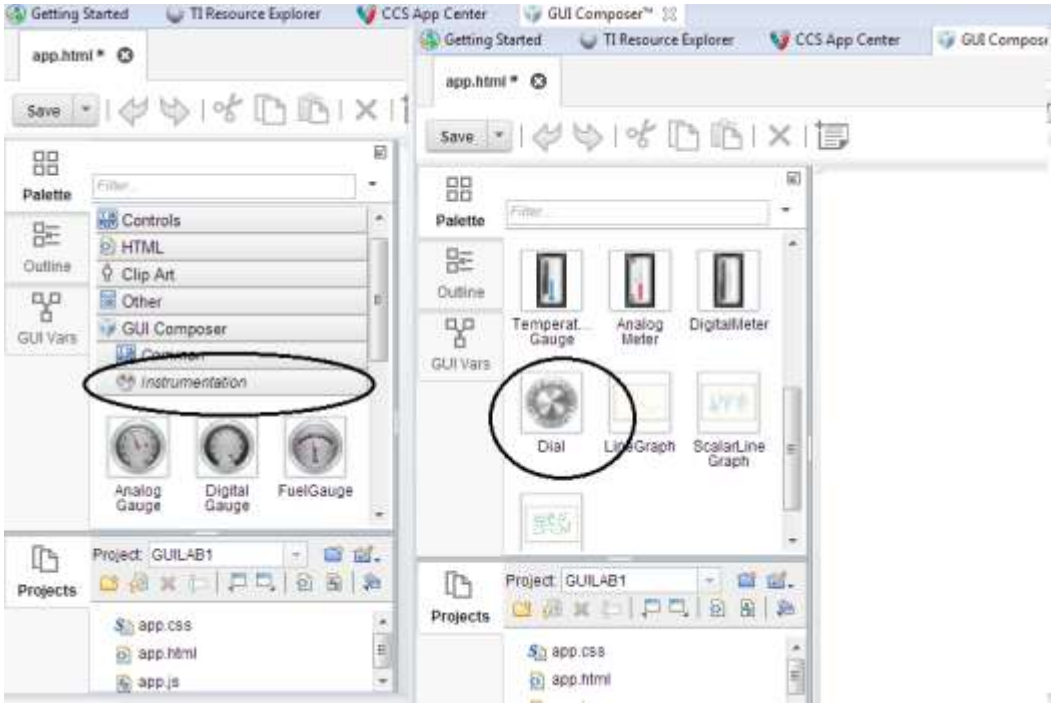
在电脑中安装好 Code Composer Studio IDE v6，检查 GUI Composer 是否已经集成安装。单击菜单栏中“View”，找到 GUI Composer™，并单击。如果之前没有安装，在 CCS 的 App Center 中找到 GUI Composer。



进入 GUI Composer 的界面。单击新建工程，输入工程名称。

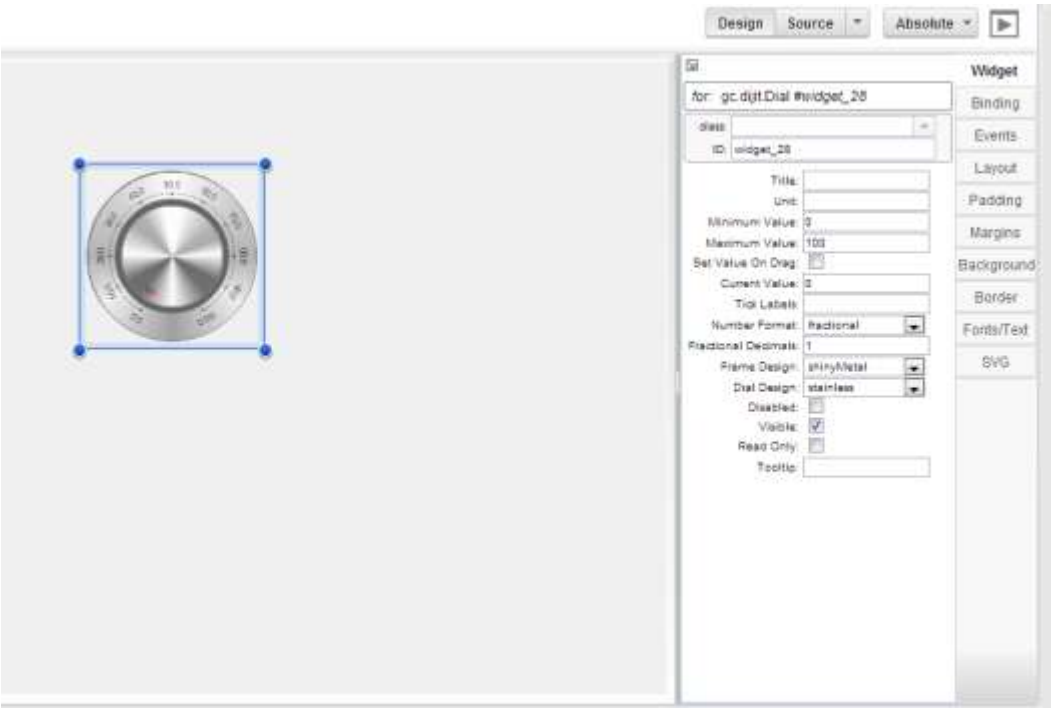


GUI Composer 的控制界面基于 html 页面。在左侧的“Palette”中选择“Instrumentation”控件，下拉窗口中有若干可视化仪表盘。选择“Dial”旋钮，按住将其拖动到右边的空白窗的合适位置，松开。



