



GR5xx FreeRTOS示例手册

版本： 3.1

发布日期： 2023-11-06

版权所有 © 2023 深圳市汇顶科技股份有限公司。保留一切权利。

非经本公司书面许可，任何单位和个人不得对本手册内的任何部分擅自摘抄、复制、修改、翻译、传播，或将其全部或部分用于商业用途。

商标声明

GOODIX 和其他汇顶商标均为深圳市汇顶科技股份有限公司的商标。本文档提及的其他所有商标或注册商标，由各自的所有人持有。

免责声明

本文档中所述的器件应用信息及其他类似内容仅为您提供便利，它们可能由更新之信息所替代。确保应用符合技术规范，是您自身应负的责任。

深圳市汇顶科技股份有限公司（以下简称“GOODIX”）对这些信息不作任何明示或暗示、书面或口头、法定或其他形式的声明或担保，包括但不限于针对其使用情况、质量、性能、适销性或特定用途的适用性的声明或担保。GOODIX对因这些信息及使用这些信息而引起的后果不承担任何责任。

未经GOODIX书面批准，不得将GOODIX的产品用作生命维持系统中的关键组件。在GOODIX知识产权保护下，不得暗或以其他方式转让任何许可证。

深圳市汇顶科技股份有限公司

总部地址：深圳市福田区腾飞工业大厦B座12-13层

电话：+86-755-33338828 邮编：518000

网址：www.goodix.com

前言

编写目的

本文档介绍如何使用和修改GR5xx SDK中的FreeRTOS示例，旨在帮助用户快速进行二次开发。

读者对象

本文适用于以下读者：

- 芯片用户
- 开发人员
- 测试人员
- 开发爱好者
- 文档工程师

版本说明

本文档为第3次发布，对应的产品为低功耗蓝牙GR5xx系列。

修订记录

版本	日期	修订内容
1.0	2023-01-10	首次发布
3.0	2023-03-30	新增支持多款芯片的相关描述
3.1	2023-11-06	更新GProgrammer、GRUart、GRToolbox获取方式。

目录

前言..... I

1 简介..... 1

2 FreeRTOS源码目录介绍..... 2

3 初次运行..... 3

 3.1 准备工作..... 3

 3.2 固件烧录..... 3

 3.3 测试验证..... 3

4 应用详解..... 6

 4.1 配置介绍..... 6

 4.1.1 配置内存管理策略..... 6

 4.1.2 配置内核..... 7

 4.2 关键代码..... 7

 4.2.1 任务创建..... 8

 4.2.2 Bluetooth LE调度详解..... 8

5 常见问题..... 12

 5.1 串口无打印..... 12

 5.2 蓝牙无广播..... 12

1 简介

FreeRTOS是一个开源的（MIT License）、轻量级的嵌入式实时操作系统，占用较少的RAM/ROM资源，具有可移植、可裁减、调度策略灵活的特点。该系统包含任务管理、时间管理、信号量、消息队列、内存管理等功能。

本文档介绍GRxx SDK中的FreeRTOS移植示例，包括示例的使用方法以及关键源码的说明。

在进行操作前，可参考以下文档。

表 1-1 文档参考

名称	描述
对应芯片开发者指南	介绍GRxx SDK以及基于SDK的应用开发和调试
Keil用户指南	Keil详细操作说明： www.keil.com/support/man/docs/uv4/
FreeRTOS Documentation	FreeRTOS的使用方法： www.freertos.org/Documentation/RTOS_book.html

2 FreeRTOS源码目录介绍

FreeRTOS源码位于目录SDK_Folder\external\freertos，包括include文件夹、portable文件夹和.c源文件。

 说明:

