MSP430F5529 LP 移植 AY-CC2564EVM 模块操作手册

杭州艾研信息技术有限公司

2014 年 11 月

申明

杭州艾研信息技术有限公司保留随时对其产品进行修正、改进和完善的权利，同时也保留在不作任何通告的情况下，终止其任何一款产品的供应的权利。用户在下订单前应及时获取相关信息的最新版本，并验证这些信息是当前的和完整的。

可通过如下方式获取最新信息、技术资料和技术支持：
技术支持电话：0571-86134572
技术支持邮箱：support@hpati.com
产品&资料下载中心：http://www.hpati.com/products/
互动论坛：http://www.hpati.com/bbs/forum.php
公司地址：浙江省杭州市西湖区留和路16号新峰商务楼B306

Aiyan Info
Do Engineering 艾研信息

MSP430F5529 LP 移植 AY-CC2564EVM 模块操作手册

版本 <1.0>

修订历史记录

日期	版本	说明	作者
<24/07/2014>	<1.0>	<创建文档>	<张恺>

目录

MSP430F5529 LP 移植 AY-CC2564EVM 模块操作手册

1. 简介

本手册简要描述的如何从 TI 官方提供的基于 MSP430 系列 MCU LaunchPad 和 AY-CC2564EVM 蓝牙模块的蓝牙通讯例程，由例程中的 F5438A 系列移植到支持 F5529 系列 MCU 上。其中包含了硬件环境的搭建，软件的安装和配置以及工程的导入和修改、联机调试等多个环节的内容。

1.1 目的

帮助刚刚开始着手从事无线或者物联网开发的工程师掌握最基本的蓝牙例程的安装调试；

1.2 范围

仅适用于由 TI 提供的 MSP-EXP430F5529 LP MCU 开发板和 TI 的第三方合作伙伴--杭州艾研信息技术有限公司所提供的 AY-CC2564EVM 蓝牙模块上使用。

1.3 定义、首字母缩写词和缩略语

1.3.1 MSP430F5529 LP：

MSP430F5529 实验板 (MSP-EXP430F5529) 是 MSP430F5529 器件的开发平台，出自最新一代的具有集成 USB 的 MSP430 器件。该实验板与 CC2520EMK 等众多 TI 低功耗射频无线评估模块兼容。实验板能帮助设计者快速使用新的 F55xx MCU 进行学习和开发，其中 F55xx MCU 为能量收集、无线传感以及自动抄表基础设施 (AMI) 等应用提供了业界最低工作功耗的集成 USB、更大的内存和领先的集成技术。实验板上的 MSP430F5529 器件可以通过 USB 进行供电和调试。

产品链接：

http://www.ti.com.cn/tool/cn/msp-exp430f5529lp?keyMatch=msp430f5529%20launchpad&tisearch=Search-CN

1.3.2 AY-CC2564EVM 蓝牙模块：

AY-CC2564EVM 是基于 TI CC256x 的双模式蓝牙 4.0 的评估板。TI 的电源技术和软件算法使得 CC256X 在蓝牙 BR/EDR/LE 多种模式都比同类产品更节能。评估板采用板载 PCB 天线，无障碍通信距离不小于 10 米；评估板提供标准的 BoosterPack 接口，可以与 TI 的 MSP430 LaunchPad 或 TIVA LaunchPad 直接互联；评估板同时提供非 BoosterPack 标准的连接接口，方便用户将评估板连接到自制的 MCU 系统上。 板上有关蓝牙芯片的应用设计可以直接拷贝到用户自己的电路板上以减少产品的开发时间。可应用于无线传感、移动设备扩展附件、工业控制、医疗保健设备和简单的音频设备。

产品链接：

http://www.hpati.com/ay_bluetooth/product_32.html

1.3.3 CCS（Code Composer Studio）：

Code Composer Studio 是一种集成开发环境 (IDE)，支持 TI 的微控制器和嵌入式处理器产品系列。Code Composer Studio 包含一整套用于开发和调试嵌入式应用的工具。它包含了用于优化的 C/C++ 编译器、源码编辑器、项目构建环境、调试器、描述器以及多种其他功能。直观的 IDE 提供了单个用户界面，可帮助您完成应用开发流程的每个步骤。熟悉的工具和界面使用户能够比以前更快地入手。Code Composer Studio 将 Eclipse 软件框架的优点和 TI 先进的嵌入式调试功能相结合，为嵌入

式开发人员提供了一个引人注目、功能丰富的开发环境。

下载链接：

http://www.ti.com.cn/tool/cn/ccstudio?intc=searchrecs&keyMatch=CCS&tisearch=Search-CN

1.3.4 Bluetopia SDK：

Stonestreet One 的 Bluetopia 最新版专门针对 TI 的 eXpressDSP 参考框架 3 (RF3) 而精心设计和优化。TI 的参考框架能适应客户的各种应用，为开发商消除初始低层次设计决策的许多麻烦，并可使之将更多精力集中在特色产品的开发上。该版本 Bluetopia 构建于 Stonestreet One 性能卓越的 DSP 蓝牙协议栈之上，后者已被众多客户及合作伙伴广泛采用。Bluetopia 适用于蓝牙版本 1.1，具有灵活的可扩展内存空间，并包括范围广泛的 API。众多对基于 DSP 产品开发至关重要的耳机与免提产品的配置文件，以及其它核心和最新发布的配置文件也都包括在内。Bluetopia 几乎可适用于任何操作系统及平台。

下载链接：

http://www.ti.com.cn/tool/cn/stonestreetone-bt-sdk?keyMatch=bluetopia&tisearch=Search-CN

1.4 参考资料

MSP-EXP430F5529LP LaunchPad™ Development Kit User's Guide.pdf

Code Composer Studio v6.0 for MSP430 User's Guide.pdf

Hardware Porting Guidelines.pdf

2. 环境搭建

环境的搭建包涵了硬件和软件两个方面的准备工作。

2.1 硬件平台搭建

硬件部分包含了 MSP430F5529 Launchpad 开发板和双模蓝牙 4.0 评估板(AY-CC2564EVM Module)。如下图所示：左侧的为 MSP430F5529 Launchpad，右侧的为双模蓝牙 4.0 评估板。

