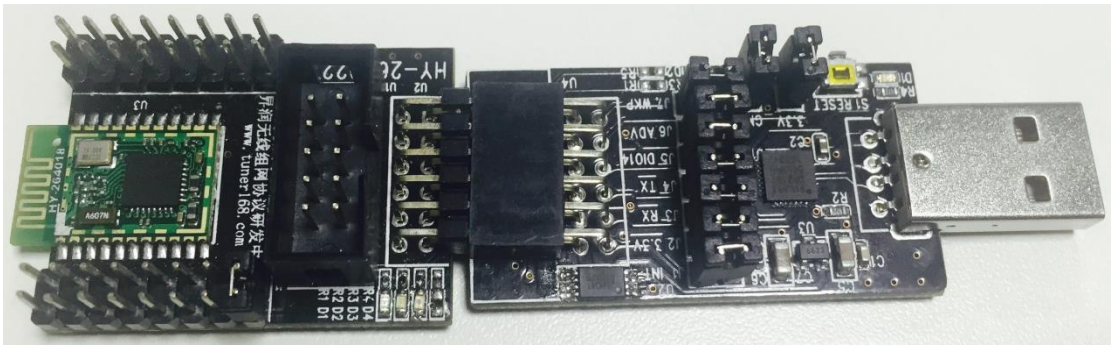
昇润 CC2640 SDK 应用入门教程四

BLE 与串口数据传输

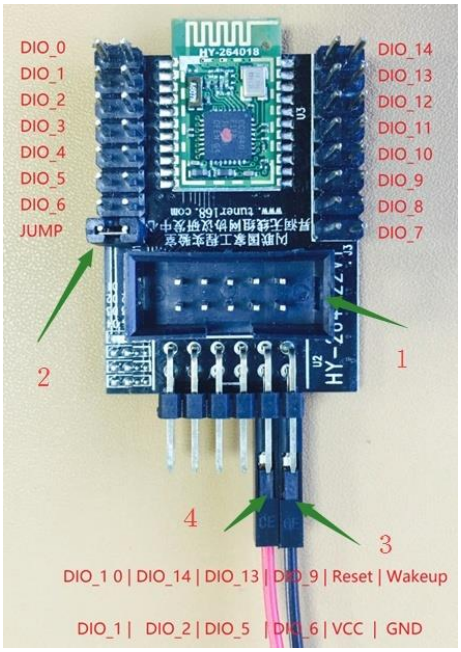
之前的实验，通过手机ＡＰＰ可以与模块进行双向收发数据，并利用数据的发送来控制ＬＥＤ的开、关、闪烁。很多老司机在问，可不可以将数据通过串口打印出来呀？串口是否可以直接发送数据给手机？答案当然是没问题，看我给大家细细道来：

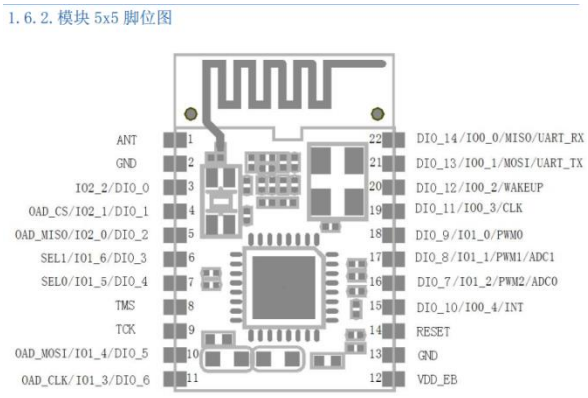
既然想串口打印，当然要有一个 USB 转 UART 的工具，老套路，先上个图：

把昇润 SDK 开发套件中的串口工具，同我们的开发目标板联上，如下图所示：



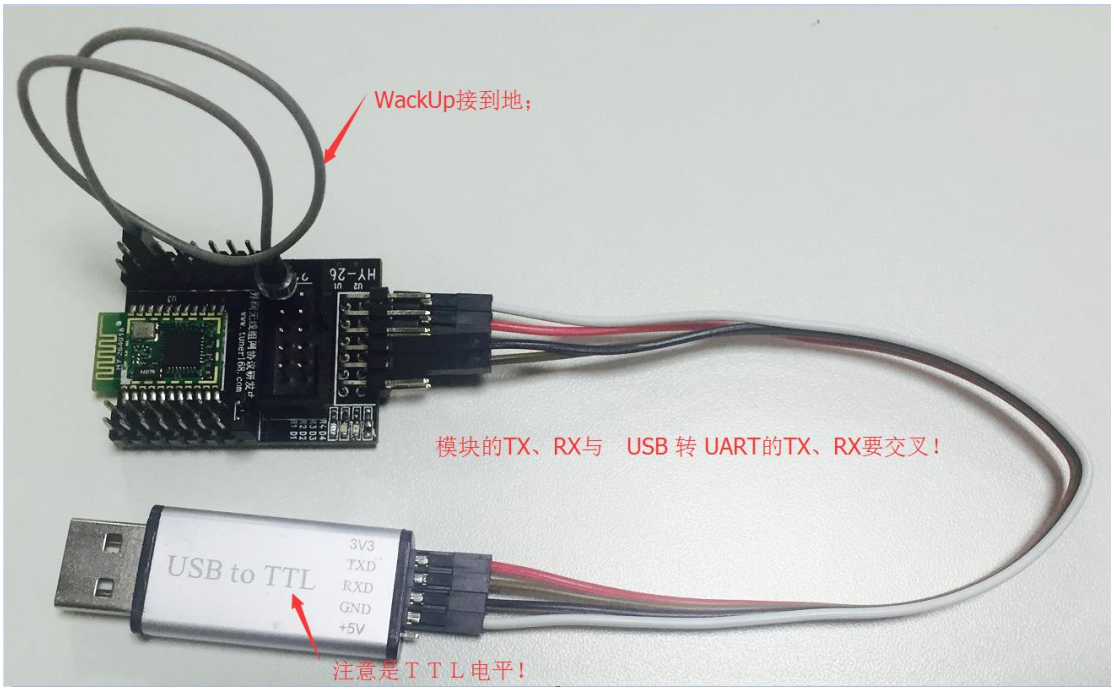
跳线什么的，大家都照着先接上，没有这个转接套件的，依下图的标示把串口要用的线接上：





怎么接呢？

依上图标示，我们将 Wakeup 接到地上；DIO13,即 UART_TX 接到 USB 转 UART TTL 电平工具的 RX;同样我们 DIO14,即 UART_RX 接到 USB 转 UART TTL 电平工具的 TX;再接上电源、地；我接啦一个给大家参考一下：