SDK_Folder为对应芯片SDK的根目录。

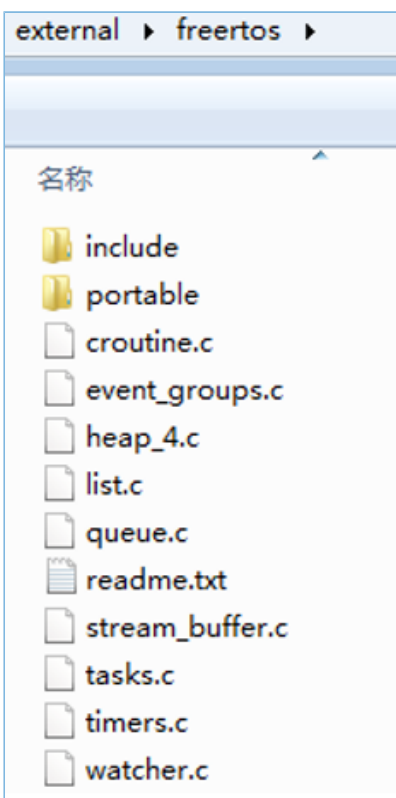


图 2-1 GR5xx SDK中的freertos文件夹

- include目录：FreeRTOS全部API，相关结构体和宏定义。
- portable目录：GR5xx平台移植代码。
- .c源文件：FreeRTOS的核心业务代码实现。

如需了解FreeRTOS的详细介绍，请访问FreeRTOS官网www.freertos.org。

 说明:

armcc/gcc/iar下移植FreeRTOS使用的文件是不同的。根据编译要求选择对应的文件，具体文件在portable目录中。

3 初次运行

本章介绍如何快速验证GR5xx SDK中的FreeRTOS示例。

3.1 准备工作

验证FreeRTOS示例之前，请完成以下准备工作。

- 硬件准备

表 3-1 硬件准备

名称	描述
开发板	对应芯片Starter Kit开发板（以下简称“开发板”）
连接线	USB Type-C（GR551x系列使用Micro USB 2.0连接线）
Android手机	Android 5.0（KitKat）及以上版本
iOS设备	支持Bluetooth LE 4.0及以上的iOS设备，如iPhone 4s及以上版本、iPad 3及以上版本

- 软件准备

表 3-2 软件准备

名称	描述
Windows	Windows 7/Windows 10操作系统
Keil MDK5	IDE工具，支持MDK-ARM 5.20 及以上版本，下载网址： www.keil.com/download/product/
GRToolbox（iOS）	Bluetooth LE调试工具，可通过App Store下载
GRToolbox（Android）	Bluetooth LE调试工具，下载网址： www.goodix.com/zh/software_tool/grtoolbox
GRUart（Windows）	串口调试工具，下载网址： www.goodix.com/zh/download?objectId=64&objectType=software
GProgrammer（Windows）	Programming工具，下载网址： www.goodix.com/zh/software_tool/gprogrammer_ble

3.2 固件烧录

用户可使用GProgrammer将`ble_app_template_freertos.bin`固件烧录至开发板。GProgrammer烧录固件的具体操作方法，请参考《GProgrammer用户手册》。

说明:

`ble_app_template_freertos.bin`位于`SDK_Folder\projects\ble\ble_peripheral\ble_app_template_freertos\build\`。

3.3 测试验证

当准备好开发板和测试所需软件后即可进行FreeRTOS示例的测试验证，本示例的测试内容包括以下两方面：

- FreeRTOS功能

- 蓝牙功能

1. 验证FreeRTOS功能

启动GRUart，打开配置的串口，查看运行结果。在GRUart的Receive Data窗口中，如果每隔一秒打印一行“APP_I: TickCount: tickcount数, Time: 月/日 时:分:秒.毫秒”，则表示FreeRTOS系统运行正常。

