GHOSTYU 谷雨

串口透传

基于 BLE 的双向透明传输

Ghostyu.com

2014/1/14

[在此处键入文档的摘要。摘要通常是对文档内容的简短总结。在此处键入文档的摘要。摘要通常是对文档内容的简短总结。]

目录

1 前言

对于传统蓝牙的透传，有几个无法避免的问题，功耗大，成本高，另外串口透传多数情况下并不需要较高的速率，使用低功耗蓝牙作为串口透传的载体，将非常合适，我们今天来在 SmartRF 开发板上，开发一个技术低功耗蓝牙的串口双向透传程序。

该串口透传，无需任何五向按键操作，完全在串口调试助手里实现。先通过《实践 5：串口控制蓝牙》中的 AT 命令完成主从机的连接，然后就可以双向透传了。

2 必要条件

A 硬件

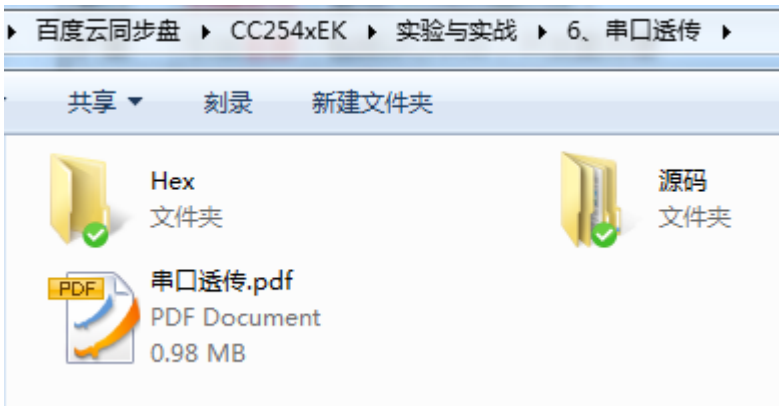
1、 两个 SmartRF 系列开发板，CC2540 或者 CC2541
2、 CC-Debugger 仿真器
3、 MiniUSB 线（用于 NewSmartRF 开发板连接 PC 的 USB）或直连串口线（用于 SmartRF 开发板连接 PC 的串口）

B 软件

1、 ble 协议栈，版本：1.3.2
2、 IAR for 8051 开发环境，版本：8.10
3、 Flash Programmer 固件烧写软件。
4、 串口调试助手。

3 文件预览

本文档的所有相关源码、说明均位于【CC254xEK\实验与实战\6、串口透传】目录下，如下图：



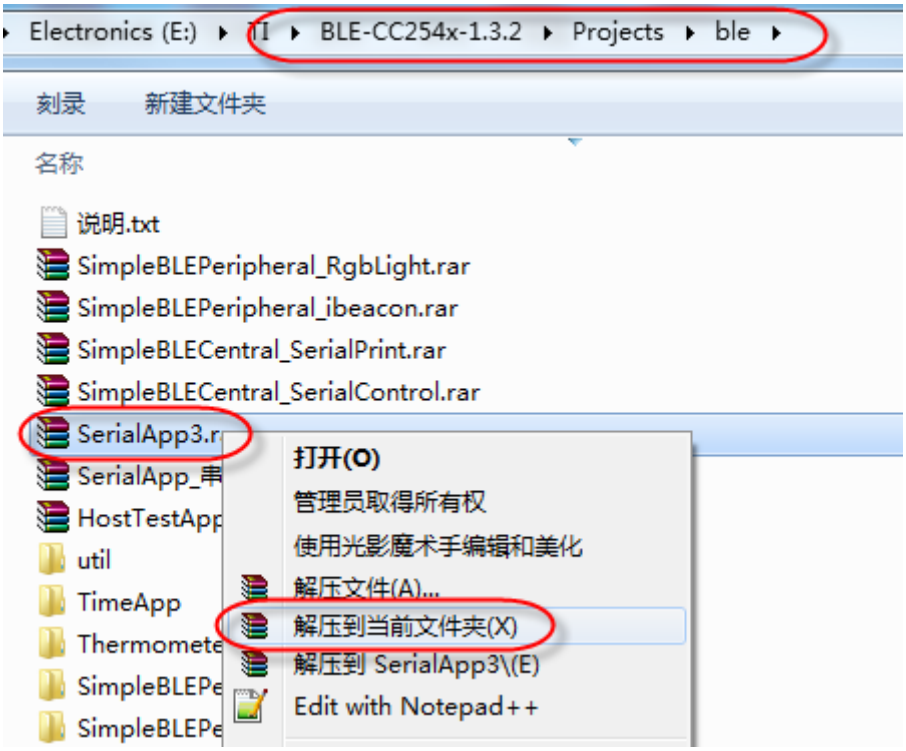
【Hex】文件夹存放我们预先编译 OK 的固件，可以直接下载到 SmartRF 系列开发板中测试运行。