当该控件被选中的时候，表盘外圈会呈现蓝色，右侧有对其进行编辑的选项，可根据实际需求对各项参数进行编辑和修改。编辑内容一方面包括控件的外观，例如在“Widget”中

可以通过下拉菜单选择改变旋钮表盘的质感等；另一方面可以编辑的则是关联的参数选项，例如在“Widget”中同样可以修改旋钮表盘的阈值范围。而在“Binding”中则是添加图形化控件所传递的对应参数。此外，与其他图形化软件类此，在此编辑页面中可以添加其他的操作控件以及对这些控件对应操作的函数。



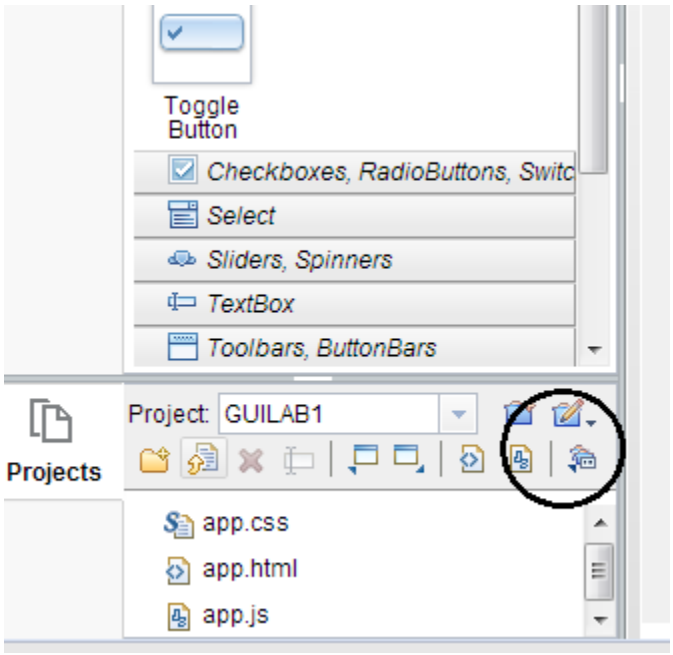
通过改变旋钮，可以改变某个特定参数的值。在本实验中，用旋钮来控制 loop 循环次数值，从而实现 LED 闪烁频率的改变。

将“blinky”工程导入至 CCS 中，该示例程序实现了板上 LED 按照参数“delay”的大小进行相应频率的闪烁。

在 GUI 界面中首先按照上文所述方法添加控制旋钮，在“Widget”控制面板中修改“Maximum Value”参数。这里因为程序中 delay 的值含义为机器周期单位，所以这里设置为 1,000,000，以保证足够的时间。在“Binding”界面中的“Value”值内填入传递参数“delay”。至此完成旋钮相关参数的基本设置。

