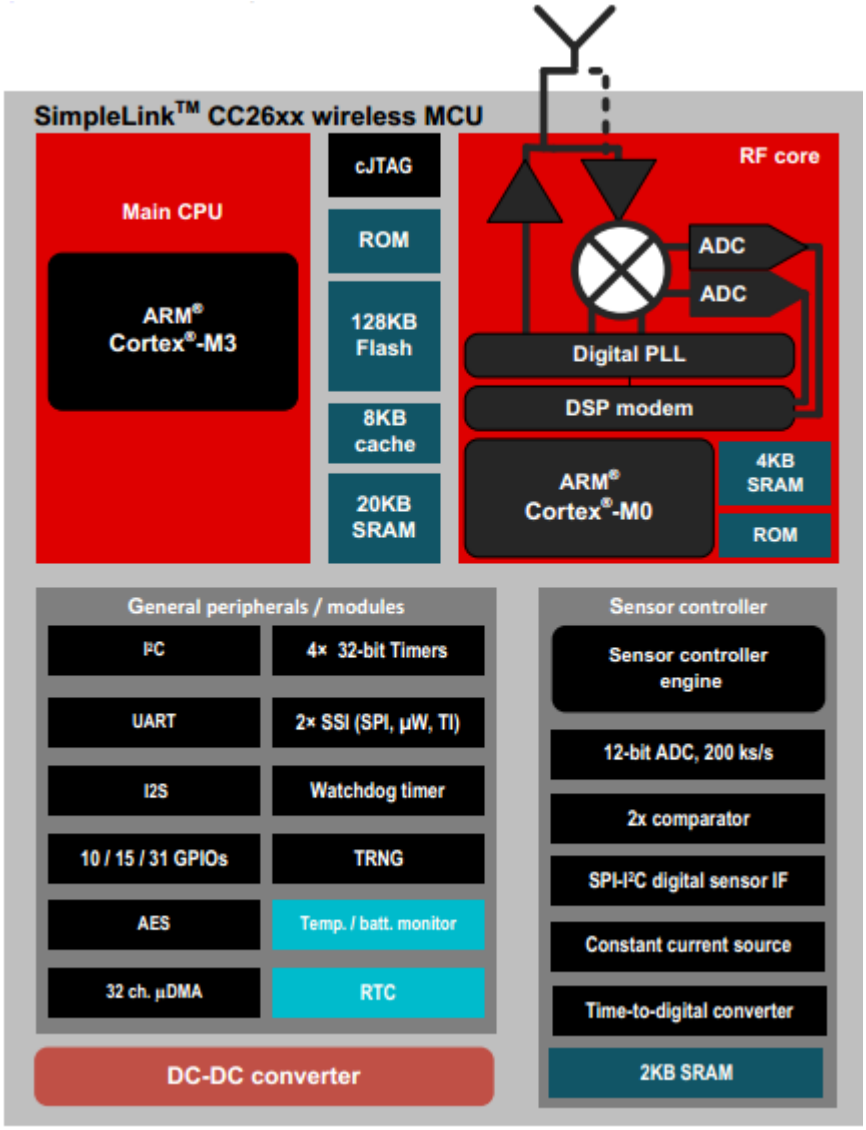
昇润 CC2640 SDK 应用入门教程五

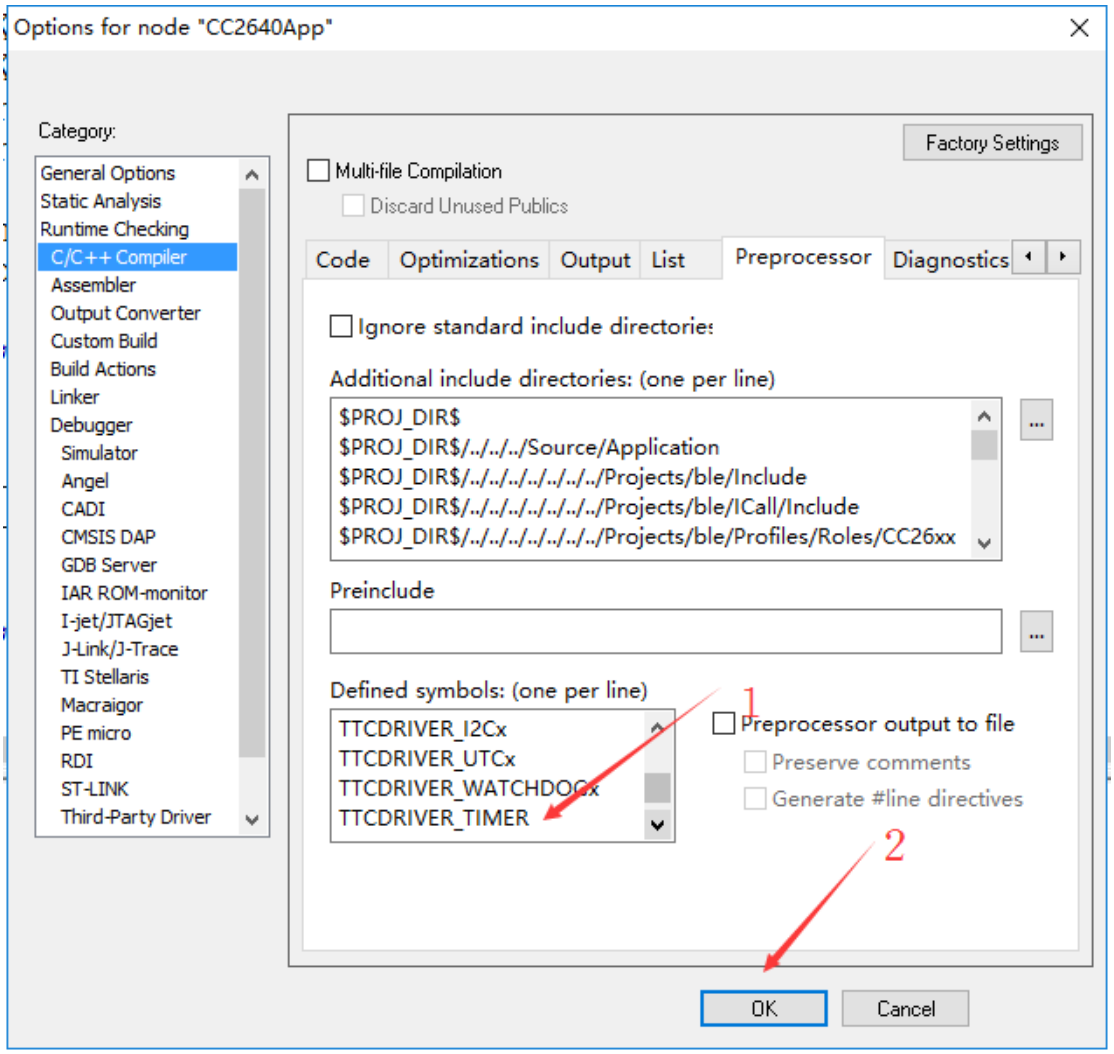
BLE 如何控制 PWM 输出

有老司机说 CC2640 有 PWM，手机ＡＰＰ可以控制ＰＷＭ输出吗？