【源码】文件夹存放的是该实践相关的源码程序

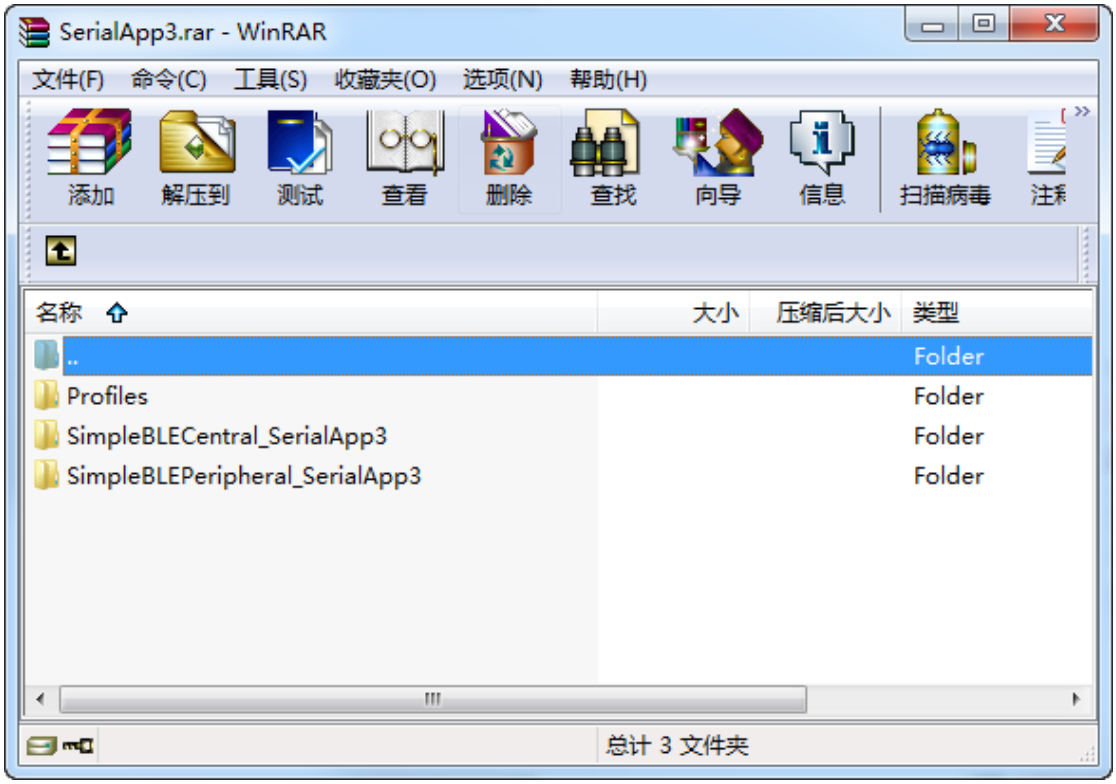
【串口透传.pdf】也就是本文档，在进行任何操作前请务必先仔细阅读。

4 源码包解压

将【\实验与实战\6、串口透传\源码\CC254x】下的压缩包，**复制到 1.3.2 版本的协议栈 projects 目录下，然后右击选择“解压到当前文件夹”，并且替换协议栈原有的源码，**如下图所示，务必注意，请勿“解压到 xxx”，否则会多一级目录，造成源码编译不通过。



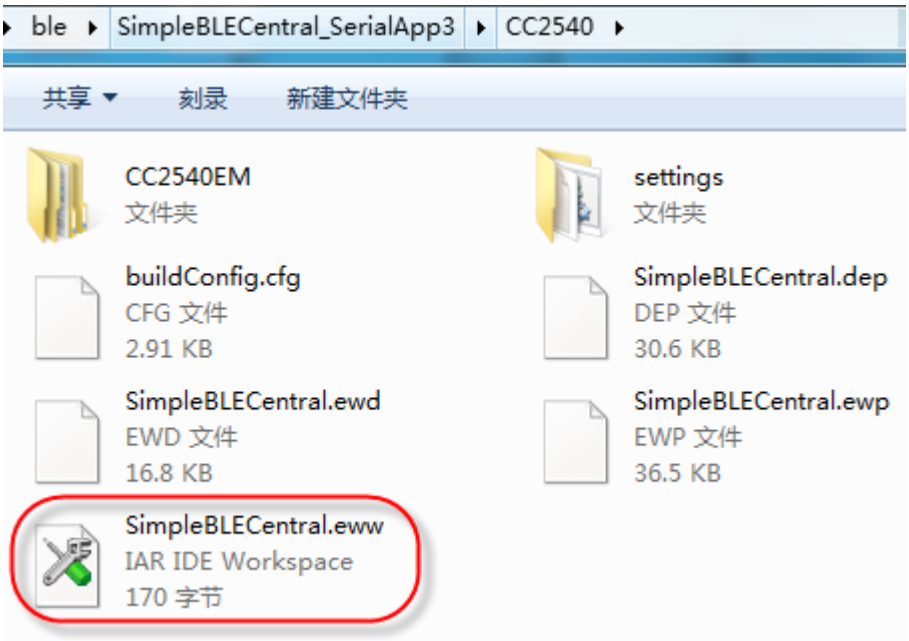
如果你打开了压缩包，你会发现有三个文件夹，如下图，主从机两个工程，我们已经重命名，但是程序中修改了 TI 提供的 SimpleProfile，所以源码包里提供了透传配套的修改后的 SimpleProfile，和源文件名相同，大家解压的时候会提醒是否替换原有文件，需要选择是。