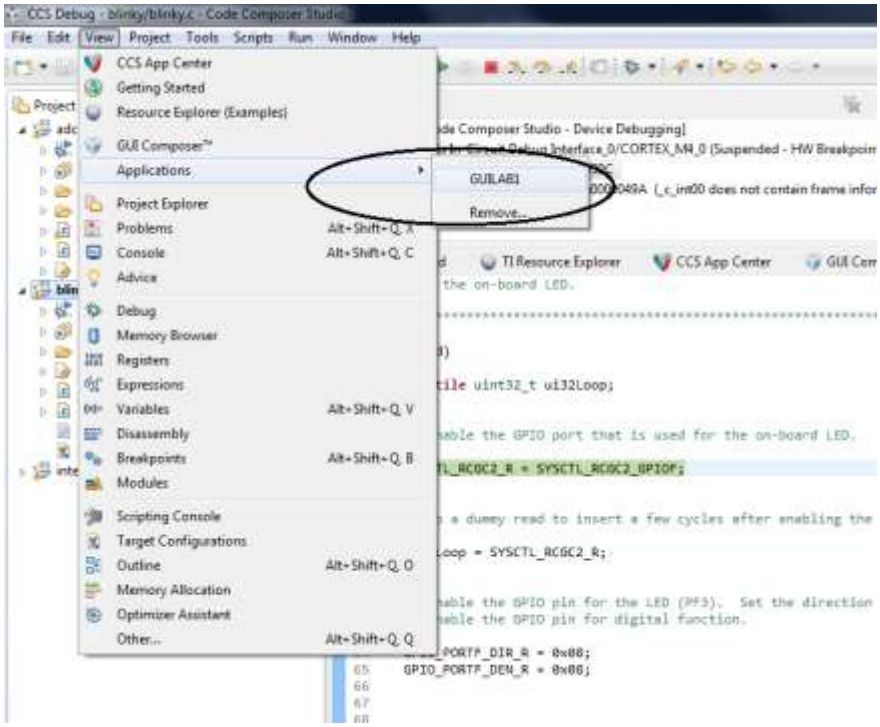
单击“Save”保存配置。单击右上角的“  ”可以预览界面。

单击“Projects”框内的“install project”，将该工程装载入 CCS 中。



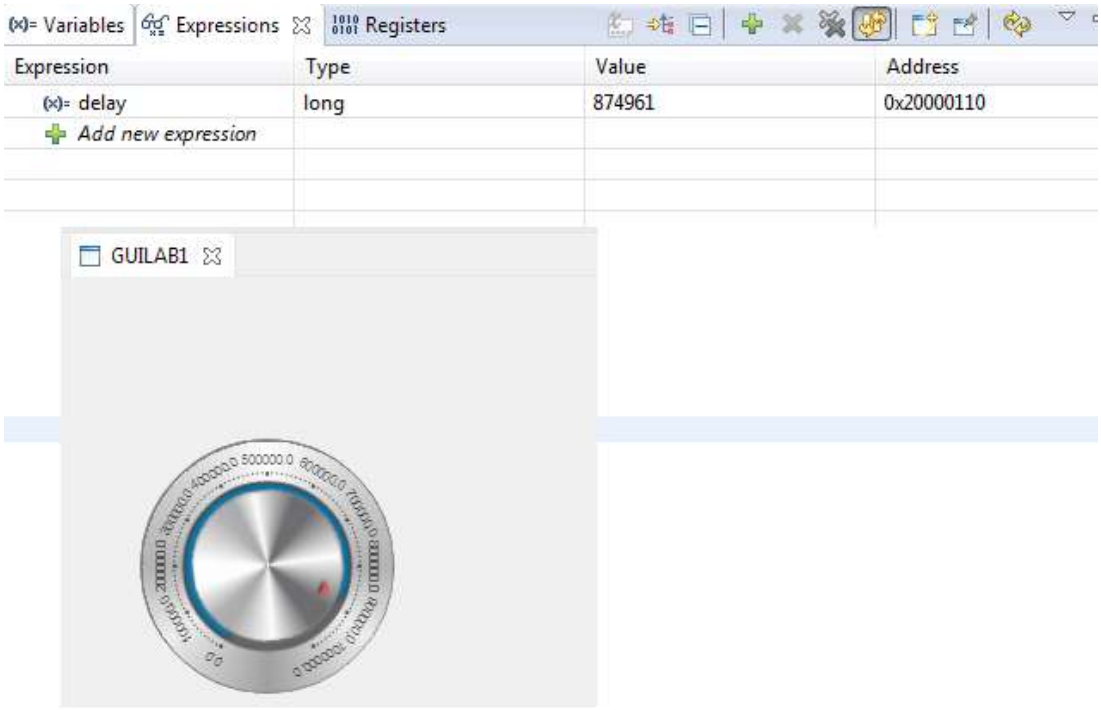
单击“  ”将 CCS 工程下载至 M4 Launchpad 中。

单击“View”→ “Applications”中之前装载的对应工程。拖动出现的窗口，使其大小适中。



单击“ ”运行程序，观察板上 LED 现象。此时 LED 似乎处于常亮状态，其实是因为闪烁的频率太快，导致肉眼观察不到。此时在 GUILAB1 窗口中向右旋转旋钮，随着旋钮指针指向数字的增大，闪烁的频率越来越慢。

如果调出 Expressions 窗口，添加“delay”变量，可以发现在旋转旋钮的时候，程序中 delay 的值会随之发生改变，这就解释了为什么只是简单改变旋钮就可以改变观察到的现象。这里的旋钮控件担当着传递参数的作用，将对应的改变值反映到程序中。熟悉 CCS 调试功能的读者可能发现如果没有 GUI，而在 Expressions 窗口中直接修改“delay”的值也可以实现类似的功能，只是 GUI 提供了一个更为方便和友好的界面。



在上述实验中，选择的旋钮控件实现了外部参数写入的功能，接下来选择一个显示计来实现参数读出显示的功能。

导入“adcsensor”工程。该工程在 blinky 闪烁 LED 的基础上添加了一路 ADC 转换，将 PE3 引脚的输入电压转换为数字量，赋值给“ui32Temp”。