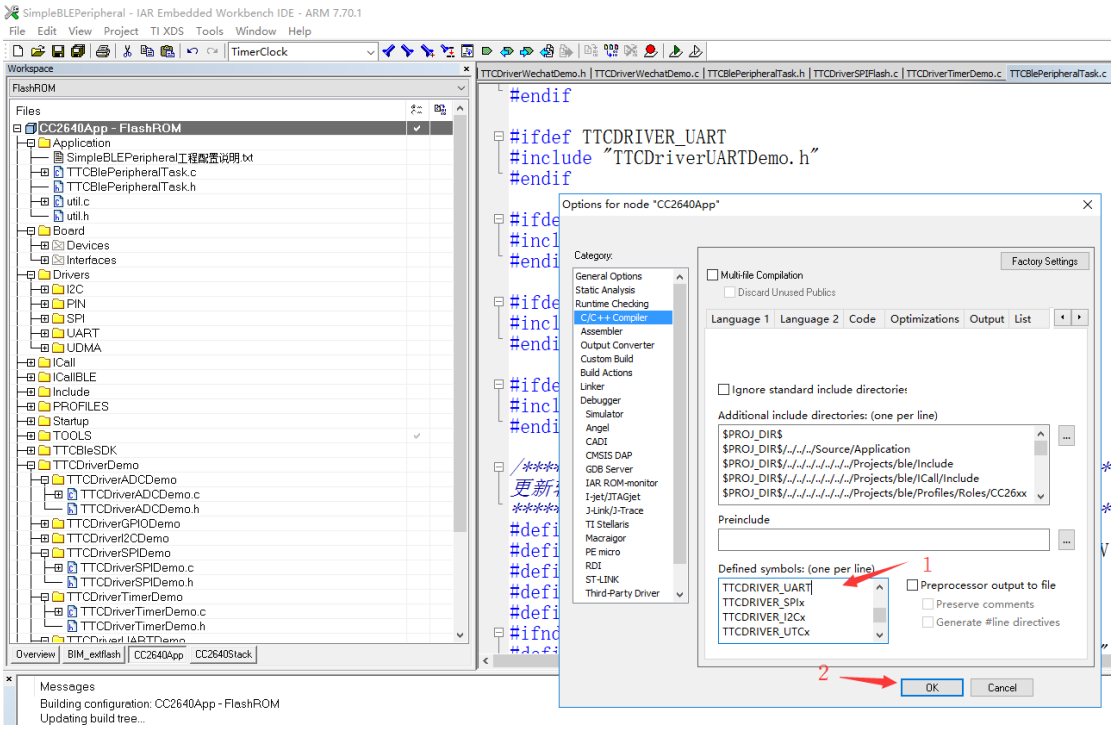


为什么要这么接呢？

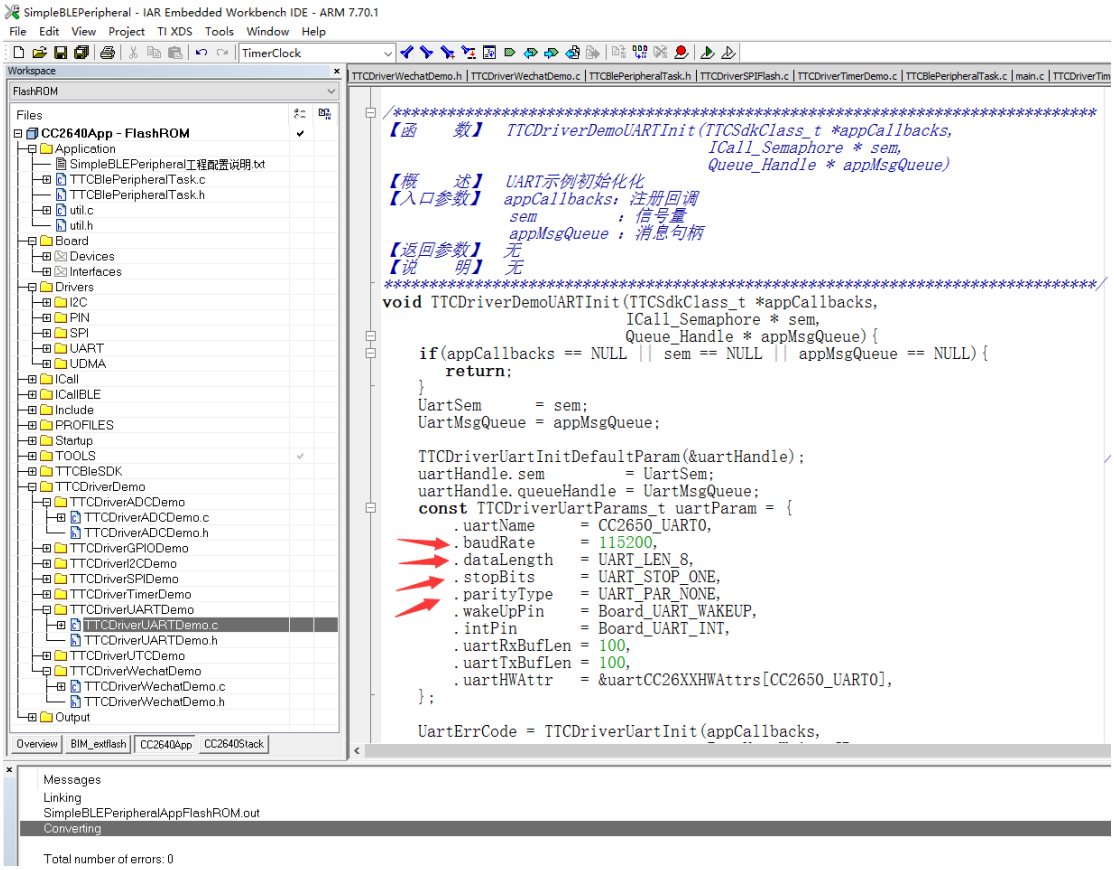
在昇润的 SDK 下的这个文件：TTCSDKBoard.h 里有定义好 IO 脚位，大家可以对照一下：

```
/* Mapping of pins to board signals using general board aliases
 *      <board signal alias>                  <pin mapping>
 */
/* UART Board */
#define Board_UART_RX                         IOID_14
#define Board_UART_TX                         IOID_13
#define Board_UART_CTS                        PIN_UNASSIGNED
#define Board_UART_RTS                        PIN_UNASSIGNED
#define Board_UART_INT                        IOID_10
#define Board_UART_WAKEUP                     IOID_12
```

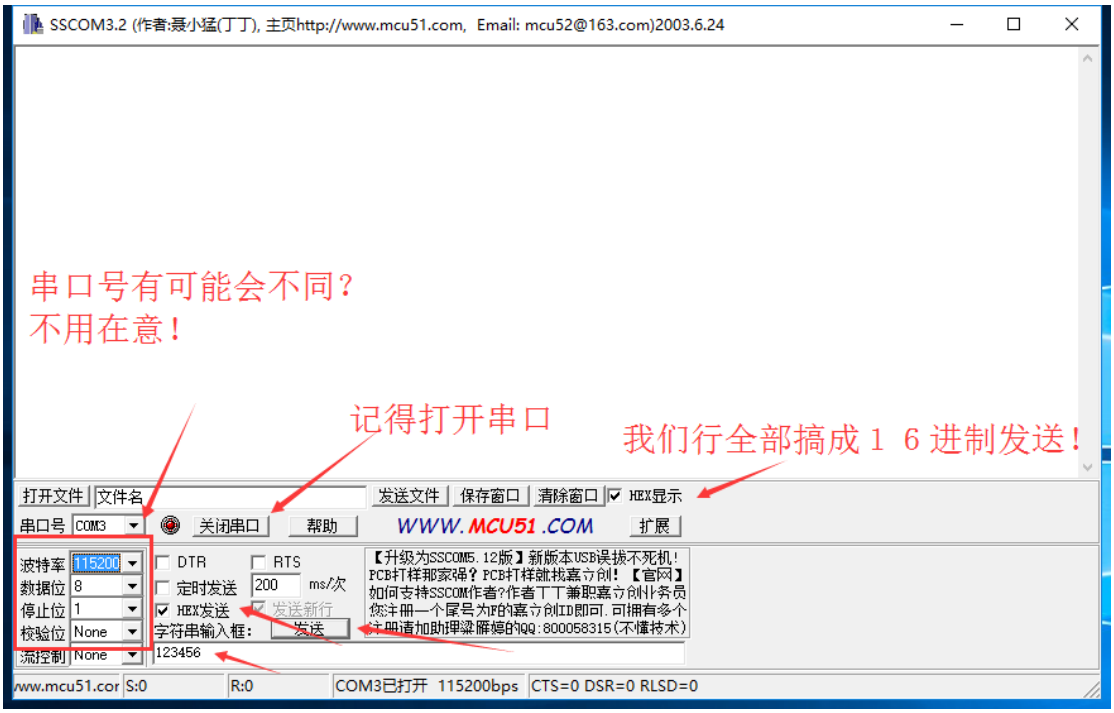
其中 WakeUp 接到地上，芯片的串口模块才会工作；硬件准备好啦，我们把开发调试器联接上，打开上个试验的工程，开始将串口的软件功能宏定义打开，如下所示：



