青风带你玩蓝牙 nRF52832 系列教程

----------作者：青风

出品论坛：　www.qfv8.com　青风电子社区

作者： 青风
出品论坛: www.qfv8.com
淘宝店：http://qfv5.taobao.com
QQ 技术群：346518370
硬件平台：青云 QY-nRF52832 开发板

2.21 蓝牙协议栈下硬件定时器的使用

上一讲，我们讲述了协议栈下软件定时器的使用，你可以设置多个软件定时器实现多任务的定时，满足绝大部分的设计需求。有人就会觉得硬件定时器 TIMER 无法再协议下使用，实际上不是这样的，协议栈下也是可以使用硬件定时器的。

在使用协议栈之后，RTC0，TIMER0 都被协议栈用去了，RTC1 被软件定时器使用了，因此相关用到了这些外设的代码也需要修改避开。

本例在匹配的SDK15.0的蓝牙串口样例基础上就行编写，使用的协议栈为：s132。

1：nRF51822 蓝牙 BLE 硬件定时器设置：

1.1 BLE 定时器声明

本例在 SDK15.0 下的串口蓝牙例子下进行修改，：

首先我们需要再主函数前定义使用哪个定时器，使用 NRF_DRV_TIMER_INSTANCE (id)中 ID 来定义，按照前面的原理介绍，这个 ID 值可以为 0,1,2 三个值，协议栈下只能使用 1，2。如果设置定时器 1 则如下代码：

```
const nrf_drv_timer_t TIMER_LED = NRF_DRV_TIMER_INSTANCE(1);//设置使用的定时器
```

其实定时器在协议栈下的使用，首先设置一个定时器，函数如下

```
//定时器初始化
void TimeInt(void)
{
    uint32_t time_ms = 1000; //设置 1s 发生一个定时器中断
    uint32_t time_ticks;
    uint32_t err_code = NRF_SUCCESS;
```

```
    //配置定时器参数
    nrf_drv_timer_config_t timer_cfg = NRF_DRV_TIMER_DEFAULT_CONFIG;
    err_code = nrf_drv_timer_init(&TIMER_LED, &timer_cfg, timer_led_event_handler);
    APP_ERROR_CHECK(err_code);
     //配置定时器，同时注册定时器的回调函数
    time_ticks = nrf_drv_timer_ms_to_ticks(&TIMER_LED, time_ms);
      //设置定时器捕获/比较 触发设备、通道、滴答时间
    nrf_drv_timer_extended_compare(&TIMER_LED,      NRF_TIMER_CC_CHANNEL0,
time_ticks, NRF_TIMER_SHORT_COMPARE0_CLEAR_MASK, true);
    //使能定时器
      nrf_drv_timer_enable(&TIMER_LED);

}
```

◎使用 nrf_drv_timer_init 函数。该函数用于定义定时器的相关配置，以及中断回调的函数：

```
#define NRFX_TIMER_DEFAULT_CONFIG
{
    .frequency= (nrf_timer_frequency_t)NRFX_TIMER_DEFAULT_CONFIG_FREQUENCY,\
    .mode = (nrf_timer_mode_t)NRFX_TIMER_DEFAULT_CONFIG_MODE,              \
    .bit_width= (nrf_timer_bit_width_t)NRFX_TIMER_DEFAULT_CONFIG_BIT_WIDTH,\
    .interrupt_priority=NRFX_TIMER_DEFAULT_CONFIG_IRQ_PRIORITY,
     .p_context= NULL                                                      \
}
```

如果结构体不设置，就默认为初始设置。如何需要设置，这对结构体赋值，比如在协议栈下给的配置需要注意修改定时器宽度为 32 位宽度**（关于定时器详细讲解请看外设篇定时器 TIMER）**，在配置文件 sdk_config.h 中修改如下：

```
5391 // <0=> 16 bit
5392 // <1=> 8 bit
5393 // <2=> 24 bit
5394 // <3=> 32 bit
5395
5396 #ifndef TIMER_DEFAULT_CONFIG_BIT_WIDTH
5397 #define TIMER_DEFAULT_CONFIG_BIT_WIDTH 3
5398 #endif
5399
```

1.2 定时器开始使能

定时器开始定时，需要在 sdk_config.h 文件中进行使能，具体需要修改的代码如下，首先是使能总的定时器：

```
5358 // <e> TIMER_ENABLED - nrf_drv_timer - TIMER periperal driver - legacy layer
5359 //==========================================================
5360 #ifndef TIMER_ENABLED
5361 #define TIMER_ENABLED 1
5362 #endif
```

然后，对需要使用的定时器模块进行使能，这里使用定时器 1：

```
5424 // <q> TIMER1_ENABLED  - Enable TIMER1 instance
5425
5426
5427 #ifndef TIMER1_ENABLED
5428 #define TIMER1_ENABLED 1
5429 #endif
5430
```

1.3 主函数及中断子函数

中断子函数，在定时器定时到了时间后，触发中断超时操作，代码如下：

```
void timer_led_event_handler(nrf_timer_event_t event_type, void* p_context)
{
    switch (event_type)
    {
        case NRF_TIMER_EVENT_COMPARE0:
                nrf_gpio_pin_toggle(LED_3);//翻转 LED
                printf("\r\OK 1s\r\n");//串口输出指示
            break;

        default:
            //Do nothing.
            break;
    }
}
```

主函数写一个测试函数，主要是加入定时器初始化函数，编写代码如下：

```
int main(void)
{int main(void)
{
    bool erase_bonds;

    // Initialize.
    uart_init();
    log_init();
    timers_init();
     TimeInt();//加入定时器初始化函数
    buttons_leds_init(&erase_bonds);
    power_management_init();
    ble_stack_init();
    gap_params_init();
    gatt_init();
    services_init();
```

```
    advertising_init();
    conn_params_init();

    // Start execution.
    printf("\r\nUART started.\r\n");
    NRF_LOG_INFO("Debug logging for UART over RTT started.");
    advertising_start();
      tx_power_set();

    // Enter main loop.
    for (;;)
    {
        idle_state_handle();
    }
}
}
```

修改后编译通过，提示 OK

2 应用与调试

2.1 下载

打开 NRFgo 进行下载，，首先整片擦除，后下载协议栈，下载完后可以下载工程，首先把工程编译一下，通过后点击 KEIL 上的下载按键。下载成功后，程序开始运行，同时开发板上广播 LED 开始广播。

2.2 测试

本实验采用手机写 nrf connect app 软件，返现服务应用名 QFDZ_time,如下图所示：

同时，打开串口助手，串口助手 1s 后输出如下，LED3 也按照 1s 时间闪：