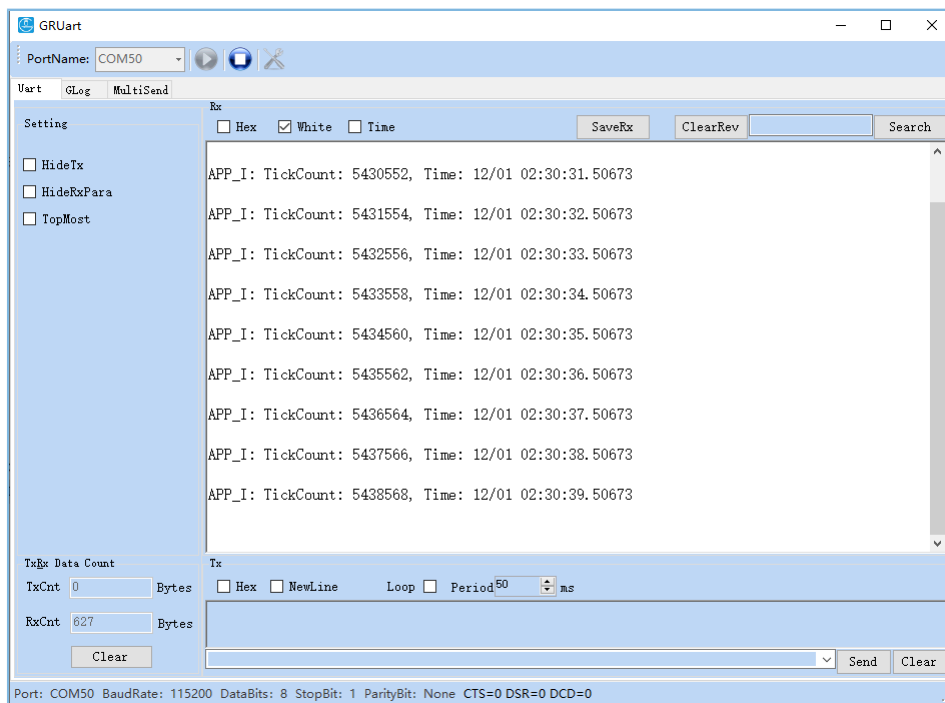


图 3-1 运行结果

2. 验证蓝牙功能

启动GRToolbox，扫描周边蓝牙设备。如果在设备列表中发现名为“Goodix_Tem_OS”的广播设备，则表示FreeRTOS应用运行正常。

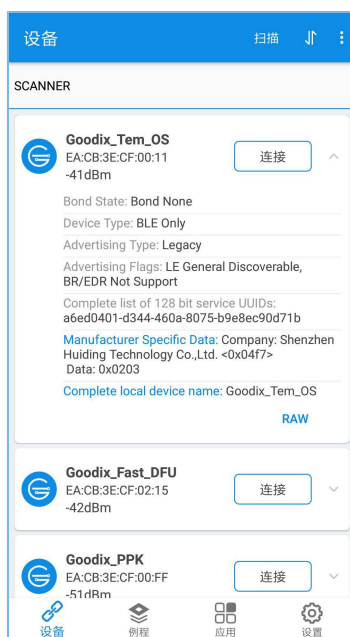


图 3-2 扫描到蓝牙设备Goodix_Tem_OS

说明:

本文中GRToolbox的截图仅供用户了解操作步骤，实际界面请参考最新版本GRToolbox。

4 应用详解

通过修改ble_app_template_freertos示例的相关配置，用户可自定义FreeRTOS应用，如：

- 修改FreeRTOS配置
- 修改示例程序配置

本章将介绍ble_app_template_freertos示例的配置和关键代码。

4.1 配置介绍

用户可根据产品需求自定义FreeRTOS的内存管理策略和FreeRTOS内核。

4.1.1 配置内存管理策略

本工程的内存管理策略采用heap_4.c。用户可将heap_4.c替换为自己所需的内存管理策略源文件。

FreeRTOS支持5个内存管理策略，分别由heap_1.c，heap_2.c，heap_3.c，heap_4.c和heap_5.c来实现。各内存管理策略源文件的描述如下：

表 4-1 FreeRTOS内存管理策略

源文件	特点
heap_1.c	<ul style="list-style-type: none">• 实现相对简单，代码量比较小。• 只能申请内存，申请成功后就不能被释放。
Heap_2.c	<ul style="list-style-type: none">• 使用最佳匹配算法。• 允许释放已分配的内存块。• 不合并相邻空闲块，可能造成内存碎片。• 多次申请和释放内存资源会造成内存碎片。
Heap_3.c	<ul style="list-style-type: none">• 封装malloc()和free()函数，使得他们具备线程保护。• 需在启动汇编文件startup_gr55xx.s中配置heap大小。• Keil集成环境“Options for Target”窗口的“Target”选项，需勾选“Use MicroLIB”，否则无法正常使用。
Heap_4.c	<ul style="list-style-type: none">• 使用最佳匹配算法。• 允许释放已分配的内存块。• 合并相邻的空闲内存块。• 多次申请和释放内存资源会造成内存碎片。
Heap_5.c	<ul style="list-style-type: none">• 使用最佳匹配算法。• 允许释放已分配的内存块。• 合并相邻的空闲内存块。• 允许内存堆跨越多个非连续的内存区。• 需要依次分别初始化内存堆。