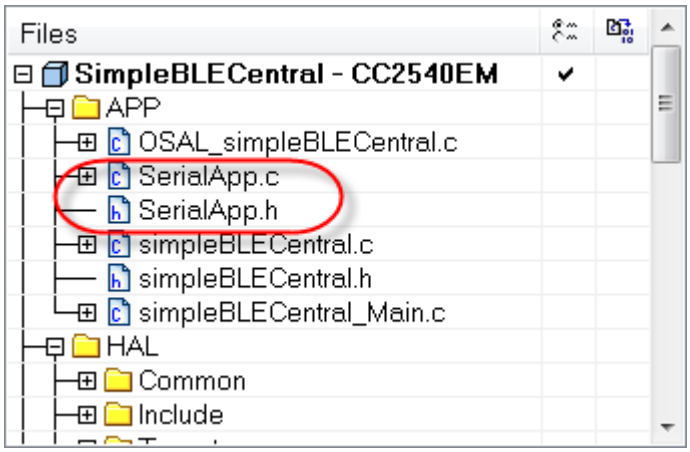
5 打开 IAR 工程

5.1 主机工程

接下来我们打开 SimpleBLECentral_SerialApp3 透传的主机工程，进入【BLE-CC254x-1.3.2\Projects\ble\SimpleBLECentral_SerialApp3\CC2540】，打开 IAR 工程，如下图，如果你使用的芯片是 CC2541，进入 CC2541 文件夹打开工程。



在 APP 目录，我们添加了两个文件，用来配置串口通信。另外，在 SIMPLEBLECENTRAL.C 文件的最后，是 GATT WRITE 函数。并且还有在《通过串口控制蓝牙》中的 AT 解析代码



首先打开 SerialApp.c 源文件，当 CC254x 接收到串口数据后会调用 sbpSerialAppCallback 函数，如下图：

```
00062: /*uart接收回调函数*/
00063: void sbpSerialAppCallback(uint8 port, uint8 event)
00064: {
00065:   uint8  pktBuffer[SBP_UART_RX_BUF_SIZE];
00066:   // unused input parameter; PC-Lint error 715.
00067:   (void)event;
00068:   int i=0;
00069:   for(i=6000;i>0;i--){
00070:     asm("nop");
00071:   }
00072:   //HalLcdWriteString("Data form my UART:", HAL_LCD_LINE_4 );
00073:   //返回可读的字节
00074:   if ( (numBytes = Hal_UART_RxBufLen(port)) > 0 ){
00075:     //读取全部有效的数据，这里可以一个一个读取，以解析特定的命令
00076:     (void)HalUARTRead (port, pktBuffer, numBytes);
00077:         if(pktBuffer[0]=='A' && pktBuffer[1]=='T'){
00078:           CommondHandle(pktBuffer, numBytes);
00079:         }else{
00080:           sbpGattWriteString(pktBuffer,numBytes);
00081:         }
00082:   }
00083:
00084: } ? end sbpSerialAppCallback ?
```

该函数接收全部的串口数据后，调用 CommondHandle 函数开始解析 AT 命令。CommondHandle 函数位于 simpleBLECentral.c 文件中。如下图程序片段，一共可以处理 7 条 AT 命令，大家可以更具需要添加更多的 AT 命令

AT

用于串口测试，如果程序运行并且串口通畅，会返回 OK

AT+ROLE?

获取当前角色，返回 Central

AT+SCAN

扫描从机，发送后 CC254x 开始 Discovery 从机，等待片刻后，返回找个的从机数量。

AT+CON[x]