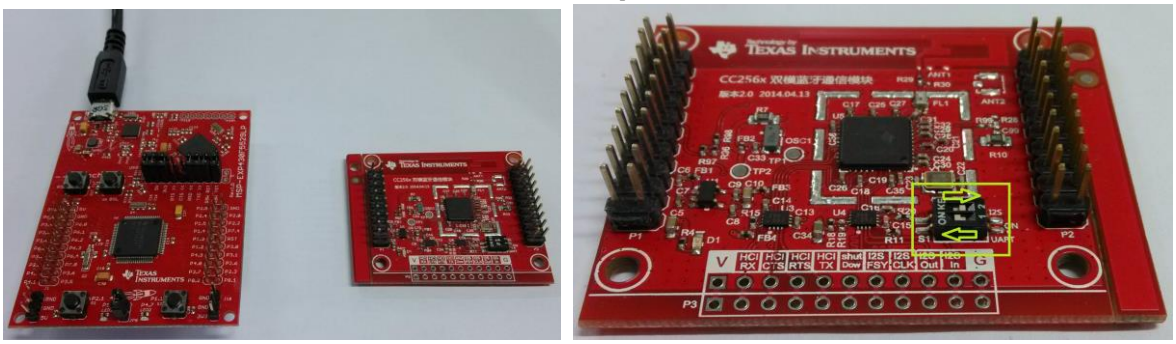

图 1、MSP430F5529 与 AY-CC2564EVM Module 展示

其中值得特别关注的是：在 AY-CC2564EVM 蓝牙模块上有一个拨码开关（上右图），**需要将 UART 档位拨至 ON 而关闭 I2S 连接**。也就是说靠下侧的拨码开关拨向左侧，上面的拨码开发拨向右侧。

2.1.1 模块插装：

两个模块上面都有一组四排 40 个插针的 BoostPack 通用接口，其中 LaunchPad 的背面有一组 BoostPack 的母口，而 CC2564EVM Module 上有一组 BoostPack 的公口。将 CC2564EVM Module 插在 LaunchPad 的背面（**注：两个模块的字体方向须一致！**）插上后如下图所示：

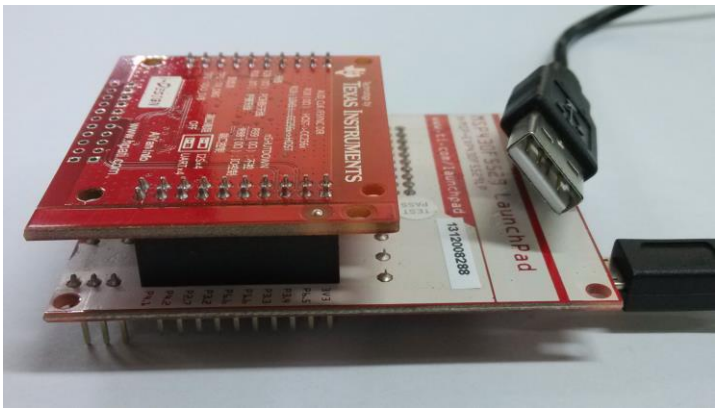

图 2、模块插装效果展示

2.1.2 连接电脑：

如图 2，LaunchPad 拥有一个 Micro USB 接口可以直接连接上电脑的 USB 口。整套模块会由 USB 供电后而自动上电，插上电脑后 LaunchPad 和 CC2564 模块上的指示灯会亮起。

2.2 软件环境创建

2.2.1 CCS 安装：

CCS 的安装参考 CCS 的用户手册即可以。值得强调的是 CCS 是一个强大的集成平台，它能够做很多 MCU、DSP 等不同平台的开发工作，根据用户的使用倾向可以选择安装自己所针对的开发平台。在本说明文档主要针对的是 MSP430 的开发，所以在安装过程中勾选上 MSP430 系列的选项。

2.2.2 Bluetopia SDK 安装：

进入 TI 官网的 SDK 安装连接后，实际下载的是一个叫 CC256XMSPBTBLESW-v1.4 R2-Setup.zip 安装压缩包文件，解压后按照正常的安装一步一步执行即可。默认安装模式会在 C:盘下的 ti 文件下生成一个 Connectivity 文件夹，其中包涵了蓝牙开发所需要的各种驱动库文件、文档说明、例程等诸多资源文件。

3. 蓝牙程序的运行

在 Bluetopia 针对蓝牙设计的系列 Bluetooth Demo 中只有 SPPLEDemo_Lite 是可以支持 MSP430F5529 芯片上运行无线蓝牙，而其他的 Demo 程序都是没有办法进行移植的。并且 SPPLEDemo_Lite 是不支持临时输入输出的 Demo 程序，也就是说再调试过程中是没有办法像其他的 Demo 程序一样可以通过 Debug UART 虚拟串口和 MCU 保持通讯、发送和接收命令。其原因在于 Debug UART 口被 F5529 用于和 CC2564 的 BT UART 通讯了，所以这部分功能被简化掉了。

3.1 SPPLEDemo_Lite 工作空间的创建：

建立专属于 Bluetooth 的 CCS 工作空间（WorkSpace），为了防止导入工程变得复杂化建议用户直接使用 MSP430 自带的 Sample 所在的路径作为工作空间。如果开发者前期的安装都是使用的默认安装那么这个工作空间的位置应该是(C:\ti\Connectivity\CC256X BT\CC256x MSP430 Bluetopia SDK\v1.4 R2\MSP430_Experimentor\Samples)。

操作方法：

1、 选中 CCS 菜单：File->Switch WorkSpace->Other...

