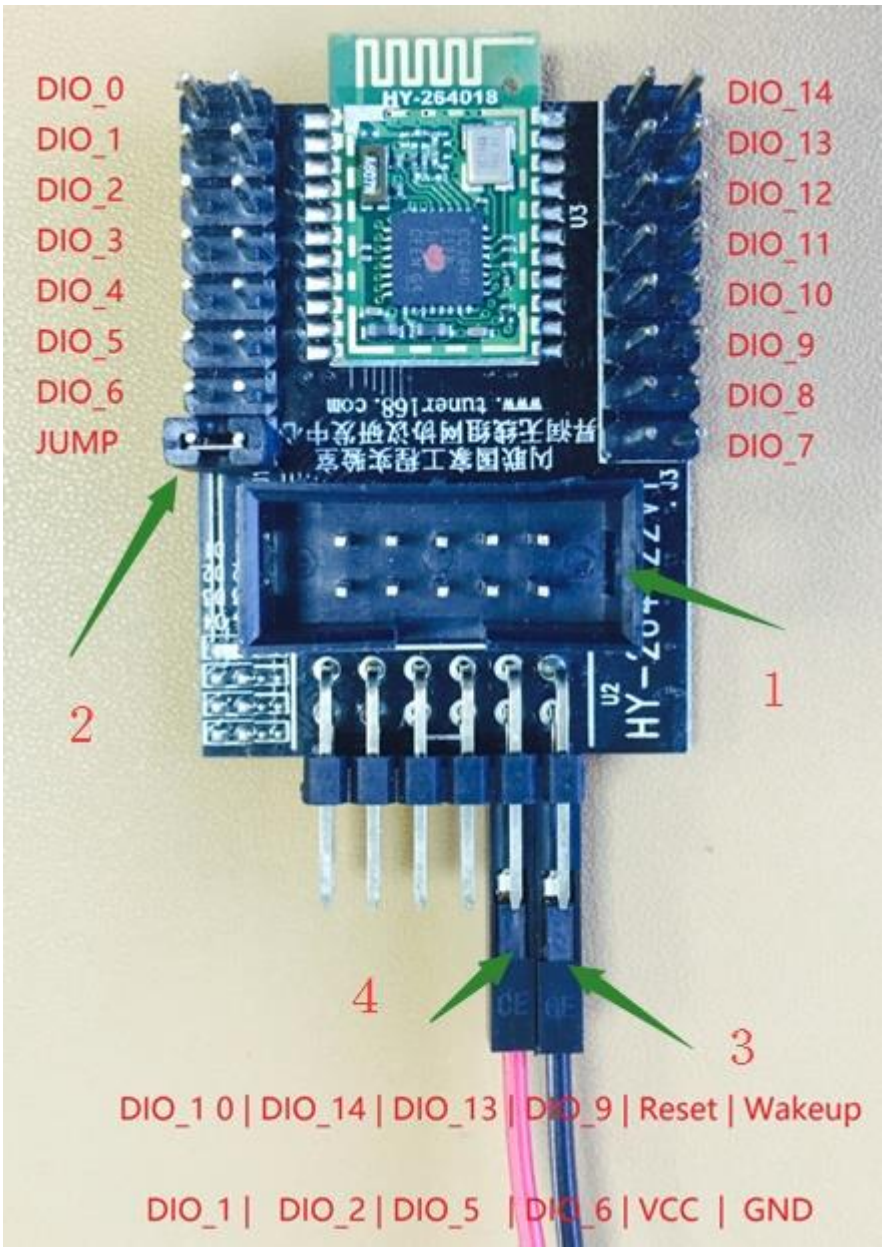
昇润 CC2640 SDK 应用入门教程二

如何使用昇润 SDK 工具实现简单的蓝牙控制

通过上次的演示说明，相信大家对蓝牙的开发环境及 CC2640 SDK 开发套件已基本熟悉。那么，这次我们教大家如何使用昇润 SDK 工具实现简单的蓝牙控制，通过 APP 实现 LED 的开、关和闪烁。

