理解并使用汇顶科技 GR551x 系列 BLE SoC 的低功耗技术

版本修订记录

版本	修订说明	日期
1.0	首次发布	2022/11/1

目录

表目录

图目录

1. 简介

低功耗的使用对于低功耗蓝牙产品而言，至关重要，而要达成低功耗也并非易事。GR551x 是基于 Bluetooth 5.1 的低功耗蓝牙系列 SoC，对应的 SDK 已经实现了一些主要的低功耗管理机制，但开发者在使用的具体过程中，也会遇到各种各样的问题，例如：

- 如何理解低功耗？
- 如何应用低功耗模式至 GR551x 的产品应用中？
- 如何调试？

为了解决以上这些疑问，才有了这篇文章的产生。

2. 低功耗的本质是什么？

低功耗本质上是耗电量低，电量的计算公式如下：

$$Q = U * I * T$$

Q 为电量，U 为供电电压，I 为供电电流，T 为供电时长；

我们能够看到，降低应用产品的功耗，可以从降低电压、电流和减少供电时间三个方面入手，对于 SoC 而言，低功耗调试是一个系统级的问题，涉及软件、硬件的联合调试。

3. 理解 PMU

为了管理芯片的功耗，GR551x 设计了一个专门的电源管理单元（PMU）。PMU 支持对主要电源模块的电源参数进行调节及控制开关，能满足特定场景下的工况需求，最大限度节约功耗。

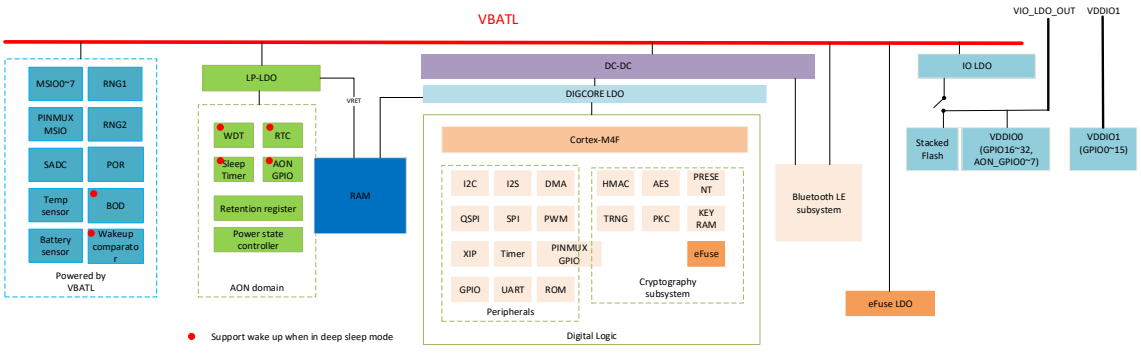

Figure 1 GR551x 电源系统框图

由上图可见，GR551x 为不同的模块提供独立可控的电源。为达到最佳的低功耗表现，GR551x SDK 可为 GR551x 系列提供最优的电源管理策略，开发者在具体产品应用中，仅需直接使用 SDK 提供的低功耗模式，即可快速实现产品的开发上市。

为了方便开发者理解，以下表格介绍了电源与主要模块的关系：

Table 1 电源与主要模块的关系

电源	供电模块	说明
VBATL	RNG1、RNG2、BOD、POR、Wake up comparator、MSIO0~7、MSIO PINMUX、SADC、Temperature sensor、Battery sensor 等	一直供电
LP-LDO	WDT、RTC、Sleep timer、AON_GPIO、Retention register、Power state controller、RAM 等	一直供电
DC-DC	RAM、Bluetooth LE subsystem、Cortex-M4F、Peripherals、Cryptography subsystem 等	在 Deep sleep mode 可以关闭，对于 RAM，在 Deep sleep mode，由于 LP-LDO 仍然可以提供支持 RAM retention 的电压，所以 RAM 的数据不会丢失。
eFuse LDO	eFuse	仅在 eFuse 量产测试，烧录 eFuse 时使用。
IO LDO	Stacked Flash、VDDIO0、VDDIO1	IO LDO 也可以为外部器件供电，其中电源设计，VDDIO0 及 VDDIO1 也可通过外部供电。

4. 理解低功耗模式

4.1 模式定义

GR551x 支持 Active、Idle、Deep sleep、Ultra deep sleep 等功耗管理模式，开发者可根据产品的使用场景进行选择和配置。

Active 模式主要用于需要产品全速处理任务的场景，最大限度发挥 SoC 的处理性能，例如智能手表需要高速刷新屏幕，在用户滑动界面等过程中，能够保证屏幕的流畅响应。

针对短时间内无任务处理的场景，可设置进入 Idle 模式，这种情况下，如果系统进入 Deep sleep 模式，Deep sleep 模式进入及退出过程，消耗电量会比 Idle 更多，例如当

主要任务处理完毕，但系统突然检测到后面会有任务，短时间段后需要一个广播，此时可临时进入 Idle 模式。

在 Deep sleep 模式下，只有 AON（Always ON）、VBATL 直供的模块有供电，其它电源均会关闭，而 RAM 数据则能保持，因此可以快速唤醒恢复至 Active 模式及时处理任务。例如在传感器应用中，需要周期性更新传感器数据，除更新数据时段外，系统都可以处于 Deep sleep 模式，从而进一步提升续航。

Ultra deep sleep 模式能够最大限度地保持低功耗，RAM 掉电，数据亦无法保持，只保留唤醒功能。唤醒过程相当于掉电重启系统，因此在唤醒后，系统将重新开始初始化流程。该模式适用于船运模式或类似场景，比如需要远洋运输的产品，能够确保用户首次启用产品时，依然可以开机使用。

