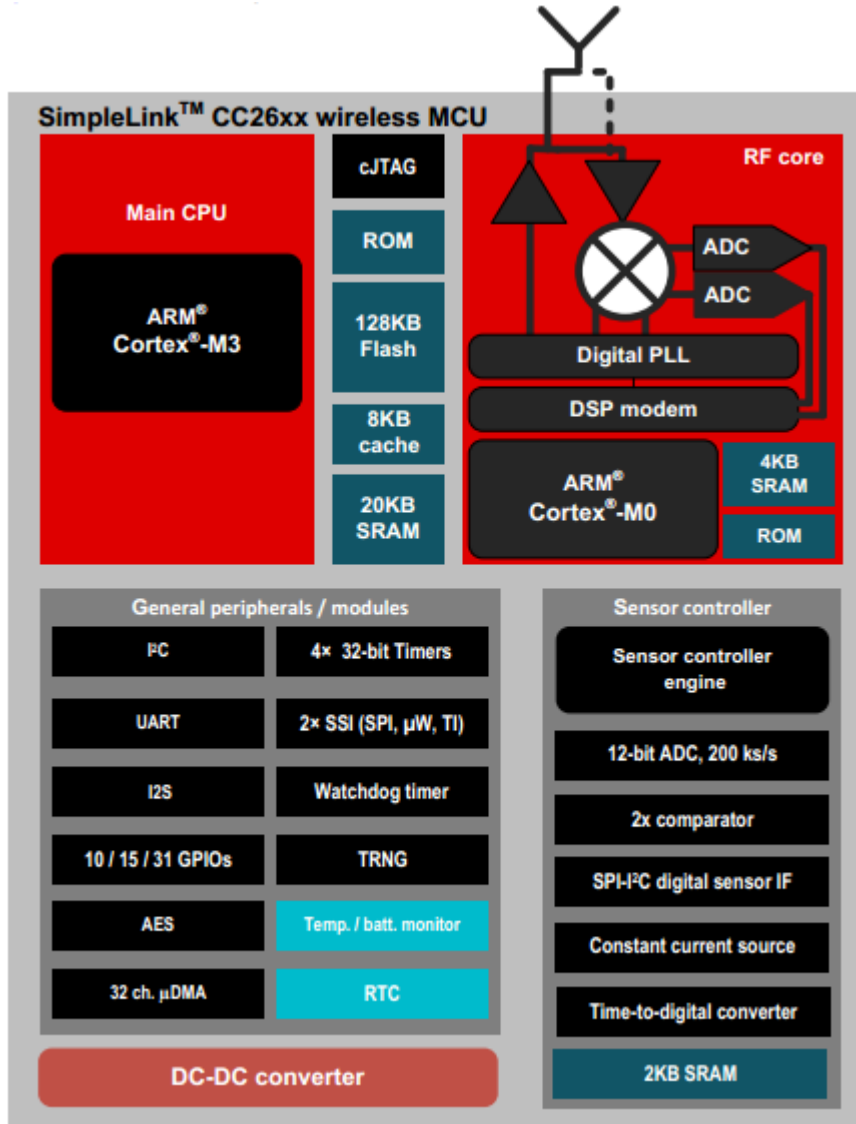


## 昇润 CC2640 SDK 应用入门教程五

BLE 如何控制 PWM 输出

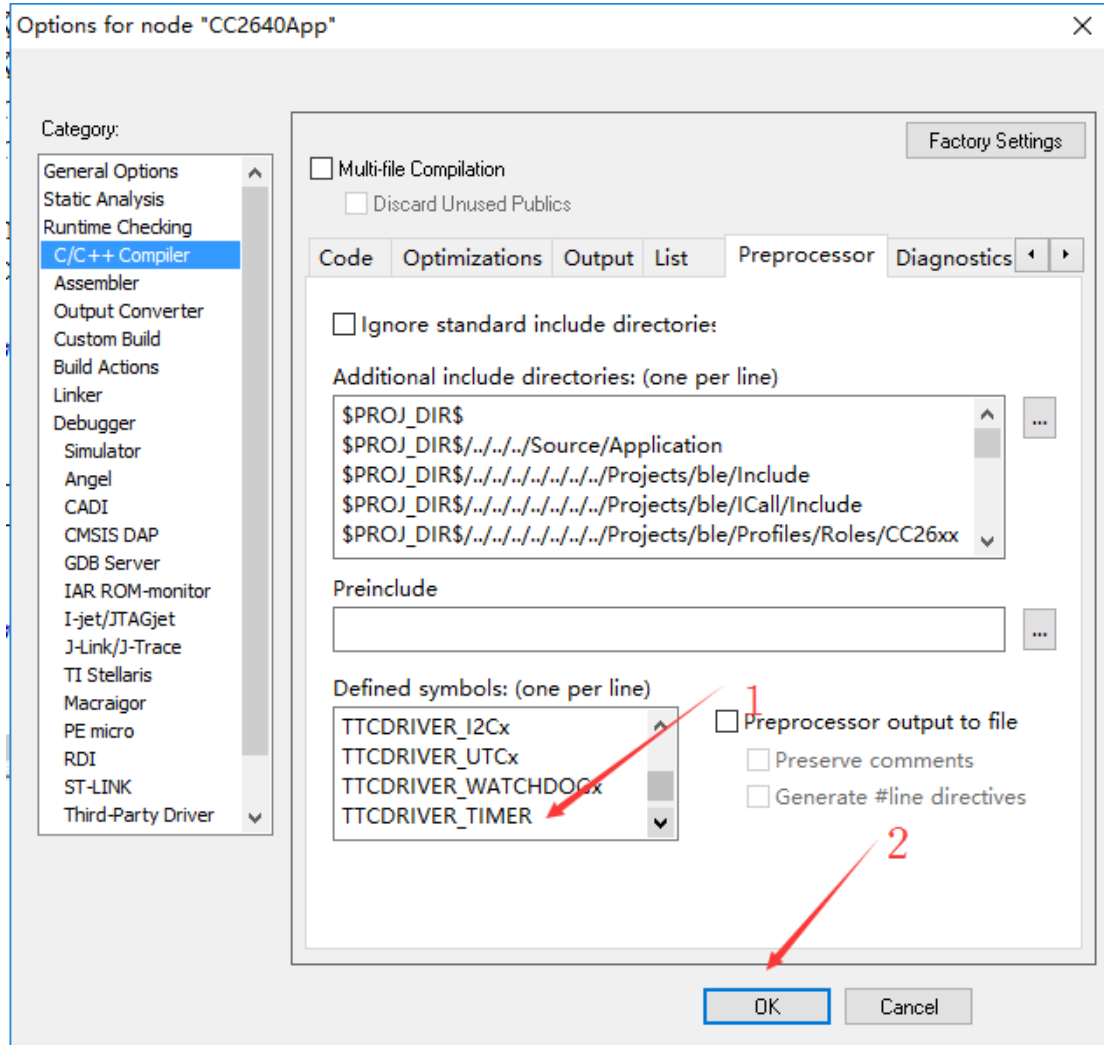
有老司机说 CC2640 有 PWM，手机 A P P 可以控制 P W M 输出吗？