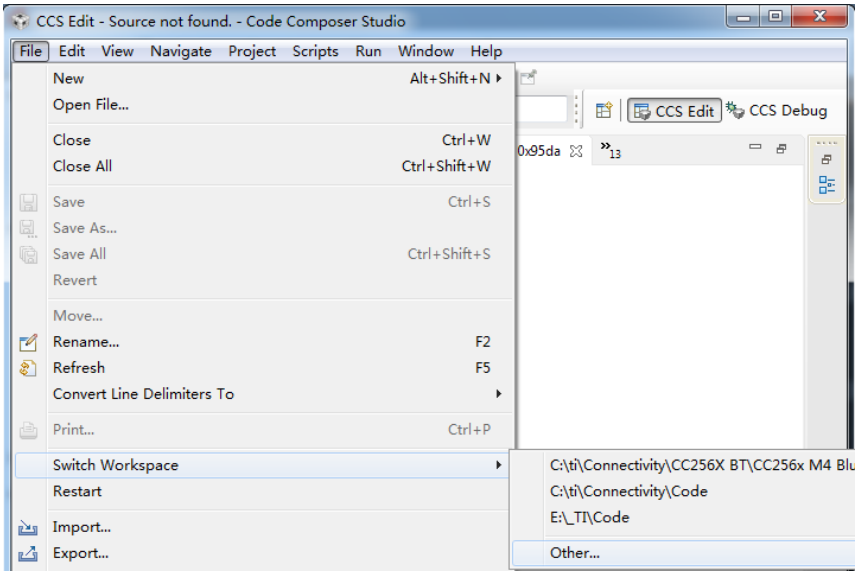

图 3、创建新的 Workspace

2、 弹出一个工作空间载入器（Workspace Launcher），点击 Browse...按钮；

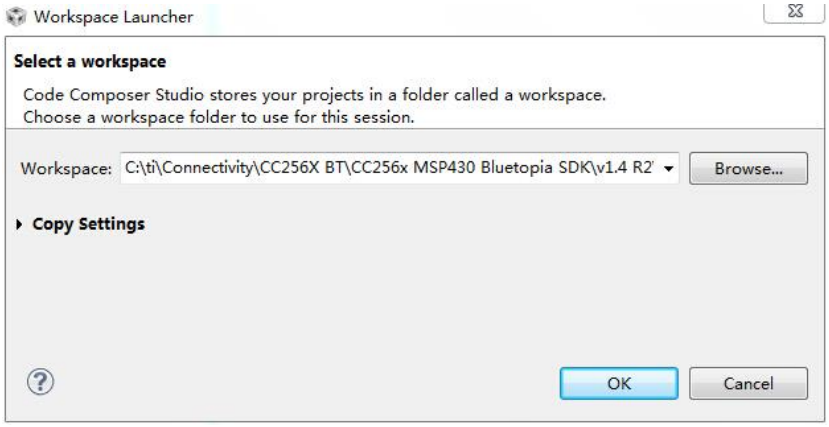

图 4、Workspace 选择提示框

3、找到（C:\ti\Connectivity\CC256X BT\CC256x MSP430 Bluetopia SDK\v1.4 R2\MSP430_Experimentor\Samples）所在的位置选中 Sample 确认即可。或者直接将绝对路径复制到 Workspace 后的输入栏中点击 OK 确认选中都可以。

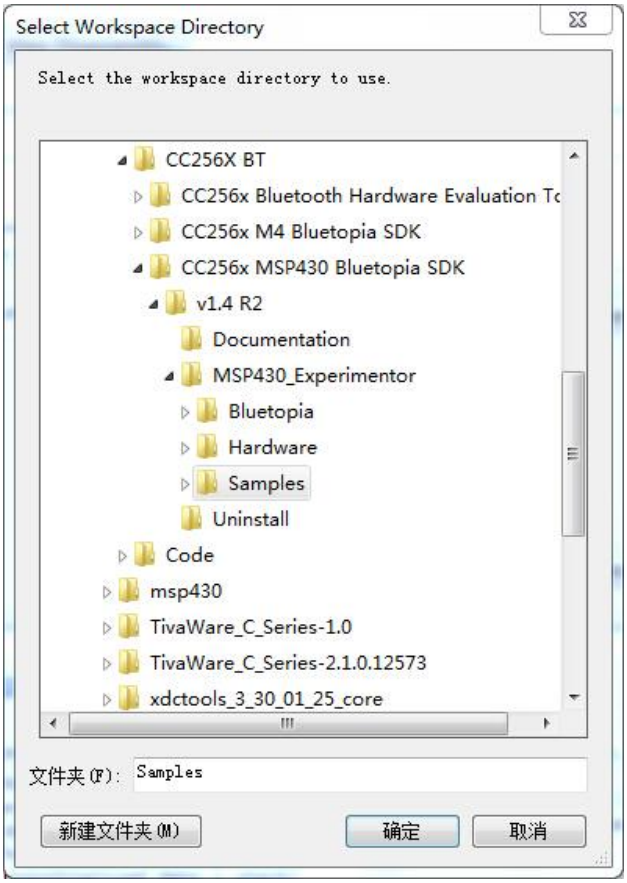

图 5、Workspace 空间选择

完成了上述步骤后，可以从 Sample 所在的系统文件夹中找到类似.metadata 的文件夹。这些都是 CCS 在创建完成 Workspace 后自动生成的文件系统。

3.2 SPPLEDemo_Lite 工程的导入

在 CCS 中导入一个已有的 project 有很多种方式，这里我介绍一种最常用的方式：

1、 CCS 菜单中选择 Project->Import CCS Project...