 说明:

源文件位于SDK_Folder\external\freertos\portable\MemMang。

4.1.2 配置内核

FreeRTOS内核由*FreeRTOSConfig.h*头文件中的宏定义进行配置，包括配置主时钟频率，最大任务优先级等。用户可修改这些宏定义来制定新的内核。FreeRTOS的常见宏定义如表 4-2 所示：

表 4-2 FreeRTOS常见宏定义

宏定义	配置
configUSE_IDLE_HOOK	1：使能IDLE任务的HOOK函数。 0：禁用IDLE任务的HOOK函数。
configUSE_TICK_HOOK	1：使能TICK中断执行HOOK函数。 0：禁用TICK中断执行HOOK函数。
configCPU_CLOCK_HZ	定义CPU的主频，单位Hz，当前平台默认为64000000。
configTICK_RATE_HZ	定义系统时钟节拍数，单位Hz，当前平台默认为1000。
configMAX_PRIORITIES	定义可供用户使用的最大优先级数。 如果定义为5，那么用户可以使用的优先级号是0、1、2、3、4，不包含5。
configMINIMAL_STACK_SIZE	定义系统任务默认最小使用的堆栈大小，单位word，当前平台默认为512 words（共2048 bytes）。
configTOTAL_HEAP_SIZE	该宏表示内存管理所使用的内存池容量大小，单位KB，当前平台默认为32 KB。 如果使用动态API，所有FreeRTOS内核中将会从该内存池中申请内存。需根据实际情况分配总量，避免引起系统运行时异常。
configPRIO_BITS	表示当前平台优先级设定占用比特位数，当前平台默认为4。
configLIBRARY_LOWEST_INTERRUPT_PRIORITY	表示当前平台支持的最小优先级，当前平台默认为15。
configLIBRARY_MAX_SYSCALL_INTERRUPT_PRIORITY	定义FreeRTOS管理的最高优先级中断。数字越小，优先级越高。 如果设置为5，那么小于5的优先级是不受FreeRTOS管控。在屏蔽中断时，小于5的优先级不被屏蔽。

说明:

- *FreeRTOSConfig.h*位于SDK_Folder\app\projects\ble\ble_peripheral\ble_app_template_freertos\Src\user。
- 更多宏配置，请参考<https://www.freertos.org/a00110.html>。

4.2 关键代码

本节主要介绍如何使用代码实现任务的创建以及Bluetooth LE调度。

4.2.1 任务创建

- 创建任务

在本示例工程中创建的任务是`print_test_task`，该任务主要负责打印信息。

路径：`ble_app_template_freertos\Src\user\main.c`

函数：`vStartTasks()`;

```
static void vStartTasks(void *arg)
{
    app_calendar_init();
    xTaskCreate(print_test_task, "print_task", APP_TASK_STACK_SIZE,
                NULL, configMAX_PRIORITIES - 1, NULL);
    xTaskCreate(dfus_schedule_task, "dfus_schedule_task", DFU_TASK_STACK_SIZE,
                NULL, configMAX_PRIORITIES - 2, NULL);
    #if APP_LOG_STORE_ENABLE
        xTaskCreate(log_store_dump_task, "log_store_dump_task",
                    LOG_STORE_DUMP_TASK_STACK_SIZE, NULL, configMAX_PRIORITIES - 3, NULL);
    #endif
    vTaskDelete(NULL);
}
```

路径：`ble_app_template_freertos\Src\user\main.c`

函数：`print_test_task()`;