连接指定的从机，x 为搜索到的从机序号，如果只扫描到一个从机，可以输入：AT+CON1 连接该从机。

AT+RSSI

获取当前 rssi 值，执行该命令后，程序会每个一秒打印一次 RSSI 值，再次发送该命令，停止 RSSI 值打印。

AT+DISCON

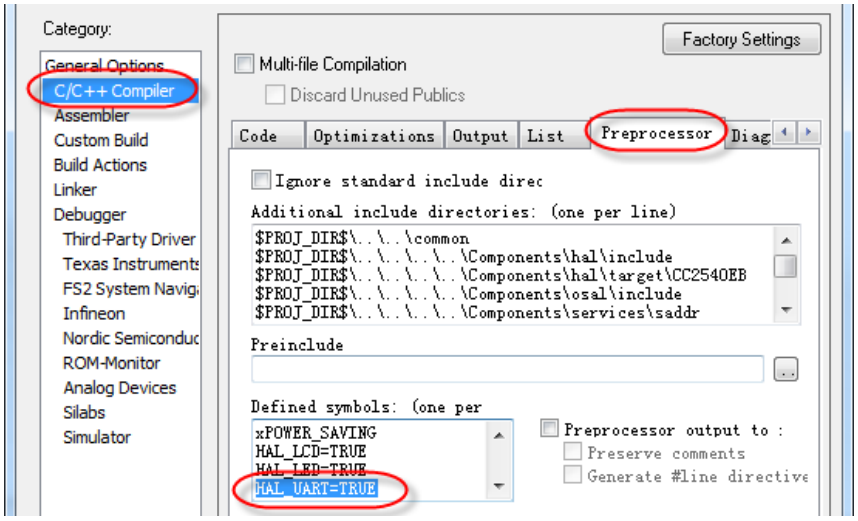
断开连接

AT+WRITE[0xXX]

向 Char1 写入特征值。如果要向从机 char 发送 0x15，输入发送命令：AT+WRITE0x15

```
01095: //AT          串口测试，返回OK
01096: //AT+ROLE?    获取当前角色
01097: //AT+SCAN     扫描从机
01098: //AT+CON[x]   连接指定的从机，x为搜索到的从机序号
01099: //AT+RSSI     获取rssi值
01100: //AT+DISCON   断开连接
01101: //AT+WRITE[0xXX]
01102: void CommondHandle(uint8 *pBuffer, uint16 length)
01103: {
01104:   if(length<2)
01105:     return ;
01106:   if(pBuffer[0]!='A' && pBuffer[1]!='T')
01107:     return ;
01108:   if(length <=4){
01109:     SerialPrintString("OK\r\n");
01110:     return ;
01111:   }
01112:   if(length>=8 && str_cmp(pBuffer+3,"ROLE?",5)==0){
01113:     SerialPrintString("Central\r\n");
01114:     return ;
01115:   }
01116:   if(length>=7 && str_cmp(pBuffer+3,"SCAN",4)==0){
01117:     simpleBLEScanning = TRUE;
01118:     simpleBLEScanRes = 0;
01119:
01120:     LCD_WRITE_STRING( "Discovering...", HAL_LCD_LINE_1 )
01121:     SerialPrintString("Discovering...\r\n");
01122:     LCD_WRITE_STRING( "", HAL_LCD_LINE_2 );
01124:     GAPCentralRole_StartDiscovery( DEFAULT_DISCOVERY_MOD
```

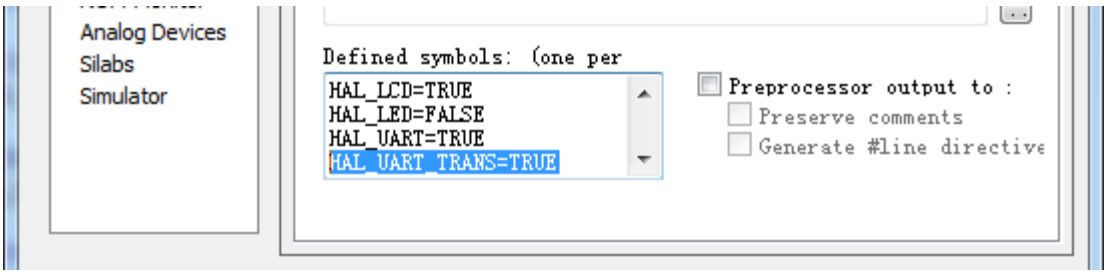
另外注意，我们已经在工程 Options 的 Preprocessor 中添加了 HAL_UART=TRUE 宏定义。否则底层的 uart 程序不会被编译到程序中。在当前 Configuration 上右击，然后选择 Options…



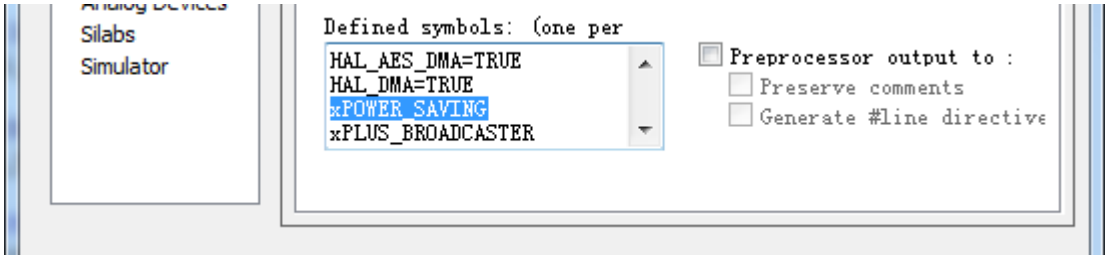
如果串口接收到的数据是非“AT”打头，那么认为是透传的数据，将直接通过 GATT Write 函数发送到从机。

```
01119: uint8 sbpGattWriteString(uint8 *pBuffer, uint16 length)
01120: {
01121:     uint8 status;
01122:     uint8 len;
01123:     if(length > 20)
01124:         len = 20;
01125:     else
01126:     len = length;
01127:     attWriteReq_t req;
01128:     req.handle = simpleBLECharHdl;
01129:     req.len = len;
01130:     req.sig = 0;//必须要填
01131:     req.cmd = 0;//必须要填
01132:     osal_memcpy(req.value,pBuffer,len);
01133:     status = GATT_WriteCharValue( simpleBLEConnHandle, &req, simpleBLETaskId );
01134:
01135:     return status;
01136: }
```