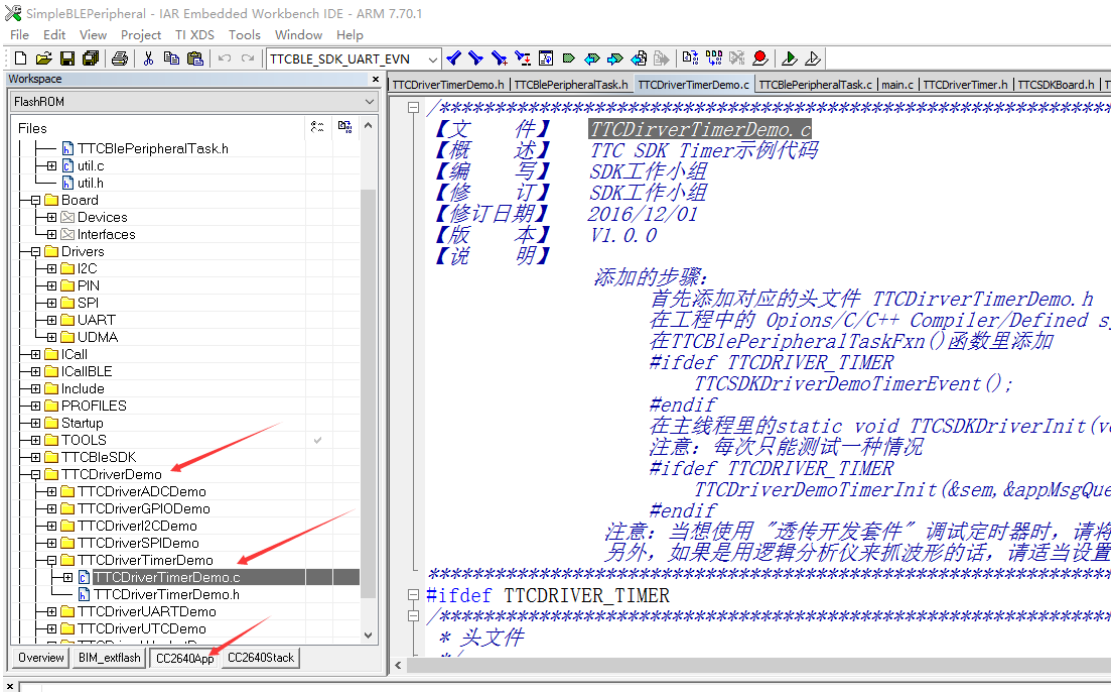
先看一下芯片模块图，是有４路３２bit 定时器，３２位的芯片均是通过定时器来实现ＰＷＭ控制，我们也来试试：

首先把定时器测试示例宏定义打开：



看一下示例子程式：

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）.

