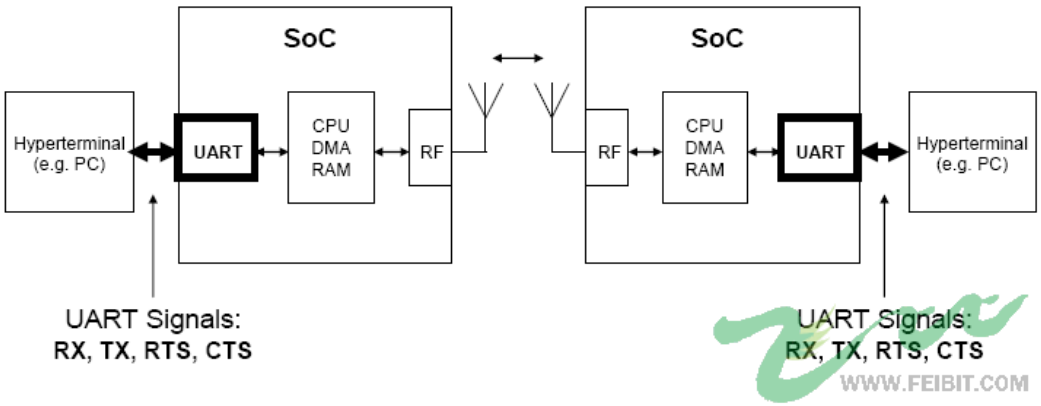
Z-STACK　串口通讯实验（SerialApp）

RS232，也称标准串口，是目前最常用的一种串行通讯接口，因其成本低廉，应用广泛而被很多嵌入式系统所采用。在 CC2530 开发板上，由于 LCD、LED 等基本外接显示信息量有限，同时串口也方便了与其他系统进行通讯，所以它无疑成为了开发者最重要的一个调试手段，同时也是其他系统升级为无线传输系统的一个重要途径。

第一节　　程序功能

实现两个节点之间的绑定与通讯，同时每个节点可与其“上位机”——所边接的 PC 串口终端，进行通讯。示意如下：

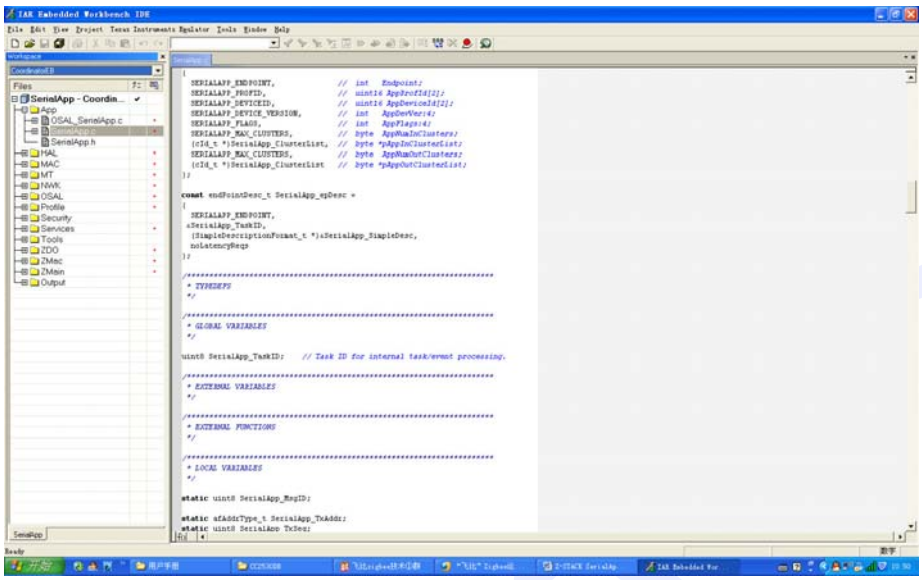


第二节　　软件编译与下载

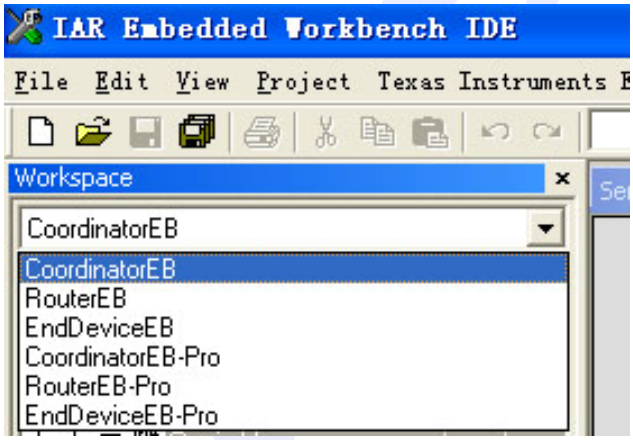
项目工程文件夹在

C:\Texas Instruments\ZStack-CC2530-2.3.0-1.4.0\Projects\zstack\Utilities\SerialApp\CC2530DB