Copyright © 2016, Texas Instruments Incorporated

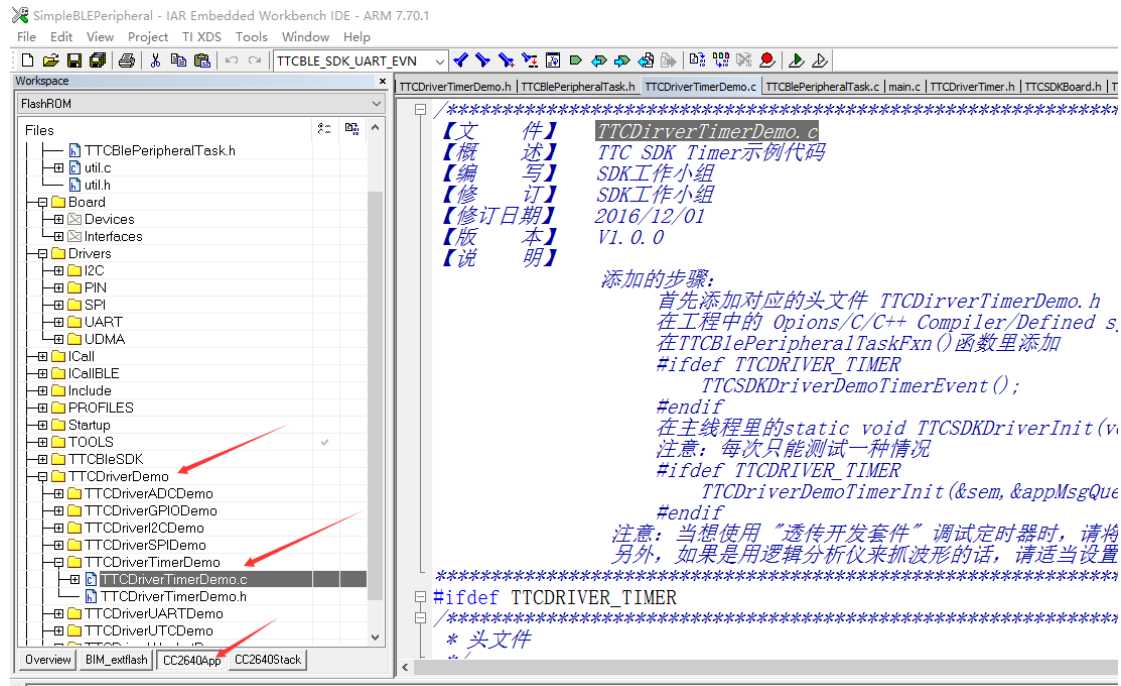
先看一下芯片模块图，是有 4 路 3 2 bit 定时器，3 2 位的芯片均是通过定时器来实现 P W M 控制，我们也来试试：

