



**Hi3861V100 / Hi3861LV100 低功耗**

## **开发指南**

文档版本 01

发布日期 2020-04-30

版权所有 © 上海海思技术有限公司2020。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

## 商标声明



**HISILICON**、海思和其他海思商标均为海思技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

## 注意

您购买的产品、服务或特性等应受海思公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，海思公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

# 上海海思技术有限公司

地址：            深圳市龙岗区坂田华为总部办公楼    邮编：518129

网址：            <https://www.hisilicon.com/cn/>

客户服务邮箱：  [support@hisilicon.com](mailto:support@hisilicon.com)



# 前言

## 概述

本文档详细描述了Hi3861V100 / Hi3861LV100 的系统低功耗模式及应用开发指导，同时提供了常见问题的处理方法。

## 产品版本

与本文档对应的产品版本如下。

| 产品名称    | 产品版本 |
|---------|------|
| Hi3861  | V100 |
| Hi3861L | V100 |



## 读者对象

本文档主要适用于以下工程师：



- 技术支持工程师
- 软件开发工程师

## 符号约定

在本文中可能出现下列标志，它们所代表的含义如下。

| 符号  | 说明                              |
|---|---------------------------------|
|  <b>危险</b> | 表示如不可避免则将会导致死亡或严重伤害的具有高等级风险的危害。 |
|  <b>警告</b> | 表示如不可避免则可能导致死亡或严重伤害的具有中等级风险的危害。 |



| 符号   | 说明   |
|--|--|
|  注意 | 表示如不可避免则可能导致轻微或中度伤害的具有低等级风险的危害。  |
| 须知   | 用于传递设备或环境安全警示信息。如不可避免则可能会导致设备损坏、数据丢失、设备性能降低或其它不可预知的结果。<br>“须知”不涉及人身伤害。 |
|  说明 | 对正文中重点信息的补充说明。<br>“说明”不是安全警示信息，不涉及人身、设备及环境伤害信息。                        |

## 修改记录

| 文档版本  | 发布日期       | 修改说明  |
|-------|------------|---|
| 01    | 2020-04-30 | 第一次正式版本发布。 <ul style="list-style-type: none"><li>新增“<a href="#">1.1 低功耗配置前提条件</a>”小节。</li><li>在“<a href="#">5 常见问题</a>”的<a href="#">不能进入休眠模式，可能是什么原因？</a>中新增协议栈（lwIP）未配置低功耗模式的说明。</li></ul> |
| 00B01 | 2020-04-03 | 第一次临时版本发布。  |



## 目录

|                    |    |
|--------------------|----|
| 前言.....            | i  |
| 1 概述.....          | 1  |
| 1.1 低功耗配置前提条件..... | 1  |
| 1.2 系统低功耗模式说明..... | 2  |
| 2 超深睡模式.....       | 4  |
| 2.1 概述.....        | 4  |
| 2.2 接口说明.....      | 4  |
| 2.3 应用示例.....      | 4  |
| 3 深睡模式.....        | 6  |
| 3.1 概述.....        | 6  |
| 3.2 接口说明.....      | 6  |
| 3.3 应用示例.....      | 7  |
| 4 浅睡模式.....        | 10 |
| 4.1 概述.....        | 10 |
| 4.2 接口说明.....      | 10 |
| 4.3 应用示例.....      | 10 |
| 5 常见问题.....        | 11 |