双击工程文件 SerialApp.eww 进入 IAR 工程界面：

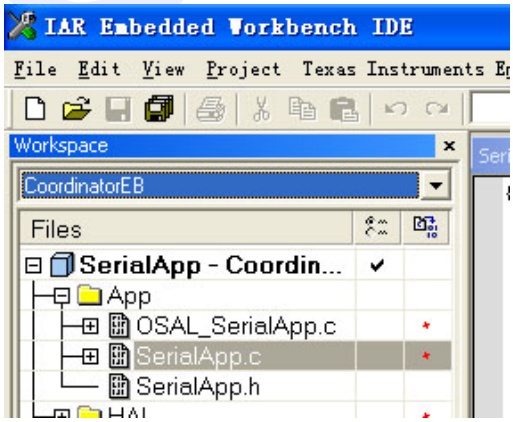


该工程包含六个子工程，如下图：

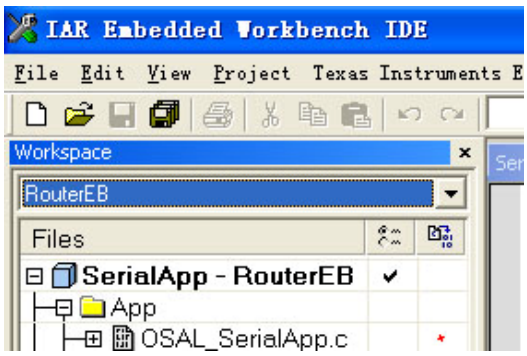


此时可选择工程编译的协议栈版本(Zigbee 2007/Zigbee Pro)和节点类型(Coordinator/Router/EndDevice)：

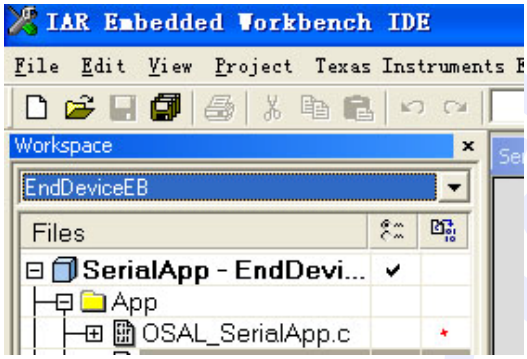
Zigbee 2007 的协调器节点：



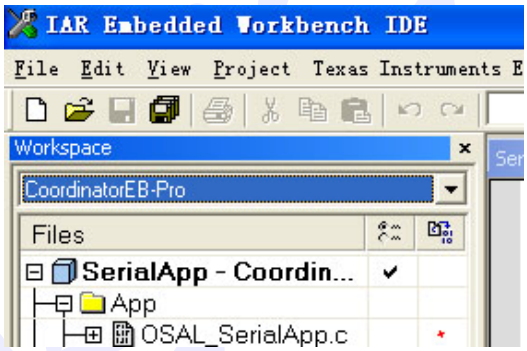