在上面的工程 Options 的 Preprocessor 中，除了要添加 HAL_UART=TRUE，还需要一个宏定义，来区分串口透传工程和一般的主从机工程，虽然我们替换了协议栈原先的 SimpleProfile 文件，但是并未破坏原来的结构，而是在代码中，添加了透传工程的宏定义。

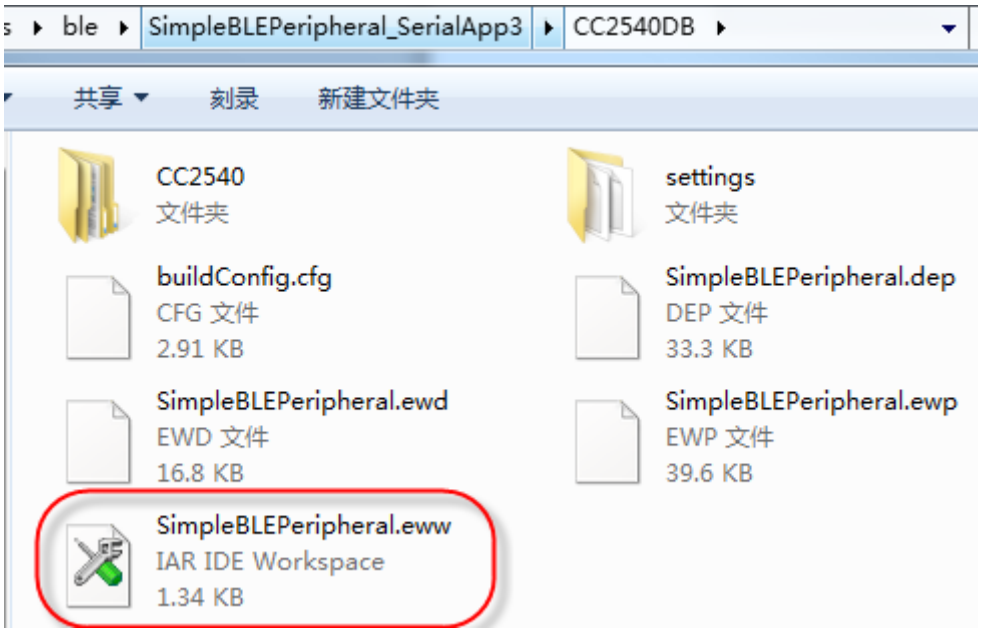


还需要设置，不能休眠。否则 UART 无法接收到完整的数据。在 1.4 的协议栈中可能有改善。在每条宏定义的前边加一个 x 表示，取消该条宏定义（其实任何字母都可以，为了改变了宏定义的名称而已）。



5.2 从机工程

从机工程和主机工程非常类似，尤其是串口代码部分，另外最大的不同是数据的发送上。



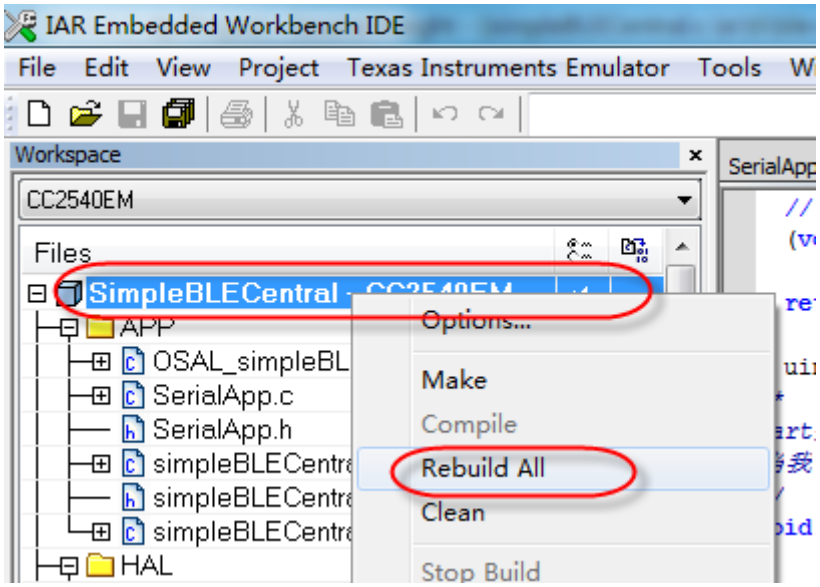
主机向从机发送数据是通过调用 GATT_WriteCharValue（GATT 的 client 主动向 service 发数据，这里的主机是 GATT 的 client），从机向主机发送数据是通过调用 GATT_Notification（GATT 的 Service 主动向 client 发送数据，这里的从机是 GATT 的 service）