首先把定时器测试示例宏定义打开：



看一下示例程式：

TTCDirverTimerDemo.c



好多功能：

功能说明：

- 1、配置任意一个定时器输出一路 PWM 信号，并让这路 PWM 在 Board\_PWM0-Board\_PWM7 脚之间来回切换输出。
- 2、为任意一个定时器配置定时中断功能，在中断回调函数里翻转 Board\_PWM1 脚的输出状态
- 3、配置定时器 CC2650\_TIMER0\_A 从 Board\_PWM0 脚输出 PWM 信号；  
配置指定的定时器为输入计数模式，信号从 Board\_PWM2 脚输入，在计数中断回调函数里翻转 Board\_PWM3 脚的输出状态。
- 4、配置定时器 CC2650\_TIMER0\_A 从 Board\_PWM0 脚输出 PWM 信号；  
配置指定的定时器 TimerName 为输入捕获模式，信号从 Board\_PWM2 脚输入，在捕获中断回调函数里翻转 Board\_PWM7，同时若是开启了 UART 驱动，则会通过 UART 打印两次捕获的时间差（此差值就是输入信



号两个边沿之间时间)。

哇，这么多，搞定一个简单点的先：

```

void TTCDriverDemoTimerInit(ICall_Semaphore * sem,
                             Queue_Handle * appMsgQueue,
                             CC2650_TimerName TimerName) {
    TTCDriverSinglePWMSignalTest(sem, appMsgQueue, TimerName); //PWM输出
    // TTCDriverSingleInterSignalTest(sem, appMsgQueue, TimerName); //定时中断
    // TTCDriverSingleCountSignalTest(sem, appMsgQueue, TimerName); //外部输入计数
    // TTCDriverSingleCatSignalTest(sem, appMsgQueue, TimerName); //外部输入捕获
}

```

看看 PWM 输出功能：

```

static void TTCDriverSinglePWMSignalTest(ICall_Semaphore * sem,
                                           Queue_Handle * appMsgQueue,
                                           CC2650_TimerName TimerName) {
    if(sem == NULL || appMsgQueue == NULL) {
        return;
    }
    TimSem = sem;
    TimMsgQueue = appMsgQueue;
    TimErrCode = TTCSDKDriver_TIM_Init(TimSem, TimMsgQueue, TimerName); //初始化 定时器
    if(TimErrCode != TTCDRIVER_INIT_SUCCESS) {
        return;
    }
    TimErrCode = TTCSDKDriver_PWMInit(TimerName, IOID_0); //配置PWM功能 信号从Board_PWM0输出
    if(TimErrCode != TTCDRIVER_INIT_SUCCESS) {
        return;
    }
    Util_constructClock(&TimerClock, //配置一个软件定时器
                       TTCDriverDemoTimerClockHandler, //定时100ms
                       100, //立即启动
                       0,
                       true,
                       TTCBLE_SDK_TIME_EVN);
}

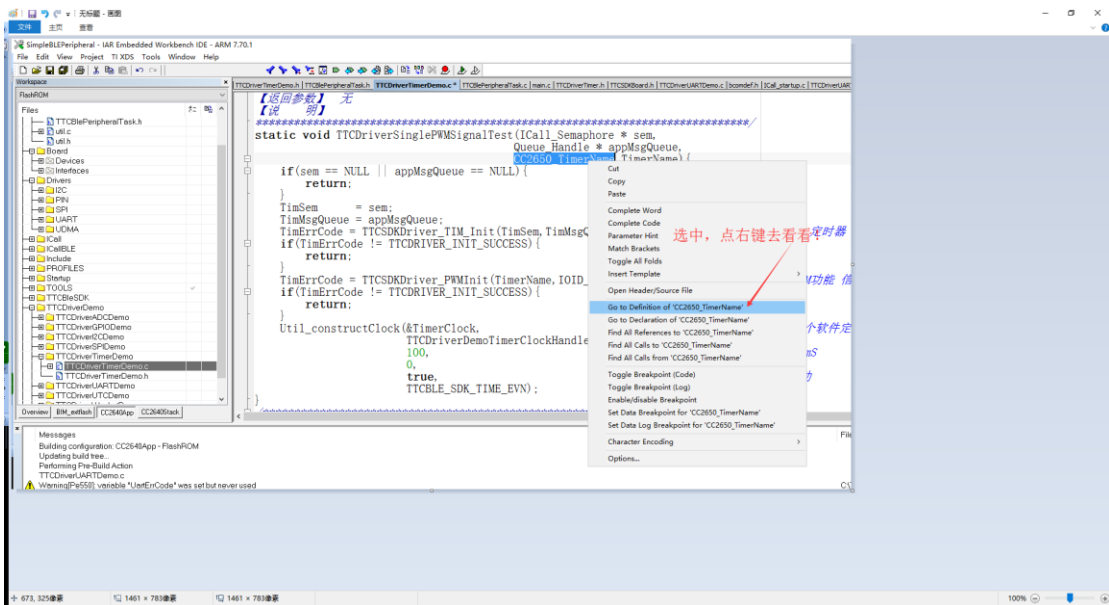
```