Zigbee 2007 的路由器节点：



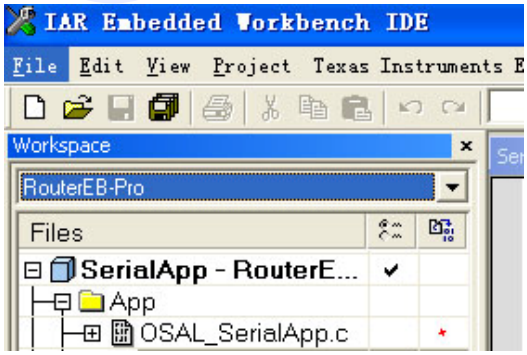
Zigbee 2007 的终端节点：



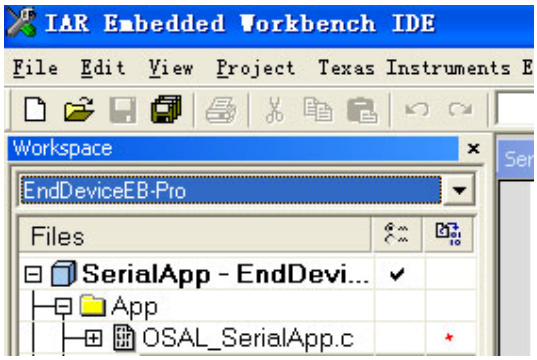
Zigbee Pro 的协调器节点：



Zigbee Pro 的路由器节点：

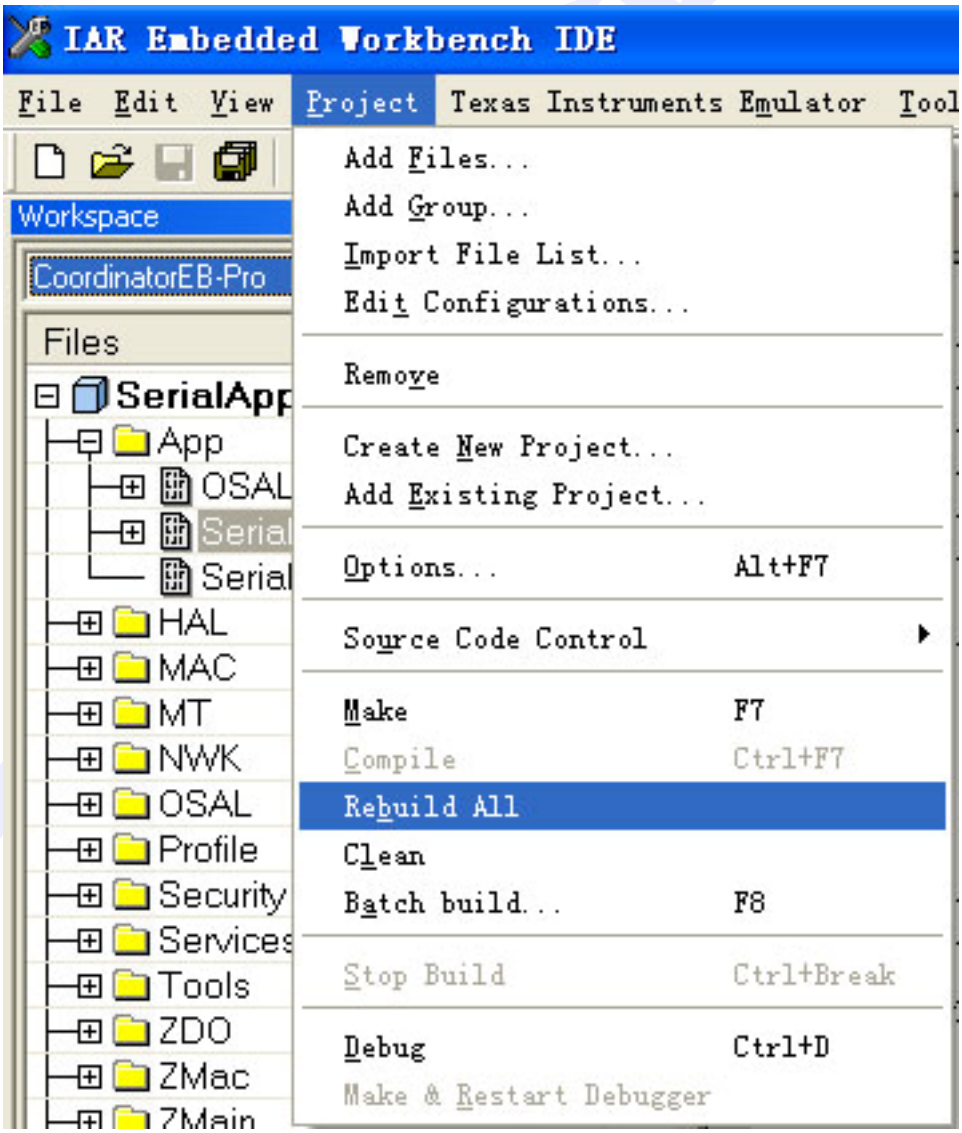


Zigbee Pro 的终端节点：

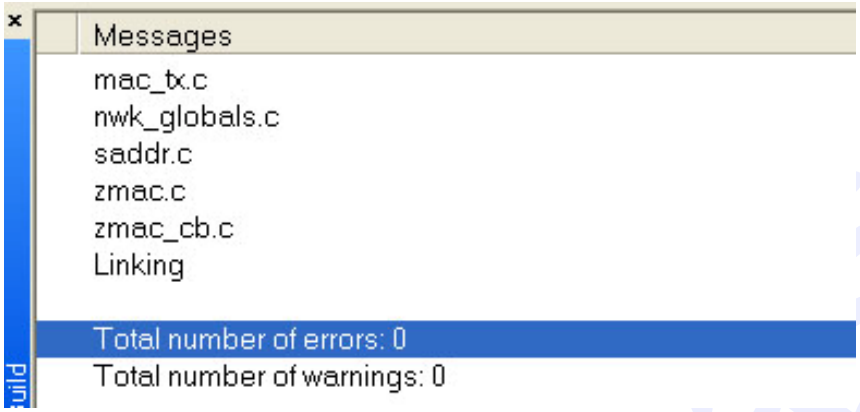