打开后编译通过，我们再找到串口例子程式的串口初始化，确认一下串口的参数：



将串口调试器的参数设定成一样：

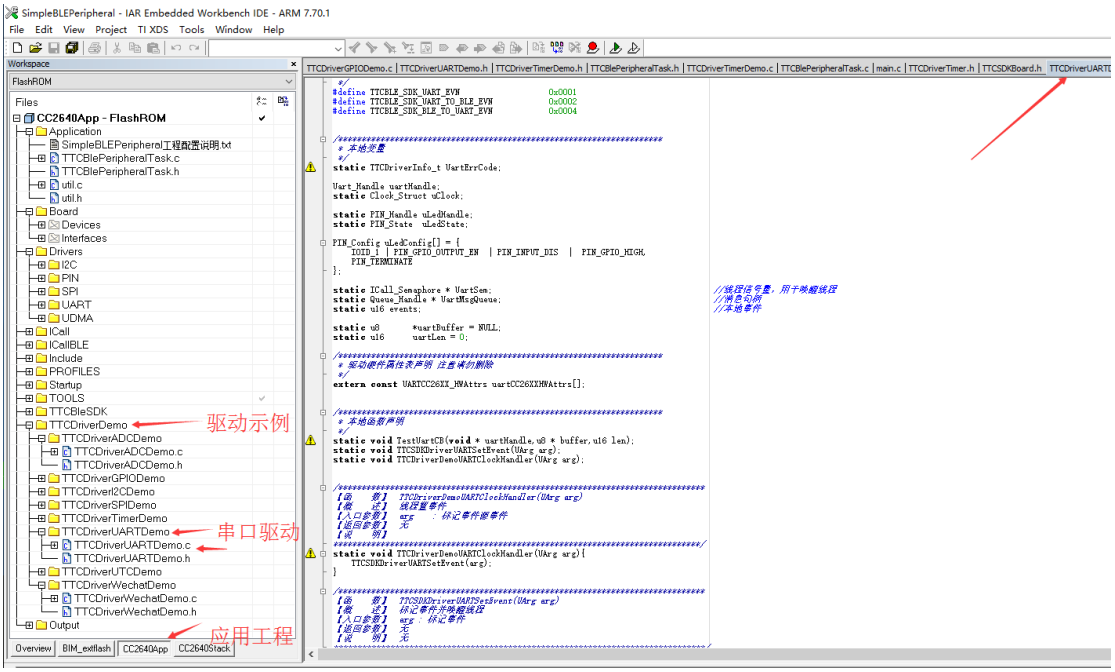


重复一下，参数一定要设成一样哦!!!

准备工作做好后，我们先来测试一下他原有的测试程试，对要把有冲突的ＩＯ改一下，串口示例代码的作用是将串口接收模块接收到的数据发回给发送模块，每发送一次数据就将

DIO0 翻转一次，这里会与上次的 GPIO 测试会有点冲突，我们改下，改到 DIO1；

首先找到 TTCDriverRARTDemo.c 这个文件：



还是 TTCDriverRARTDemo.c 这个文件，先把 UART 接收回调函数里的 IO 改过来：

```
/*****************************************************************************
【函    数】  TestUartCB(TTCDriverUartState_t uartState,u8 * buffer,u16 len)
【概    述】  UART接收回调函数
【入口参数】
【返回参数】  无
【说    明】  无
*****************************************************************************/
static void TestUartReadCB(void * uartHandle,u8 * buffer,u16 len){
    uartBuffer = buffer;
    uartLen    = len;
//    static u8 ubf[50] = {0};
//
//    memcpy(ubf,buffer,len);

    if(uLedHandle != NULL){
        TTCDriverIOSetOutputVaule(&uLedHandle,IOID_1,~TTCDriverIOGetOutputValue(IOID_1));
    }
```

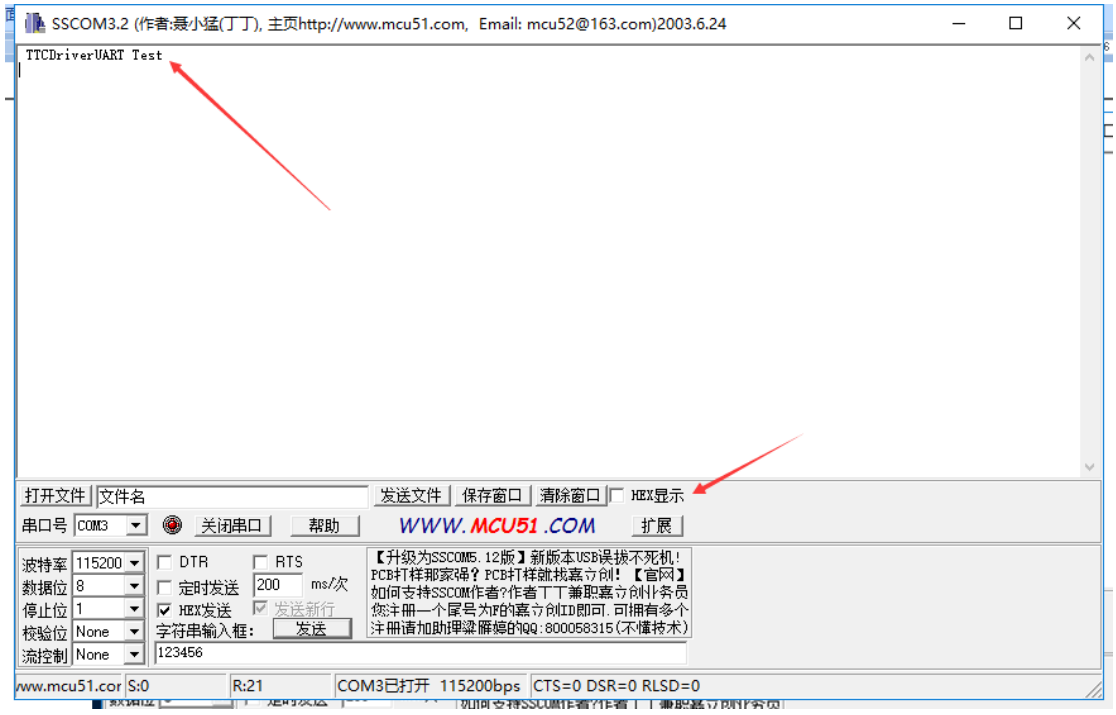
再把 PIN_Config 改一下：

```
/*****************************************************************************
 * 本地变量
 */
static TTCDriverInfo_t UartErrCode;

Uart_Handle uartHandle;
static Clock_Struct uClock;

static PIN_Handle uLedHandle;
static PIN_State  uLedState;

PIN_Config uLedConfig[] = {
    IOID_1 | PIN_GPIO_OUTPUT_EN | PIN_INPUT_DIS | PIN_GPIO_HIGH,
    PIN_TERMINATE
};
```

将这个#if 0 改为 1 打开这个 I O 的配置功能：

```
*****************************************************************************/
void TTCDriverDemoUARTInit(TTCSdkClass_t *appCallbacks,
                           ICall_Semaphore * sem,
                           Queue_Handle * appMsgQueue){
    if(appCallbacks == NULL || sem == NULL || appMsgQueue == NULL){
    UartSem      = sem;
    UartMsgQueue = appMsgQueue;

    TTCDriverUartInitDefaultParam(&uartHandle);                          //不是必
    uartHandle.sem         = UartSem;
    uartHandle.queueHandle = UartMsgQueue;
    const TTCDriverUartParams_t uartParam = {

    UartErrCode = TTCDriverUartInit(appCallbacks,
                                    TestUartWritetCB,                    //write
                                    TestUartReadCB,                      //readC
                                    &uartHandle,
                                     uartParam);

     UartErrCode = TTCDriverUartWrite(&uartHandle,"TTCDriverUART Test\r\n",strlen("TTCDr

     #if 1                                                               //当测试
       TTCBleSDKManagerInfo_t Err;
       Err = TTCDriverIOOpen(&uLedHandle,&uLedState,(const PIN_Config *)uLedConfig);

       if(MANAGER_INFO_REQUEST_IO_SUCCESS == Err){

         asm("nop");
       }
```