首先上图，目标板：



如上图示，利用开发套件中的 HY-264022V1 开发目标板：

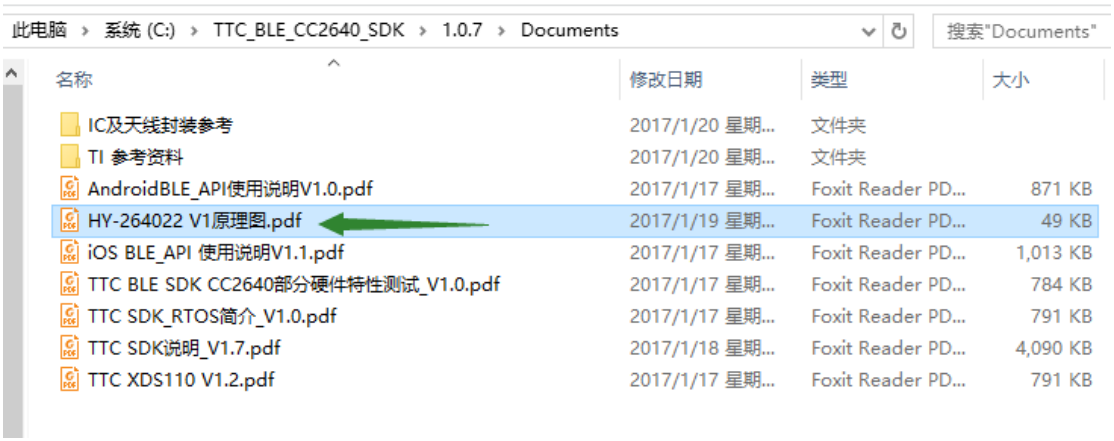
１：接好Ｊ Ｔ Ａ Ｇ调试电缆，请注意方向；

２：把Ｌ Ｅ Ｄ Ｄ１的跳线帽接上，该Ｌ Ｅ Ｄ是通过Ｄ Ｉ Ｏ ０来控制；

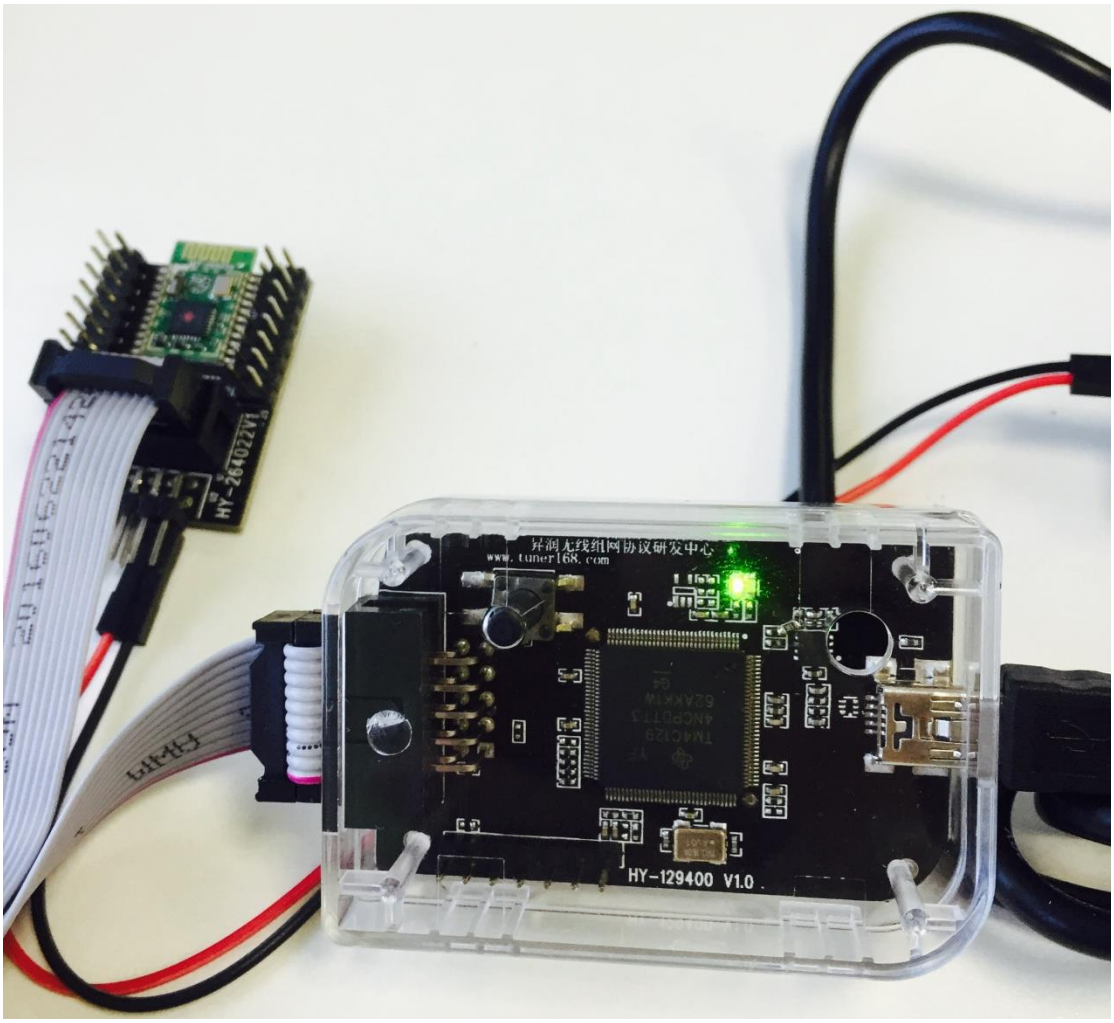
３：外部供电电源的负极接上；

４：外部供电电源的 1.8V-3.6V 电源正极接上；

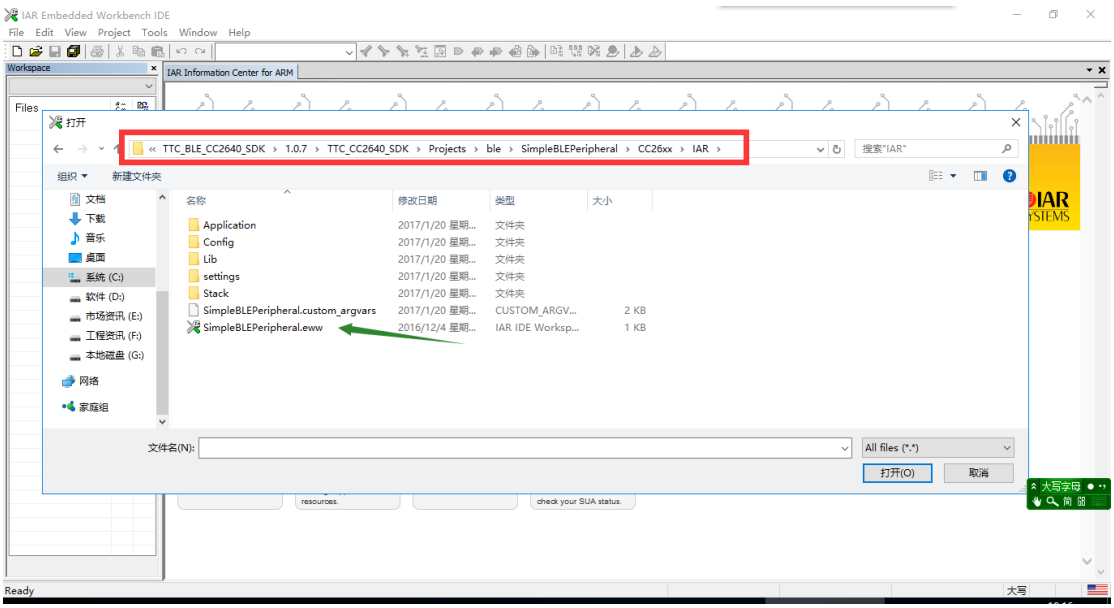
注：ＨＹ－２６４０２２Ｖ１的原理图在安装好的ＳＤＫ的说明文档中，如下图所示：



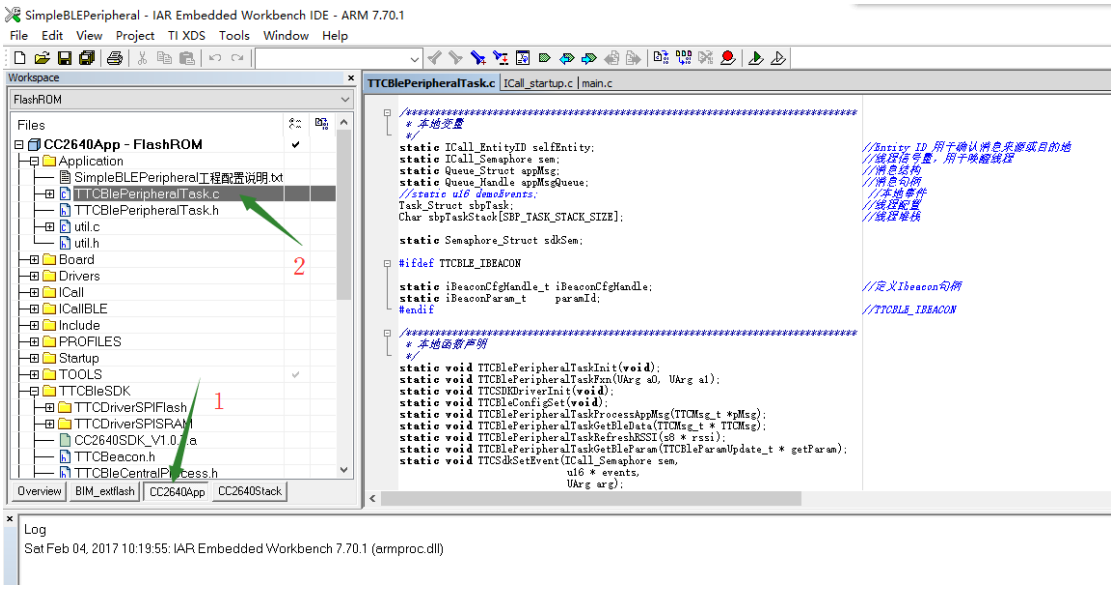
连接上 TTC XDS110 也就是ＨＹ－１２９４００：



打开 IAR ARM 7.7 ，再打开程ＳＤＫ的从机工程:



打开工程后，还是先编译蓝牙协议栈，并下载ＯＫ!再选择应用的工程１，打开我们将要改写的 TTCBlePeripheralTask.c　２



改写之前，我们先验证手机与蓝牙模块之前能进行正常的数据传输：先编译蓝牙应用程序、将程式下载至目标板、全速运行，或是取掉调试器，断开外接电源，重新上电，我们回顾一下上次的蓝牙联线过程，并验证数发送接收，ＳＤＫ刚安装好后，原始默认是接收到任何数据，均返回；