、 **配置 Zigbee Pro 的协调器节点：**

选择 CoordinatorEB-Pro，选择 Project—>Rebuild All 编译，如下图：

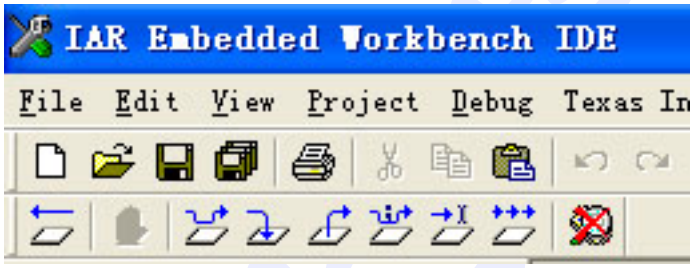


编译完成后，屏幕下方编译信息栏有如下信息提示（Errors 0，Warnning 0）：



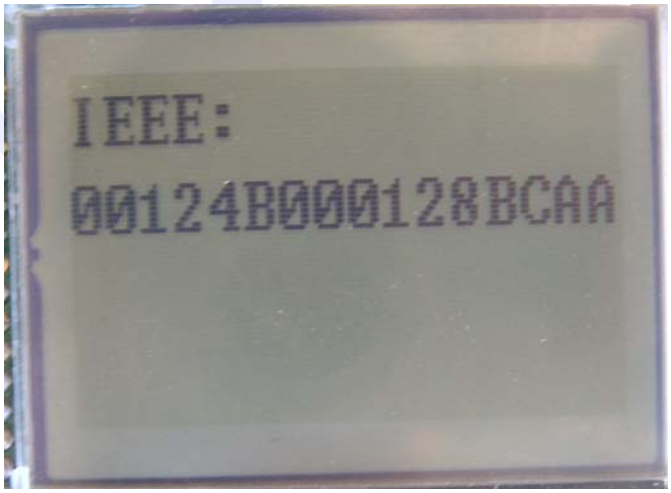
连接好 CC DEBUGGER 和目标板，开启目标板电源，按下 CC DEBUGGER 的 Reset 键，此时 CC DEBUGGER 的指示灯应为绿色。点击  开始向目标板下载程序。