设定一个定时器在 IOID\_0 输出 PWM。看看 S D K 有多少个定时器：

```

*****
est(ICall_Semaphore * sem,
    Queue_Handle * appMsgQueue,
    CC2650_TimerName TimerName) {
NULL) {

```



```

DE - ARM 7.70.1
Help
K_UART_EVN
TTCDriverTimerDemo.h | TTCBlePeripheralTask.h | TTCDriverTimerDemo.c * | TTCBlePeripheralTask.c | main.c | TTCDriverTimer.h | TTCSDKBoard.h | TTCDriverUARTDemo
CC2650_TIMER3_A 和 CC2650_TIMER3_B 对应使用 CC2650_TIMER3定时器
所有PWM均可同时运行
*/
typedef enum {
    CC2650_TIMER0_A = 0,
    CC2650_TIMER0_B = 1,
    CC2650_TIMER1_A = 2,
    CC2650_TIMER1_B = 3,
    CC2650_TIMER2_A = 4,
    CC2650_TIMER2_B = 5,
    CC2650_TIMER3_A = 6,
    CC2650_TIMER3_B = 7,
    CC2650_TIMERSIZE,
} CC2650_TimerName;

```

原来 S D K 把 4 路 3 2 位的定时分为了 8 路 1 6 位的；想了解更多关于定时器的设置可以仔细看看头文件中的说明：

```

【编写】 SDK工作小组
【修订】 SDK工作小组
【修订日期】 2016/12/3
【版本】 V1.0.9
【说明】 定时器说明:
        在工程中添加以下头文件
        #include "TTCDriverTimer.h"
        static Timer_Handle timerHandle;
        TTCDriverTimerInit_t timerInitParam = {
            .sem = &sem,
            .queueHandle = &appMsgQueue,
            .timerName = (CC2650_TimerName) (CC2650_TIMER0_A),
        };
        TTCDriverTimerInit (&TTCBlePeripheralTaskC1s,
            timerInitParam,
            &timerHandle);
        注意: 初始化时必须定义一个Timer_Handle。否则无法操作定时器。
1. PWM说明
typedef struct{
    u32 pwmTimerValue :16; //PWM 定时初值
    u32 pwmMatchValue :16; //PWM 匹配值
    u32 pwmPolarity :2; //PWM输出极性 (CC2650_PWM)
    u32 pwmAndCCP :1; //PWM与输出使能 (TRUE: 1)
    u32 pwmIdlePinState :1; //PWM闲置状态设置 (1: PWM)
    u32 retain :28; //保留
} TTCDriverTimerPwmParams_t;
pwmTimerValue : 载入定时器初值
pwmMatchValue : 匹配值
pwmPolarity : PWM输出极性 (默认为正极性, 即高电平有效)
pwmAndCCP : PWM输出比较波形使能, 该功能需要CC2650_TIMERn_A与CC2650_TIMERn_B一起使用
            使能后, CC2650_TIMERn_A的引脚输出比较波形。
            若CC2650_TIMERn_A输出高, 且CC2650_TIMERn_B输出高, 则CCP波形输出高。
            若CC2650_TIMERn_A或CC2650_TIMERn_B其中一个有输出低, 则CCP波形输出低。
pwmIdlePinState: 当PWM处于闲置状态时, 将会输出指定的电平状态

```

还有 S D K 说明文档:



看一下原来的定时器事件：

```
/*
【函数】 TTCSDKDriverDemoTimerEvent(UArg arg)
【概述】 标记事件并唤醒线程
【入口参数】 arg：标记事件
【返回参数】 无
【说明】 无
*/
void TTCSDKDriverDemoTimerEvent(void) {
    if(events & TTCBLE_SDK_TIME_EVN) {
        events &= ~TTCBLE_SDK_TIME_EVN;
        TTCDriverSinglePWMSignalDispose(CC2650_TIMER0_A);
    }
}
```

原程式看看：

```
/*
【函数】 TTCDriverSinglePWMSignalDispose(CC2650_TimerName TimerName)
【概述】 实现将一路PWM定时在PWM0-PWM7之间来回切换输出
【入口参数】 无
【返回参数】 无
【说明】 测试时请检查IOID_0 - IOID_7 处于可用状态(没被申请), 另外需要实际查看
        一下对应的脚有没有接什么外部的电路(“透传套件”上的跳帽, 还有下载线需要拔掉).
*/
static void TTCDriverSinglePWMSignalDispose(CC2650_TimerName TimerName) {
    static u8 cntt = 0;
    u8 i, j;
    cntt++;
    if(cntt == 8) {
        cntt = 0;
    }
    TTCDriverTimerPwmParams_t param = {
        48000,
        24000,
        PWM_OUTPUT_NOT_INVERTED,
        false,
        0
    };
    i = TimerName/2;
    j = TimerName%2;
    TimErrCode = TTCDriverTimerPwmSetParam(&Handle[i][j],
        param,
        Board_PWM0+cntt,
        true);

    if(TimErrCode != TTCDRIVER_INIT_SUCCESS) {
        return;
    }
    Util_startClock(&TimerClock);
}
```