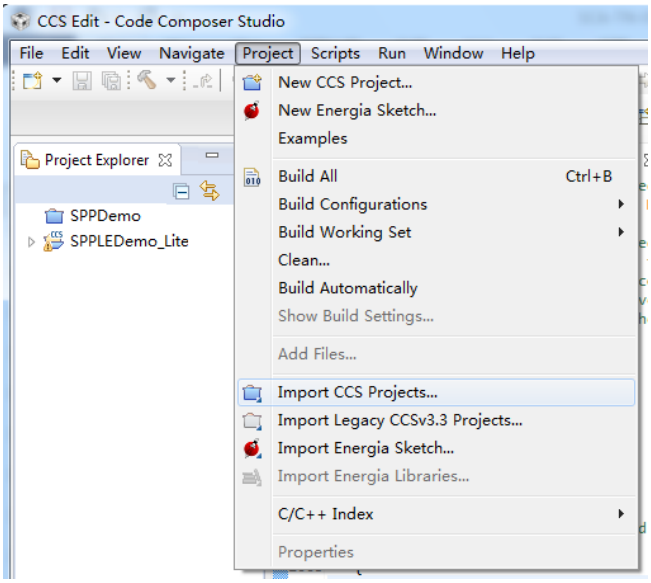

图 5、导入 Project

2、点击 Browse...按键，打开文件系统；

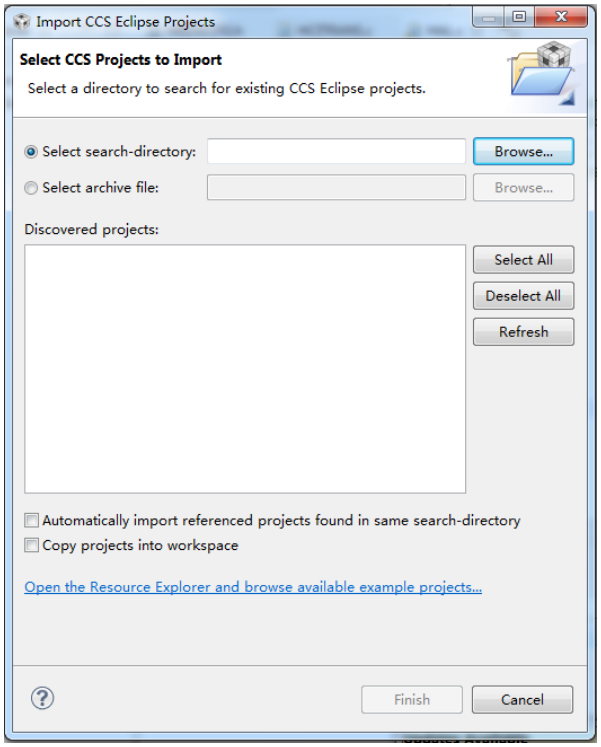

图 6、导入 CCS Project 提示框

2、 选择 SPPLEDemo_Lite 文件夹，确认返回；即可完成例程的导入工作。

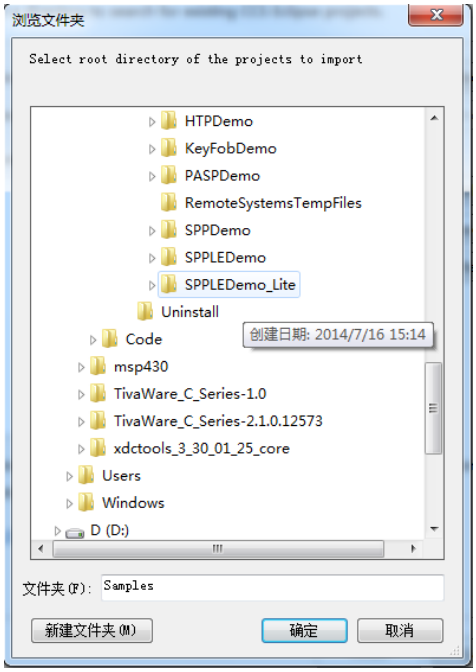

图 7、选择 SPPLEDemo_Lite 例程

3.3 SPPLEDemo_Lite 工程的移植修改

3.3.1 工程属性打开

打开工程属性框。右键工程名称弹出列表选中最下方的 Properties 选项

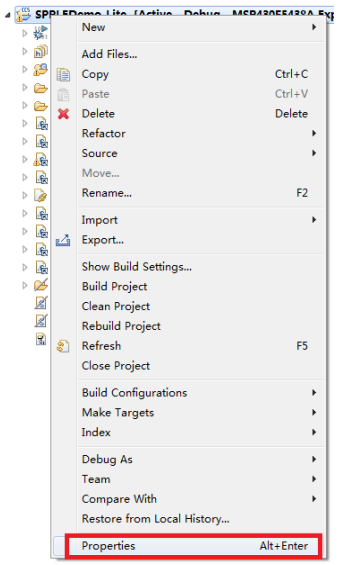

图 8、打开 Project 工程属性

3.3.2 修改芯片型号、编译器版本

选中属性框左侧的 General 选项，在右侧 General 选项框中将 MSP430F5438 原有的芯片名称换成 MSP430F5529。然后调整 Compiler Version 的版本，调整到最高的版本上。