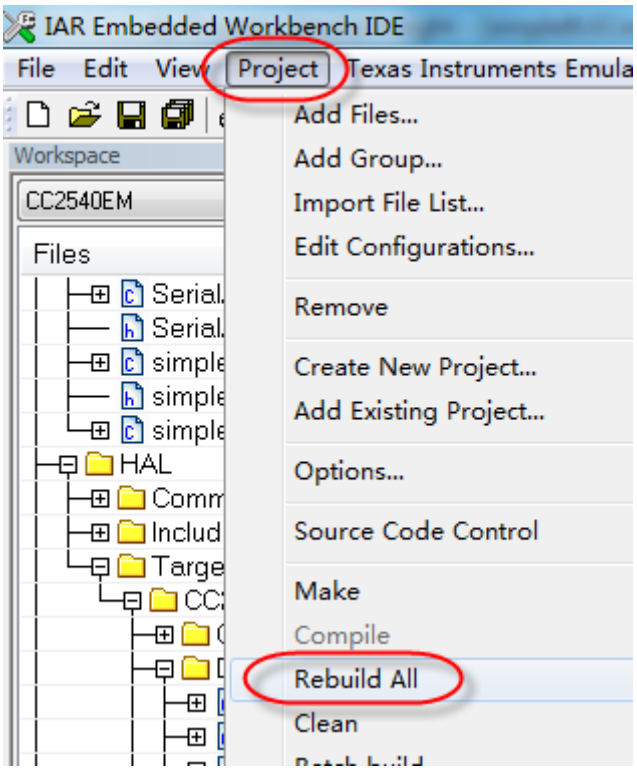
```
00082: void sbpSerialAppSendNoti(uint8 *pBuffer,uint16 length)
00083: {
00084:   uint8 len;
00085:   if(length > 20)
00086:     len = 20;
00087:   else
00088:     len = length;
00089:   static attHandleValueNoti_t pReport;
00090:   pReport.handle=0x2E;
00091:   pReport.len = len;
00092:   osal_memcpy(pReport.value, pBuffer, len);
00093:   GATT_Notification( 0, &pReport, FALSE );
00094: }
```

6 编译下载

在当前 Configuration 上右击，然后选择 Rebuild All，重新编译整个工程。或者选择菜单

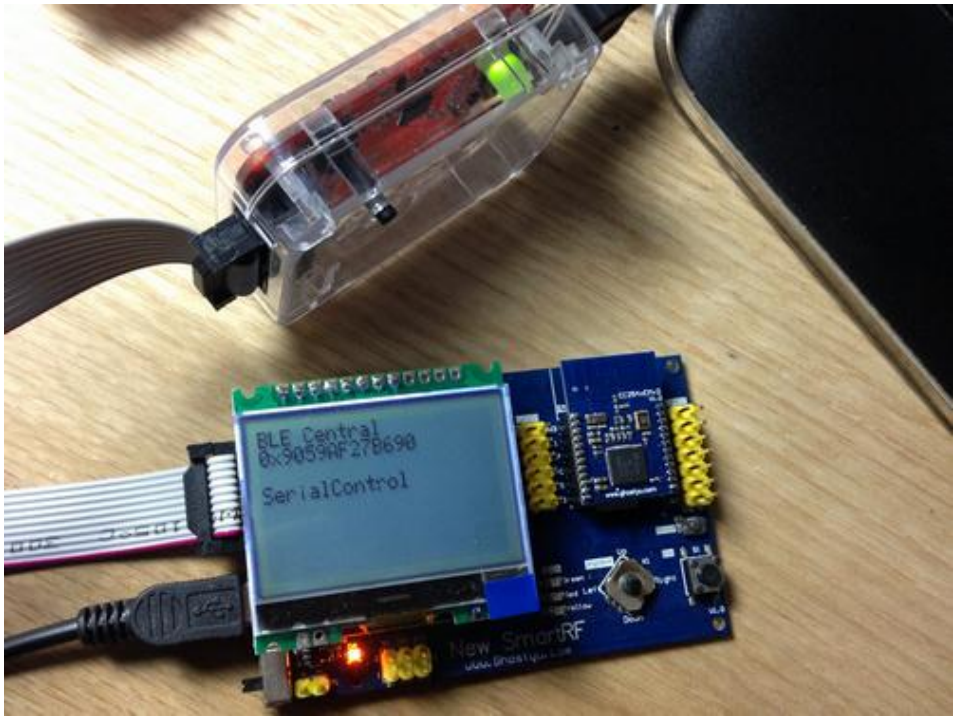
Project/Rebuild All。效果相同。





如果源码解压的位置正确，并且使用的是 1.3.2 的 ble 协议栈和 8.10 的 IAR 编译器，不会出现任何编译问题（可能有部分警告，一般是变量声明了，但未使用）。