这样就可以实现串口自发自收，大家测试一下，首先一运行：

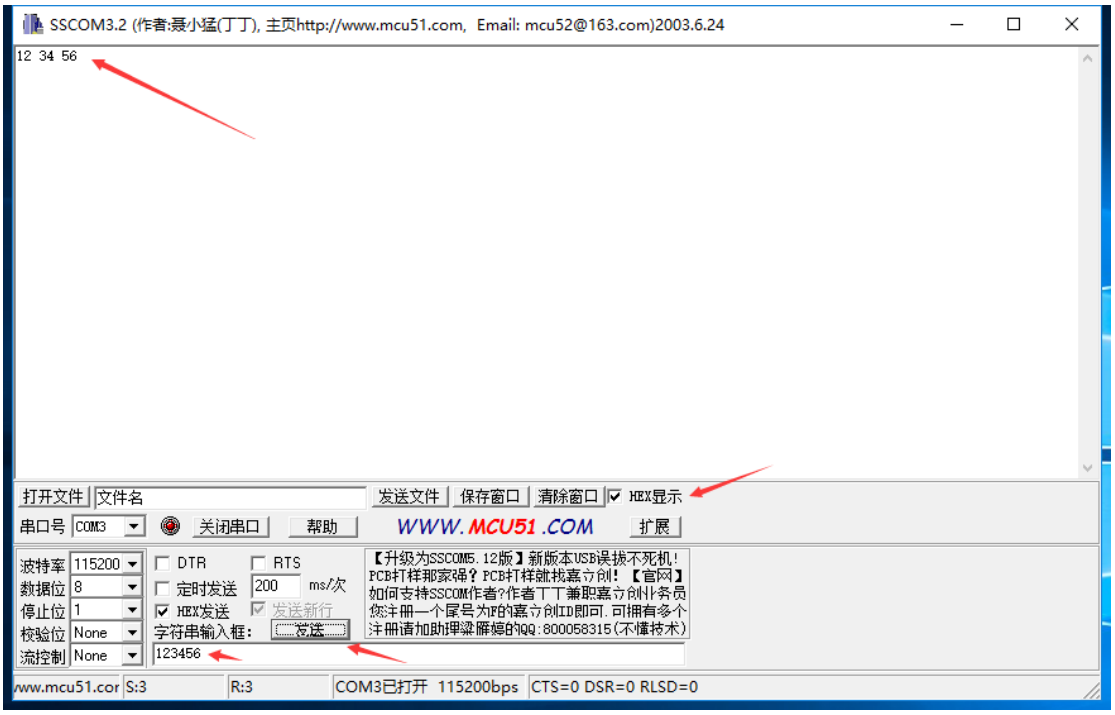


串口测试程式会输出：TTCDriverUART Test 并换行；

因为测试程式在初始化串口时就输出啦这行字符，如下所示；

```
/*********************************************************************/
void TTCDriverDemoUARTInit(TTCSdkClass_t *appCallbacks,
                           ICall_Semaphore * sem,
                           Queue_Handle * appMsgQueue){
    if(appCallbacks == NULL || sem == NULL || appMsgQueue == NULL){
       return;
    }
    UartSem      = sem;
    UartMsgQueue = appMsgQueue;

    TTCDriverUartInitDefaultParam(&uartHandle);                    //不是必须的
    uartHandle.sem          = UartSem;
    uartHandle.queueHandle = UartMsgQueue;
    const TTCDriverUartParams_t uartParam = {
        .uartName     = CC2650_UART0,
        .baudRate     = 115200,
        .dataLength   = UART_LEN_8,
        .stopBits     = UART_STOP_ONE,
        .parityType   = UART_PAR_NONE,
        .wakeUpPin    = Board_UART_WAKEUP,
        .intPin       = Board_UART_INT,
        .uartRxBufLen = 100,
        .uartTxBufLen = 100,
        .uartHWAttr   = &uartCC26XXHWAttrs[CC2650_UART0],
    };

    UartErrCode = TTCDriverUartInit(appCallbacks,
                                    TestUartWritetCB,              //writeCB
                                    TestUartReadCB,                //readCB
                                    &uartHandle,
                                     uartParam);

     UartErrCode = TTCDriverUartWrite(&uartHandle,"TTCDriverUART Test\r\n",strlen("TTCDriverUART Test\r\n"));
```