该函数实现以1秒为延时的循环打印。其中`vTaskDelay`函数的单位为毫秒。

```
static void print_test_task(void *arg)
{
    while (1)
    {
        app_rtc_get_time(&g_calendar_time);
        APP_LOG_INFO("TickCount: %d, Time: %02d/%02d %02d:%02d:%03d\r\n",
                     xTaskGetTickCount(),
                     g_calendar_time.mon, g_calendar_time.date,
                     g_calendar_time.hour, g_calendar_time.min, g_calendar_time.sec,
                     g_calendar_time.ms);
        app_log_flush();
        vTaskDelay(1000);
    }
}
```

4.2.2 Bluetooth LE调度详解

本节介绍在FreeRTOS下，Bluetooth LE协议栈与Bluetooth LE应用如何进行调度。

进入`main()`函数后，在执行FreeRTOS的任务调度之前，需要完成：

1. 实现各种硬件外设的初始化。

2. 声明运行Bluetooth LE协议栈需使用的内存（heaps_table）。
3. 完成Bluetooth LE协议栈的初始化。

Bluetooth LE协议栈初始化后会启用BLE_IRQ以及BLE_SDK_IRQ两个中断。

- 将Bluetooth LE协议栈的Bluetooth LE Event通知给Bluetooth LE应用。

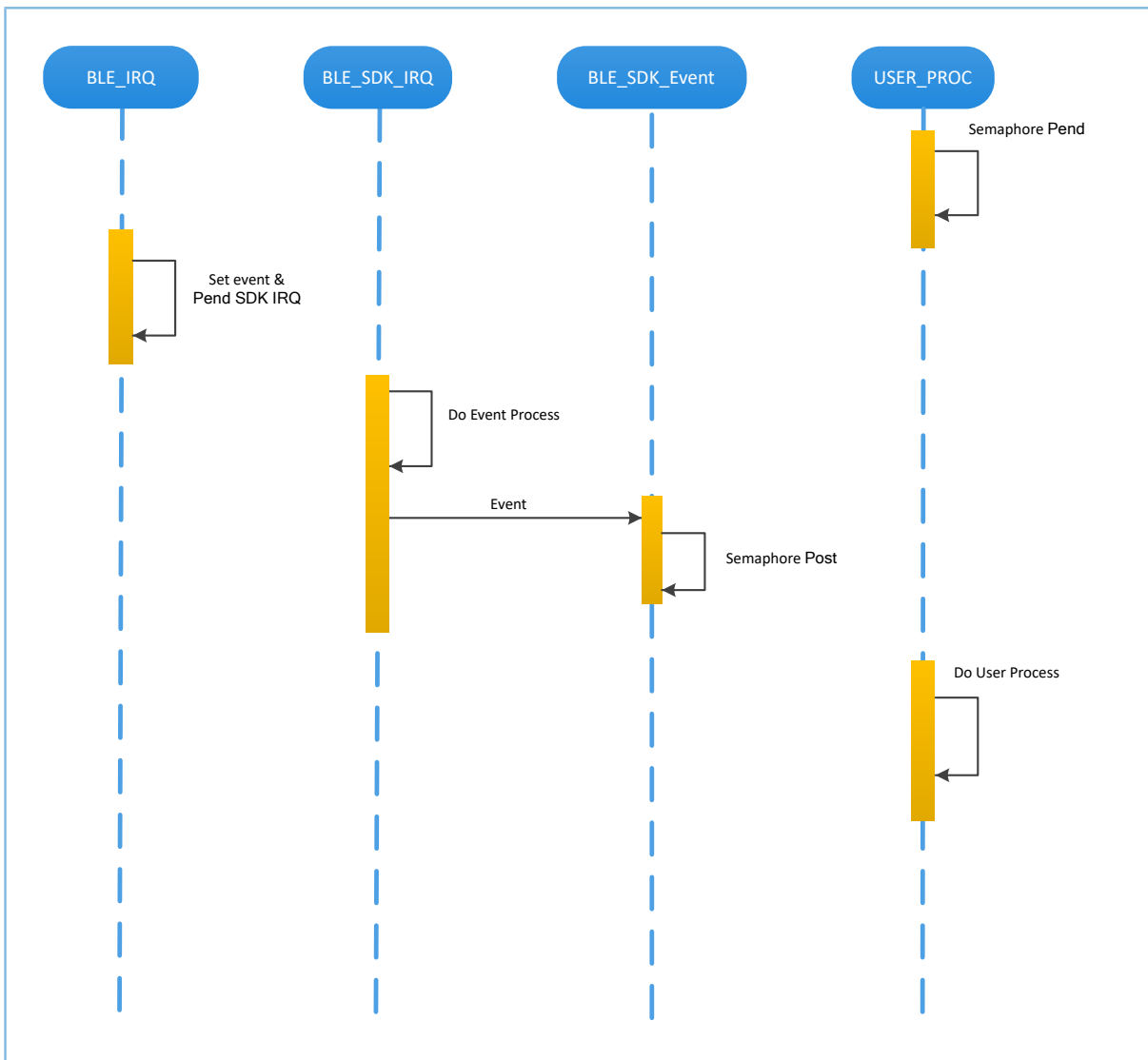


图 4-1 Bluetooth LE协议栈将Bluetooth LE Event通知给Bluetooth LE应用

在图 4-1中，当Bluetooth LE Baseband收到数据包后，会触发BLE_IRQ中断，BLE_IRQ_Handler产生Bluetooth LE Event并将BLE_SDK_IRQ中断置为Pending态；在BLE_SDK_IRQ_Handler执行期间，Bluetooth LE Event会被处理，并通过BLE_SDK_Event函数将部分Bluetooth LE Event通知到Bluetooth LE应用。

关于BLE_SDK_Event函数实现的建议：

1. Event函数是在中断处理函数（BLE_SDK_IRQ_Handler）中被调用的，因此建议不要在Event函数进行耗时操作，否则可能会延迟用户任务的执行。

- 如果Event函数中有需要Bluetooth LE应用及时处理的数据或状态信息，建议使用信号量机制在用户任务中完成业务逻辑的处理，即在用户任务等待信号量（Pend），在Event函数中释放（Post）信号量。
 - 如果Event函数中的数据较多、处理较耗时、需有序处理，建议开发者采用消息队列将数据缓存后转由其他任务处理。
 - 在BLE_SDK_Event函数中，如果需要使用系统API，则请调用FreeRTOS中FromISR结尾的API，且禁止在BLE_SDK_Event函数中执行等待信号量的操作。
- 将Bluetooth LE应用层的请求发给Bluetooth LE协议栈。