*//48000000/48000 = 1000Hz*

*//定时器的句柄 即指定此PWM的  
//PWM的参数  
//指定PWM信号的输出脚  
//立即使能定时器*

8个 I O 上来回变化, 而且不断重复, 这么多, 复制一下: 我们简单点:

```
static void TTCDriverSinglePWMSignalDisposeSet(CC2650_TimerName TimerName, u32 PWM_MatchValue) {
    u8 i, j;

    TTCDriverTimerPwmParams_t param = {
        48000,
        PWM_MatchValue,
        PWM_OUTPUT_NOT_INVERTED,
        false,
        0
    };
    //param.pwmMatchValue
    i = TimerName/2;
    j = TimerName%2;
    TimErrCode = TTCDriverTimerPwmSetParam(&Handle[i][j],
        param,
        Board_PWM7,
        true);

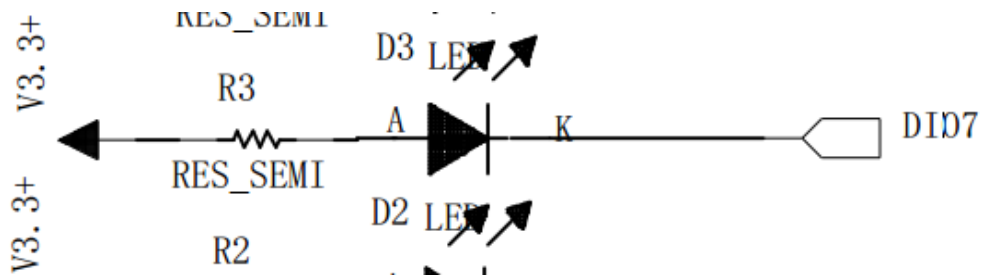
    if(TimErrCode != TTCDRIVER_INIT_SUCCESS) {
        return;
    }
    //Util_startClock(&TimerClock);
}
```

*//48000000/48000 = 1000Hz*

*//定时器的句柄 即指定此PWM信号是基于哪个  
//PWM的参数  
//指定PWM信号的输出脚  
//立即使能定时器*

把占空比的部分放出来, 由外面的参数来决定;

把输出 I O 固定一下, 开发板的第三个 L E D 还没有用起来, D I O 7 试试:



执行完毕，又去执行，我们先关掉，第更新一次 A P P 的数据，我们就改写一次 P W M 的占空比：

透传 A P P 的自定义界面有滑条，我们来试试看：



进去把 R 这个滑条的最大，最小设定一下，确定；最大为什么要设 0xbb80？因为这里设定满格是 48000 呀：

```

static void TTCDriverSinglePWMSignalDisposeSet(CC2650_TimerName TimerName, u32 PWM_MatchValue) {
    u8 i, j;
    TTCDriverTimerPwmParams_t param = {
        48000, //48000000/48000 = 1000Hz
        PWM_MatchValue, //24000,
        PWM_OUTPUT_NOT_INVERTED,
        false,
        0
    };
    //param.pwmMatchValue
    i = TimerName/2;
    j = TimerName%2;
    TimErrCode = TTCDriverTimerPwmSetParam(&Handle[i][j], param);
}

```

是什么东东，我们确认一下：



```

/* PWM参数设置 */
typedef struct{
    u32  pwmTimerValue   : 16;
    u32  pwmMatchValue   : 16;
    u32  pwmPolarity     : 2;
    u32  pwmAndCCP       : 1;
    u32  pwmIdlePinState : 1;
    u32  retain          : 28;
} TTCDriverTimerPwmParams_t;

```

//PWM 定时初值  
 //PWM 匹配值  
 //PWM输出极性(CC2650\_PWMpolarity)  
 //PWM与输出使能(TRUE: 与输出使能。FALSE: 与输出使能)  
 //PWM闲置状态设置(1: PWM闲置状态输出高电平, 0: 输出低电平)  
 //保留

先把 BLE 收到的二个字节数据拼到一个变量里，用这个变量来作为 PWM 的占空比，二个文件之前要传送参数，搞成一个全局变量：

```

#ifdef FEATURE_OAD
#define OAD_PACKET_SIZE 2 //((OAD_PACKET_SIZE))
#endif

unsigned int BLEpwmMatchValue = 0;

unsigned char BLE_RxBuffer[20] = {0};
uint8_t BLE_RxBufferLen = 0;

//extern UART_Handle uartHandle;