我们再来测试一个是不是，真的发什么，就回传什么呀！清屏，并改为 1 6 进制先试一下：

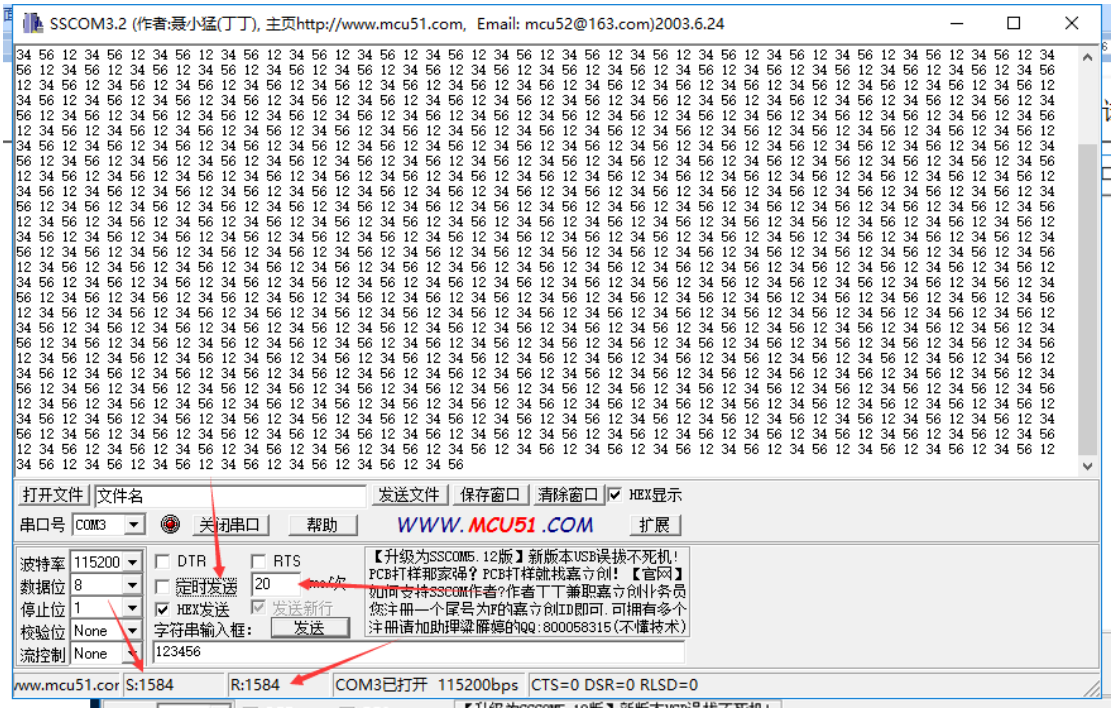


真的可以哦！这是为什么呢？

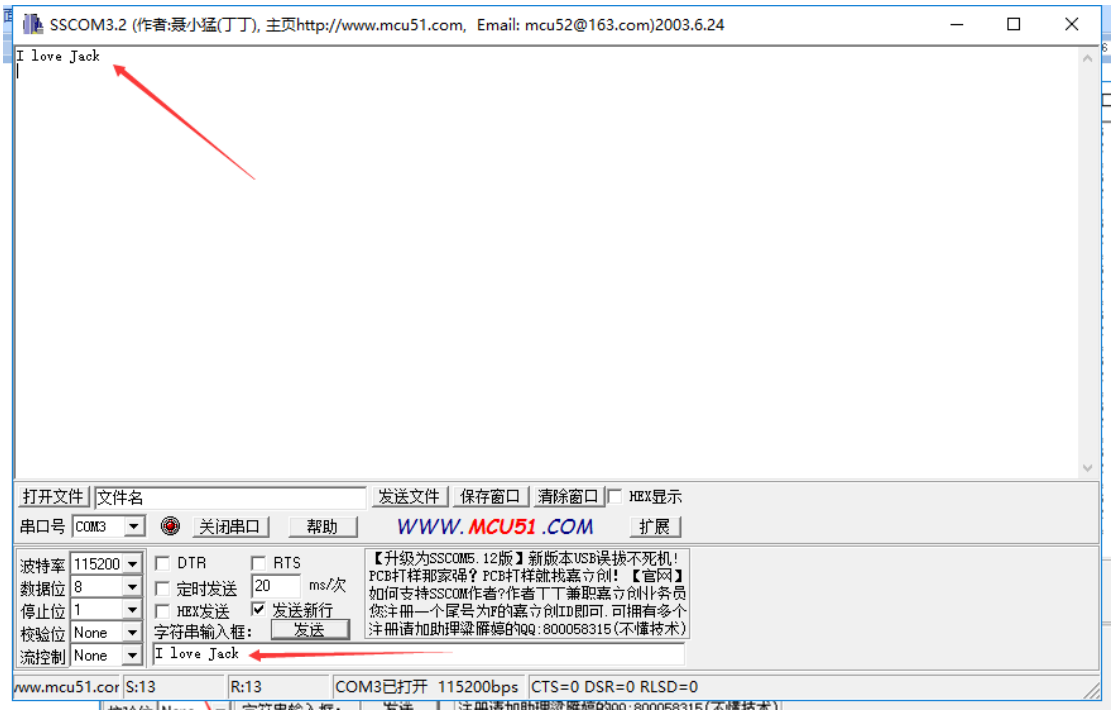
```
/*****************************************************************************
【函    数】   TestUartCB(TTCDriverUartState_t uartState,u8 * buffer,u16 len)
【概    述】   UART接收回调函数
【入口参数】
【返回参数】   无
【说    明】   无
*****************************************************************************/
static void TestUartReadCB(void * uartHandle,u8 * buffer,u16 len){
    uartBuffer = buffer;
    uartLen    = len;
//    static u8 ubf[50] = {0};
//
//    memcpy(ubf,buffer,len);

    if(uLedHandle != NULL){
        TTCDriverIOSetOutputVaule(&uLedHandle,IOID_1,~TTCDriverIOGetOutputValue(IOID_1));
    }

    TTCDriverUartWrite(uartHandle,buffer,len);//把串口接收到数据，又发还给串口
```