首先打开ＡＰＰ：

扫描，并选择要联线的设备；　　提示联接设备；

联线成功后，我们在ＡＰＰ中选择：数传，发送１２３４５６７８９０，此时可经看到ＲＸ也会显示１２３４５６７８９０，证实发任何数据，模块会将数据回传回来！

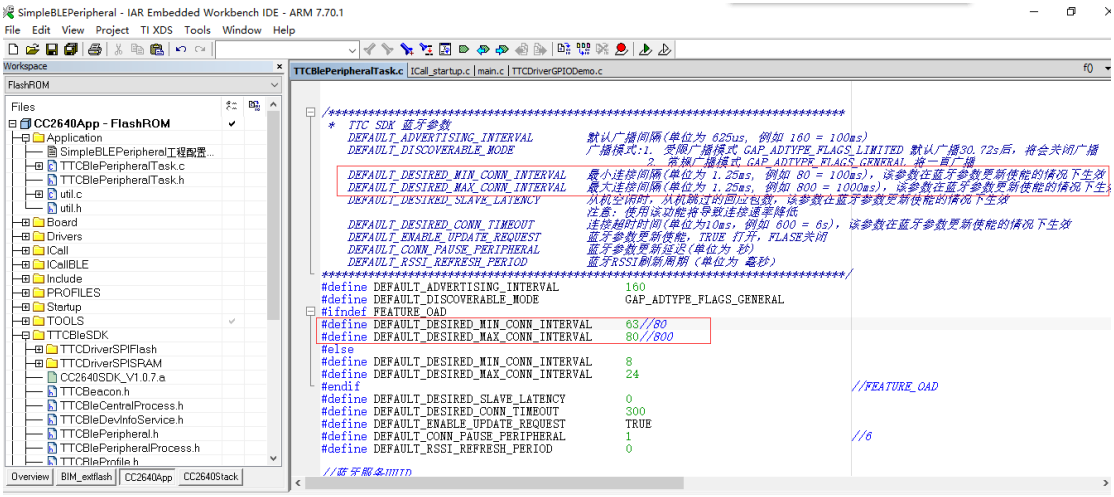
刚才打开的文件中的函数如下：

```
/*********************************************************************************
【函    数】  TTCBlePeripheralTaskGetBleData(TTCMsg_t * TTCMsg)
【概    述】  处理蓝牙消息
【入口参数】  TTCMsg : 接收蓝牙数据消息结构体
【返回参数】  无
【说    明】  无
*********************************************************************************/
static void TTCBlePeripheralTaskGetBleData(TTCMsg_t * TTCMsg){
    TTCData_t * TTCData = TTCMsg->pValue;
    TTCBleProfileSetParameter(TTCBLE_PROFILE_CHAR2,       //收到蓝牙数据后将数据通过UUID 1002特征发送回给主机
                              TTCData->len,
                              TTCData->pValue);
    ICall_free(TTCData->pValue);
    ICall_free(TTCData);
}
```

如果把这条屏蔽掉，就不会回传数据，可以测试看看：

```
/*********************************************************************************
【函    数】  TTCBlePeripheralTaskGetBleData(TTCMsg_t * TTCMsg)
【概    述】  处理蓝牙消息
【入口参数】  TTCMsg : 接收蓝牙数据消息结构体
【返回参数】  无
【说    明】  无
*********************************************************************************/
static void TTCBlePeripheralTaskGetBleData(TTCMsg_t * TTCMsg){
    TTCData_t * TTCData = TTCMsg->pValue;
//    TTCBleProfileSetParameter(TTCBLE_PROFILE_CHAR2,       //收到蓝牙数据后将数据通过UUID 100.
//                              TTCData->len,
//                              TTCData->pValue);
    ICall_free(TTCData->pValue);
    ICall_free(TTCData);
}
```

通过修正以下这二个参数，我们可以来控制蓝牙的联线间隔，参数与计算方式大家可以参考注释：



DEFAULT_DESIRED_MIN_CONN_INTERVAL

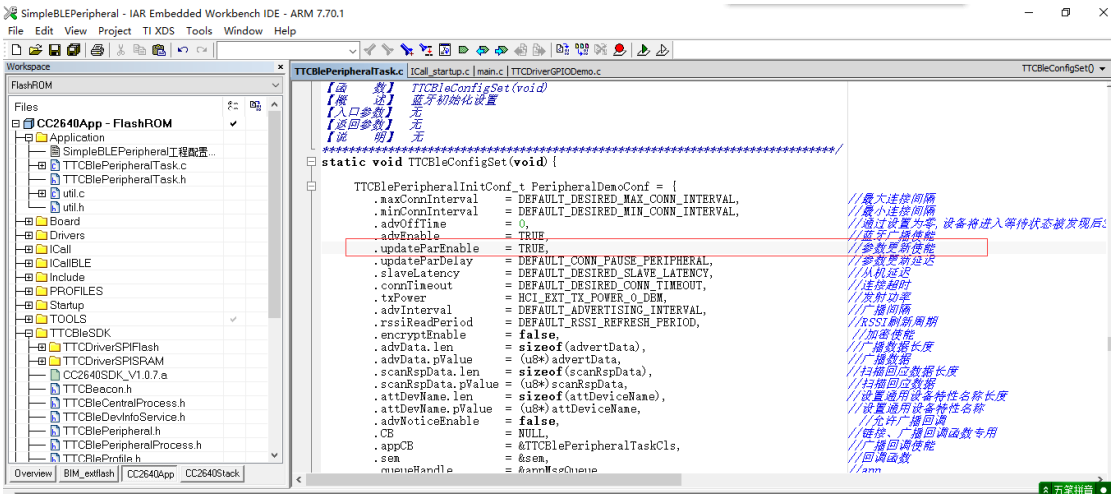
最小连接间隔(单位为 1.25ms，例如 80 = 100ms)，该参数在蓝牙参数更新使能的情况下生效

DEFAULT_DESIRED_MAX_CONN_INTERVAL

最大连接间隔(单位为 1.25ms，例如 800 = 1000ms)，该参数在蓝牙参数更新使能的情况下生效

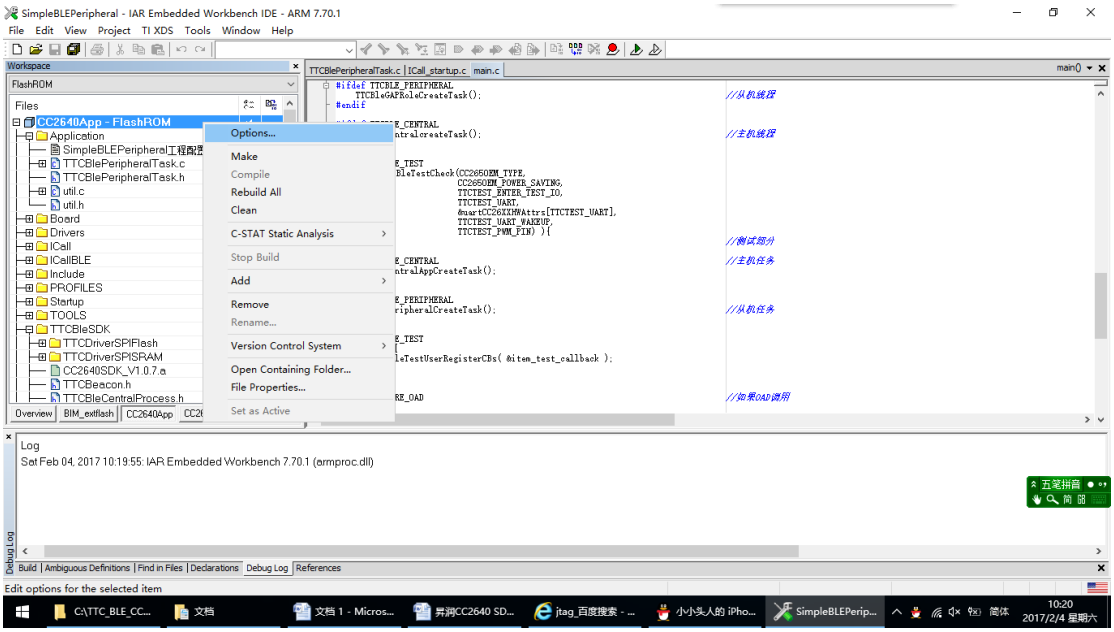
连接间隔最大值与最小值之间的差值，依苹果手机的规定，必须大于２１，苹果一般情况下均是选择最大值;因此现在的联线间隔是：８０ｘ1.25＝100ｍＳ;ＳＤＫ默认关闭软件加密，因此每次最大通讯长度是２０字节;