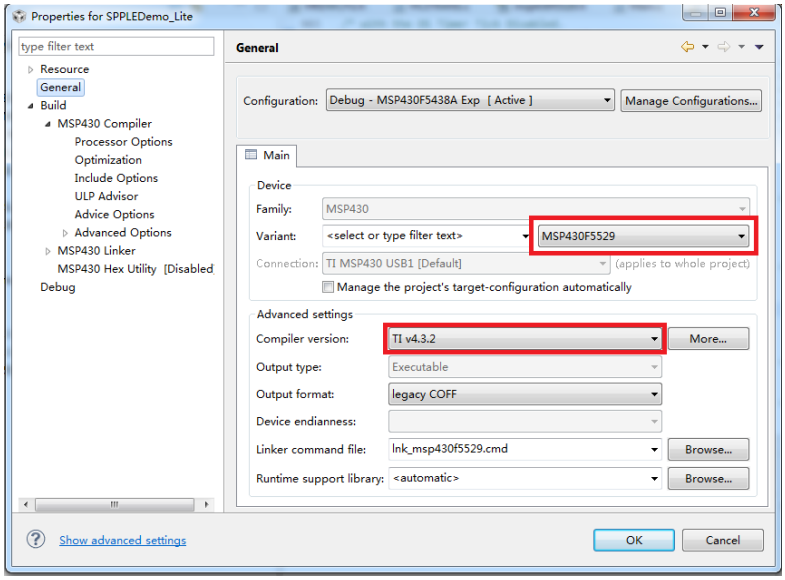

图 9、配置芯片类型和修改编译器版本

3.3.3 切换数据存储模式

将 Build->MSP430 Compiler->Processor Options 内的 Data memory model 更换到 small 模式下。

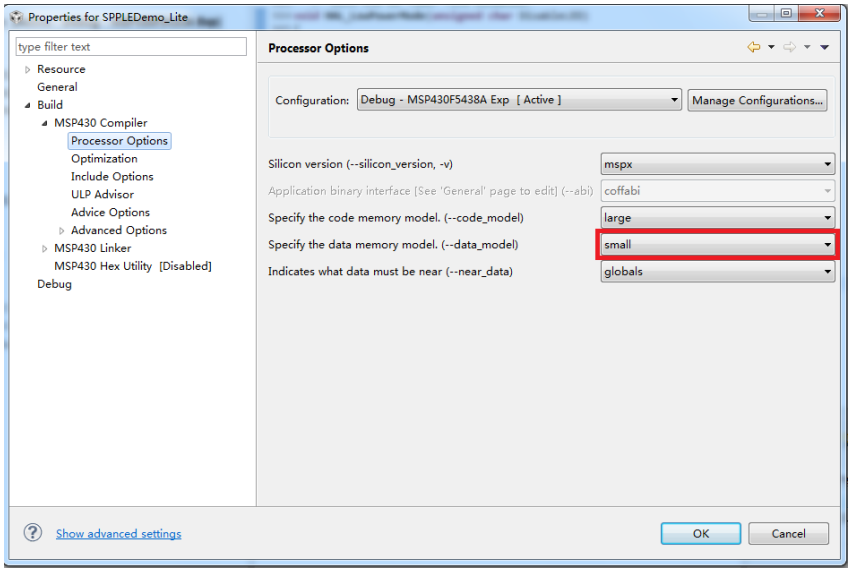

图 10、修改数据存储模式

3.3.4 修改引用头文件库

调整 Build->Include Options 下如图所示的引入路径将原有的 MSP430F5438 修改为 MSP430F5529

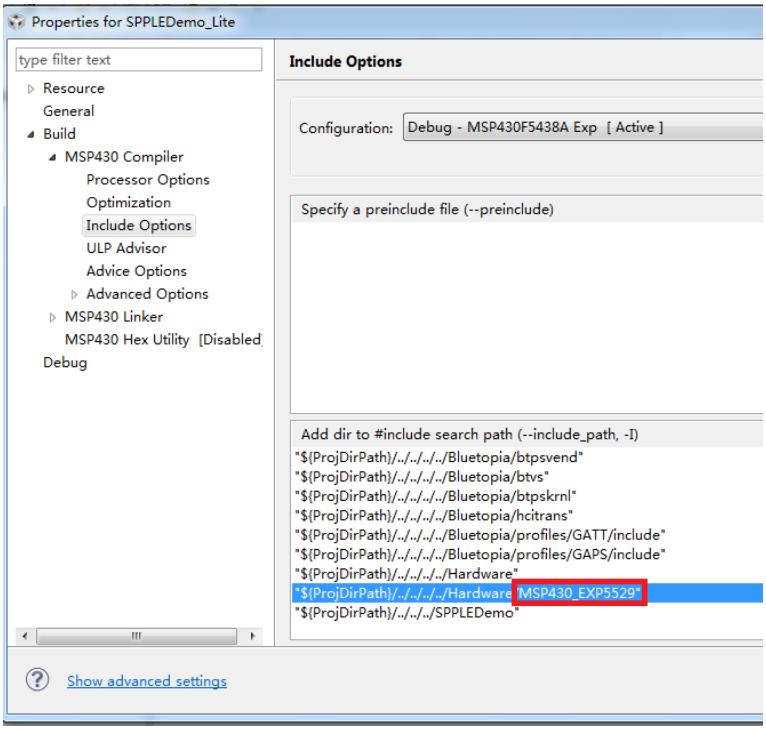

图 11、修改硬件库函数文件夹名称

3.3.5 添加 CC256XB 的预编译宏定义

在 Build->MSP430 Compiler->Advanced Options->Predefined Symbols 下添加“__SUPPORT_CC256XB_PATCH__”预编译宏定义名。原因在于我们使用的 CC256XB 系列。

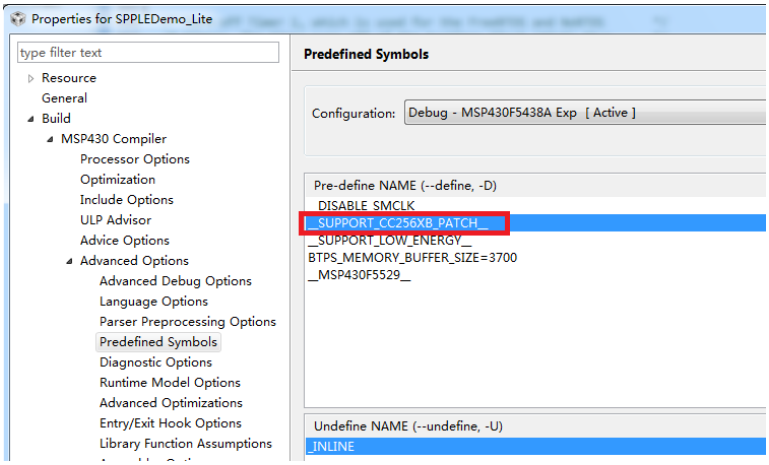

图 12、添加 CC256XB 系列预编译宏定义

3.4 SPPLEDemo_Lite 的代码修改

代码的修改可以参考 Bluetopia SDK 包中的帮助文档 C:\ti\Connectivity\CC256X BT\CC256x MSP430 Bluetopia SDK\v1.4 R2\Documentation 下面的“Hardware Porting Guidelines.pdf“。上面讲述了 MCU 由 MSP430F5438 转化为 MSP430F5529 需要注意和调整的所有细节。在这里我们抽取出了它的主要内容加以简单描述，便于快速调整开发。

首先我们先将 AY-CC2564EVM Module 使用的端口罗列出来：

CC256x 引脚信号	LaunchPad 定义	LP1	LP1X	LaunchPad 定义	CC256x 引脚信号
+3.3V	+3.3V	O	O	+5V	
	Analog In	O	O	GND	
HCI_TX	UART_RX	O	O	Analog In	
HCI_RX	UART_TX	O	O	Analog In	
nSHUTD	GPIO	O	O	Analog In	
	Analog In	O	O	Analog In	
	SPI CLK	O	O	Analog In	
	GPIO	O	O	Analog In	
	IIC_SCL	O	O	Reserved	
	IIC_SDA	O	O	Reserved	

CC256x 引脚信号	LaunchPad 定义	LP2X	LP2	LaunchPad 定义	CC256x 引脚信号
	PWM OUT/GPIO	O	O	GND	GND
	PWM OUT/GPIO	O	O	PWM OUT/GPIO	
	PWM OUT/GPIO	O	O	SPI CS	
HCI_CTS	PWM OUT/GPIO	O	O	GPIO	
HCI_RTS	TimerCap/GPIO	O	O	RST	
	TimerCap/GPIO	O	O	SPI_MOSI	
	GPIO	O	O	SPI_MISO	
	GPIO	O	O	GPIO	
	GPIO	O	O	GPIO	
	GPIO	O	O	GPIO	