下载完成后，进度条消失，左上角出现调试窗口：

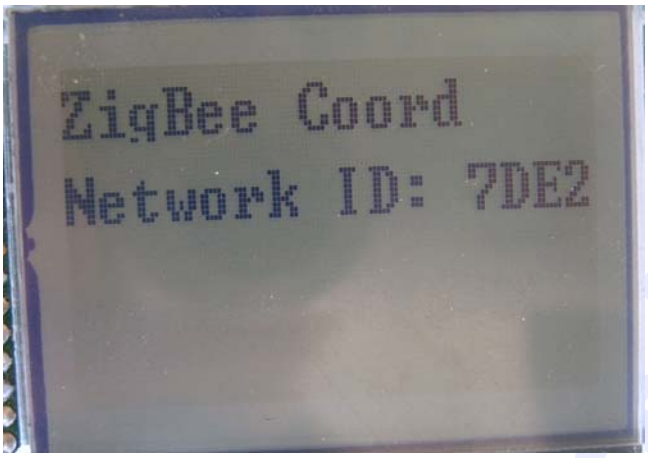


点击  ，退出调试状态，拔除目标板上的 DEBUG 线，重启目标板电源或按下 Reset 键。LCD 屏幕显示如下：

显示本节点 IEEE 地址：

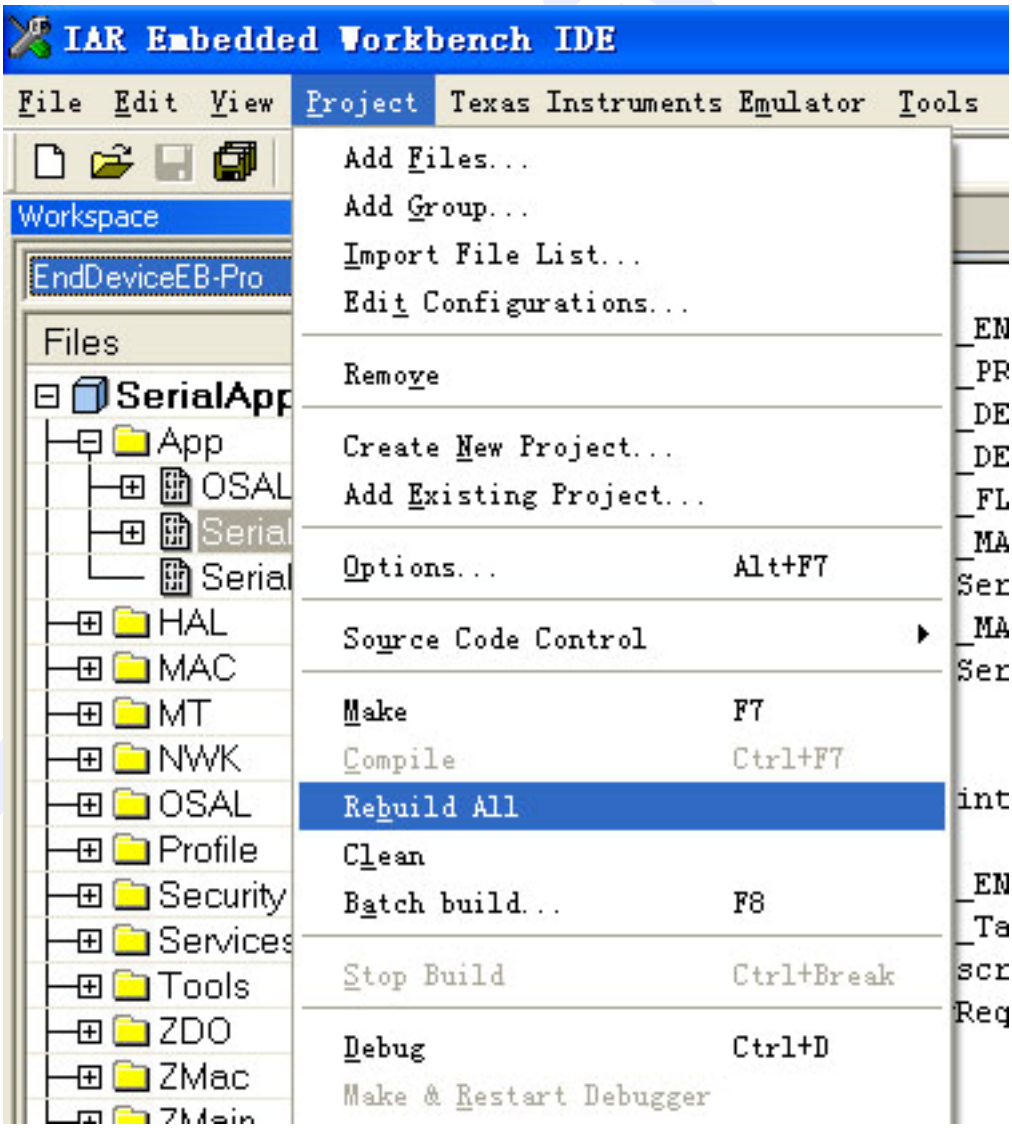


显示本节点功能类型：

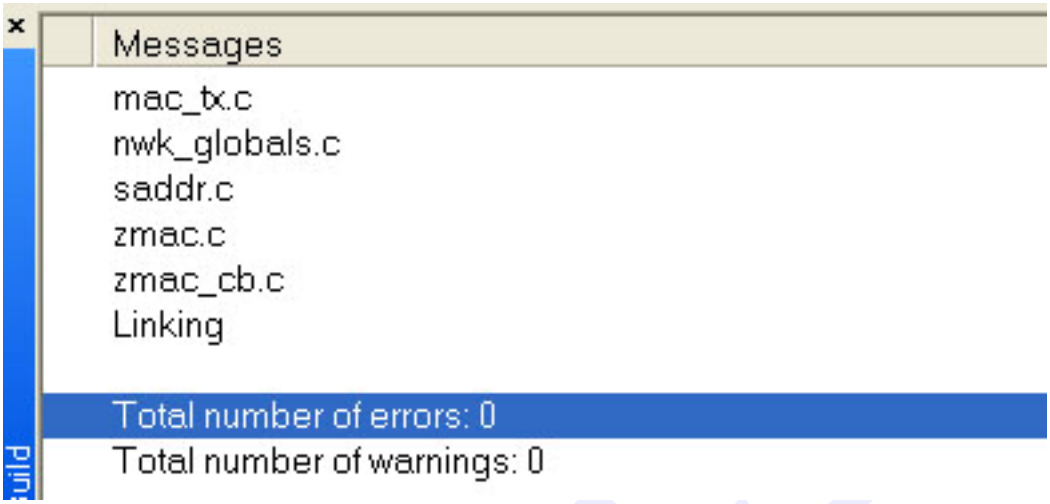


、 **配置 Zigbee Pro 的终端节点：**