为什么将这二个参数更新就会有效呢？大家可以看一下这里：

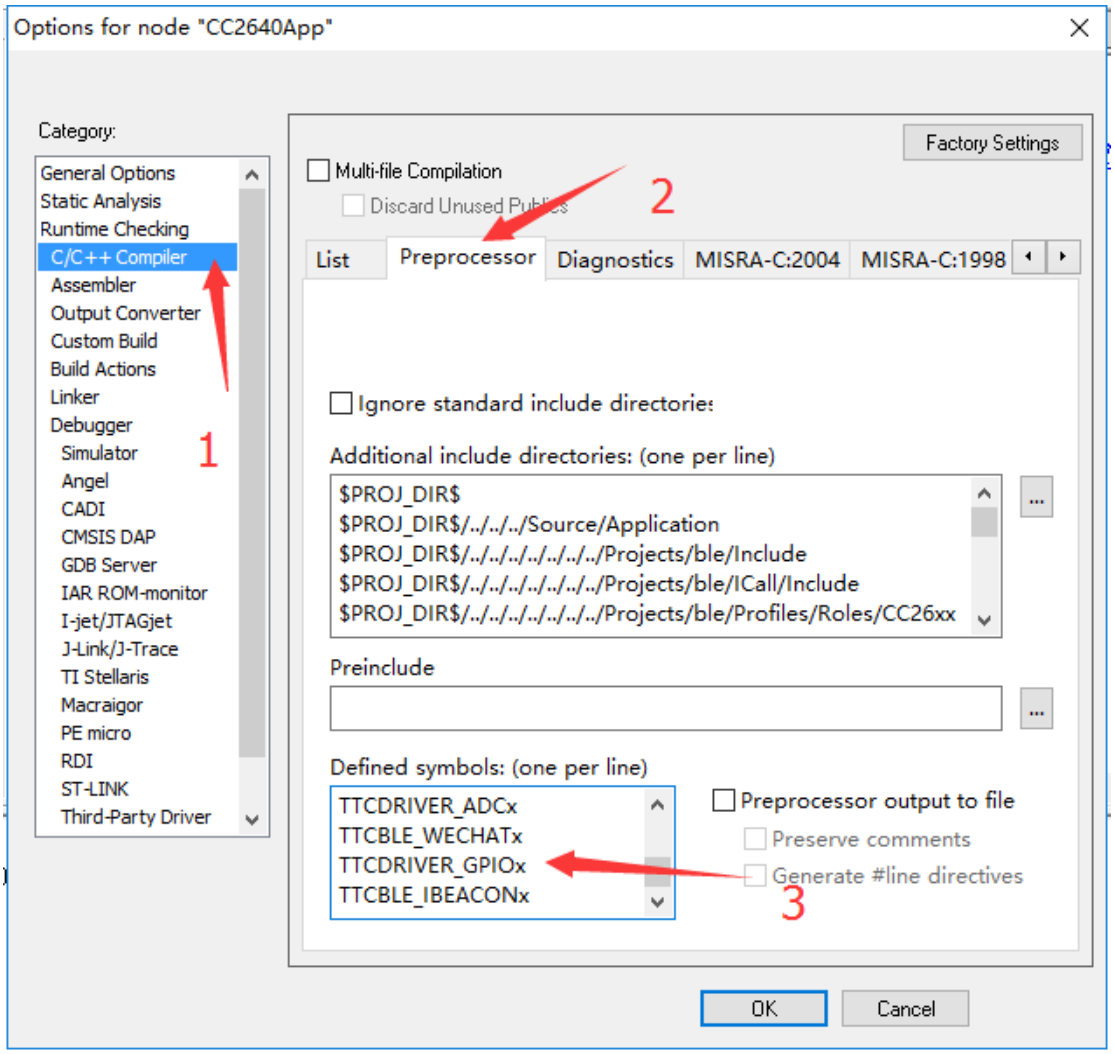


工程在蓝牙初始化时就是.updateParEnable = TRUE, “参数更新使能”

经过上面的验证,我们开始来打开ＧＰＩＯ,通过接收数据来控制ＩＯ的状态,我们先找到项目的可选项:



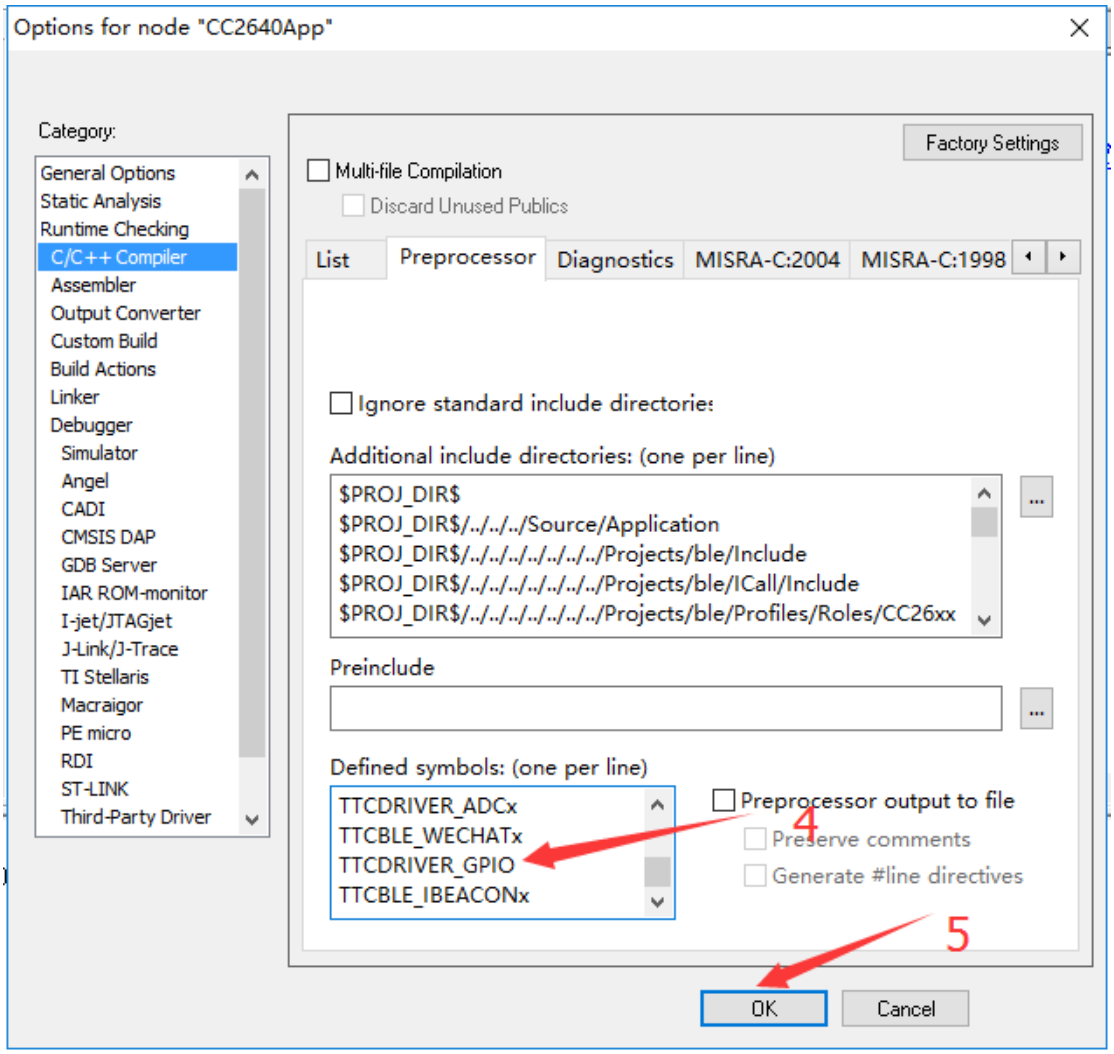
再打开可选项:



1、先选择 c++ compiler 编译选项；

2、再先择 preprocessor 预处理设置；

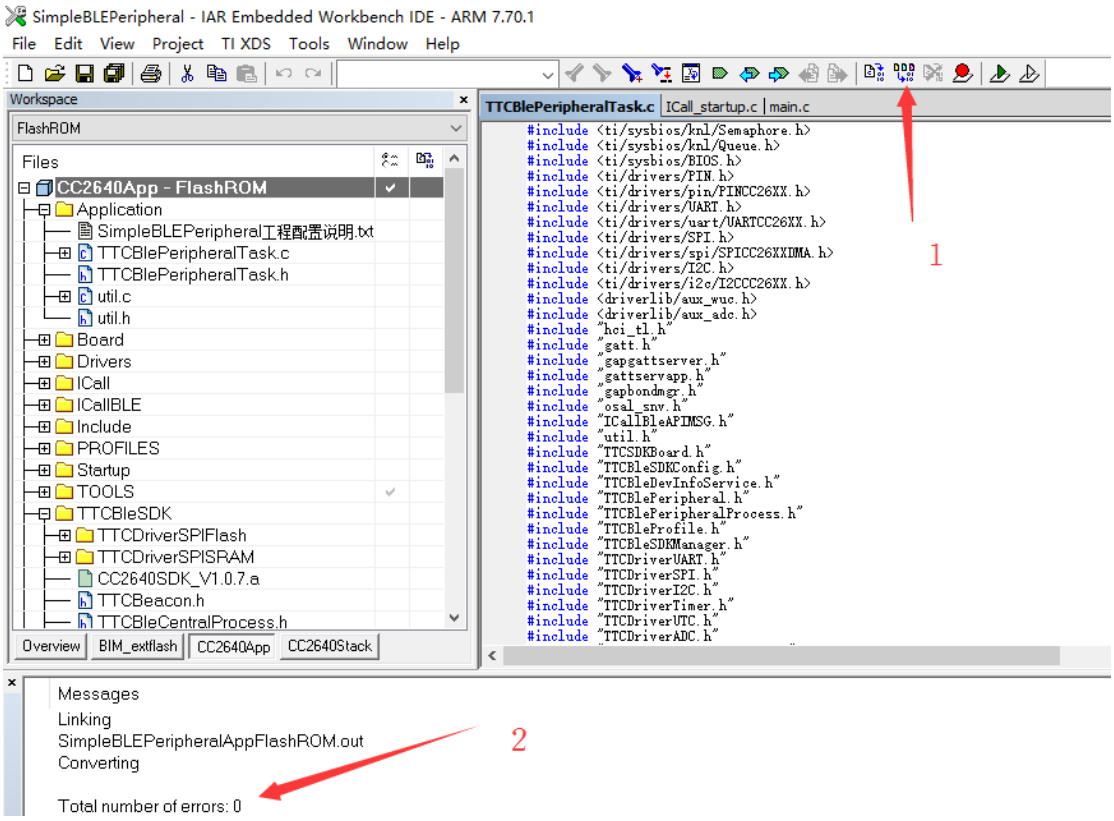
3、找到ＳＤＫ写好的ＧＰＩＯ宏定议开关项



4：我们打开ＧＰＩＯ的宏定议；

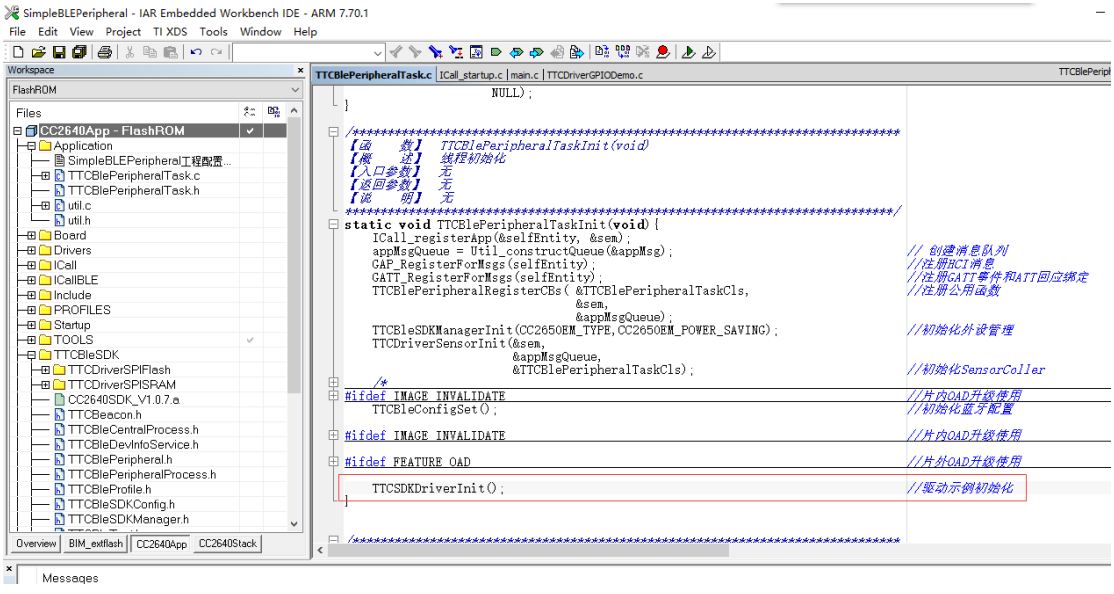
5：确认ＯＫ

我们再编译一下：

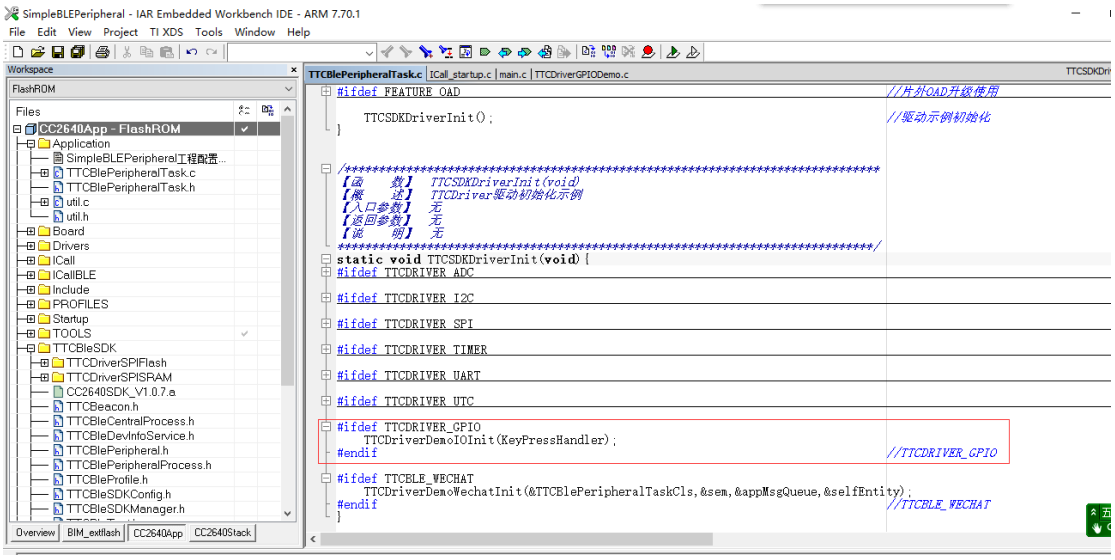


确定ＳＤＫ打开ＧＰＩＯ后，编译正确。

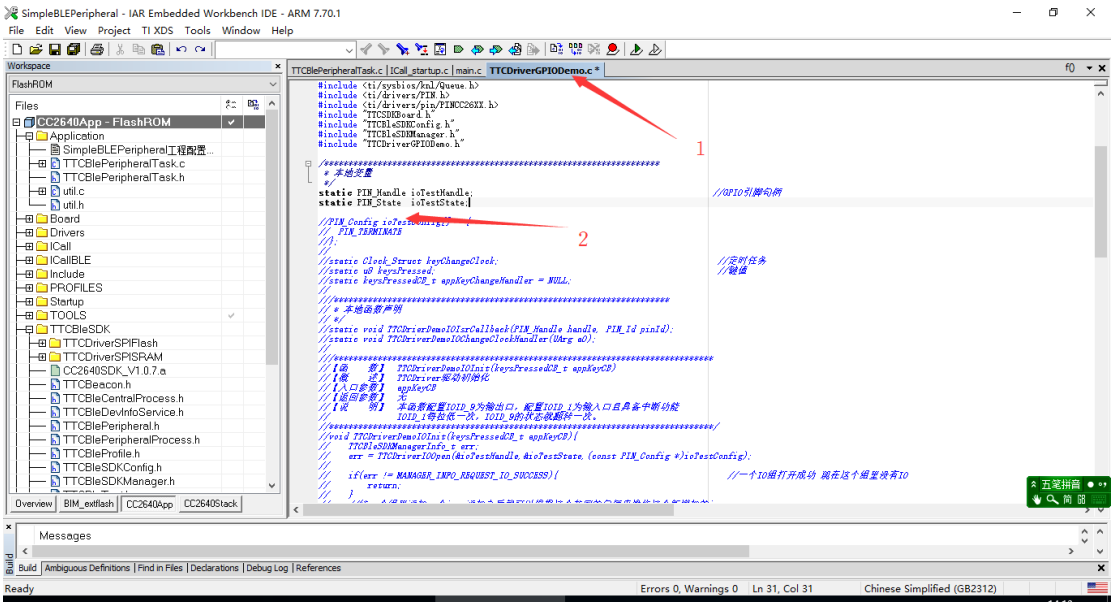
找到设备的初始设定点：



进去找到ＧＰＩＯ的初始化部分：

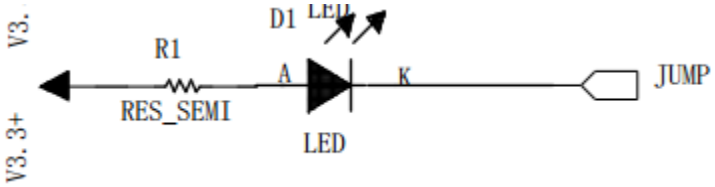


进入这个函数所在的文档 TTCDriverGPIODemo.c　1 步，将下面的全部屏蔽掉 2 步：

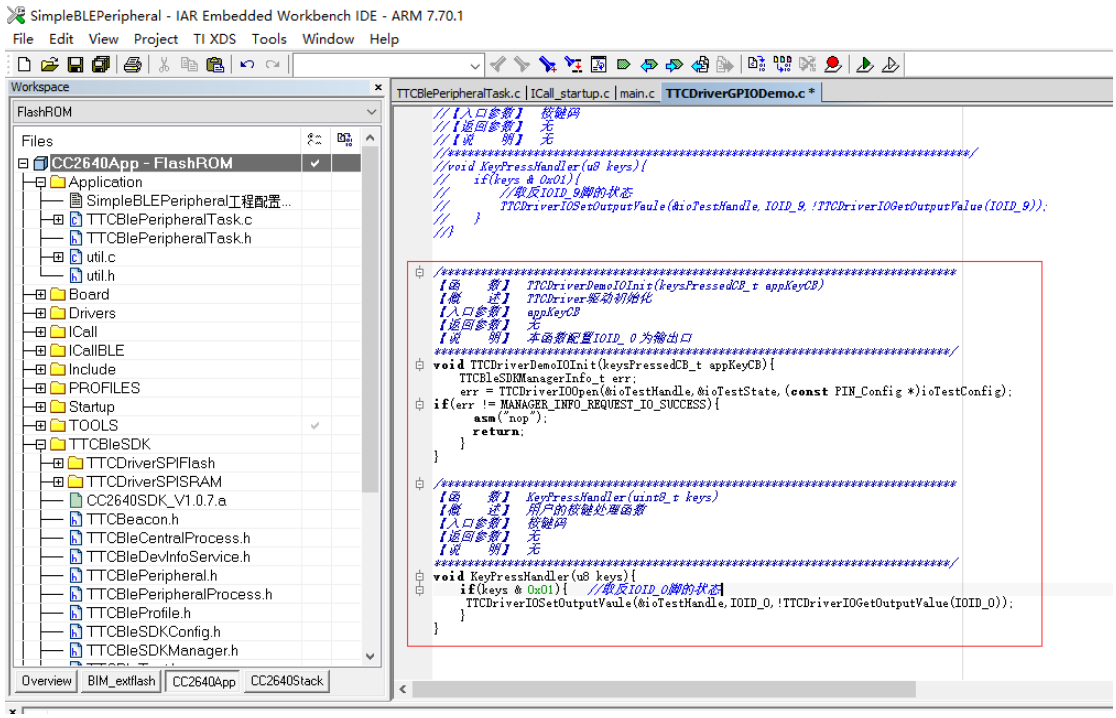



```
PIN_Config ioTestConfig[] = {
  IOID_0 | PIN_GPIO_OUTPUT_EN | PIN_INPUT_DIS | PIN_GPIO_HIGH,
  PIN_TERMINATE
};
```

在ＩＯ配置中将，IOID_0 配置为输出态，并输出为高电平，即关闭ＬＥＤ;原理如下：



利用ＳＤＫ　ＧＰＩＯ演示代码，我们修改一下：




```
/*****************************************************************************
【函    数】  TTCDriverDemoIOInit(keysPressedCB_t appKeyCB)
【概    述】  TTCDriver驱动初始化
【入口参数】  appKeyCB
【返回参数】  无
【说    明】  本函数配置IOID_0为输出口
*****************************************************************************/
void TTCDriverDemoIOInit(keysPressedCB_t appKeyCB){
    TTCBleSDKManagerInfo_t err;
    err = TTCDriverIOOpen(&ioTestHandle,&ioTestState,(const PIN_Config *)ioTestConfig);
if(err != MANAGER_INFO_REQUEST_IO_SUCCESS){
      asm("nop");
      return;
    }
}

/*****************************************************************************
【函    数】  KeyPressHandler(uint8_t keys)
【概    述】  用户的按键处理函数
【入口参数】  按键码
【返回参数】  无
【说    明】  无
*****************************************************************************/
void KeyPressHandler(u8 keys){
    if(keys & 0x01){  //取反IOID_0脚的状态
    TTCDriverIOSetOutputVaule(&ioTestHandle,IOID_0,!TTCDriverIOGetOutputValue(IOID_0));
    }
}
```