具体模式说明如下：

Table 2 系统模式介绍

模式	时钟源	描述
Active	32MHz, 32KHz/32.768KHz	代码执行状态，所有的电源域及时钟都是开启的，可以快速处理任务，但耗电较高 。
Idle	32MHz, 32KHz/32.768KHz	开启所有的电源域及时钟，Cortex-M4F 时钟暂停，进入 WFI（Wait For Interrupt）状态，等待中断恢复代码运行；其它闲置状态的蓝牙子系统及外设，相关时钟关闭。
Deep sleep（也称为“Sleep”）	32KHz/32.768KHz	只有 AON（Always ON）、VBATL 直供的模块供电，其它电源关闭，高速时钟关闭（除了 RNG1、RNG2、RTC 时钟），支持 RAM retention；系统支持唤醒，唤醒后，系统会执行 warm boot 流程，在 warm boot 流程中进行睡眠前现场恢复，从睡眠前的最后运行指令位置继续运行。除了 AON 区域的外设，其它外设睡眠唤醒后需要软件进行恢复。
Ultra deep sleep	32KHz/32.768KHz	只有 AON、VBATL 直供的模块有供电，其它所有模块关闭，RAM 掉电；系统支持唤醒，唤醒后，系统执行 cold boot 流程。

4.2 外设睡眠介绍

4.2.1 外设睡眠唤醒流程

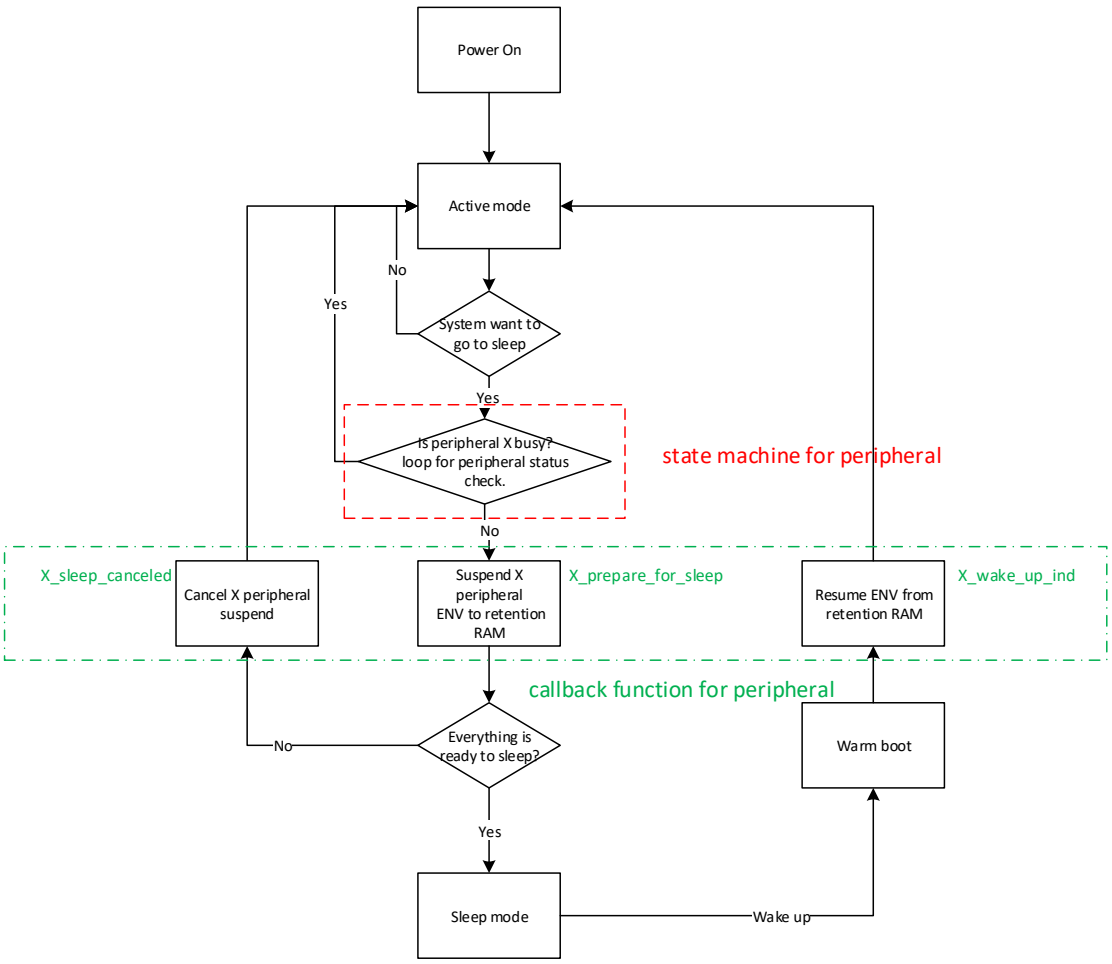

Figure 2 外设睡眠唤醒流程

4.2.2 外设状态机(state machine)

在进入 sleep 模式后，主要外设的电源会断掉，为了让相关外设能够正常的恢复运行，GR551x 的 SDK，有做一套针对主要外设的状态机，该状态机主要用于记录外设运行时的状态，如果运行过程中，正在进行数据通信，使用外设，那么此时是无法进入睡眠的，如果强行进入睡眠就会打乱正常的功能运行，等待外设空闲时，外设可以进入睡眠状态，为了能够快速的让外设睡眠，并唤醒之后能够正常工作，外设在睡眠前需要保存相关的寄存器数据到 retention RAM，在被唤醒后，再从 retention RAM 中恢复外设相关的寄存器。

这里以 UART 为例：

文件：GR551x_SDK_V1.7.0\drivers\inc\gr55xx_hal_uart.h

```
/**
 * @brief HAL UART State enumerations definition
```

```
 * @note  HAL UART State value is a combination of 2 different substates: gState and RxState.
 */
typedef enum
{
  HAL_UART_STATE_RESET     = 0x00U,  /**< Peripheral is not initialized.
                                      Value is allowed for gState and RxState */
  HAL_UART_STATE_READY     = 0x10U,  /**< Peripheral initialized and ready for use.
                                      Value is allowed for gState and RxState */
  HAL_UART_STATE_BUSY      = 0x14U,  /**< An internal process is ongoing.
                                      Value is allowed for gState only */

...
} hal_uart_state_t;
```