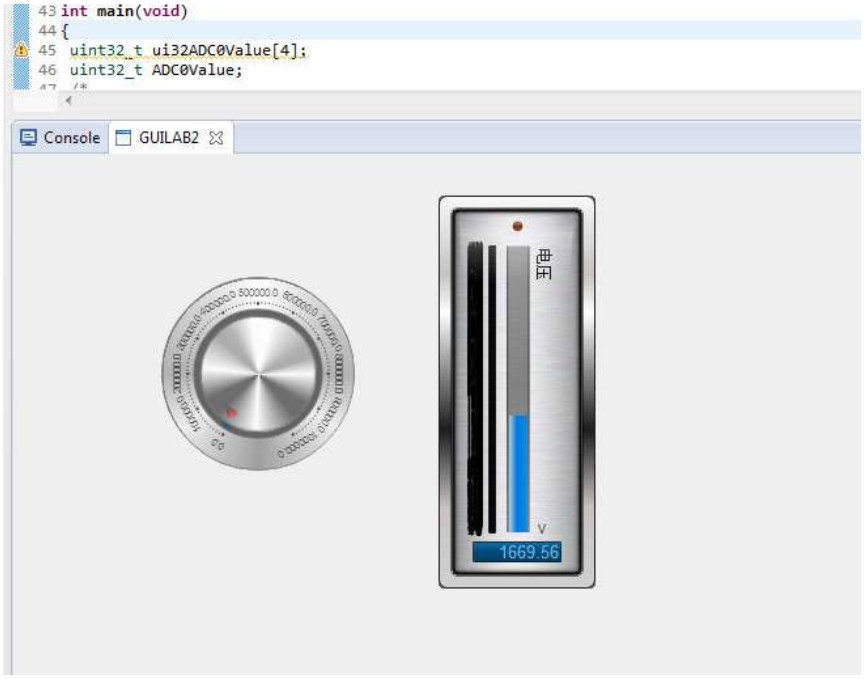
在前面实验的 GUI 工程中，拖动添加“Analog Meter”，在“Widget”中对范围做适当的调整，同时在“Binding”中填入需要观察的变量“ui32Temp”。

单击“install projects”保存好的 GUI 工程装载入 CCS。

单击“  ”将 CCS 工程下载至 M4 Launchpad 中。

单击“View”→ “Applications”中之前装载的对应工程。拖动出现的窗口，使其大小适中。

单击“  ”运行程序，观察 GUI 对话框中 Analog Meter 条形柱的变化，发现随着 PE3 引脚输入电压的变化，条形柱的高度会随之发生改变。



这里大家会产生一个疑问，显示的值并不是预期的电压值，而似乎更像是一个数字量。的确，在原始程序中，ui32Temp 变量只是简单地获取 ADC0 的原始值，未作任何更改，因而其值为 0-4095 区间（M4 内部为一 12 位的 ADC）。那么如何得到实际的电压值呢？毕竟现在看来显示的似乎并不是一个名副其实的 Analog Meter。