# 1 概述

## 1.1 低功耗配置前提条件

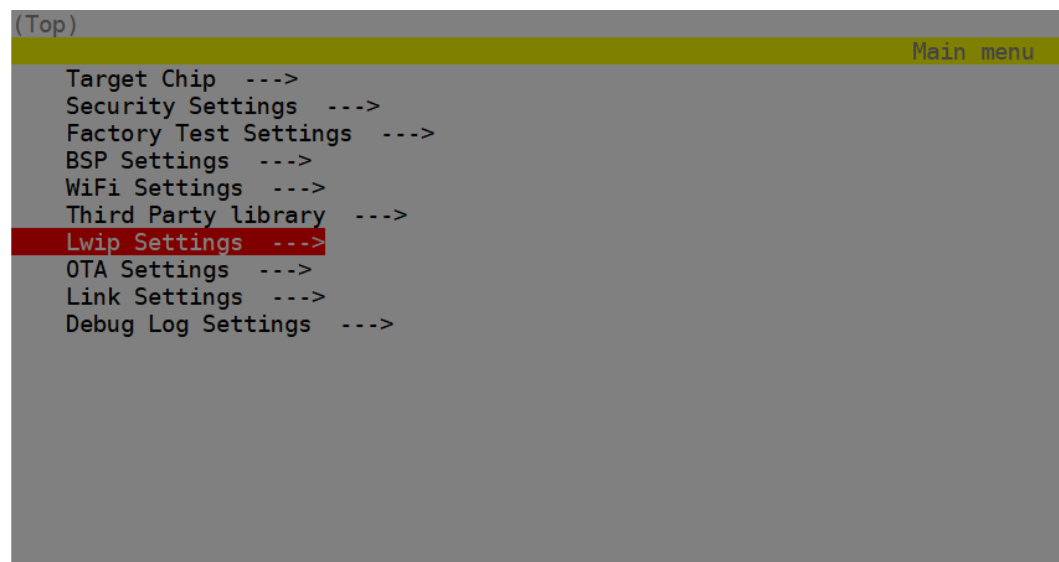
## 1.2 系统低功耗模式说明

## 1.1 低功耗配置前提条件

使用低功耗场景时，需要在SDK的menuconfig配置中打开协议栈（lwip）低功耗模式，具体配置步骤如下：

- 步骤1** 在SDK的代码目录中运行“./build.sh menuconfig”，选中“Lwip Settings --->”（如图1-1所示）后按“Enter”键。

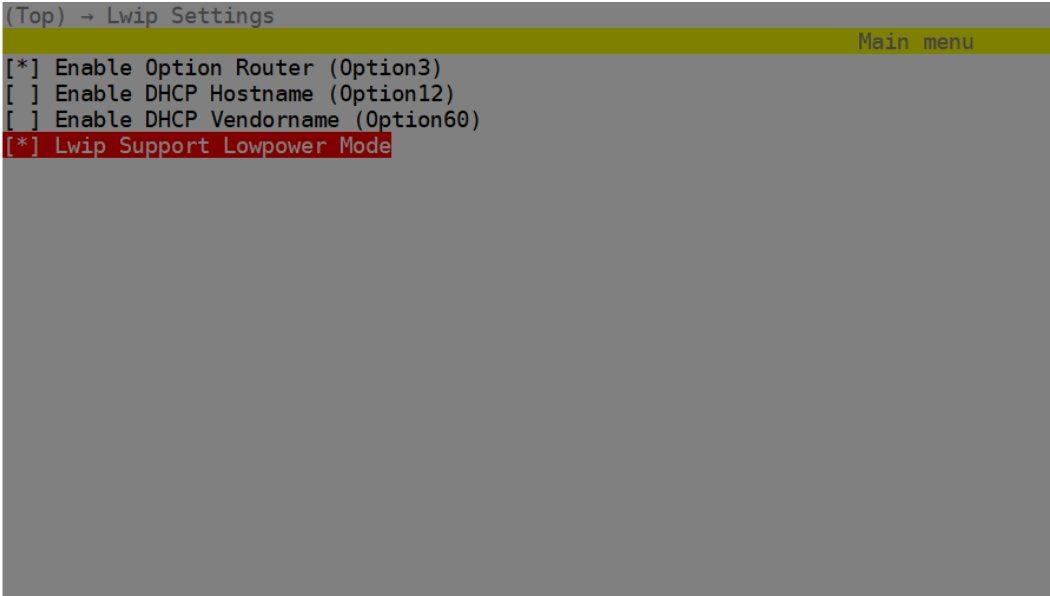
图 1-1 menuconfig 界面



- 步骤2** 选中“Lwip Support Lowpower Mode”后按“Enter”键（前方“[ ]”中显示“\*”），此时表示将协议栈（lwip）低功耗模式功能选中（如图1-2所示）。