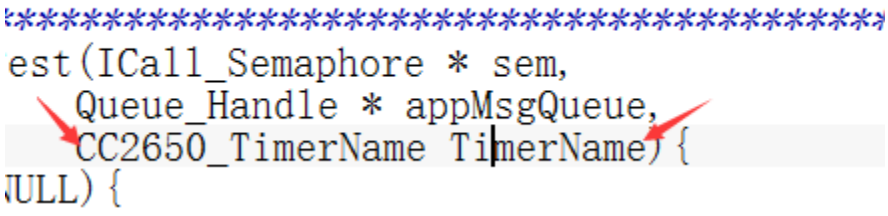
哇，这么多，搞定一个简单点的先：

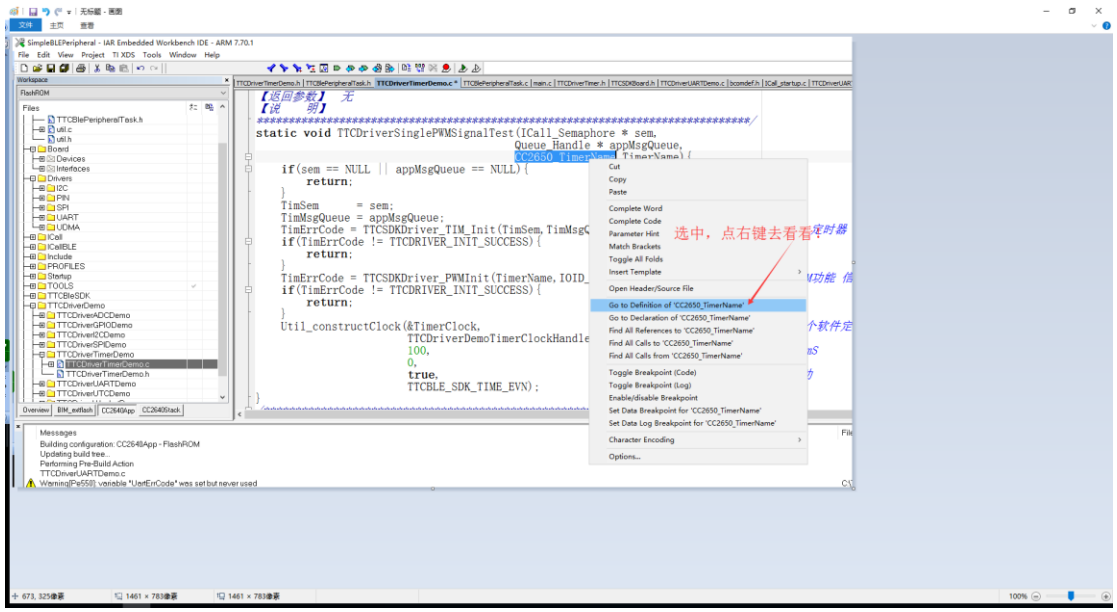
```
/*********************************************************************************/
void TTCDriverDemoTimerInit(ICall_Semaphore * sem,
                            Queue_Handle * appMsgQueue,
                            CC2650_TimerName TimerName){

    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
}
```

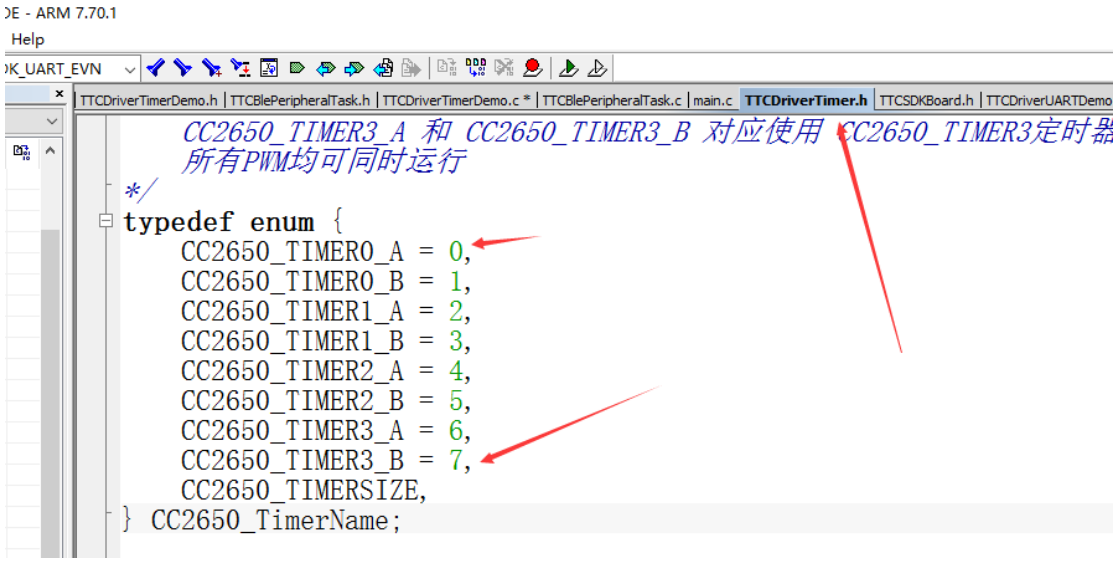
看看 PWM输出功能：

```
/*********************************************************************************/
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,
                        100,                                                    //定时100mS
                        0,
                        true,                                                   //立即启动
                        TTCBLE_SDK_TIME_EVN);
}
```