```

数据拼一下,方向别搞错：

```

//用户事件处理代码开始:
if(demoEvents & 0x01){
    demoEvents &= ~0x01;
    KeyPressHandler(0x01); //翻转一次 GPIO 0

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

    BLEpwmMatchValue = (u32)BLE_RxBuffer[1];
    BLEpwmMatchValue |= (u32)BLE_RxBuffer[0]<<8;

    TTCDriverDemoTimerSetEvent(TTCBLE_SDK_TIME_EVN);
}
//用户事件处理代码结束;

```

拼好后，激活定时器事件：

```

*****
【函数】 TTCSDKDriverDemoTimerEvent(UArg arg)
【概述】 标记事件并唤醒线程
【入口参数】 arg : 标记事件
【返回参数】 无
【说明】 无
*****
void TTCSDKDriverDemoTimerEvent(void){
    if(events & TTCBLE_SDK_TIME_EVN){
        events &= ~TTCBLE_SDK_TIME_EVN;
        // TTCDriverSinglePWMSignalDispose(CC2650_TIMER0_A);
        TTCDriverSinglePWMSignalDisposeSet(CC2650_TIMER0_A, BLEpwmMatchValue);
    }
}

```

指定一下定时器      参数放这！

编译看看,不行！发现没有定义变量，函数！不对呀！明明定啦！

检查一下头文件 TTCDriverTimerDemo.h

```

/*****
【函数】 TtCDriverDemoTimerSetEvent (UArg arg)
【概述】
【入口参数】 无
【返回参数】 无
【说明】 无
*****/
extern void TtCDriverDemoTimerSetEvent (UArg arg);

```

把事件宏移到头文件：

```

#include TtCDriverTimer.h
/*****
* 常量及宏定义
*/
#define TTCBLE_SDK_TIME_EVN 0x0008

```

再编译看看,不行! TtCDriverTimerDemo.c 原来就定义是静态的啦, 先屏掉:

```

static Queue_Handle * TimMsgQueue;
static ul6 events;
/*****
* 本地函数声明
*/
static void TtCDriverDemoTimerClockHandler (UArg arg);
//static void TtCDriverDemoTimerSetEvent (UArg arg);
static TtCDriverInfo_t TtCSDKDriver_PWMInit (CC2650_TimerName TimerName);
static TtCDriverInfo_t TtCSDKDriver_Inter_Init (CC2650_TimerName TimerName);
static TtCDriverInfo_t TtCSDKDriver_Cat_Init (CC2650_TimerName TimerName);
static TtCDriverInfo_t TtCSDKDriver_Count_Init (CC2650_TimerName TimerName);

```

再编译看看,还是不行! 找找!

```

#include TtCDriverTimerDemo.h

#include "bcomdef.h"
#include "TtCBleProfile.h"

```

把这二个头文件包到 TtCDriverTimerDemo.c, 把全局变量引进来:

```
RT_EVN
TTCDriverTimerDemo.h | TTCBlePeripheralTask.h | TTCDriverTimerDemo.c | TTCBlePeripheralTask.c | main.c | TTCDriverTimer.h | TTCSDKBoard.h | TTCDriverUARTDemo.c | bcomdef.h * | Icall

// ICall and Dispatch - Messages IDs (0x80 - 0x8F)
#define ICALL_EVENT_EVENT          0x80 //!< ICall Event message
#define ICALL_CMD_EVENT           0x81 //!< ICall Command Event message
#define DISPATCH_CMD_EVENT        0x82 //!< Dispatch Command Event message

/** @) End BLE_MSG_IDS */

/*
 * TYPEDEFS
 */

//! BLE Generic Status return: @ref BLE_STATUS_VALUES
typedef Status_t bStatus_t;

/** @) End GAP_MSG_EVENT_DEFINES */

/*
 * System Events
 */

/*
 * Global System Messages
 */

/*
 * MACROS
 */

// TI Base 128-bit UUID: F000XXXX-0451-4000-B000-000000000000
#define TI_BASE_UUID_128( uuid )  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xB0, \
                                0x00, 0x40, 0x51, 0x04, LO_UINT16( uuid ), HI_UINT16( uuid ), 0x00, 0xF0

/*
 * GLOBAL VARIABLES
 */
extern unsigned int BLEpwmMatchValue;

/*
 * FUNCTIONAL
 */
```

还是不行？问题在哪呢？

```
#ifndef TTCDRIVER_Timer
#include "TTCDriverTimerDemo.h"
#endif
```