文件：GR551x_SDK_V1.7.0\components\app_drivers\src\app_uart.c

```
static bool uart_prepare_for_sleep(void)
{
  hal_uart_state_t state;

  for (uint8_t i = 0; i < APP_UART_ID_MAX; i++)
  {
    if (s_uart_env[i].uart_state == APP_UART_ACTIVITY)
    {
      state = hal_uart_get_state(&s_uart_env[i].handle);
/*检查 UART 是否处于 busy 状态，如果处于 busy，无法进入睡眠，不需要做睡眠前准备*/
      if ((state != HAL_UART_STATE_RESET) && (state != HAL_UART_STATE_READY))
      {
/*检查结果处于 busy 状态，直接返回*/
        return false;
      }
/*检查结果不处于 busy，可以正常准备睡眠，suspend UART 寄存器*/
      GLOBAL_EXCEPTION_DISABLE();
      hal_uart_suspend_reg(&s_uart_env[i].handle);
      GLOBAL_EXCEPTION_ENABLE();
      #ifdef APP_DRIVER_WAKEUP_CALL_FUN
      s_uart_env[i].uart_state = APP_UART_SLEEP;
```

```
        #endif
    }
  }
  return true;
}
```

4.2.3 回调函数

为了实现睡眠前外设的 suspend 及 resume 操作，SDK 定义了一个 app_sleep_callbacks_t 函数数组，专门用于外设的 suspend 及 resume 的回调。

文件：GR551x_SDK_V1.7.0\components\app_drivers\src\app_pwr_mgmt.c

```
struct pwr_env_t
{
  app_sleep_callbacks_t *pwr_sleep_cb[APP_SLEEP_CB_MAX];
  wakeup_priority_t wakeup_priority[APP_SLEEP_CB_MAX];
  bool is_pwr_callback_reg;
};
```

文件：

GR551x_SDK_V1.7.0\components\app_drivers\inc\app_pwr_mgmt.h

```
typedef struct
{
  bool (*app_prepare_for_sleep)(void);   /**<Peripherals prepare sleep function . */
  void (*app_sleep_canceled)(void);      /**<Peripherals cancel sleep function . */
  void (*app_wake_up_ind)(void);         /**< Resume peripherals when used function . */
} app_sleep_callbacks_t;
```

4.2.4 外设睡眠支持情况

在 GR551x 的 SDK 中，针对 APP example，已经默认实现了 app_sleep_callbacks_t，但是针对 HAL 的示例工程暂时没有支持，**这也是为什么原厂一直推荐客户使用 APP example，而不推荐 HAL example 的原因。**

HAL example 路径（不推荐）：GR551x_SDK_V1.7.0\projects\peripheral

APP example 路径（推荐）：GR551x_SDK_V1.7.0\projects\peripheral_app

参考文件	外设	xx_prepare_for_sleep 支持	xx_sleep_canceled 支持	xx_wake_up_ind 支持
app_adc.c	ADC	√	√	√
app_aes.c	AES	√	√	√
app_comp.c	COMP	√	√	√
app_dma.c	DMA	√	√	√
app_dual_tim	Dual Timer	√	√	√
app_gpiote.c	GPIO	√	√	√
app_hmac.c	HMAC	√	√	√
app_i2c.c	I2C	√	√	√
app_i2s.c	I2S	√	√	√
app_pkc.c	PKC	√	√	√
app_pwm.c	PWM	√	√	√
app_qspi.c	QSPI	√	√	√
app_rng.c	RNG	√	√	√
app_spi.c	SPI	√	√	√
app_systick.c	systick	√	√	√
app_tim.c	Timer	√	√	√
app_uart.c	UART	√	√	√

说明:

以上是与开发者紧密相关的外设，在 APP 中未看到的外设及蓝牙子系统，已经在 SDK 库中包含相关的睡眠唤醒机制。

4.3 模式切换介绍

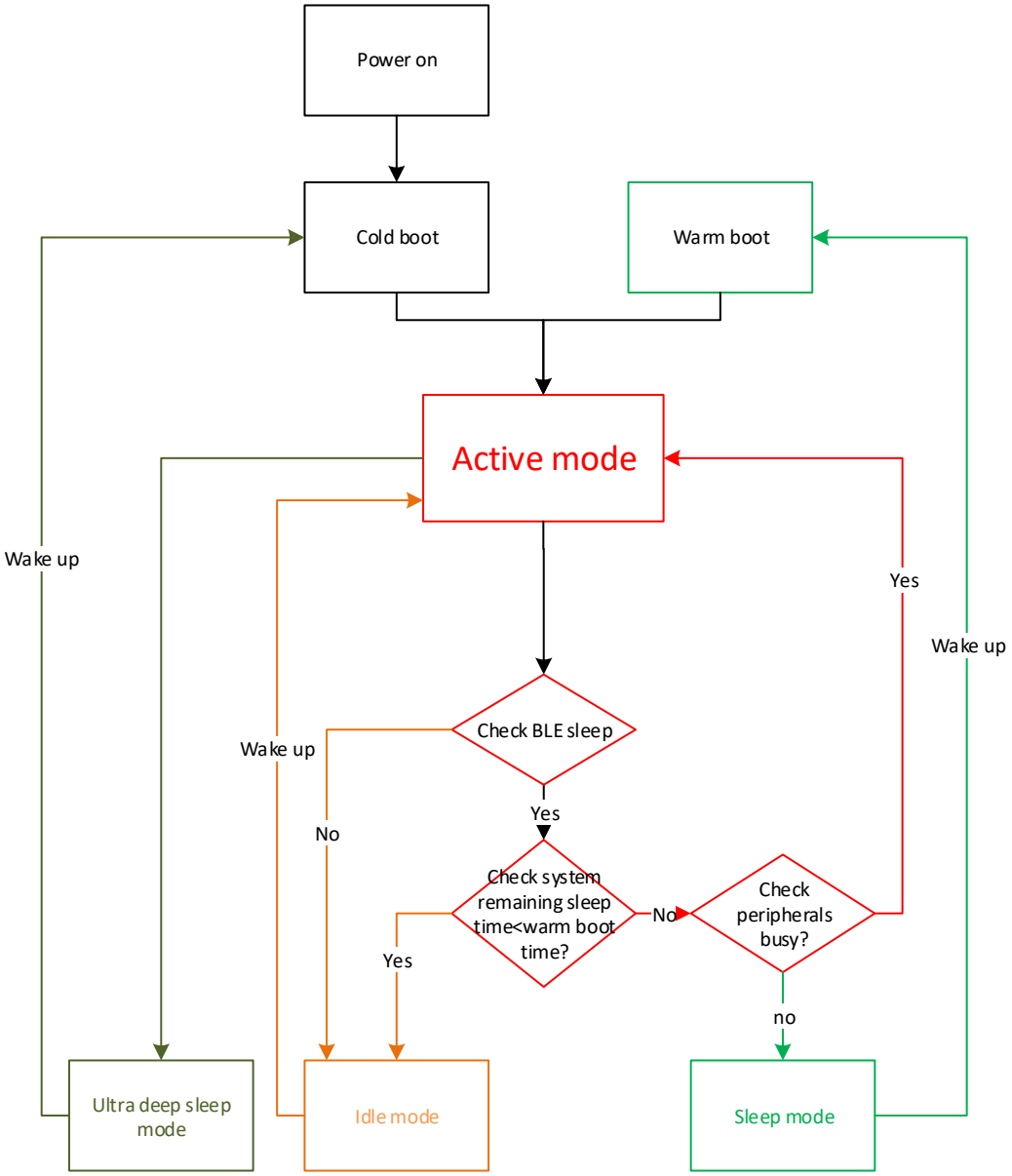

Figure 3 模式切换主要流程

1. 系统上电之后，进入 cold boot 流程，正常进入 active 模式，此模式下，可调用 pwr_mgmt_mode_set()接口，设置想进入的模式，该接口支持 active，idle，sleep 的设置，如果想进入 ultra deep sleep 模式，需要使用 pwr_mgmt_ultra_sleep()接口。
2. 进入 active 模式下，正常运行任务，系统在运行过程中会周期性检测是否 BLE 模块需要进入睡眠，如果确认 BLE 模块处于睡眠状态，那么会确认 BLE 剩余的睡眠时间是否比 warm boot 的时间更长，如果更长，为了省电，进一步走“check peripherals busy”流程，如果更短，那么会进入 idle 模式省电。

3. 如果走到“check peripherals busy”流程，睡眠管理机制会判断是否有外设处于 busy 状态，如果没有就可以正常的睡眠，如果仍然处于 busy，会继续处于 active 模式下周期检测是否进入所设置的模式。
4. 对于 sleep 模式而言，在唤醒时，会进入”warm boot”流程，对于 ultra deep sleep mode 而言，会进入 cold boot 模式。

4.4 相关概念说明

4.4.1 cold boot

Cold boot 就是 POR(Power On Reset)第一次给系统上电复位的状态，所有的 PMU，时钟，外设等都需要重新进行初始化，启动时间较长，第一次给系统上电，以及处于 ultra deep sleep 模式被唤醒，都会进入到该状态。

4.4.2 warm boot

Warm boot 意味着已经有正常启动过一次，进入到 sleep 模式时，只是关闭了系统的部分功能，比如外设，Cortex-M4F 等，且有数据暂存在 RAM 中，所以进入该流程，只需要进行恢复操作，而不需要对所有模块进行重新初始化，启动时间很短，通常在 1 ms 以内。

4.4.3 WFI

WFI 是 ARM 定义的一个指令，针对 Cortex-M4F 而言，当不再忙时，可以通过 WFI 进入低功耗状态，低功耗状态下，Cortex-M4F 的 clock 会被关闭，所以耗电会被正常情况下低很多，进入低功耗状态，可以被外部中断唤醒，正常进入 active 模式。

5. 开始使用 GR551x 的低功耗模式

在 GR551x 的 SDK 中，默认支持非 RTOS 的 example 以及 RTOS 的 example，开发者可参考如下关键步骤增加低功耗的功能。

5.1 使用非 RTOS example

在 GR551x SDK 中，默认提供了一个 PCS（Power Consumption Service）工程，该示例工程中，已经实现了低功耗，开发者可以用于测量功耗数据。

工程路径：GR551x_SDK_V1.7.0\projects\ble\ble_peripheral\ble_app_pcs

5.1.1 设置模式

a. Active/Idle/Sleep 模式

文件：

GR551x_SDK_V1.7.0\projects\ble\ble_peripheral\ble_app_pcs\Src\platform\user_periph_setup.c

```
void app_periph_init(void)

{
```

```
...
  pwr_mgmt_mode_set(PMR_MGMT_SLEEP_MODE);
}
```

说明：

这里要特别注意，设置模式之后，并不会马上进入对应的模式，这里只是设置系统想进入的模式，系统会在 pwr_mgmt_schedule()中进行策略判断，根据执行情况，选择进入对应模式的时机点。

以上 pwr_mgmt_mode_set()接口，可支持如下参数：

文件：GR551x_SDK_V1.7.0\components\sdk\gr55xx_pwr.h

```
/**@brief power manager model. */
typedef enum
{
  PMR_MGMT_ACTIVE_MODE = 0x0,      /**< Full speed state. */
  PMR_MGMT_IDLE_MODE,              /**< Idle state. */
  PMR_MGMT_SLEEP_MODE,             /**< Deep sleep state. */
} pwr_mgmt_mode_t;
```

b. Ultra deep sleep 模式

调用 pwr_mgmt_ultra_sleep()接口，该接口调用后，系统会直接进入 ultra deep sleep 模式；

5.1.2 初始化 BLE 协议栈

文件：GR551x_SDK_V1.7.0\projects\ble\ble_peripheral\ble_app_pcs\Src\user\main.c

```
 ble_stack_init(&s_app_ble_callback, &heaps_table);
```

说明：

如不使用 BLE 功能，可不调用该接口。

5.1.3 调用 schedule

文件：GR551x_SDK_V1.7.0\projects\ble\ble_peripheral\ble_app_pcs\Src\user\main.c

```
while (1)
{
    pwr_mgmt_schedule();
}
```

在非 RTOS 环境下，必须调用 pwr_mgmt_schedule()，方便 SDK 调度，切换至对应的模式。

5.2 使用 RTOS example

在 GR551x SDK 中，默认提供了一个 ble_app_template_freertos 工程，该示例工程中，默认已加入了低功耗管理的代码。

工程路径：GR551x_SDK_V1.7.0\projects\ble\ble_peripheral\ble_app_template_freertos

5.2.1 设置模式

文件：GR551x_SDK_V1.7.0\projects\ble\ble_peripheral\ble_app_template_freertos\Src\platform\user_periph_setup.c

```
void app_periph_init(void)
{
…
  pwr_mgmt_mode_set(PMR_MGMT_SLEEP_MODE);
}
```