自然而然我们有一个有效的解决方案：在 CCS 源代码中新建一个名为 VolTemp 的变量，将其赋值为：

$$VolTemp = \frac{ui32Temp}{4096} \times 3.3V$$

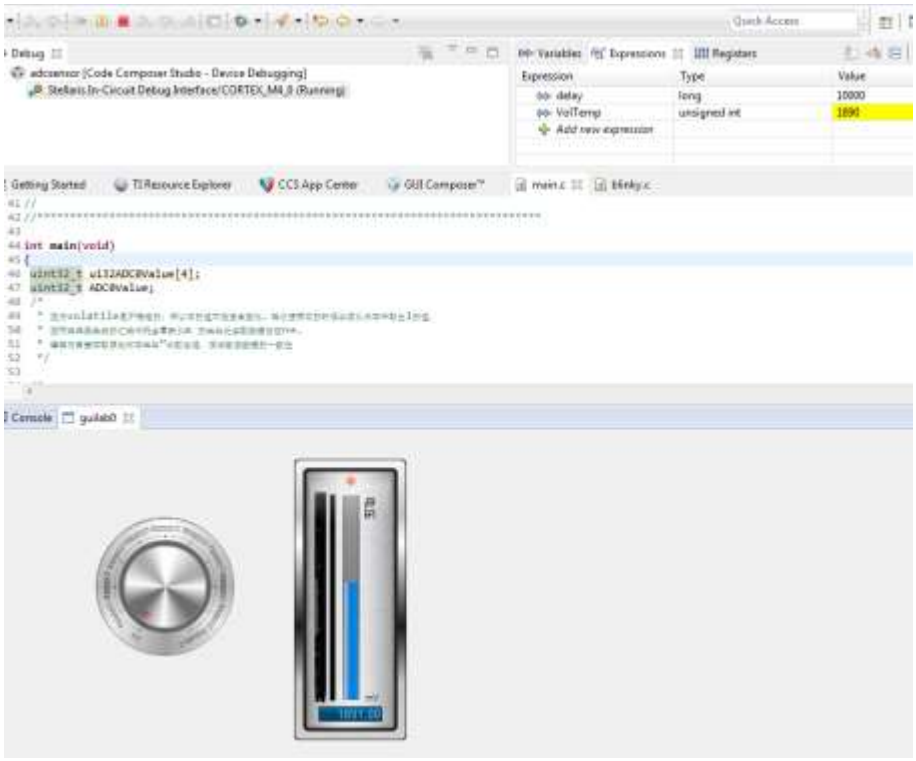
其中，3.3V 为 ADC 的参考电压值。

在 CCS 源代码中添加

```
    VolTemp = ui32Temp*3300/4096;
```

然后将 GUI 中 Analog Meter 的 Binding 值由原先的 ui32Temp 改为 VolTemp。需要注意的是在 CCS 中如果显示小数，需要有额外的操作，所以为简化程序，这里以 mV 为单位，避免小数点操作。

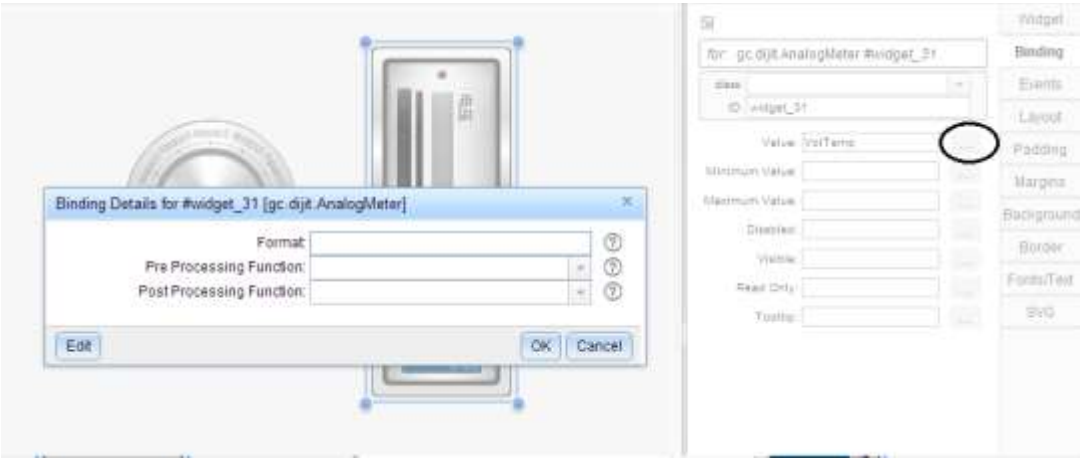
生成新的应用文件并装载至 CCS 中，运行，得到图示结果。



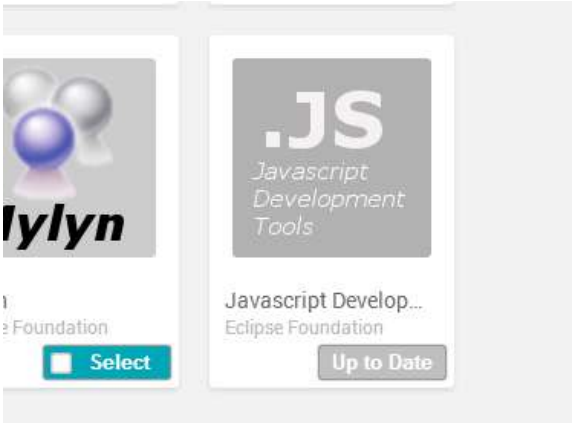
这样，显示的数值就和实际的电压值对应上了。

实际上，还有一种更为推荐的方法，适用于用户不希望源代码进行更改，而同时达到 GUI 界面的显示数据更符合现实生活中人们所能够理解的值。

打开 GUI 工程的编辑界面，单击“Binding”的“Value”右侧的按钮，弹出图示对话框。在弹出的对话框中看到，针对 Value 参数提供了两个处理函数。一个是“Pre Processing Function”，即在 GUI 界面中对 Value 参数在使用前对其进行的预处理。另一个对应的是“Post Processing Function”，顾名思义，为在使用 Value 参数结束后对其进行的后期处理。针对 GUI 提供的这个功能，可以对需调用显示的“ui32Temp”值进行预处理，将其转换为对应的电压值，函数不妨命名为“DigtalToAanlog”。同时为使显示能够保持，添加后期处理函数，不妨命名为“AnlaogHold。”注意将 Value 值改回为原先的“ui32Temp”。