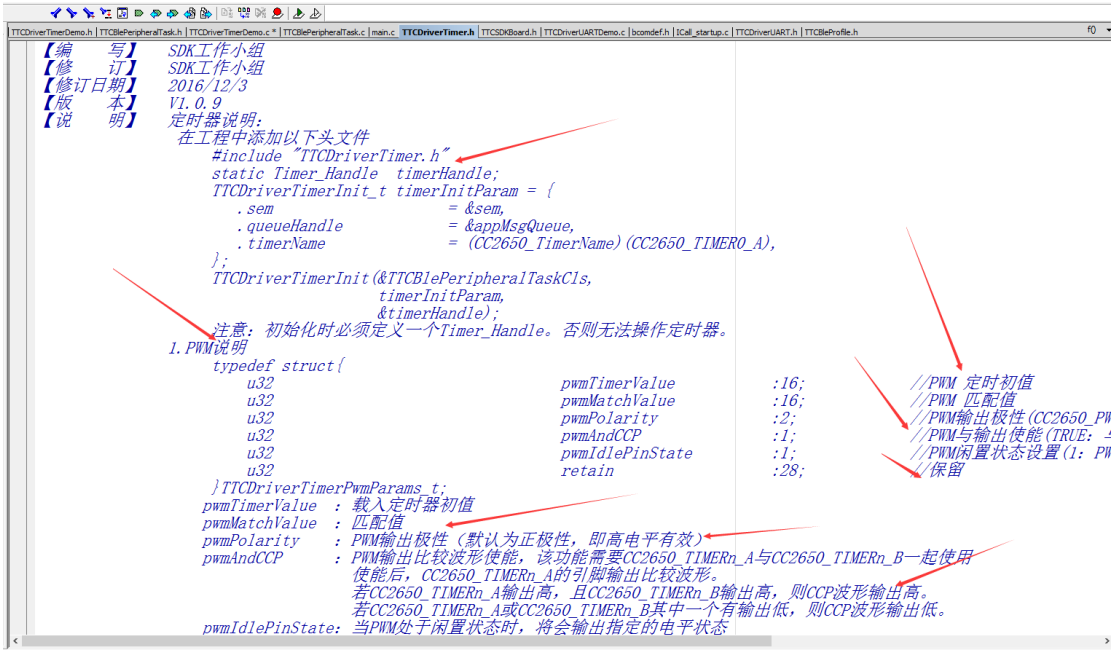
设定一个定时器在 IOID_0 输出 PWM。看看 SDK 有多少个定时器：



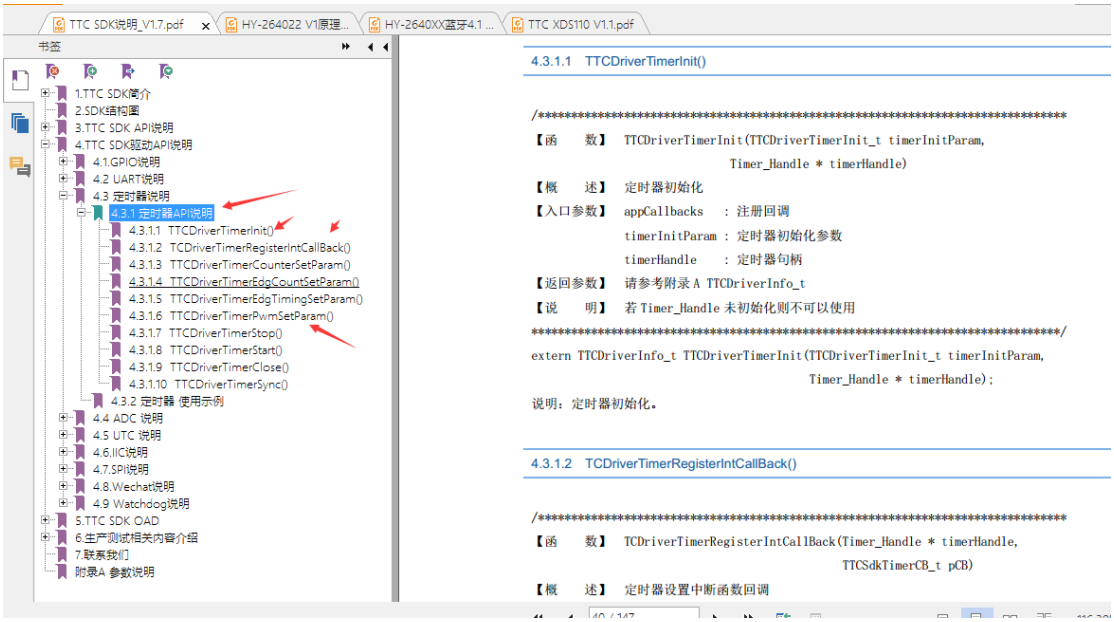




原来ＳＤＫ把４路３２位的定时分为了８路１６位的；想了解更多关于定时器的设置可以仔细看看头文件中的说明：



还有ＳＤＫ说明文档:



说得很仔细，发点时间过一下，定时器的特殊用法都有！好啦，这里不费话，来实际的：

原有 TTCDriverSinglePWMSignalTest 是指定在 IOID_0 输出 PWM,与之前的试验有点冲突，

我们改一下：

```
 *  【说    明】  无
 ******************************************************************************/
void TTCDriverDemoTimerInit(ICall_Semaphore * sem,
                            Queue_Handle * appMsgQueue,
                            CC2650_TimerName TimerName){
    TTCDriverSinglePWMSignalOutSet(sem,appMsgQueue,TimerName,IOID_7);

//    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
}
#endif
```

把ＩＯ由外面的指定，可以指定不同的ＩＯ来测试ＰＷＭ的输出：

```
 ******************************************************************************/
static void TTCDriverSinglePWMSignalOutSet(ICall_Semaphore * sem,
                                           Queue_Handle * appMsgQueue,
                                           CC2650_TimerName TimerName,u8 IO_Pin){
    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,IO_Pin);                       //配置PWM功能 信号从I
    if(TimErrCode != TTCDRIVER_INIT_SUCCESS){
        return;
    }
    Util_constructClock(&TimerClock,                                           //配置一个软件定时器
                        TTCDriverDemoTimerClockHandler,
                        100,                                                   //定时100mS
                        0,
                        true,                                                  //立即启动
                        TTCBLE_SDK_TIME_EVN);
}
```

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】
【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信
//PWM的参数
//指定PWM信号的输出脚
//立即使能定时器
```

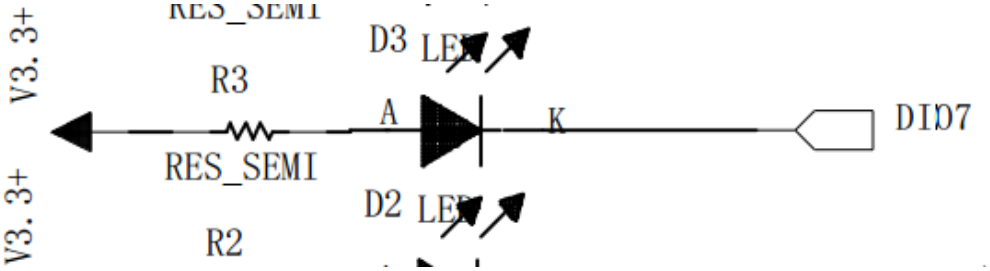
8个IO上来回变化，而且不断重复，这么多，复制一下：我们简单点：

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

    TTCDriverTimerPwmParams_t param = {
        48000,
        PWM_MatchValue, //24000,
        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IO固定一下，开发板的第三个LED还没有用起来，DIO7试试：



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

透传ＡＰＰ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],                   //定时器的句柄 即指定此PWM信号是
                                           param);                          //PWM的参数
```

是什么东东，我们确认一下：

```
/* PWM参数设置 */
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_PWMPolarity)
    u32  pwmAndCCP       : 1;           //PWM与输出使能(TRUE：与输出使能。 FALSE：上
    u32  pwmIdlePinState : 1;           //PWM闲置状态设置(1：PWM闲置状态输出高电平,
    u32  retain          : 28;          //保留
} TTCDriverTimerPwmParams_t;
```

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                         ((OA
#endif

unsigned int BLEpwmMatchValue = 0;

unsigned char        BLE_RxBuffer[20] = {0};
ul(BLE_RxBuffer)E_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ＧＰＩＯ０

          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);
        }

        //用户事外理代码结束;
    }
```

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 TTCDirverTimerDemo.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：