选择 EndDeviceEB-Pro，选择 Project—>Rebuild All 编译，如下图：

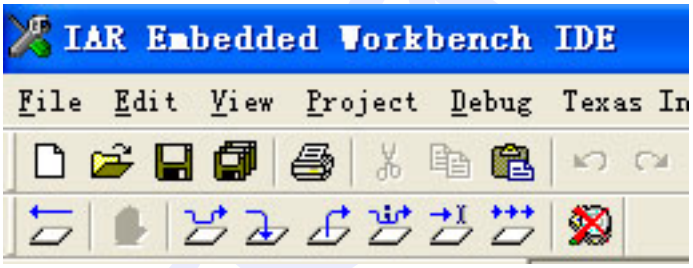


编译完成后，屏幕下方编译信息栏有如下信息提示（Errors 0，Warnning 0）：



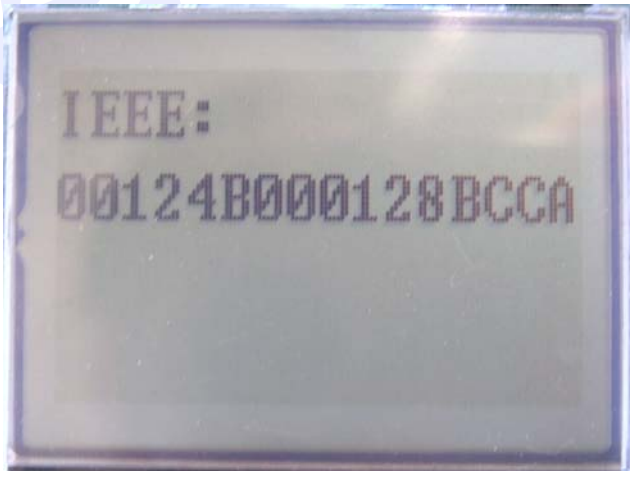
连接好 CC DEBUGGER 和目标板，开启目标板电源，按下 CC DEBUGGER 的 Reset 键，此时 CC DEBUGGER 的指示灯应为绿色。点击  开始向目标板下载程序。