/***

【函　　数】　TTCDriverDemoIOInit(keysPressedCB_t appKeyCB)

【概　　述】　TTCDriver 驱动初始化

【入口参数】　appKeyCB

【返回参数】　无

【说　　明】　本函数配置 IOID_0 为输出口

***/

```
void TTCDriverDemoIOInit(keysPressedCB_t appKeyCB){
    TTCBleSDKManagerInfo_t err;
    err     =     TTCDriverIOOpen(&ioTestHandle,&ioTestState,(const     PIN_Config
*)ioTestConfig);
    if(err != MANAGER_INFO_REQUEST_IO_SUCCESS){
          asm("nop");
      return;
    }
}
```

/***

【函　　数】　KeyPressHandler(uint8_t keys)

【概　　述】　用户的按键处理函数

【入口参数】　按键码

【返回参数】　无

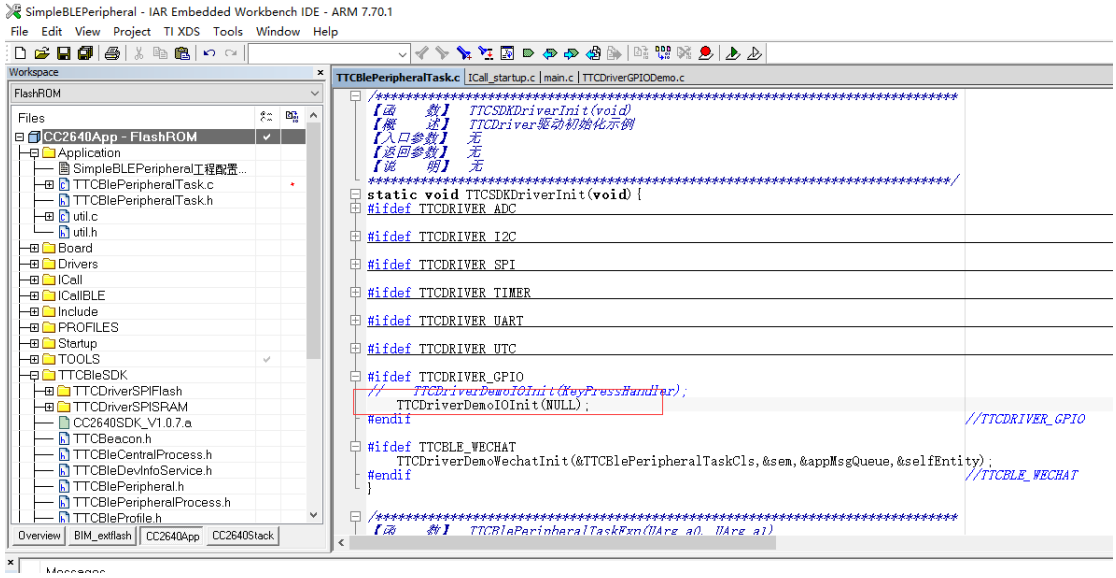
【说　　明】　无

**********/

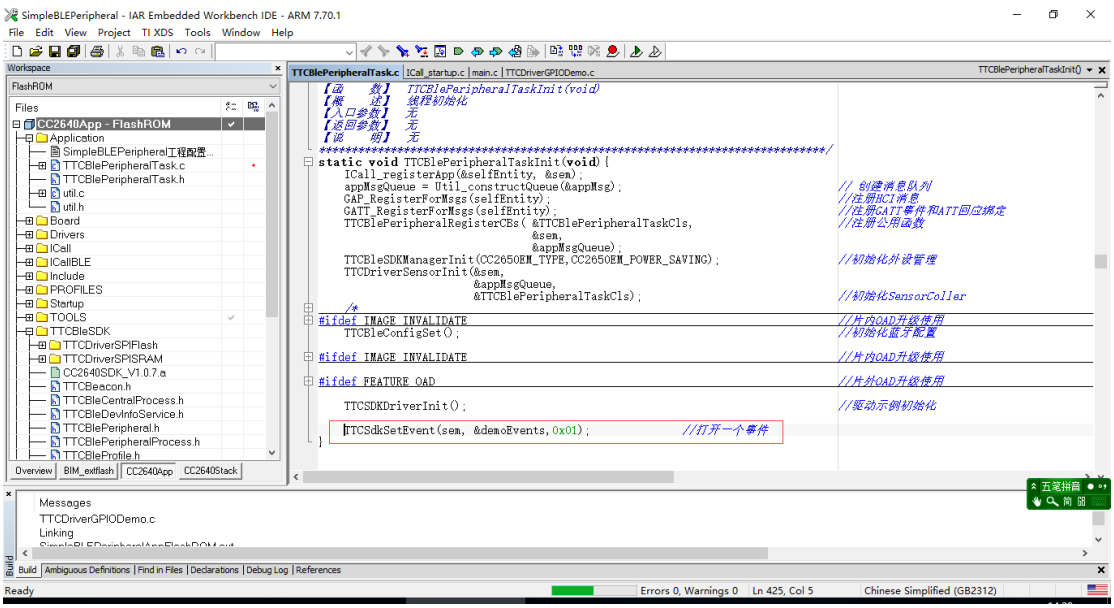
```
void KeyPressHandler(u8 keys){
    if(keys & 0x01){   //取反 IOID_0 脚的状态

TTCDriverIOSetOutputVaule(&ioTestHandle,IOID_0,!TTCDriverIOGetOutputValue(IOID_0));
    }
}
```