```
#ifdef TTCDRIVER_TIMER
/*****************************************************************************************
 * 常量及宏定义
 */
#define TTCBLE_SDK_TIME_EVN                    0x0008
```

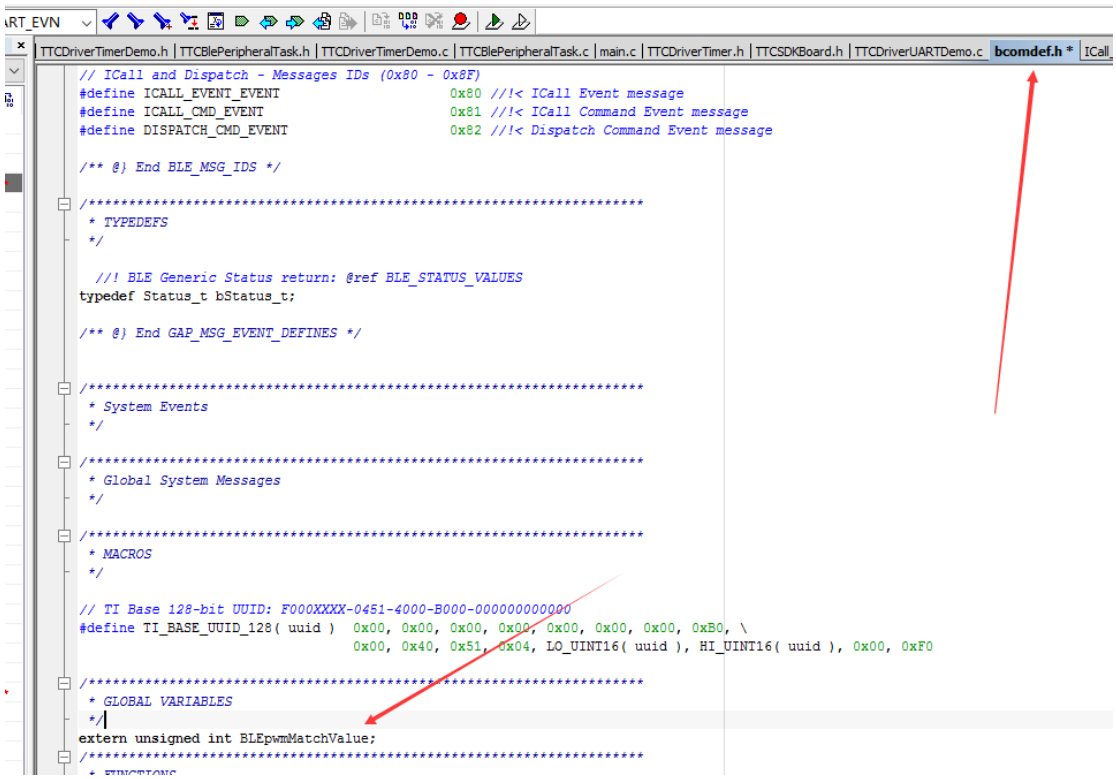
再编译看看,不行！TTCDirverTimerDemo.c 原来就定义是静态的啦，先屏掉：

```
static Queue_Handle * TimMsgQueue;
static u16 events;

/*****************************************************************************************
 * 本地函数声明
 */
static void TTCDriverDemoTimerClockHandler(UArg arg);
//static void TTCDriverDemoTimerSetEvent(UArg arg);
static TTCDriverInfo_t TTCSDKDriver_PWMInit(CC2650_TimerName TimerNam