连接 CC-Debugger 仿真器和开发板，准备烧写程序，注意，仿真器连接 OK 后务必按仿真器复位按钮，待 CC-Debugger 指示灯变绿后，再进行下一步操作。另外还需要连接开发板的与 PC 的串口（程序下载仅连接仿真器即可，连接仿真器的 usb，是准备下步骤的 uart 通信用。），使用 IAR 下载，或者使用 Flash Programmer 将程序下载到开发板中。

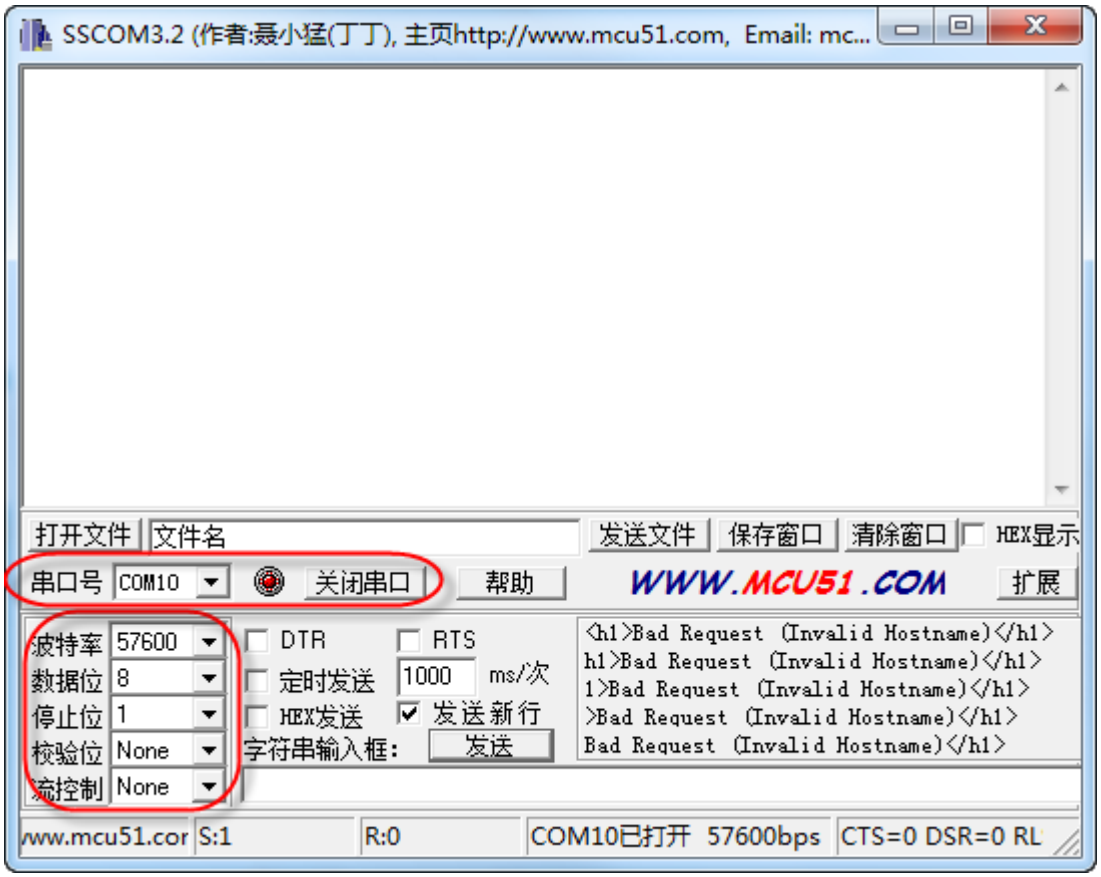


从机工程的烧写方法与主机工程完全相同。

最后连接如下

7 测试

按照上图连接开发板和电脑。然后打开串口调试助手。打开开发板对应的虚拟串口。如下图所示的参数设置。



7.1 主机连接测试

本节中的操作与《串口控制蓝牙》完全相同，使用简单的 AT 命令来控制蓝牙的链接过程。

1、测试程序与串口，发送 AT 指令。

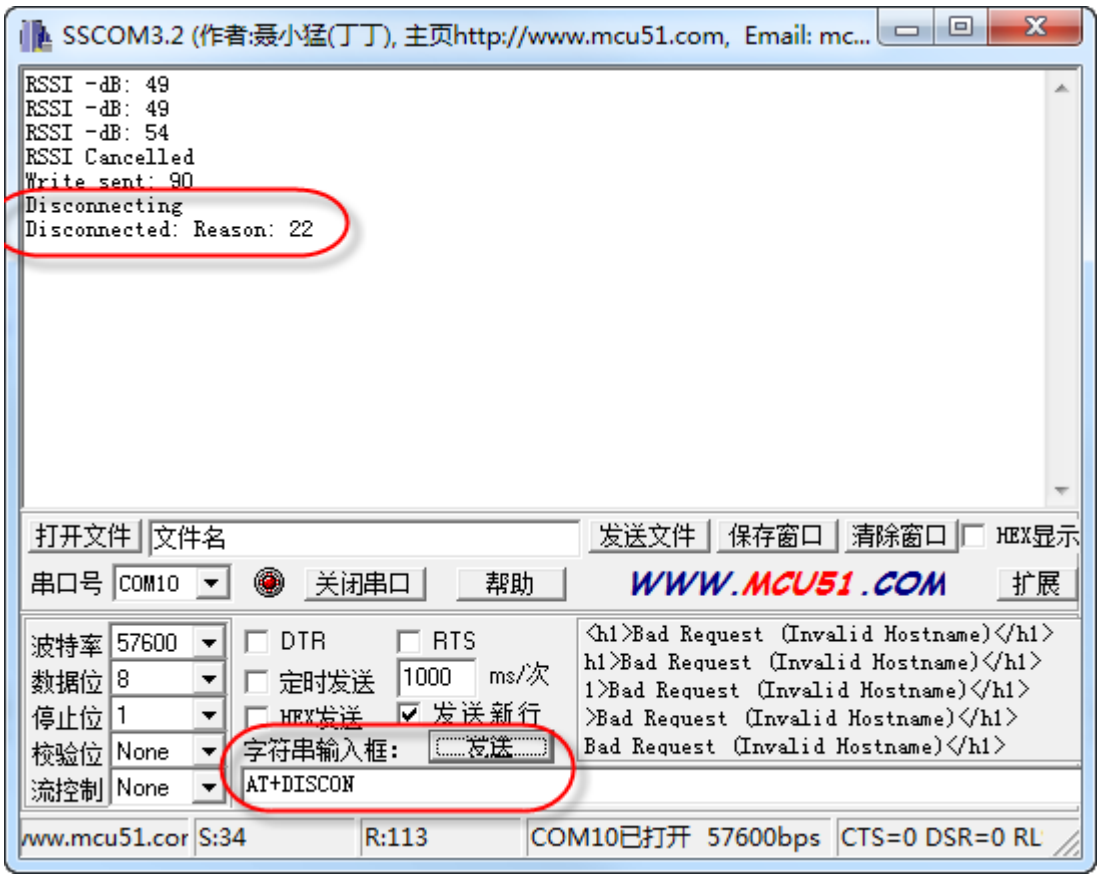
2、扫描从机，输入：AT+SCAN，点击发送