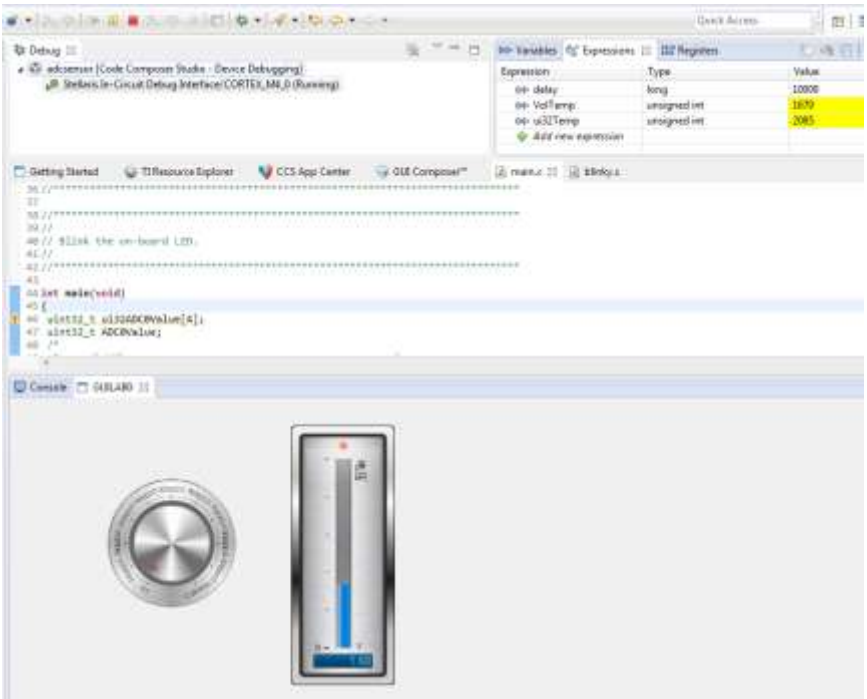
然后在 Projects 中双击“app.js”文件，对之前设置的预处理函数进行编辑。这个文件为 JavaScript 文件，如果无法正常打开需预先安装相应的开发工具。这个工具在 CCS 的 App Center 中可以方便找到。



在 app.js 文件中已经生成了对应的函数，在函数空白处填入需要对 Value 参数进行的处理。保存修改，同之前的操作装载该应用至 CCS。注意：此处的计算和参数都基于 PC 进行，所以可以比较放心地使用小数点运算。可以用 V 作为单位显示，需同时将“Widget”中的最大值做对应的调整。

```
/*
 * This file is provided for custom JavaScript logic that your HTML files might need.
 * GUI Composer includes this JavaScript file by default within HTML pages authored in GUI Composer.
 */
require(["dojo/ready"], function(ready){
    ready(function(){
        // logic that requires that Dojo is fully initialized should go here

    });
});

function DigitalToAnalog( valueFromTarget) {
    // return valueFromTarget/2;

    valueFromTarget = valueFromTarget*3.3/4096;
    return valueFromTarget;
}

function AnalogHold( valueToTarget) {
    // return valueToTarget*2;
    return valueToTarget;
}
```

运行并观察现象。同理，对于 LED 闪烁频率调节，可以将程序中比较难理解的 delay 值转换为 ms 单位，便于理解和观察，在此不做赘述。



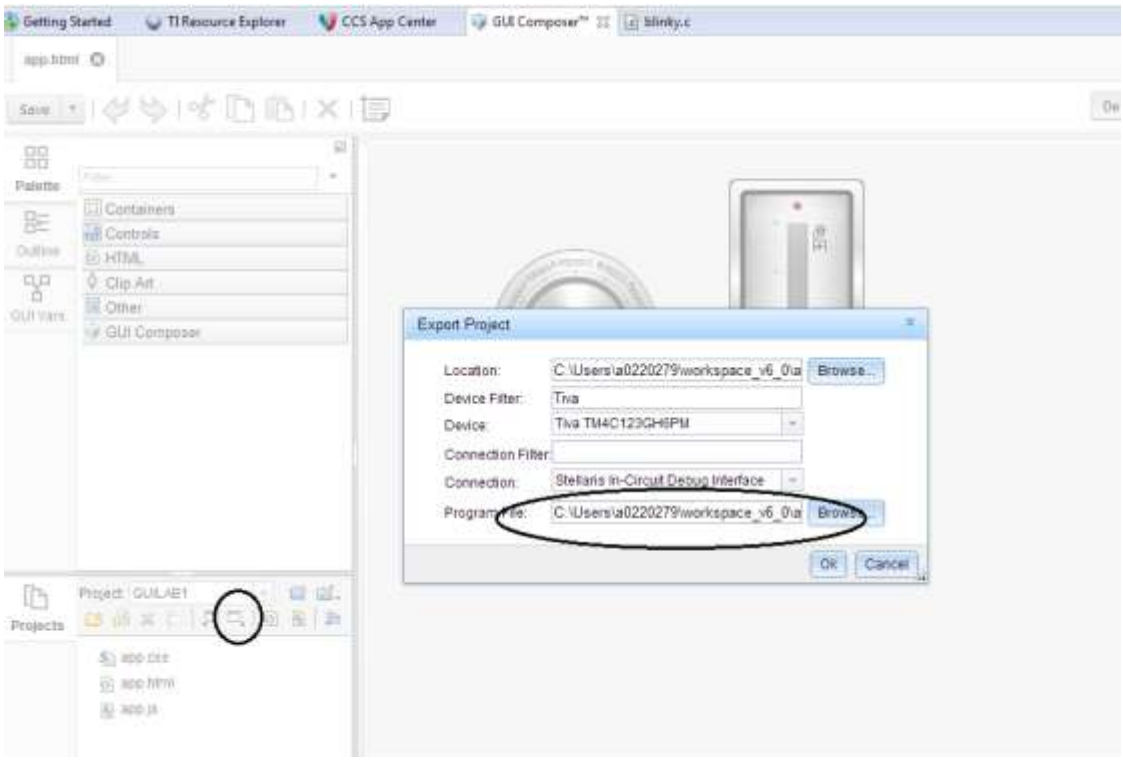
通过以上的两个简单实验，可以了解和掌握如何利用 GUI Composer 更为形象地观察程序内的参数。GUI Composer 内也提供了更多的控件窗口供开发者选择，丰富可视化界面。