说明：

这里要特别注意，设置模式之后，并不会马上进入对应的模式，这里只是设置系统想进入的模式，系统会在 FreeRTOS 中 tasks.c 文件的 prvIdleTask 进行策略判断，prvIdleTask 中已经默认集成了 schedule 的接口 vPortEnterDeepSleep，该接口的功能与非 RTOS 的 pwr_mgmt_schedule()接口功能类似，系统根据执行情况，选择进入对应模式的时机点。

以上 pwr_mgmt_mode_set()接口，可支持如下参数：

文件：GR551x_SDK_V1.7.0\components\sdk\gr55xx_pwr.h

```
/**@brief power manager model. */
typedef enum
{
  PMR_MGMT_ACTIVE_MODE = 0x0,      /**< Full speed state. */
  PMR_MGMT_IDLE_MODE,              /**< Idle state. */
  PMR_MGMT_SLEEP_MODE,             /**< Deep sleep state. */
} pwr_mgmt_mode_t;
```

5.2.2 初始化 BLE 协议栈

文件：
GR551x_SDK_V1.7.0\projects\ble\ble_peripheral\ble_app_template_freertos\Src\user\main.c

```
ble_stack_init(&s_app_ble_callback, &heaps_table);          /*< init ble stack*/
```

说明：

在 FreeRTOS 环境下，必须调用协议栈初始化，否则无法正常使用功耗模式，主要原因是 FreeRTOS 有复用 BLE 协议栈的 Timer。

5.2.3 调用 schedule

文件：
GR551x_SDK_V1.7.0\projects\ble\ble_peripheral\ble_app_template_freertos\Src\user\main.c

```
int main(void)
{
  app_periph_init();                                  /*<init user periph .*/
  ble_stack_init(&s_app_ble_callback, &heaps_table);          /*< init ble stack*/
  xTaskCreate(vStartTasks, "create_task", 512, NULL, 0, NULL);   /*< create some demo tasks via freertos */
  vTaskStartScheduler();                              /*< freertos run all tasks*/
  for (;;);                                     /*< Never perform here */
}
```

6. 如何在低功耗模式下调试

6.1 功耗测量准备

对低功耗调试而言，合适的工具对于低功耗调试至关重要，这里以内部调试的 GRPPK 工具为例，该工具可测量电流波形，配合 GR5515 Starter Kit 进行测试。

为什么仅测量电流，因为电源设计一旦锁定，电压基本上是不变的，测量的电流可以表征功耗情况，开发者可以根据自身的情况选择电流测量工具。

6.2 测量过程

6.2.1 硬件环境

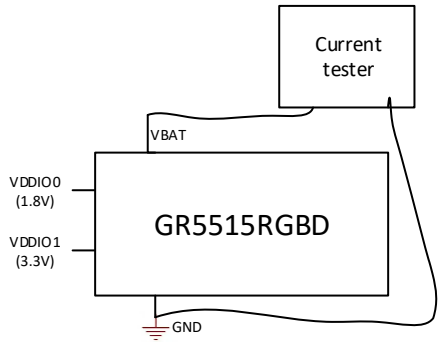

Figure 4 电流测试系统框图

采用 GR5515 Starter Kit，硬件设置如下：

Table 3 GR5515 Starter Kit 测量设置

名称	设置值	说明
VBAT	3.3V	
VDDIO0	1.8V	
VDDIO1	3.3V	J6(2 连接 3), J65 (3 连接 4), J68(1 连接 2)

6.2.2 软件环境

采用 SDK1.7.0 的 PCS 工程，默认设置，详细操作步骤可参考《GR551x Power Consumption Profile 示例手册》” 测试验证”章节。