下载完成后，进度条消失，左上角出现调试窗口：

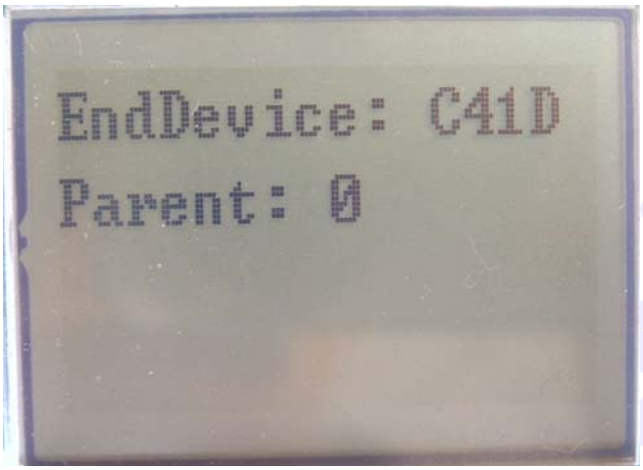


点击  ，退出调试状态，拔除目标板上的 DEBUG 线，重启目标板电源或按下 Reset 键。LCD 屏幕显示如下：

显示本节点 IEEE 地址：



显示本节点功能类型：

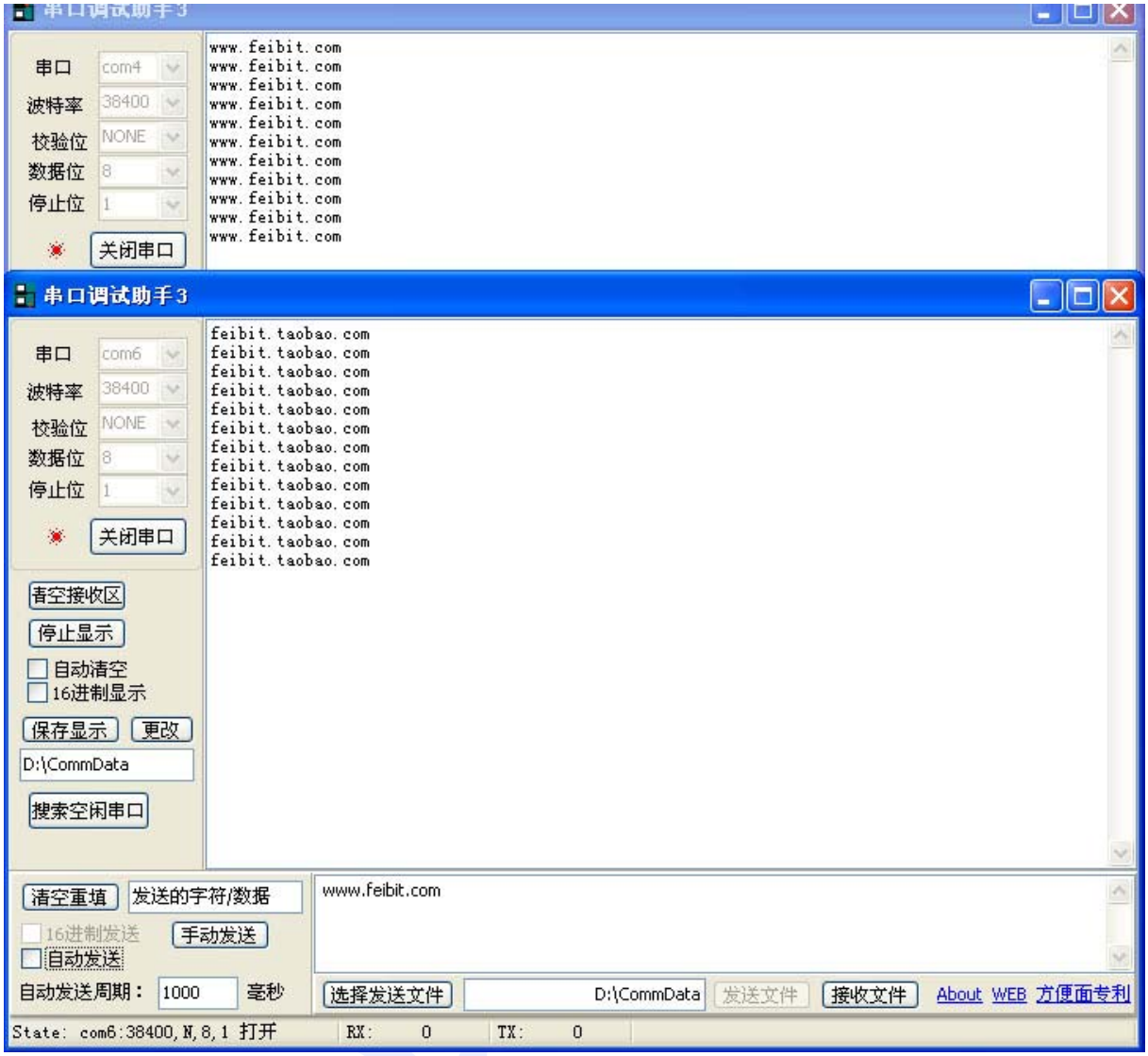


第三节 软件演示

程序下载完成后，按下任意一个节点的 U1(Joystick)右键进行绑定申请，然后立即按下另外一个节点的 U1（Joystick）右键进行绑定确认。此时，两个节点的红色 LED 灯－LED1 同时点亮，表示绑定成功。