原来在串口接收回调中把收到的数据又发送回去啦！

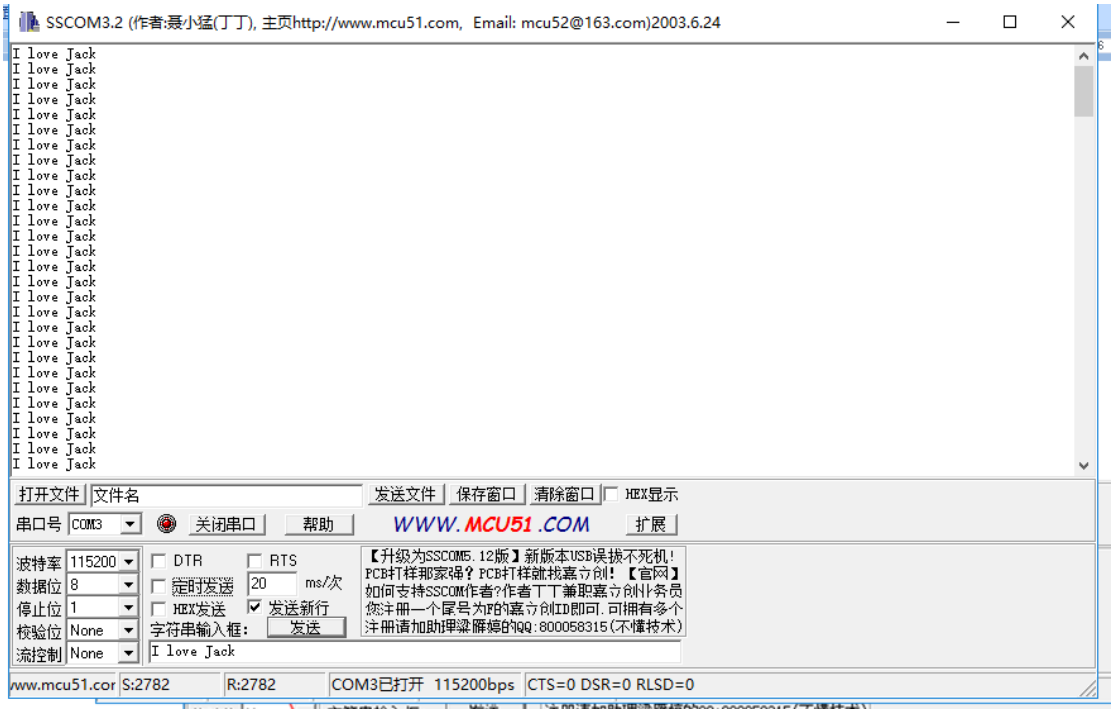
定时传会不会死呀？试试：



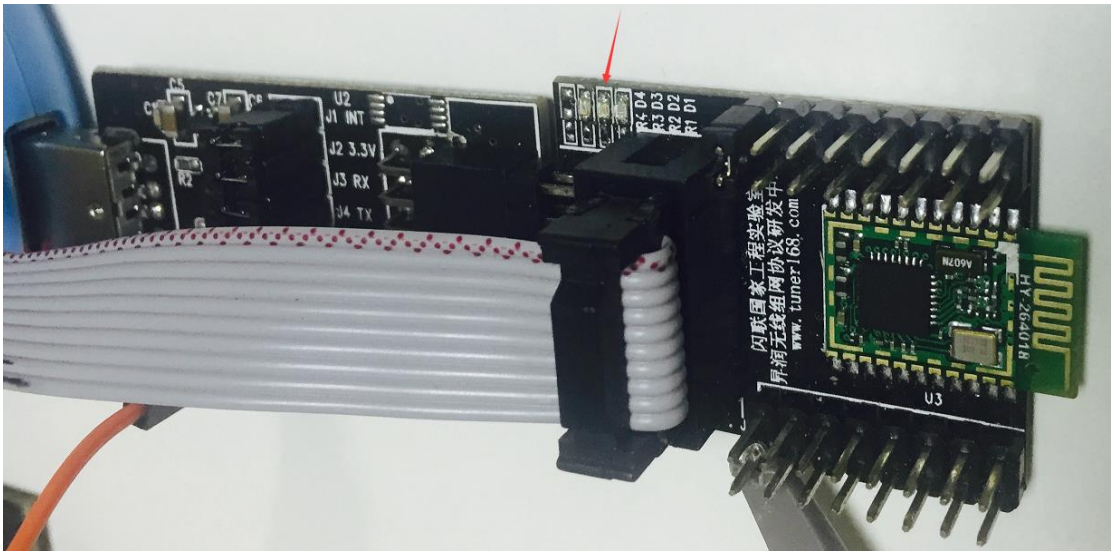
字符行不行呀？



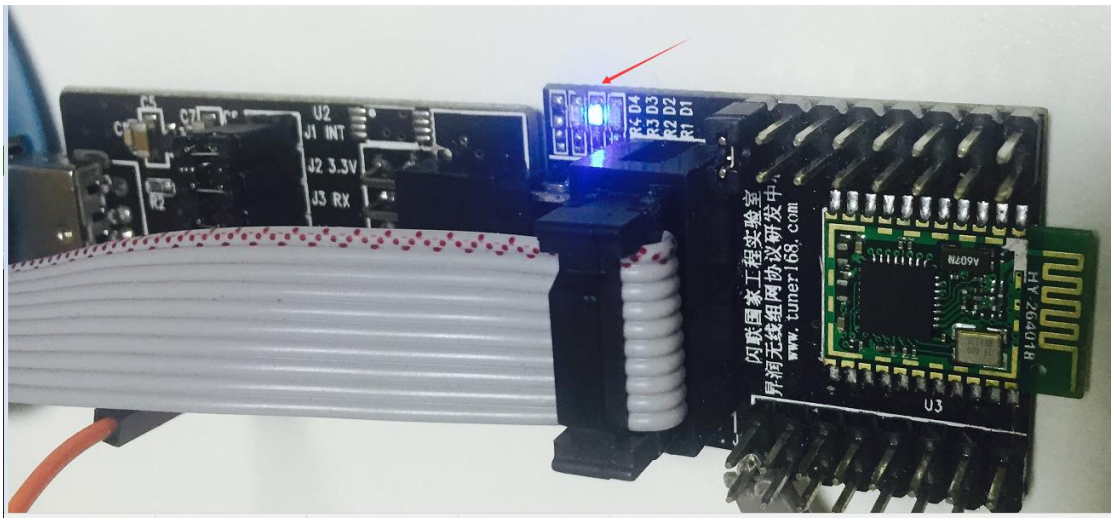
定时发一下：



ＩＯ是不是真的会翻转呀？现在是关啦！



再发一下，是会亮哦！



单独串口测试没有问题啦，把ＢＬＥ收发移进来吧！

首先在 TTCDriverRARTDemo.c 这个文件中，增加二个变量，保存串口接收到的数据指针及收到的数据长度：

```
static u8       *uartBuffer = NULL;
static u16      uartLen = 0;
```

在接收回调中，将数据存到变量中，标注 1；激活串口任务，标注 2；

```
/*****************************************************************************
【函    数】  TestUartCB(TTCDriverUartState_t uartState,u8 * buffer,u16 len)
【概    述】  UART接收回调函数
【入口参数】
【返回参数】  无
【说    明】  无
*****************************************************************************/
static void TestUartReadCB(void * uartHandle,u8 * buffer,u16 len){
    uartBuffer = buffer;
    uartLen    = len;
//    static u8 ubf[50] = {0};
//
//    memcpy(ubf,buffer,len);

    if(uLedHandle != NULL){
        TTCDriverIOSetOutputVaule(&uLedHandle,IOID_1,~TTCDriverIOGetOutputValue(IOID_1))
    }

    TTCDriverUartWrite(uartHandle,buffer,len);//把串口接收到数据，又发还给串口

    TTCSDKDriverUARTSetEvent(TTCBLE_SDK_UART_TO_BLE_EVN);
}
```

1

2

在串口事件中，将数传通过ＢＬＥ传送给手机ＡＰＰ，程式如下修改：

```
/*****************************************************************************
【函    数】  TTCSDKDriverUARTEvent(void)
【概    述】  Demo UART 事件处理
【入口参数】  无
【返回参数】  无
【说    明】  无
*****************************************************************************/
void TTCSDKDriverUARTEvent(void){
//    if(events & TTCBLE_SDK_UART_EVN){
//        events &= ~TTCBLE_SDK_UART_EVN;
//        TTCDriverUartWrite(&uartHandle,"TTCDriverUART Test\r\n",strlen("TTCDriverUART Test\r\n"));
//        Util_startClock(&uClock);
//    }

    if(events & TTCBLE_SDK_UART_TO_BLE_EVN){
        events &= ~TTCBLE_SDK_UART_TO_BLE_EVN;
        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
                                  uartLen,
                                  uartBuffer);
    }
}
```

2

1

标注 1：之前在接收回调保存的数所指针，及数据长度；

标注 2：这个函数是在上次试验中的使用过的向手机传送数据的函数；

同样，我们要把ＢＬＥ接收到的数据也先保存起来，也是先申请好变量：

```
unsigned char        BLE_RxBuffer[20] = {0};
u16        BLE_RxBufferLen = 0;
```

这个要在 TTCBlePeripheralTask.c 这个文件中进行，然后ＢＬＥ接收数据地方把数据保存起来：

```
/*********************************************************************************
【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);

    memcpy(BLE_RxBuffer,TTCData->pValue,TTCData->len);
    BLE_RxBufferLen = TTCData->len;

    TTCSdkSetEvent(sem, &demoEvents,0x01);              //打开一个事件

    ICall_free(TTCData->pValue);
    ICall_free(TTCData);
}

/*********************************************************************************/
```

标注 1 ：复制数据；

标注 2 ：保存接收到的数据长度；

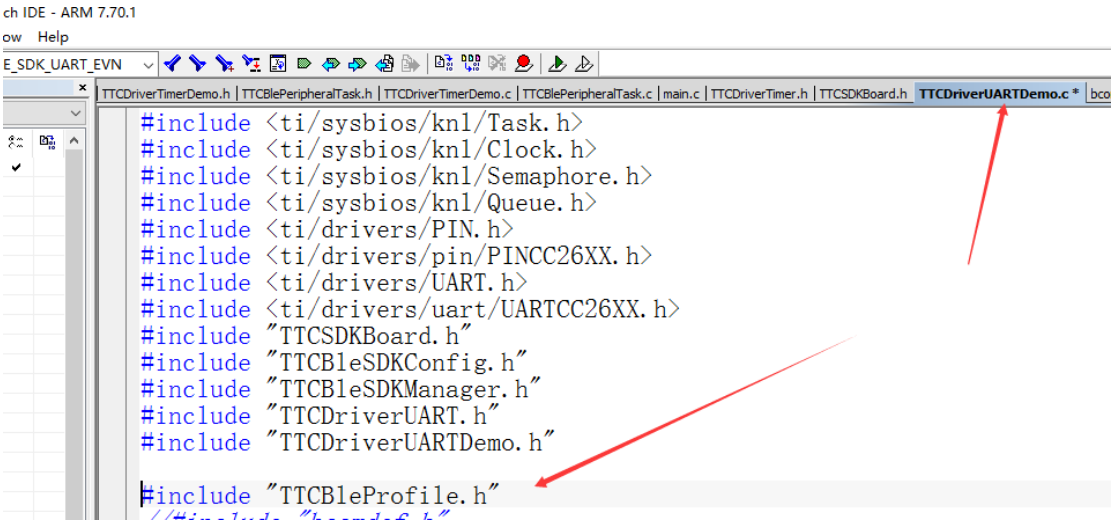
我们在激活的事件中，把数据传送给串口：

```
        if (errno == ICALL_ERRNO_SUCCESS){

            #ifdef TTCBLE_IBEACON
                  TTCBeaconProcessStack(&selfEntity);
            #else
                  TTCBlePeripheralProcessStack(&selfEntity);
            #endif

            while (!Queue_empty(appMsgQueue)){                                  //接收消息
                TTCMsg_t *pMsg = (TTCMsg_t *)Util_dequeueMsg(appMsgQueue);
                if (pMsg){
                    TTCBlePeripheralTaskProcessAppMsg(pMsg);                    //处理消息
                    ICall_free(pMsg);                                           //释放内存
                }
            }
        }
        //用户事件处理代码开始：

        if(demoEvents & 0x01){
          demoEvents &= ~0x01;
          KeyPressHandler(0x01);//翻转一次ＧＰＩＯ０

          TTCDriverUartWrite(&uartHandle,BLE_RxBuffer,BLE_RxBufferLen);//把BLE接收到数据，又发给串口
```