工程目录： GR551x_SDK_V1.7.0\projects\ble\ble_peripheral\ble_app_pcs

6.2.3 测量结果

i. 电流波形预览

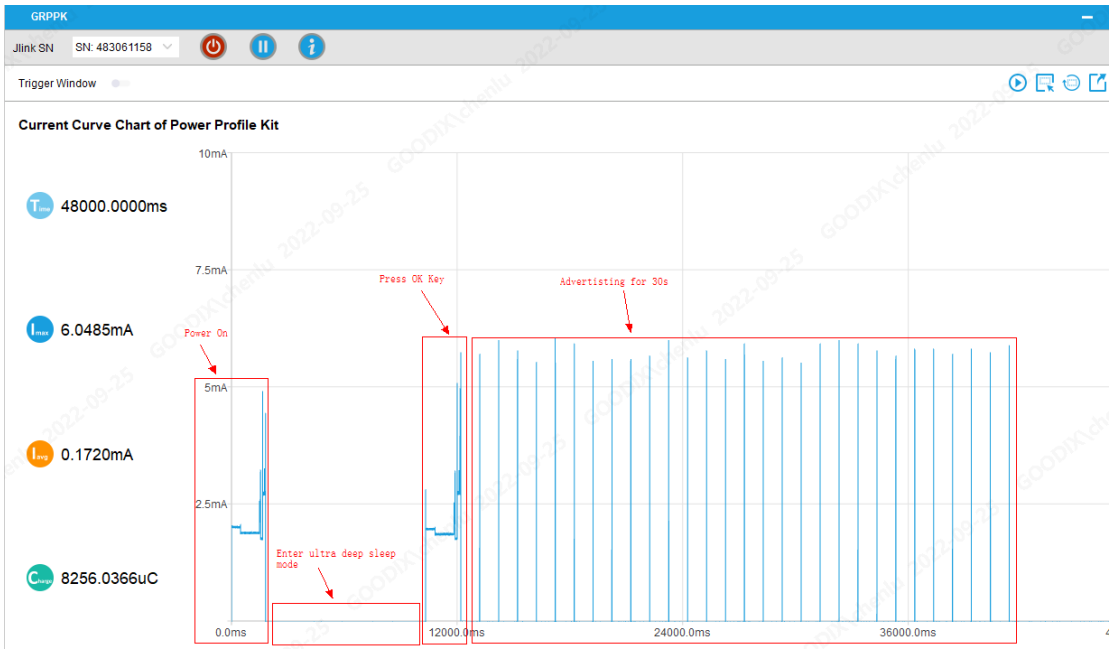

Figure 5 PCS 工程操作电流波形（1）

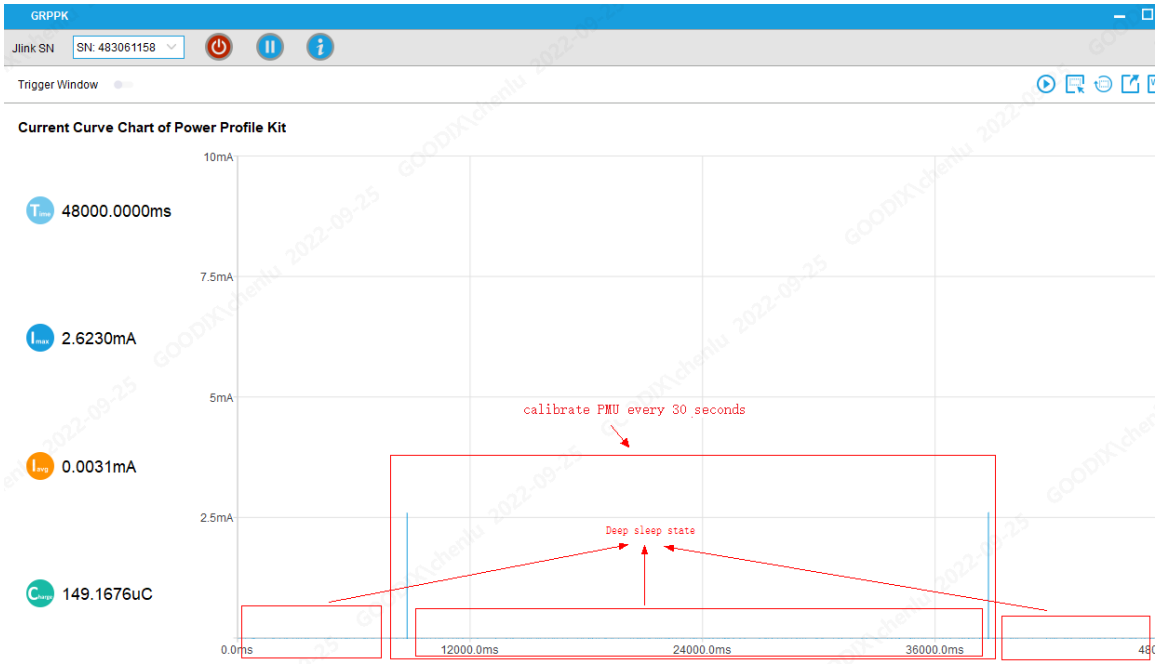

Figure 6 PCS 工程操作电流波形（2）

a. PCS 工程默认上电之后，先进入 active，然后直接进入 ultra deep sleep 模式。

b. 按“OK”键 3s，系统进入 30s 的广播。
c. 30s 广播结束后，会进入 Deep sleep，在此过程中，开发者会看到 30s 为周期的电流脉冲（产生该脉冲的原因是系统每 30s 会进行一次 PMU 校准）。