static TTCDriverInfo_t TTCSDKDriver_Inter_Init(CC2650_TimerName Timer
static TTCDriverInfo_t TTCSDKDriver_Cat_Init(CC2650_TimerName TimerNa
static TTCDriverInfo_t TTCSDKDriver_Count_Init(CC2650_TimerName Timer
```

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 TTCDirverTimerDemo.c，把全局变量引进来：



还是不行？问题在哪呢？

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

原来ＳＤＫ有坑呀！这谁挖的呀！没有区分大小写呀！

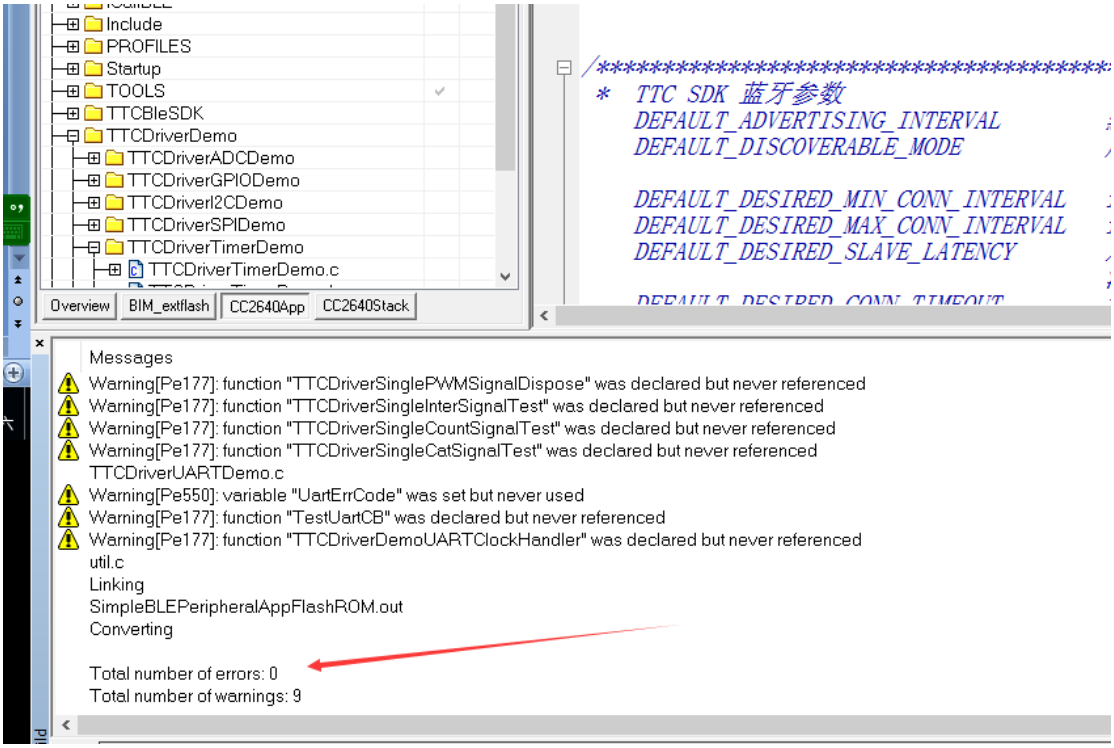
改过来！

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

还面还有好几个都是这样！都改改：

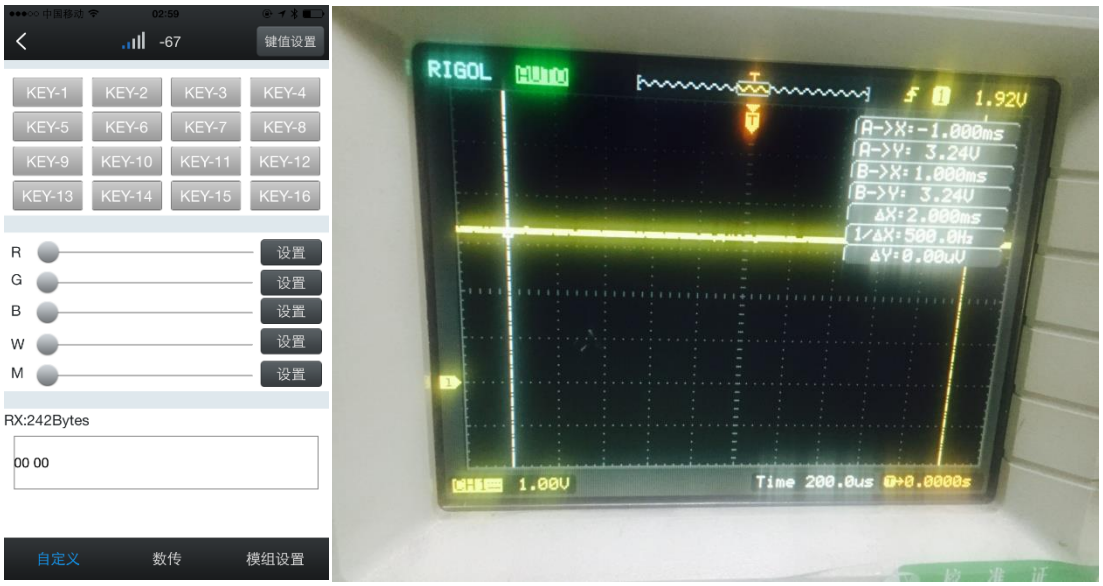
```
#ifdef TTCDRIVER_GPIO
#include "TTCDriverGPIODemo.h"
#endif