用 USB-RS232 串口转接线连接节点和 PC，在 PC 端启动串口调试助手。设置对应的 COM 口和波特率 38400。即可开始通信。

如图：



第四节　　程序解析

4．1　　申请绑定与绑定确认

首先，由某节点触发 Joystick 右键，即 HAL_KEY_SW_2，对如何通过查询电平、确认按键事件，并调用相应的按键处理函数的过程有疑问的读者请参见上述文章。在按键处理函数－－SerialApp_HandleKeys 中，

```
if ( keys & HAL_KEY_SW_2 )
    {
      HalLedSet ( HAL_LED_4, HAL_LED_MODE_OFF );
```

```
      // Initiate an End Device Bind Request for the mandatory endpoint
      txAddr.addrMode = Addr16Bit;
      txAddr.addr.shortAddr = 0x0000; // Coordinator
      ZDP_EndDeviceBindReq( &txAddr, NLME_GetShortAddr(),
                            SerialApp_epDesc.endPoint,
                            SERIALAPP_PROFID,
                        SERIALAPP_MAX_CLUSTERS, (cId_t *)SerialApp_ClusterList,
                        SERIALAPP_MAX_CLUSTERS, (cId_t *)SerialApp_ClusterList,
                            FALSE );
    }
```

此处发起绑定请求，等待其他节点应答，而如果有一个节点也按了 Joystick 右键，同样发出了绑定请求，则本节点收到一个 End_Device_Bind_rsp 的信息，并在 SerialApp_ProcessZDOMsgs 函数中进行了处理，如下代码：

```
/*********************************************************************
* @fn      SerialApp_ProcessZDOMsgs()
*
* @brief   Process response messages
*
* @param   none*
* @return  none
*/
static void SerialApp_ProcessZDOMsgs( zdoIncomingMsg_t *inMsg )
{
  switch ( inMsg->clusterID )
  {
    case End_Device_Bind_rsp:
      if ( ZDO_ParseBindRsp( inMsg ) == ZSuccess )
      {
        // Light LED
        HalLedSet( HAL_LED_4, HAL_LED_MODE_ON );
      }
      break;
...
  }
}
```

4．2　　“串口终端 1”的数据，如何被“节点 1”所接收，并且发送出去的？

串口数据是由哪层来负责的呢？——HAL。。。恩，猜对了。但这个肯定不是靠猜的，其中的过程就不讲了。让我们从主循环(osal_start_system)的 Hal_ProcessPoll 函数找下去(用 source insight

的同学可以用 ctrl +)，Hal_ProcessPoll ==> HalUARTPoll ==> HalUARTPollDMA

这个 HalUARTPollDMA 函数里最后有这样一句话：dmaCfg.uartCB(HAL_UART_DMA-1, evt)； 对 dmaCfg.uartCB 这个函数进行了调用，ctrl / 搜索这个 dmaCfg.uartCB，发现 SerialApp_Init 函数有两句话：

```
uartConfig.callBackFunc          = SerialApp_CallBack;
 HalUARTOpen (SERIAL_APP_PORT, &uartConfig);
```

此处将 dmaCfg.uartCB 这个函数注册成为 SerialApp_CallBack，也就是说 SerialApp_CallBack 函数每次循环中被调用一次，对串口的内容进行查询，如果 DMA 中接收到了数据，则调用 HalUARTRead，将 DMA 数据读至数据 buffer 并通过 AF_DataRequest 函数发送出去，注意：送出去的信息的 CLUSTERID(信息簇 ID)号为 SERIALAPP_CLUSTERID1。

总结一下这个过程：串口数据==>DMA 接收==>主循环中通过 SerialApp_CallBack 查询==>从 DMA 获取并发送到空中。

4．3　节点 2 在收到空中的信号后，如何传递给与其相连的串口终端？

节点 2 从空中捕获到信号后，在应用层上首先收到信息的就是 SerialApp_ProcessEvent 这个函数了，它收到一个 AF_INCOMING_MSG_CMD 的事件，并通知 SerialApp_ProcessMSGCmd，执行以下代码

```
switch ( pkt->clusterId )
  {
  // A message with a serial data block to be transmitted on the serial port.
  case SERIALAPP_CLUSTERID1:
... ...
      // Transmit the data on the serial port.
 if ( HalUARTWrite( SERIAL_APP_PORT, pkt->cmd.Data+1, (pkt->cmd.DataLength-1) ) )
      {
        // Save for next incoming message
        SerialApp_RxSeq = seqnb;
        stat = OTA_SUCCESS;
      }
... ...
```

这样就将从空中获取到的信息，传给了串口终端 2。