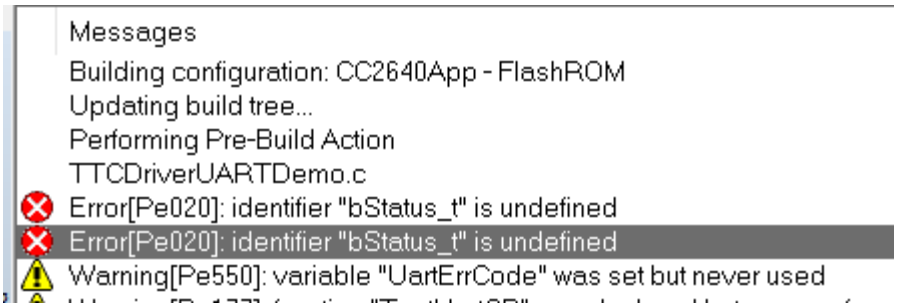
先编译一下，看可行不？

```
Messages
Building configuration: CC2640App - FlashROM
Updating build tree...
Performing Pre-Build Action
TTCDriverUARTDemo.c
Warning[Pe223]: function "TTCBleProfileSetParameter" declared implicitly
Error[Pe020]: identifier "TTCBLE_PROFILE_CHAR2" is undefined
Warning[Pe550]: variable "UartErrCode" was set but never used
Warning[Pe177]: function "TestUartCB" was declared but never referenced
Warning[Pe177]: function "TTCDriverDemoUARTClockHandler" was declared but never referenced
Error while running C/C++ Compiler

Total number of errors: 1
Total number of warnings: 4
```