图 13、AY-CC2564EVM Module 端口对应表

其中能够看到橙黄色标注的是 3.3V、GND、HCI_TX、HCI_RX、nSHUTD、HCI_CTS、HCI_RTS 七个端口。由于都是使用了 BoostPack 的通用接口，CC2564 和 F5529 的对应关系如下表所示：

表 1、F5529 与 CC2564 BoostPack 管脚对应表

MSP430F5529	CC2564
3.3V	3.3V
GND	GND
P3.4	HCI_TX
P3.3	HCI_RX
P1.6	nSHUTD
P1.4	HCI_CTS

P1.3	HCI_RTS

3.4.1　创建属于 F5529 自己的硬件引用库文件

我们必须根据上面分析，结合 F5529 特定的接口管脚配置硬件端口初始化。首先需要创建属于 F5529 自己的硬件初始化函数库。操作方式如下：

3.4.1.1新建引用库文件

在 C:\ti\Connectivity\CC256X BT\CC256x MSP430 Bluetopia SDK\v1.4 R2\MSP430_Experimentor\Hardware 文件夹下新建一个叫做 MSP430_EXP5529 的文件夹，将 ez430 下的 HAL.c、HAL.h、HRDWCFG.h 三个源文件拷贝到新建的 MSP430_EXP5529 文件夹下。

3.4.1.2修改引用库路径

右键工程中的 MSP430F5438 文件夹选中 Rename...选项，

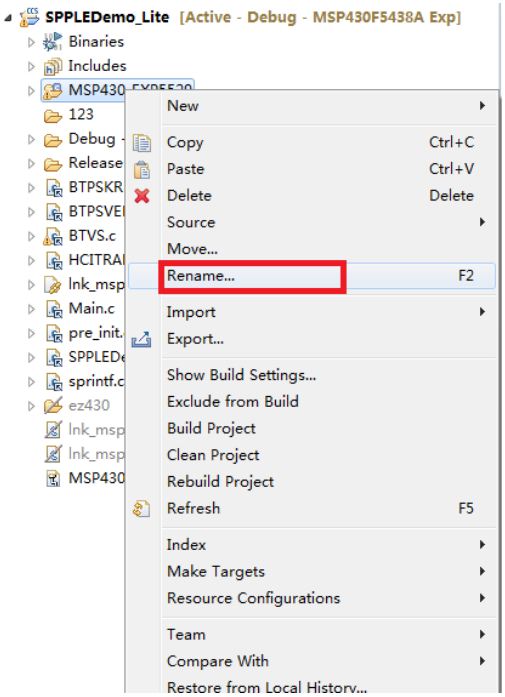

图 14、修改引用文件夹名称

将原来的 MSP430F5438 修改为 MSP430F5529；

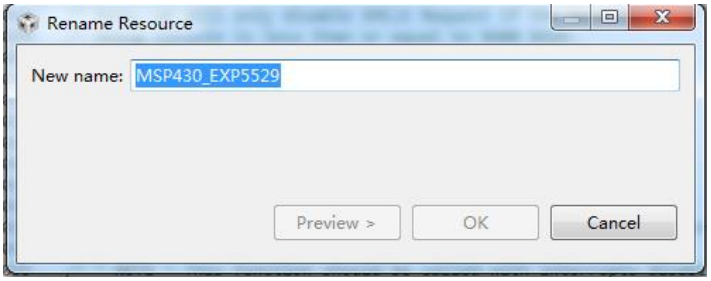

图 15、文件夹修改为 MSP430_EXP5529

3.4.1.3删除其中的三个源代码文件；

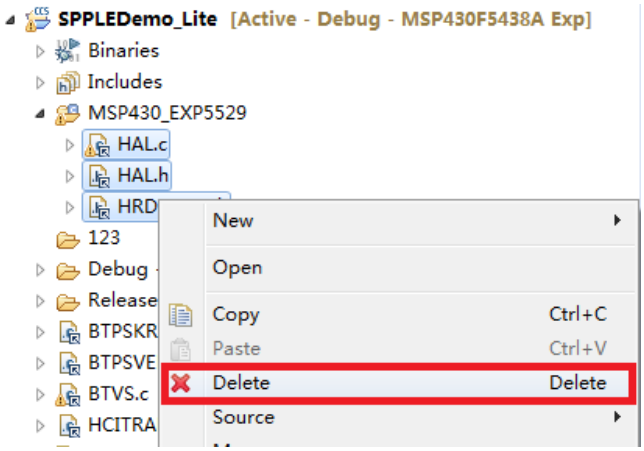

图 16、删除内部关联源码

3.4.1.4重新导入 MSP430F5529 文件夹中的三个源码文件；

可以通过右键工程选择 Add Files...添加到工程中，并移动到 5529 文件夹下。甚至可以直接通过文件夹拖动到 5529 文件夹中。

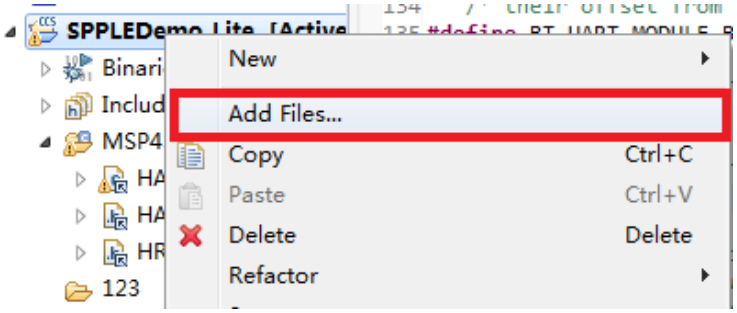

图 17、导入新的关联源码

3.4.2 修改 HRDWCFG.h

它是 MSP430F5529 硬件配置头文件。其中包含了供电、BT UART 连接等的管脚定义。

3.4.2.1 nSHUTD 定义修改

将源代码中关于 BT_DEVICE_RESET_PORT_BASE、BT_DEVICE_RESET_PORT_PIN 修改为如下的代码。

```
#define BT_DEVICE_RESET_PORT_BASE       ((unsigned char *)&P1IN)
#define BT_DEVICE_RESET_PORT_PIN        (BIT6)
```