原来 S D K 有坑呀！这谁挖的呀！没有区分大小写呀！

改过来！

```
#ifndef TTCDRIVER_TIMER
#include "TTCDriverTimerDemo.h"
#endif
```

还面还有好几个都是这样！都改改：

```
#ifndef TTCDRIVER_GPIO
| #include "TTCDriverGPIDemo.h"
| #endif

#ifndef TTCDRIVER_I2C
| #include "TTCDriverI2CDemo.h"
| #endif

#ifndef TTCDRIVER_SPI
| #include "TTCDriverSPIDemo.h"
| #endif

#ifndef TTCDRIVER_TIMER
| #include "TTCDriverTimerDemo.h"
| #endif

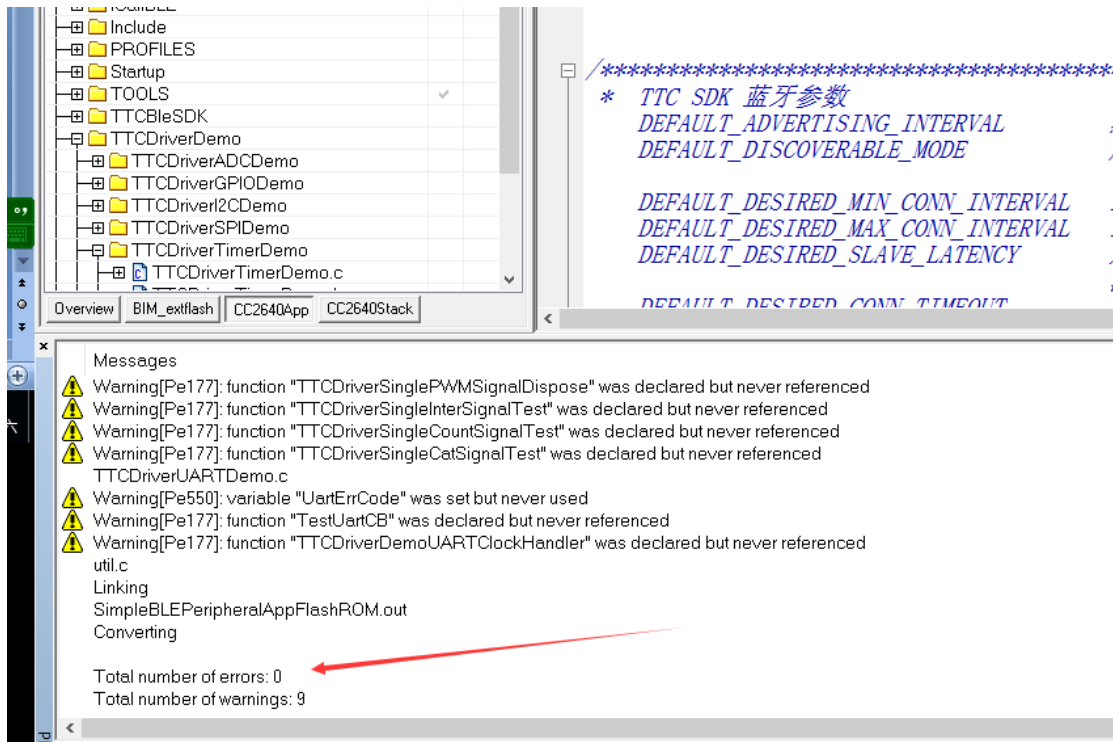
#ifndef TTCDRIVER_UART
| #include "TTCDriverUARTDemo.h"
| #endif

#ifndef TTCDRIVER_UTC
| #include "TTCDriverUTCdemo.h"
| #endif

#ifndef TTCBLE_WECHAT
| #include "TTCDriverWechatDemo.h"
| #endif

#ifndef TTCBLE_IBEACON
| #include "TTCBeacon.h"
| #endif
```

我再来！O啦！

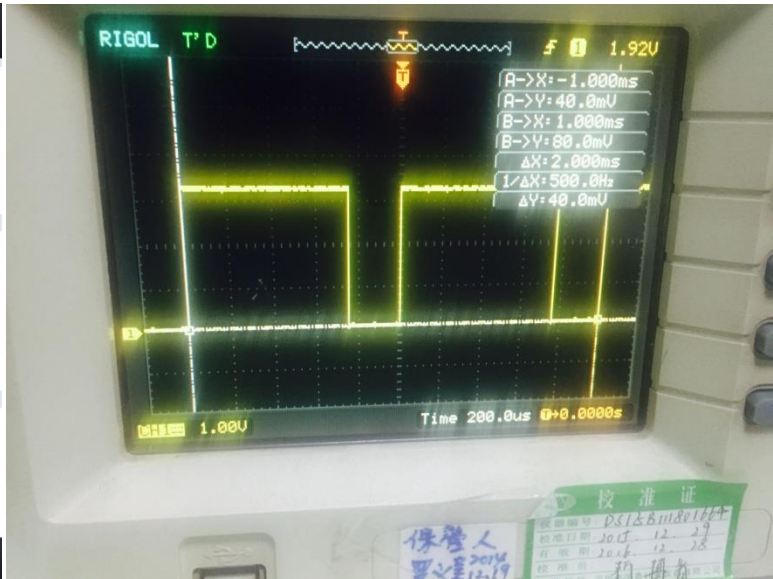
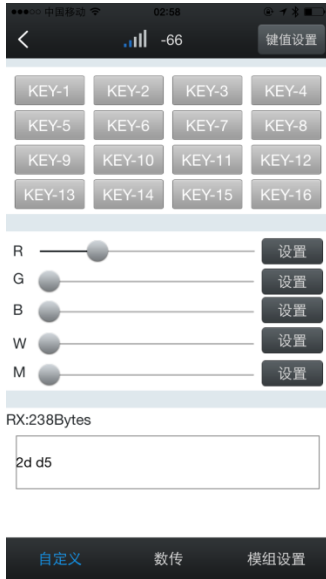