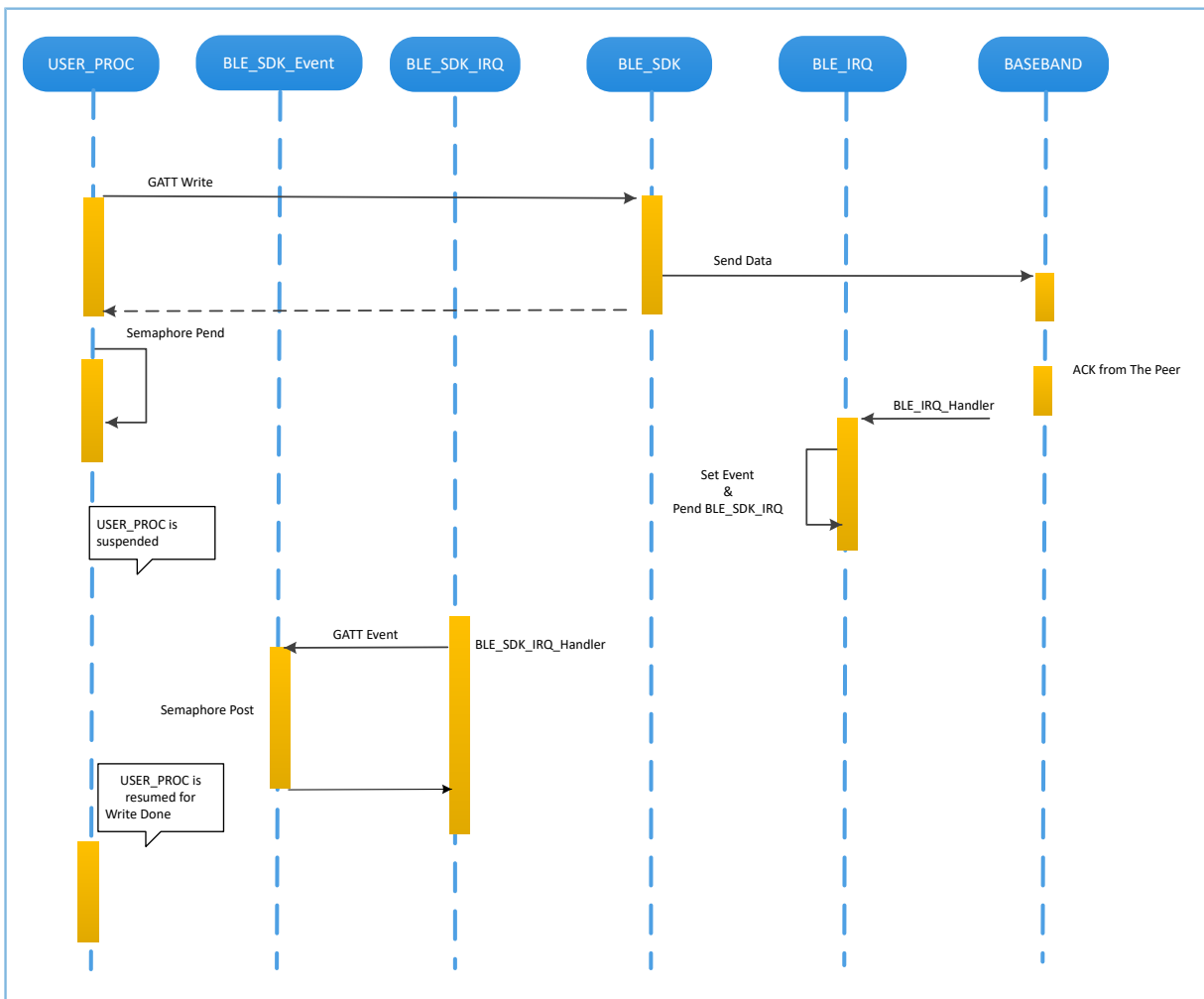


图 4-2 处理Bluetooth LE应用向Bluetooth LE协议栈发起的请求

在图 4-2 中，Bluetooth LE应用使用GATT API完成WRITE数据到对端设备。该操作涉及到与对端设备交互，且并不能立刻获得操作结果。Bluetooth LE应用需要等待Bluetooth LE协议栈的处理结果。开发者可根据Bluetooth LE应用的业务逻辑需求，使用信号量将异步函数调用转换为同步函数调用：

- 用户任务调用GATT API后，使用信号量（Semaphore Pend）接口将该任务挂起。
- Bluetooth LE协议栈将来自Bluetooth LE应用的数据发送完成后，等待对端的Ack。

3. Bluetooth LE Baseband收到来自对端的Ack后，会触发BLE_IRQ中断。
4. BLE_IRQ_Handler产生Bluetooth LE Event并将BLE_SDK_IRQ中断置为Pending态。
5. 在BLE_SDK_IRQ_Handler执行期间，该Bluetooth LE Event会被处理，并调用对应的BLE_SDK_Event函数。
6. 在BLE_SDK_Event函数中执行（Post）信号量操作来释放阻塞的信号量。

此时用户任务恢复运行，获得WRITE数据的操作结果。

一般情况下，开发者只需要关心应用层功能以及与用户进行交互的Event函数接口的业务逻辑实现，Bluetooth LE协议栈对开发者而言是透明的。GR5xx SDK编程模型，请参考对应芯片开发者指南。

5 常见问题

本章描述了在验证及应用FreeRTOS示例时，可能出现的问题、原因及处理方法。

5.1 串口无打印

- 问题描述

程序运行后，串口终端无任何打印信息。

- 问题分析

串口设置存在问题，如波特率不正确，则串口工具将不能正确显示收到的数据。

- 处理方法

请检查串口线连接是否正常，COM端口号是否选择正确，波特率等是否正确配置，建议先使用SDK默认固件进行开发环境检测。

5.2 蓝牙无广播

- 问题描述

程序运行后，手机无法搜索到广播。

- 问题分析

因固件没有正常运行，必定导致蓝牙没有广播。

- 处理方法

可以尝试复位或重新下载默认固件，同时检查天线端情况。