#ifdef TTCDRIVER_I2C
#include "TTCDriverI2CDemo.h"
#endif

#ifdef TTCDRIVER_SPI
#include "TTCDriverSPIDemo.h"
#endif

#ifdef TTCDRIVER_TIMER
#include "TTCDriverTimerDemo.h"
#endif

#ifdef TTCDRIVER_UART
#include "TTCDriverUARTDemo.h"
#endif

#ifdef TTCDRIVER_UTC
#include "TTCDriverUTCDemo.h"
#endif

#ifdef TTCBLE_WECHAT
#include "TTCDriverWechatDemo.h"
#endif

#ifdef TTCBLE_IBEACON
#include "TTCBeacon.h"
#endif
```

我再来！O啦！

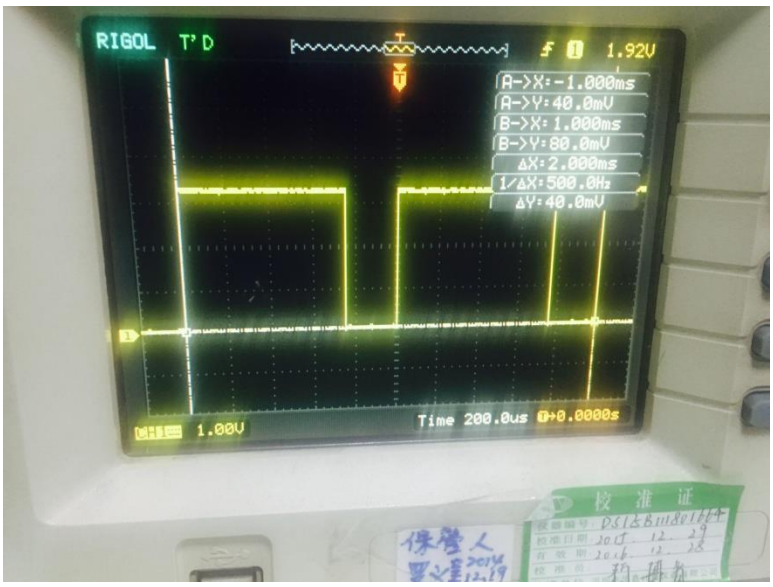


仿真，没问题，联上手机，试试滑条:

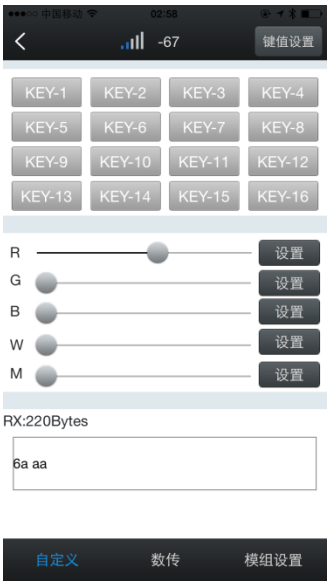
滑条输出 0x0000 看看,示波器测试 IO 一直为高：



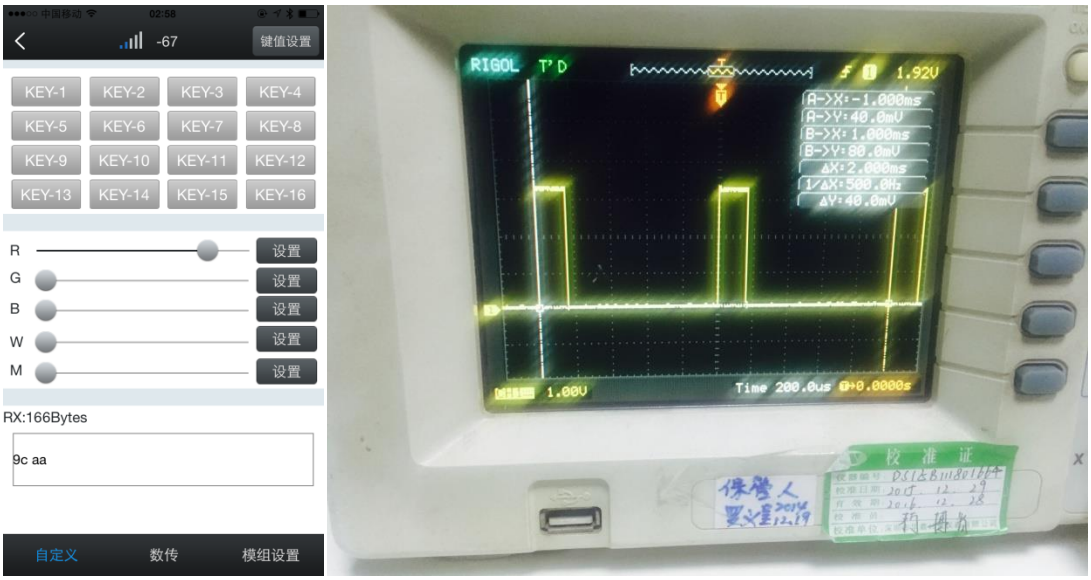
滑条输出 0x2dd5 看看,示波器测试 IO:



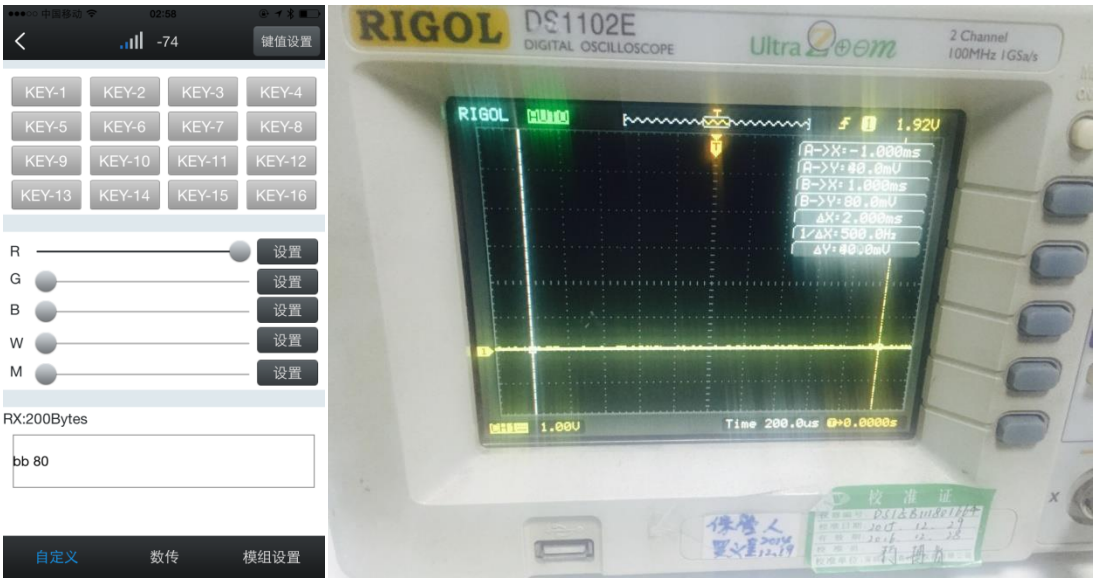
滑条输出 0x6aaa 看看,示波器测试 IO:



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４８０００的量程！