仿真，没问题，联上手机，试试滑条：

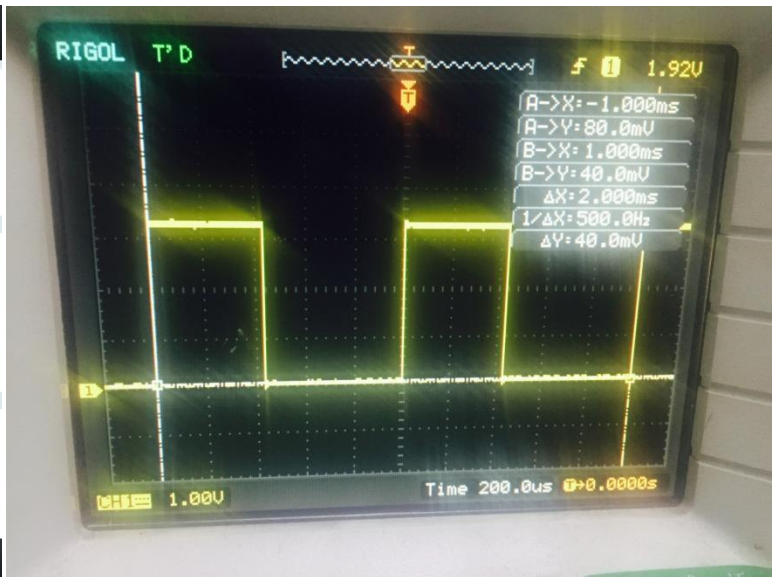
滑条输出 0x0000 看看,示波器测试 IO 一直为高：



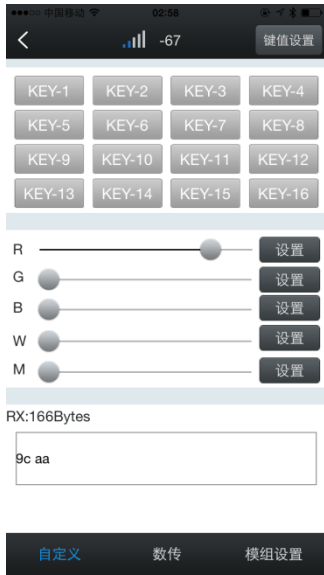
滑条输出 0x2dd5 看看,示波器测试 IO:



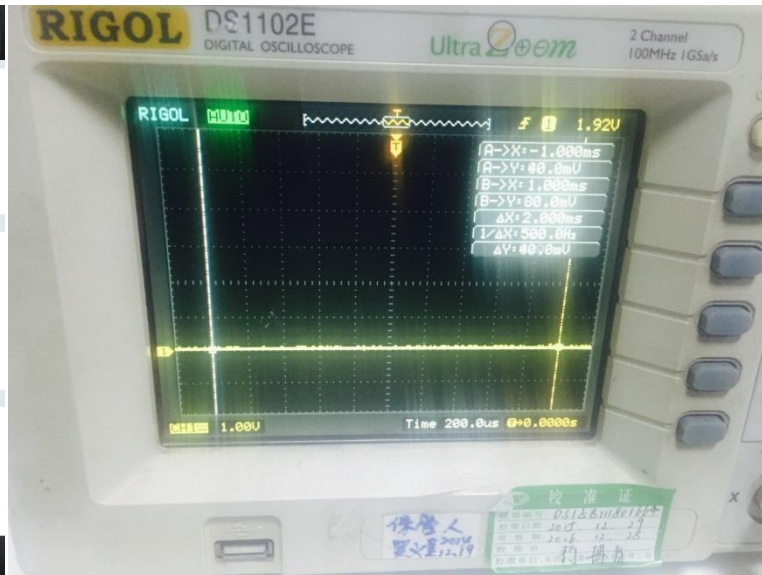
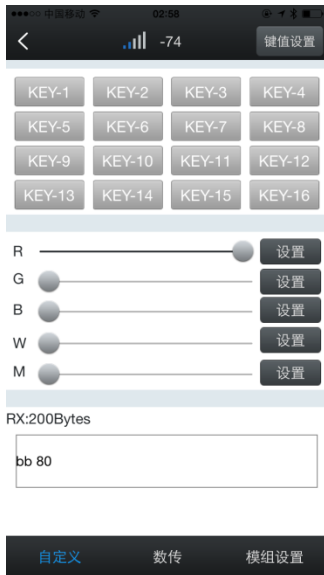
滑条输出 0x6aaa 看看,示波器测试 IO:



滑条输出 0x9caa 看看,示波器测试 IO:



滑条输出 0xbb80 看看,示波器测试 IO 输出低电平:



反复调试滑条,可以得到每一段的数据,记得,不要超出 4 8 0 0 0 的量程!