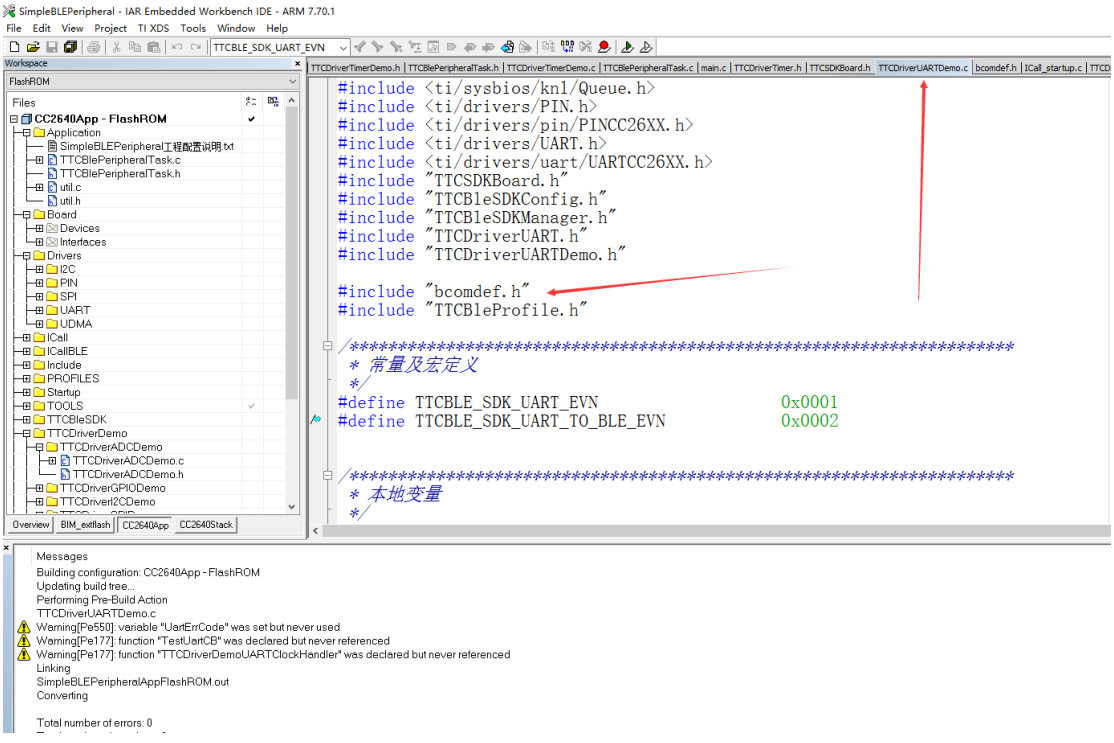
没有定义？好吧！应该是头文件没有进来，加上：



再来，晕，还是不行，看看：

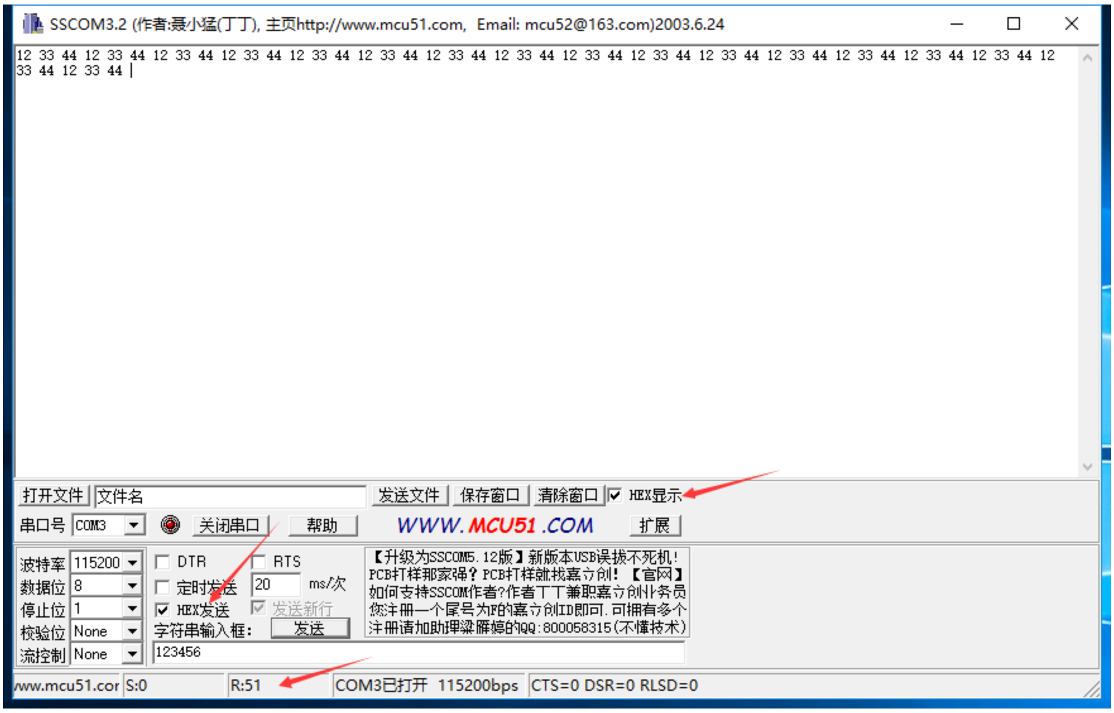


类型没有？加上，再来：



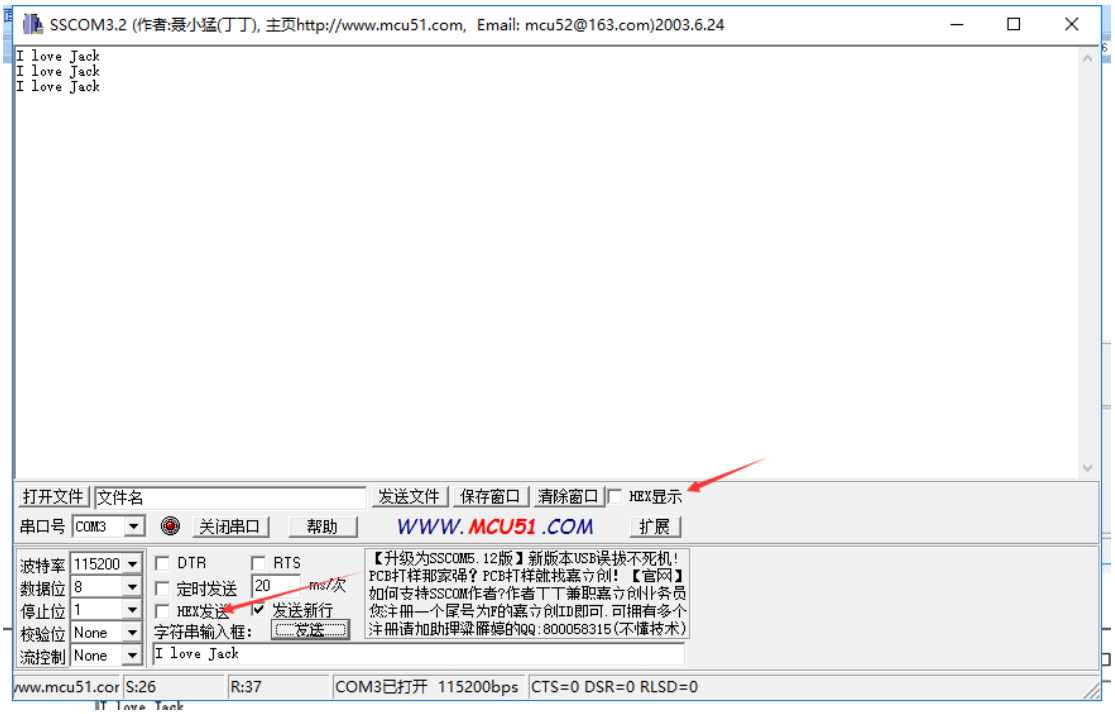
好像可以，我下载到目标板来试试，先用手机ＡＰＰ联上目标板，进到透传。

我们先测试１６进制模式：

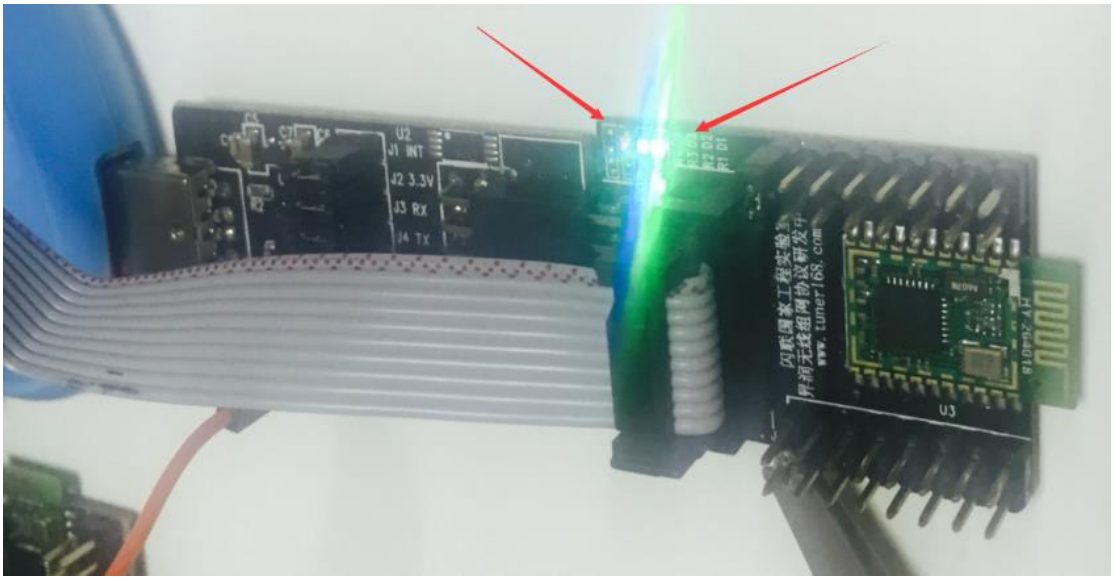

SSCOM3.2 (作者:聂小猛(丁丁), 主页http://www.mcu51.com, Email: mcu52@163.com)2003.6.24
12 33 44 12 33 44 12 33 44 12 33 44 12 33 44 12 33 44 12 33 44 12 33 44 12 33 44 12 33 44 12 33 44 12 33 44 12 33 44 12 33 44 12 33 44 12 33 44 12 33 44
打开文件
文件名
发送文件
保存窗口
清除窗口
HEX显示
串口号 COM3
关闭串口
帮助
WWW.MCU51.COM
扩展
波特率 115200
数据位 8
停止位 1
校验位 None
流控制 None
DTR
RTS
定时发送
20 ms/次
HEX发送
发送新行
字符串输入框:
发送
123456
www.mcu51.cor S:0
R:51
COM3已打开 115200bps
CTS=0 DSR=0 RLSD=0

中国移动
01:35
-71
清除数据
RX:51Bytes
是否显示多行
12 33 44
TX:51Bytes
123344
定时发送 间隔 200 ms(20~4000ms)
是否显示ASCII码值
发送3个字节
自定义
数传
模组设置

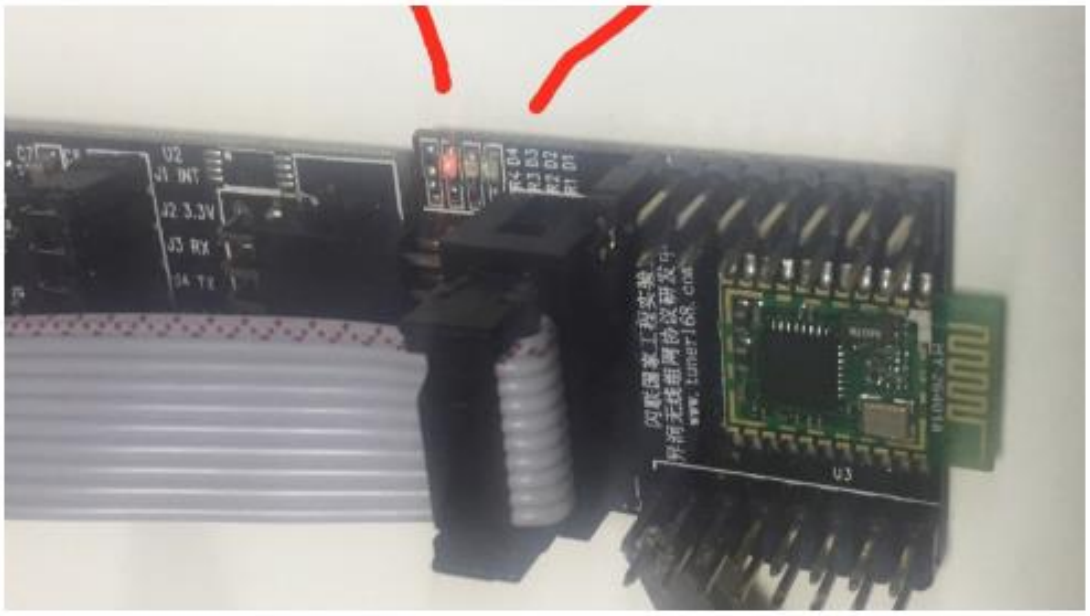
字符模式，测试看看：



看看，二个不同ＩＯ的播转功能还有没有：



还是可以翻转的：



串口与ＡＰＰ通讯功能就到这里！