3.4.2.2 CTS/RTS 定义修改

查找代码中关于下面所示代码的定义将原有管脚定义修改成如下所示的端口：

```
#define BT_UART_FLOW_RTS_PIN_BASE       ((unsigned char *)&P1IN)
#define BT_UART_FLOW_CTS_PIN_BASE       ((unsigned char *)&P1IN)

#define BT_UART_CTS_IV                  (PORT1_VECTOR)
```

```
#define BT_UART_CTS_IVR                 (P1IV)
#define BT_UART_RTS_PIN                 (BIT4)
#define BT_UART_CTS_PIN                 (BIT3)
#define BT_UART_CTS_INT_NUM             (P1IV_P1IFG3)
```

3.4.2.3 HCI_TX/HCI_RX 修改

同样的方法把 UART 的管脚定义修改为下列所示的源码：

```
#define BT_UART_MODULE_BASE             ((unsigned char *)&UCA0CTLW0)
#define BT_UART_IV                      (USCI_A0_VECTOR)
#define BT_UART_IVR                     (UCA0IV)
#define BT_UART_PIN_PORT_BASE           ((unsigned char *)&P3IN)
#define BT_UART_PIN_TX                  (BIT3)
#define BT_UART_PIN_RX                  (BIT4)
```

3.4.3 修改 HAL.c

3.4.3.1 LED 设置修改

LED 的端口设定结合 MSP430F5529 LaunchPad 的管脚配置将原来的管脚更换为现在的管脚：

```
static void ConfigureBoardDefaults(void)
{
        // LED init
        P1OUT = 0;
        P1DIR = 0xFF;
        P4OUT = 0;
        P4DIR = 0xFF;
}
static void ConfigureLEDs(void)
{
        P1SEL &= ~(BIT0);
        P1DIR |= (BIT0);

        P4SEL &= ~(BIT7);
        P4DIR |= (BIT7);
}
static void ToggleLED(int LEDID)
{
        if(P1OUT & BIT0)
        {
          P1OUT &= ~BIT0;
          P4OUT |= BIT7;
        }
        else
        {
          P1OUT |= BIT0;
          P4OUT &= ~BIT7;
        }
}
static void SetLED(int LED_ID, int State)
{
        Byte_t Mask;

        if(LED_ID == 1)
```

```
            {
            Mask = BIT1;
            } else{
            Mask = BIT7;
            }

            if(State){
                  if(LED_ID == 1){
                        P1OUT |= Mask;
                  }else{
                        P4OUT |= Mask;
                  }
            }else{
                  if(LED_ID == 1){
                        P1OUT &= ~Mask;
                  }else{
                        P4OUT &= ~Mask;
                  }
            }
      }
```

3.4.3.2 注释掉 HAL_ConfigureHardware()中的部分代码

由于没有啦 Debug UART 通讯功能，那所对应的 UART 初始化配置和中断使能等函数就没有必要初始化和使能了，可以将其注释掉。

```
      //   /* Configure the UART-USB Port for its default configuration
*/
      //   HAL_CommConfigure(BT_DEBUG_UART_BASE, BT_DEBUG_UART_BAUDRATE, 0);
      //   GPIOPinTypeUART(BT_DEBUG_UART_PIN_BASE, BT_DEBUG_UART_PIN_TX_MASK,
      BT_DEBUG_UART_PIN_RX_MASK);
      //
      //   /* Enable Debug UART Receive Interrupt.
*/
      //   UARTIntEnableReceive(BT_DEBUG_UART_BASE);
```

3.4.3.3 删除 DEBUG_UART_INTERRUPT()

同样的方法可以删除或者注释掉 Debug UART 中断响应函数。

3.4.3.4 清除或注释 HAL_ConsoleRead()、HAL_ConsoleWrite()中代码；

3.4.3.5 注释掉 CTS_ISR()中的 P1IV_P1IFG2

```
      __interrupt void CTS_ISR(void)
      {
       switch(BT_UART_CTS_IVR)
      {
      //      case P1IV_P1IFG2:
      //         /* This is the accelerometer interrupt pin on the ez430 so
just*/
      //         /* exit LPM3 if this interrupt occurs.
*/
      //         LPM3_EXIT;
```

```
//          break;
…
```

到这里，我们完成了 SPPLEDemo 移植到 MSP430F5529 LP 平台所有的工程建立、修改配置、代码调整的内容。接下来可以对整个工程进行编译工作，首先右键点击工程在弹出菜单中选择 Clean Project 清除原有的工程 Debug 过程中生成过程文件，然后重新编译工程 Build Project。