ii. Ultra deep sleep 模式电流

Figure 7 Ultra deep sleep 模式电流

iii. Sleep 模式电流

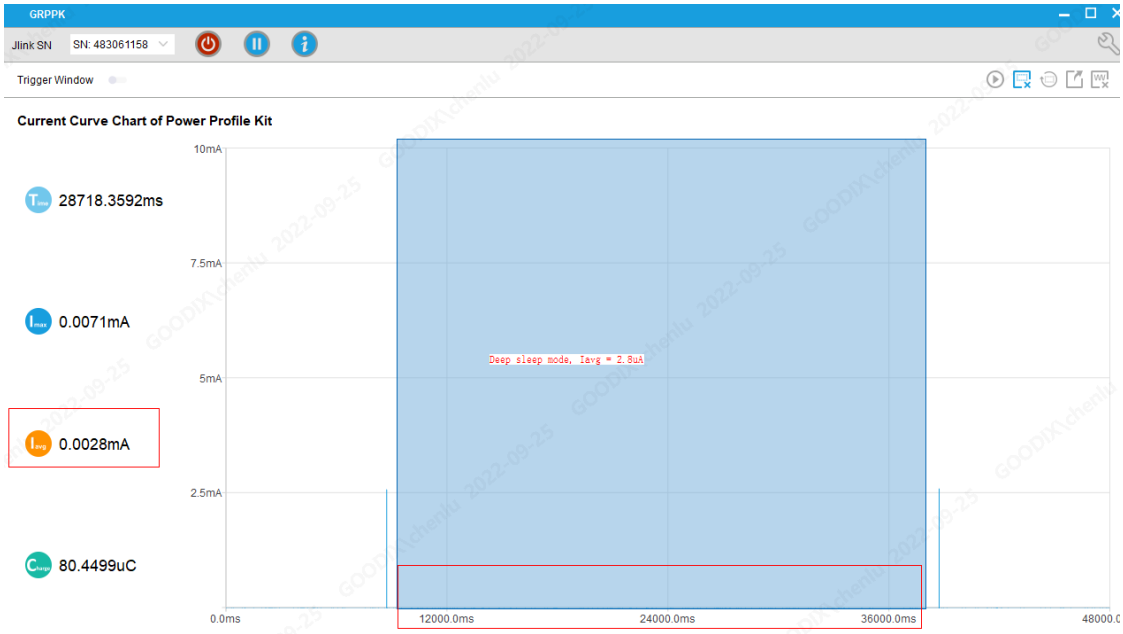

Figure 8 Deep sleep 模式电流

在调试功耗时，优先关注 Deep sleep 模式电流，Deep sleep 模式电流正常的情况下，说明外设电平，GPIO 配置等基本正确。

iv. 模式电流汇总

名称	测试值
Ultra deep sleep	1.7uA
Deep sleep	2.8uA

说明：

测试值随具体芯片存在一定的波动，上表为实验所用 GR5515 Starter Kit 的测量值，仅做参考。

6.3 GPIO/MSIO/AON_GPIO 调试

对于 GPIO/MSIO/AON_GPIO， 在睡眠情况下的状态，对功耗的影响比较大，也是功耗调试非常重要的环节；

6.3.1 GPIO/MSIO/AON_GPIO 默认态说明

基于 PCS 工程，默认只有 Flash，以及 UART0（作为 log 打印使用），在 Deep sleep 模式下，对应的 GPIO 设置情况如下表格：

Table 4 PCS 工程 GPIO/MSIO/AON_GPIO 默认配置

编号	GPIO 类型	I/O 引脚	所属电压域	Direction	Pull Down/ Pull Up/No Pull	外设
1	GPIO	GPIO_0	VDDIO1	Input	PD	
2	GPIO	GPIO_1	VDDIO1	Input	PD	
3	GPIO	GPIO_2	VDDIO1	Input	PD	
4	GPIO	GPIO_3	VDDIO1	Input	PD	
5	GPIO	GPIO_4	VDDIO1	Input	PD	
6	GPIO	GPIO_5	VDDIO1	Input	PD	
7	GPIO	GPIO_6	VDDIO1	Input	PD	
8	GPIO	GPIO_7	VDDIO1	Input	PD	
9	GPIO	GPIO_8	VDDIO1	Input	PD	
10	GPIO	GPIO_9	VDDIO1	Input	PD	
11	GPIO	GPIO_10	VDDIO1	Input	PU	UART0 log
12	GPIO	GPIO_11	VDDIO1	Input	PU	UART0 log
13	GPIO	GPIO_12	VDDIO1	Input	PD	
14	GPIO	GPIO_13	VDDIO1	Input	PD	
15	GPIO	GPIO_14	VDDIO1	Input	PD	
16	GPIO	GPIO_15	VDDIO1	Input	PD	
17	GPIO	GPIO_16	VDDIO0	Input	PD	
18	GPIO	GPIO_17	VDDIO0	Input	PD	
19	GPIO	GPIO_18	VDDIO0	Input	PU	Flash QSPI0_CS
20	GPIO	GPIO_19	VDDIO0	Input	PU	Flash QSPI0_IO3
21	GPIO	GPIO_20	VDDIO0	Input	NP	Flash QSPI0_CLK
22	GPIO	GPIO_21	VDDIO0	Input	PU	Flash QSPI0_IO2
23	GPIO	GPIO_22	VDDIO0	Input	PU	Flash QSPI0_IO1
24	GPIO	GPIO_23	VDDIO0	Input	PU	Flash QSPI0_IO0
25	GPIO	GPIO_24	VDDIO0	Input	PD	
26	GPIO	GPIO_25	VDDIO0	Input	PD	
27	GPIO	GPIO_26	VDDIO0	Input	PD	
28	GPIO	GPIO_27	VDDIO0	Input	PD	
29	GPIO	GPIO_28	VDDIO0	Input	PD	
30	GPIO	GPIO_29	VDDIO0	Input	PD	
31	GPIO	GPIO_30	VDDIO0	Input	PD	
32	GPIO	GPIO_31	VDDIO0	Input	PD	
33	MSIO	MSIO0	VBATL	Input	PD	
34	MSIO	MSIO1	VBATL	Input	PD	
35	MSIO	MSIO2	VBATL	Input	PD	
36	MSIO	MSIO3	VBATL	Input	PD	
37	MSIO	MSIO4	VBATL	Input	PD	
38	MSIO	MSIO5	VBATL	Input	PD	

41	AON_GPIO	AON_GPIO0	VDDIO1	Input	PD	
42	AON_GPIO	AON_GPIO1	VDDIO1	Input	PD	
43	AON_GPIO	AON_GPIO2	VDDIO1	Input	PD	
44	AON_GPIO	AON_GPIO3	VDDIO1	Input	PD	
45	AON_GPIO	AON_GPIO4	VDDIO1	Input	PD	
46	AON_GPIO	AON_GPIO5	VDDIO1	Input	PD	
47	AON_GPIO	AON_GPIO6	VDDIO1	Input	PD	
48	AON_GPIO	AON_GPIO7	VDDIO1	Input	PD	

6.3.2 GPIO/MSIO/AON_GPIO 的调试方式

i. 睡眠回调函数说明

在文件 GR551x_SDK_V1.7.0\components\sdk\gr55xx_pwr.h 中，提供了一个可以用获取睡前 IO 配置是否正确的接口，该回调接口会在睡眠前最后的窗口被调用，可以理解为针对 GPIO 的睡眠设置状态，开发者可在该接口保存 IO 配置状态，并在 Active 模式下通过 UART 打印 log。

pwr_mgmt_register_io_dump_func()

该接口需要在设置模式前进行初始化，如文件：GR551x_SDK_V1.7.0\projects\ble\ble_peripheral\ble_app_pcs\Src\platform\user_periph_setup.c

```
void app_periph_init(void)
{
…
  wkup_key_init();
  pwr_mgmt_register_io_dump_func(get_gpio_status_sleep);
  pwr_mgmt_mode_set(PMR_MGMT_SLEEP_MODE);
}
```

说明：

IO 配置的核心是不产生漏电回路，更多关于 GPIO 的功耗调试，可以参考《GR551x 睡眠模式及功耗测量说明》的“IO 管脚配置”。

ii. 获取 GPIO/MSIO/AON_GPIO 口的设置状态

在开发者社区的技术博客中，提供了一篇博客《GR551x 如何快速通过寄存器读取确定 GPIO，AON_GPIO，MSIO 口设置状态》，介绍如何通过直接读取寄存器，确认 GPIO/MSIO/AON_GPIO 的状态。

iii. 开关 IO LDO

GPIO/MSIO/AON_GPIO 的电源框图如下，调试 Deep sleep 模式下的 GPIO 漏电时，可以关闭 IO_LDO 快速确认 GPIO、AON_GPIO 是否有漏电，这里也取决于具体的电源方案，默认情况下，SDK 代码中，在 Deep sleep 模式下，IO LDO 处于开启状态，推荐 IO LDO 在应用阶段开启。