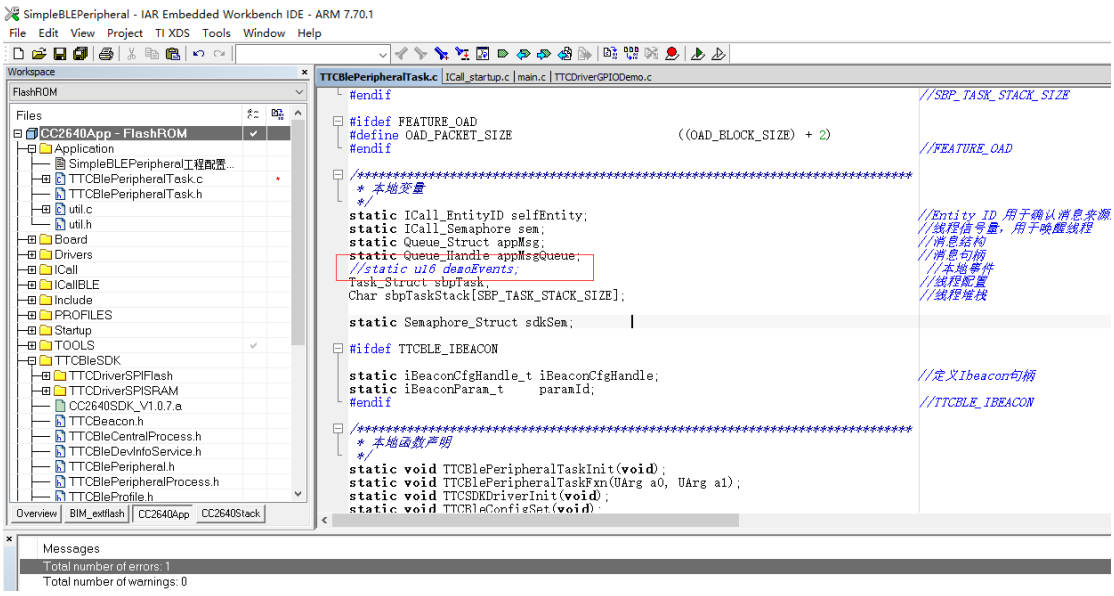
在从机线程文件中，我们把ＧＰＩＯ的初设定改一下：



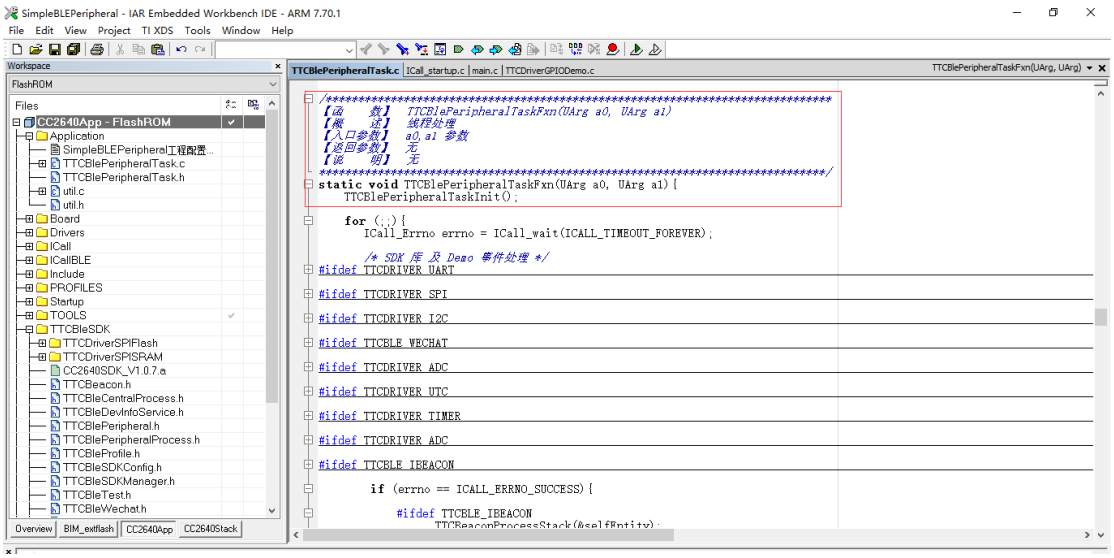
再打开一个事件：

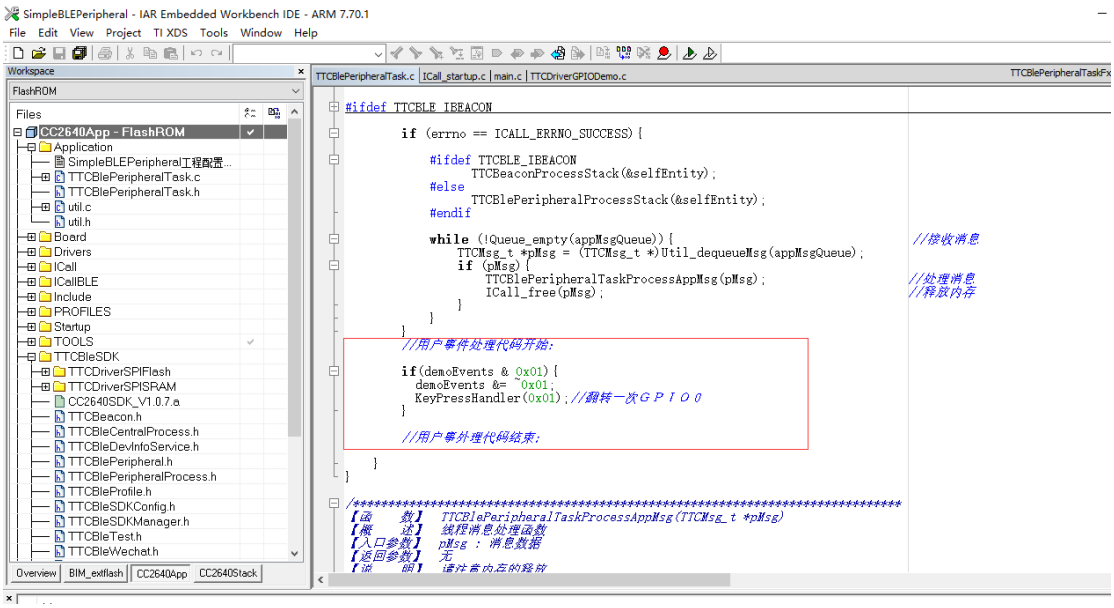


要把这个变量放出来，测试一下：

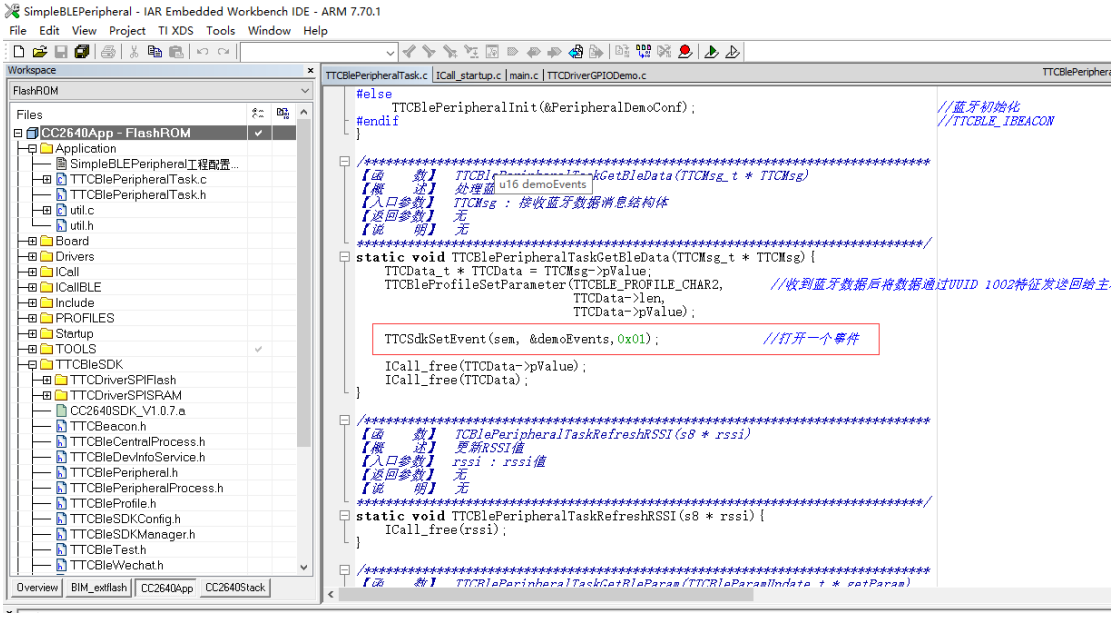


在线程处理函数中，增加用户事件：





然后在数据接收外理函数中，增加使能用户事件：



这样我们每发送一次数据，就可以将ＧＩＰＯ０取反一次；

在ＡＰＰ中将，设定好数据发送间隔，并使能定时发送，这时ＬＥＤ就按可以闪烁啦！

如果在接收数据那里将接收到的数据进解析，那么就可以防止别人操作啦！