以上的过程其实还是更接近于开发调试，基于软件开发平台 CCS。接下来将介绍利用 GUI Composer Runtime 脱离 CCS 开发环境单独运行可视化界面。

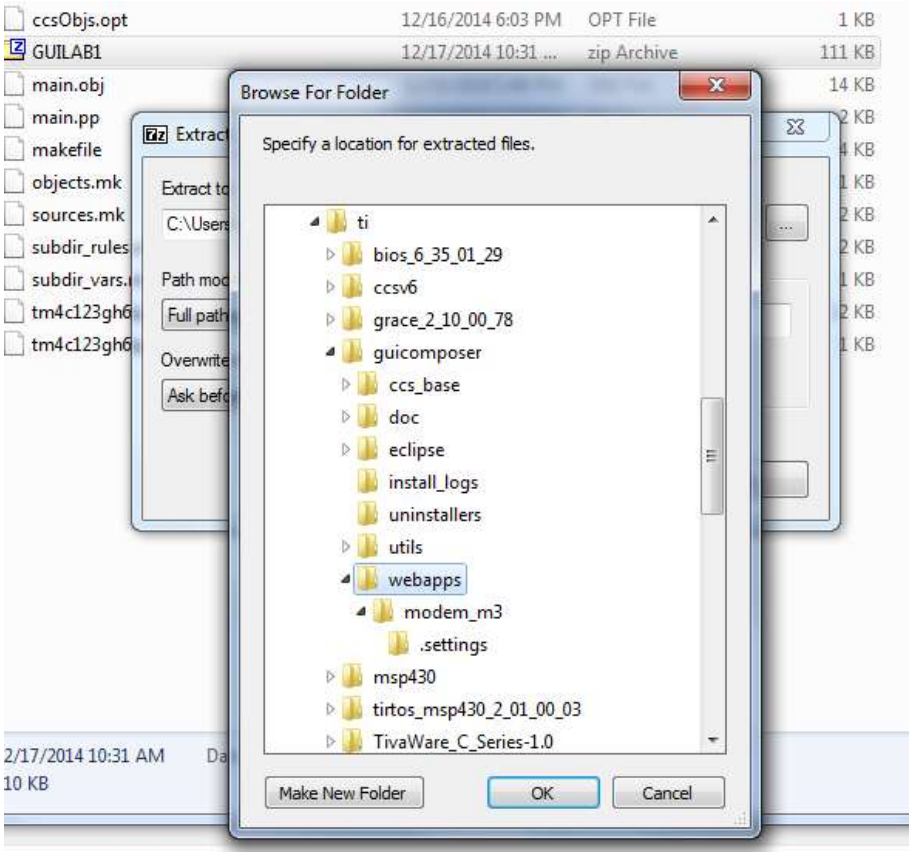
在此之前安装好 GUI Composer Runtime。

在 CCS 中生成可执行文件。

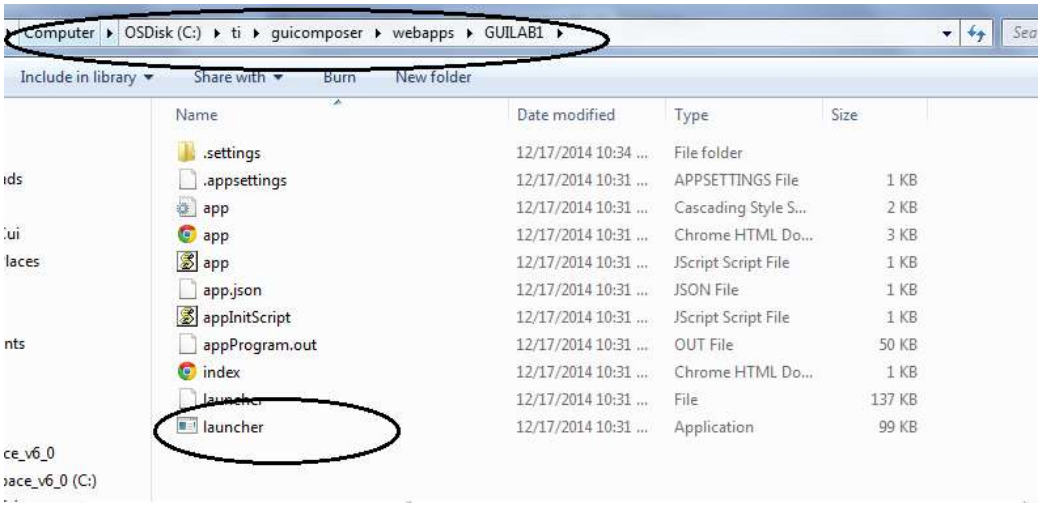
在之前我们都是将 GUI 应用装载进 CCS，现在我们将调试好的 GUI 应用导出。单击“Projects”中的“Export”按键，在弹出的对话框中依次填入导出压缩包的位置（可任意位置），器件型号，仿真器类型和程序文件（必须为需下载至板卡中的工程对应的.out 文件，注意，如果在该步骤前对 CCS 工程进行了修改需重新进行编译，保证对应的.out 文件是最新的）。



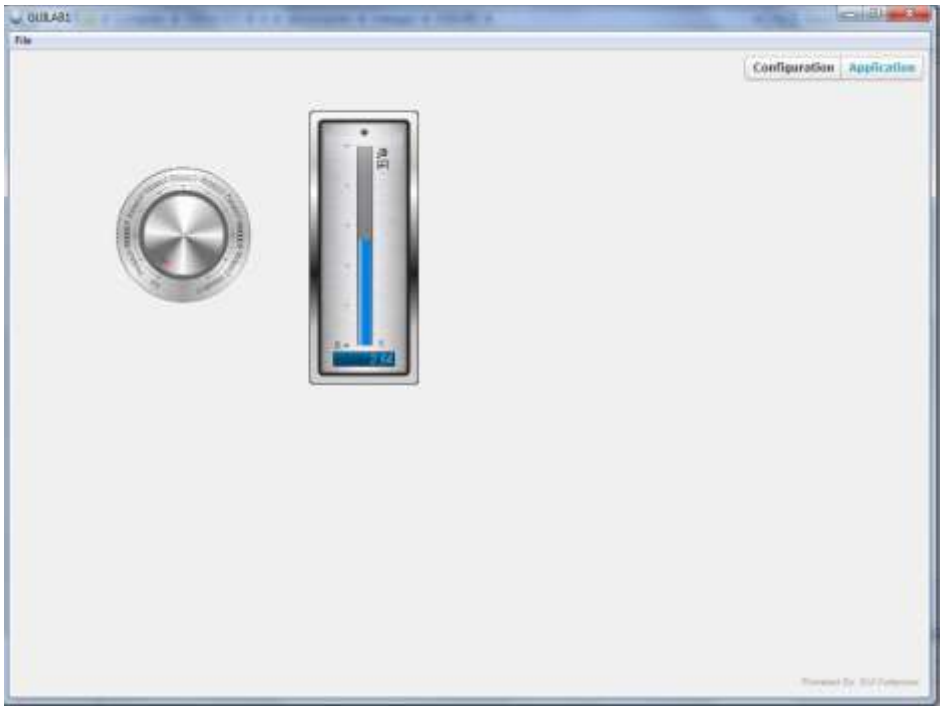
找到导出的压缩包，并将其解压到 GUI Composer 安装目录下的 webapps 文件夹下，如图所示。



找到解压文件的位置，在 Webapps 文件夹下可以看到导出的包含应用程序的文件夹。在里面找到名为“Launcher”的可执行文件，双击打开。



在设备初始化完成后，屏幕显示应用程序界面，在界面里改变旋钮位置观察板上 LED 闪烁情况。同时改变 PE3 引脚输入电压值，观察界面电压表显示状态。



- 欢迎发送邮件 regina-cui@ti.com 或 cuimeng1919@163.com 讨论交流
- 下载安装以及了解更多关于 GUI Composer，访问 GUI Composer wiki：http://processors.wiki.ti.com/index.php/Category:GUI_Composer
- 上文中涉及的资料部分来源于网上公开资料，如涉及版权问题，请及时和作者联系，将第一时间删除相关内容