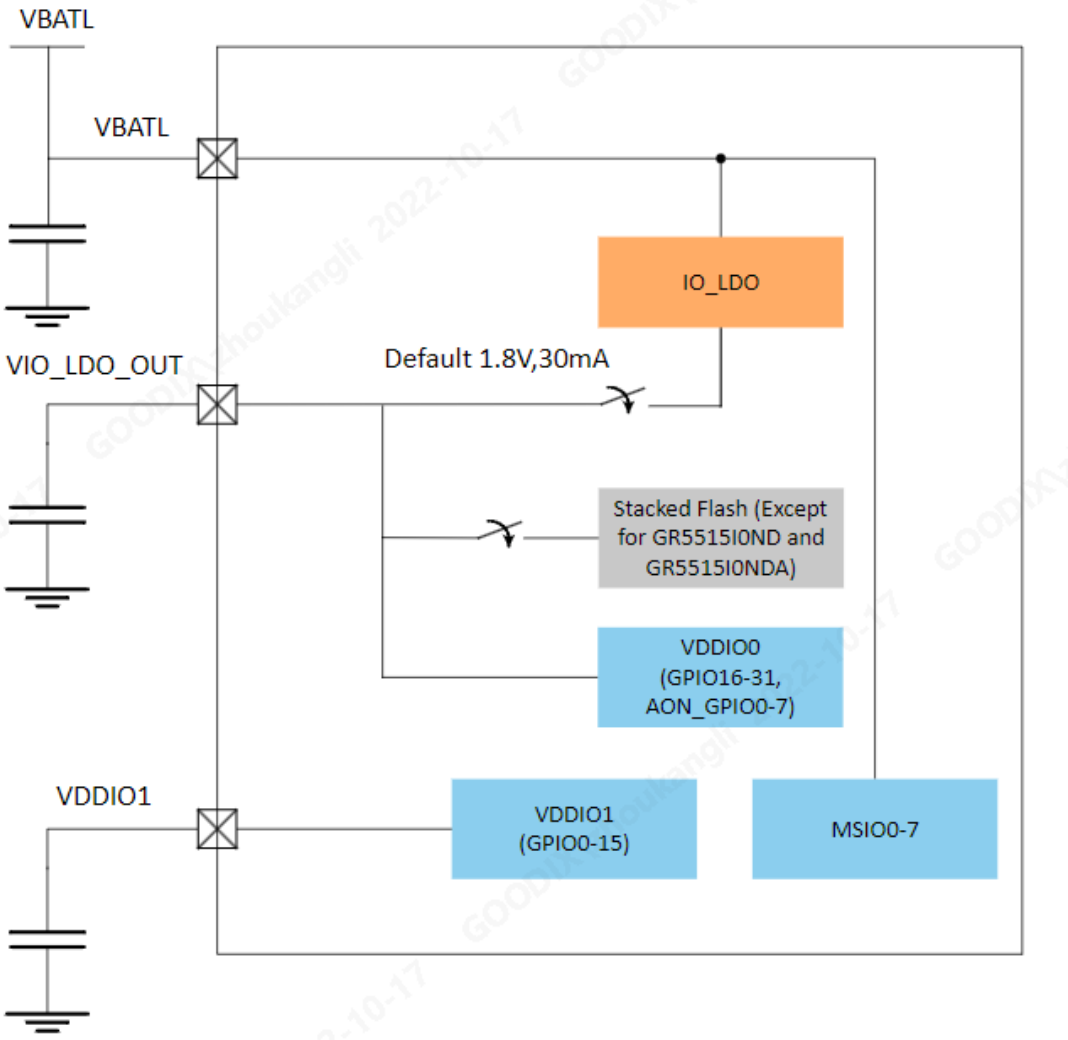

Figure 9 GPIO/AON_GPIO/MSIO 电源框图

为了快速定位 GPIO、AON_GPIO、MSIO 漏电，最好按照 Table 4 PCS 工程 GPIO/MSIO/AON_GPIO 默认配置检查；

```
void disable_io_ldo(void)
```

```
{
  AON->MEM_N_SLP_CTL |= AON_MEM_CTL_SLP_TRN_OFF_IO_LDO_EN;
}
```

可在 main.c 的 main()函数中调用，都会生效；

注意：

该接口可能会造成 Flash 代码无法正常执行，仅推荐客户用于调试使用。

6.4 低功耗蓝牙调试

关于低功耗蓝牙的主要影响参数，主要参数如下图，可以参考 PCS 工程，配合 GRToolbox 进行试验调试，选择合适的参数，如何使用 PCS 工程，可以参考《GR551x Power Consumption Profile 示例手册》，关于详细的参数原理介绍，可以参考《GR551x BLE Stack 用户指南》。

Figure 10 低功耗功耗测试场景参数

Tx Power 会影响蓝牙的功耗，关于详细如何设置蓝牙功耗，可以查看开发者社区的帖子“GR551X SDK 开发中如何设置发射功率(Tx Power)”。

7. 参考文件

《GR551x Datasheet》

《GR551x 睡眠模式及功耗测量说明》

《GR551x Power Consumption Profile 示例手册》

《GR551x BLE Stack 用户指南》

《GR551x 硬件设计指南》

《GR551x 开发者指南》

“GR5515 SK BASIC RevE”

J-Link Commander - SEGGER Wiki

8. 专业术语

名称	说明
ADC	Analog to Digital Converter
AON	Always-on
WDT	Always-on Watchdog Timer
Bluetooth LE	Bluetooth Low Energy
BOD	Brown-out Detector
LDO	Low Dropout
LPD	Low Power Domain
PMU	Power Management Unit
POR	Power-on Reset
RNG	RING Oscillator

名称	说明
SoC	System-on-Chip
SDK	Software Development Kit
GPIO	General-purpose input/output
XIP	eXecute In Place
I2C	Inter-Integrated Circuit,
I2S	Inter-IC Sound
RAM	Random-access memory
DMA	Direct memory access
PWM	Pulse-width modulation
ROM	Read-only memory
UART	Universal asynchronous receiver / transmitter
QSPI	Quad serial peripheral interface
TRNG	True Random Number Generator
PKC	Public Key Cryptography
HMAC	Hash-Based Message Authentication Codes
AES	Advanced Encryption Standard
SADC	Sense Analog-to-Digital Convert