3、连接从机，输入：AT+CON1，点击发送

4、获取 RSSI 信号值，输入：AT+RSSI，点击发送，再次发送停止。

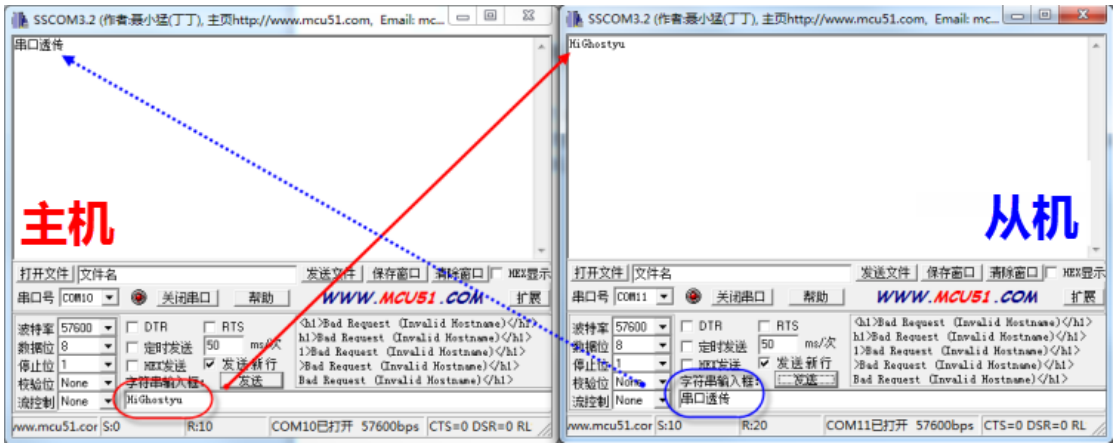
5、向 Char1 写入一个数，如 0x5a，输入：AT+WRITE0x5A

6、断开连接，输入：AT+DISCON



7.2 透传测试

使用 7.1 节中的 AT 命令连接到从机后，就可以串口透传了。
主机向从机发送数据，如果发送到主机串口的字符串不是 “AT”开头，会被直接发送到从机中，而发送到从机串口的任意数据都会被透传到主机。



8 智能机与 CC254x 之间的透传

8.1 iOS 设备
8.2Android 设备

联系我们：
刘雨 tel:15861666207
网站：http://www.ghostyu.com
技术支持：http://www.ghostyu.com/bbs
在线文档：http://www.ghostyu.com/wiki
官网店铺：http://ghostyu.taobao.com