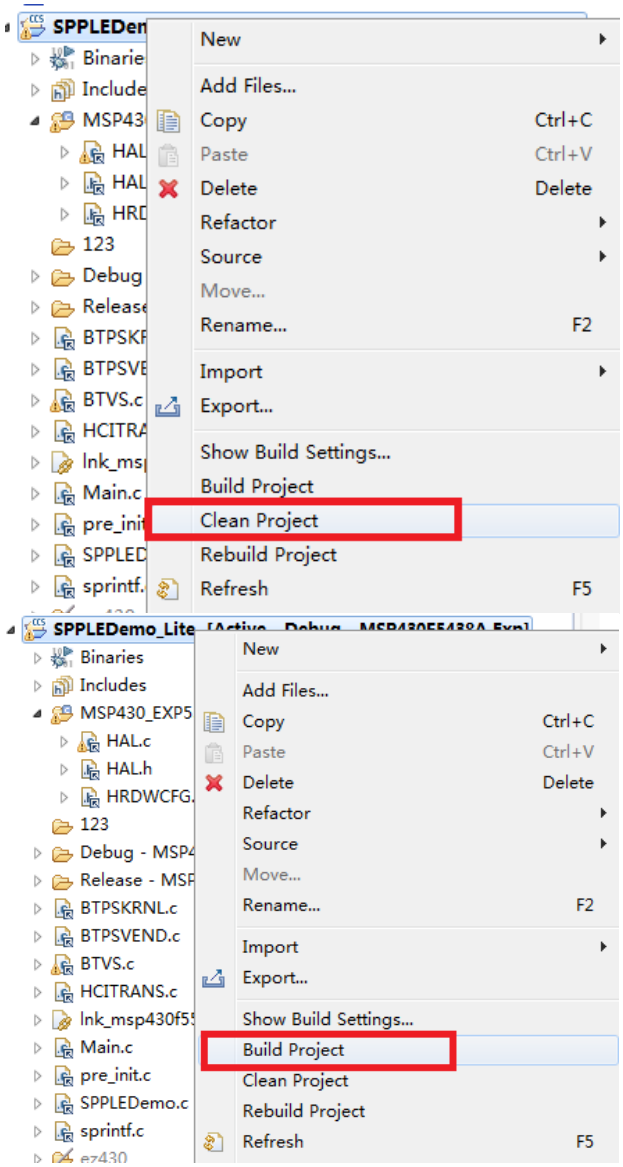

图 18、Project 工程清理和重新编译

如果完成了编译而且未发现 Errors，表示整个 project 编译通过并生成了 SPPLEDemo_Lite.out 的文件。若没有成功，请查找相应的报错信息，并根据上文的配置方式调整配置或者 Project 内容直至编译完成。

最后将 F5529 和 CC2564EVM 模块板链接上 PC 机，点击 CCS 界面中的绿色小虫子（Debug）将编译生成的文件仿真烧写到 LP 中。

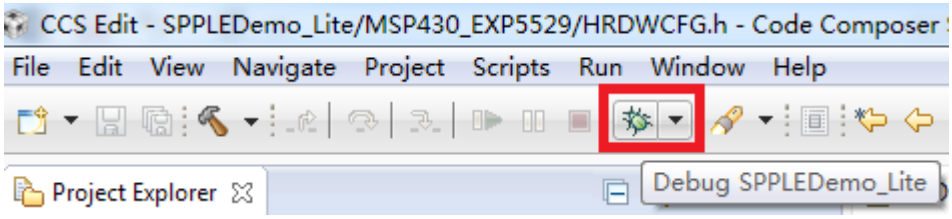

图 19、Project 的仿真烧写到目标板

完成烧写后，程序会自动跳转到程序入口主函数（Main）的第一行可执行代码上等待执行命令，接下来点击调试功能框中的绿色三角按钮（Resume）执行运行操作，程序就会开始正常的运行起来。如图 20 所示：

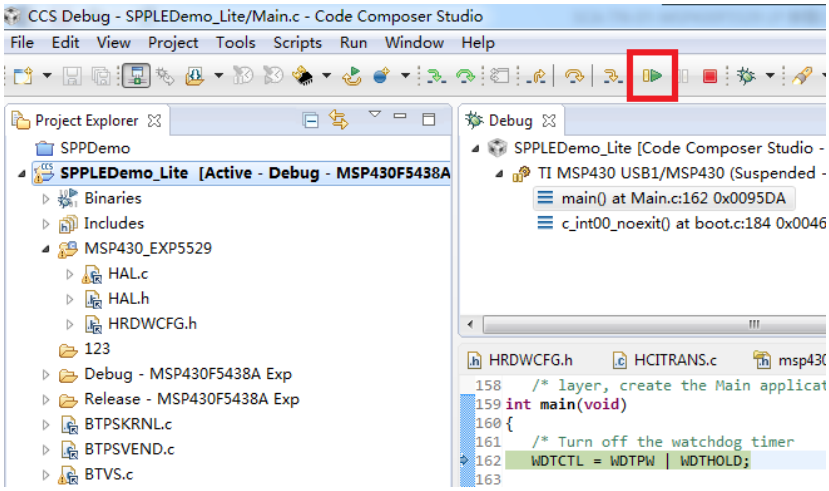

图 20、烧写完成后的结果

3.5 SPPLEDemo_Lite 的验证测试

成功的完成了工程的烧录后，接下来就可以测试无线蓝牙模块是否真正的可以完成无线的链接和数据传输功能。

操作步骤如下：

3.5.1 硬件接线和链接

如果已经能够完成程序的烧写说明链接方式正确，只要保持烧写时候的状态即可（**注：点击 Resume 后不要复位和断电重新连接**）。如下图所示：

图 21、硬件连接示意图

3.5.2 手机端 APP 安装

目前我们采用的是 Android 平台的 Mi 2s 手机作为测试样机（iOS 平台尚未做测试）。在测试手机上安装一个蓝牙串口的 APP 程序。安装方式和普通的手机应用一样，可以从第三方手机应用管理软件中搜索安装也可以打开浏览器直接下载安装。安装完成后，会在手机桌面找到一个蓝牙串口的图标。

下载链接：

http://apk.gfan.com/Product/App234389.html

图 22、蓝牙串口 APP 安装后效果

3.5.3 开启手机蓝牙

打开手机设备管理功能，启动蓝牙功能并搜索蓝牙设备。经过一段时间的搜索便能够找到可用设备
SPPLEDemo_Lite。

图 23、启动蓝牙发现蓝牙模块

3.5.4 蓝牙配对连接

在可用设备中发现 SPPLEDemo_Lite 后,点击这个设备会弹出一个蓝牙配对的对话框提示输入配对所需的 PIN 密码。请输入 0000，如下如所示：

图 24、蓝牙配对 PIN 密码输入

等待片刻后 SPPLEDemo_Lite 便会由可用设备变换为已配对设备。如下图所示：

图 25、完成蓝牙连接配对

3.5.5 蓝牙串口连接

完成手机和蓝牙模块的配对后，可以打开蓝牙串口 APP。它的界面比较简单，单击 APP 左下角的连接按钮后就会弹出一个可以链接的蓝牙设备列表其中就包涵我们刚刚配对成功的蓝牙模块。选中后便会直接连接上。

图 26、蓝牙串口连接模块

3.5.6　数据收发测试

直接在 text 对话框中输入一些简单的字符串，点击界面右下角的发送（只有真正的连上蓝牙模块才能发送字符串）。便会在界面中央看到由手机发送给蓝牙模块的数据已经从蓝牙模块返回来的数据。本例程实现的就是一个简单的接收即发送的测试。

图 27、无线蓝牙通讯示意效果