图 1-2 配置协议栈（lwip）低功耗功能示例



**步骤3** 保存协议栈（lwip）低功耗配置，重新编译代码（参考《Hi3861V100 / Hi3861LV100 SDK开发环境搭建 用户指南》）。

----结束

## 1.2 系统低功耗模式说明

Hi3861V100/Hi3861LV100支持3种系统低功耗模式（如表1-1所示），用户可根据实际应用场景选用对应的低功耗策略（默认未使能任何低功耗模式）。

表 1-1 低功耗模式说明

| 模式    | 芯片状态   | 特点  |
|-------|--|---|
| 超深睡模式 | <ul style="list-style-type: none"><li>• CPU下电，RAM下电，所有外设下电。</li><li>• 该模式仅支持GPIO3/5/7/14高电平唤醒。</li></ul> | <ul style="list-style-type: none"><li>• 不支持Wi-Fi协议规定的节能模式。</li><li>• 唤醒后系统重新启动（类似上电重启）。</li><li>• 不保持AP连接。</li><li>• 不支持GPIO输出电平保持。</li><li>• 在深睡/浅睡/非低功耗模式下配置进入。</li></ul> |



| 模式   | 芯片状态  | 特点   |
|------|---|--|
| 深睡模式 | <ul style="list-style-type: none"><li>• CPU下电，RAM不下电，GPIO、RTC以外其他外设下电。</li><li>• 该模式仅支持GPIO/SDIO/Wi-Fi唤醒。</li></ul> | <ul style="list-style-type: none"><li>• 支持Wi-Fi协议规定的节能模式。</li><li>• 保持AP连接（Station关联AP后）。</li><li>• GPIO/SDIO业务正常，其他外设休眠时不能正常工作。</li><li>• 支持GPIO输出电平保持。</li><li>• 与浅睡模式互斥，不能同时设置。</li></ul>     |
| 浅睡模式 | CPU不下电，RAM不下电，所有外设可控下电。   | <ul style="list-style-type: none"><li>• 支持Wi-Fi协议规定的节能模式。</li><li>• 保持AP连接（Station关联AP后）。</li><li>• 主要依赖Wi-Fi子系统的低功耗，关闭Wi-Fi模块电路，包括RF。</li><li>• 支持所有外设正常工作。</li><li>• 与深睡模式互斥，不能同时设置。</li></ul> |

## 说明

详细功耗指标请参见《Hi3861V100 / Hi3861LV100 开发板功耗 测试指南》。





# 2 超深睡模式

## 2.1 概述

## 2.2 接口说明

## 2.3 应用示例

## 2.1 概述

超深睡模式下仅保留IO唤醒相关模块供电，IO管脚配置值恢复为芯片默认值，其他模块均下电，内存信息不保留，Wi-Fi连接断开。超深睡模式由用户调用接口后直接进入超深睡模式，接口参数可以指定唤醒IO。对应IO为高电平会将系统从超深睡模式中唤醒，重新启动进入非低功耗模式。如果在超深睡模式下依然有保持IO输出电平状态的需求，则需要通过硬件电路设计实现。

## 2.2 接口说明

### 说明

具体API描述请参见《Hi3861V100 / Hi3861LV100 API 开发参考》。

表 2-1 超深睡模式接口说明

| 接口名称                          | 描述                             |
|-------------------------------|--------------------------------|
| hi_lpc_enable_udsleep         | 使能进入超深睡模式，同时设置唤醒IO。            |
| hi_lpc_get_udsleep_wakeup_src | 唤醒后获取唤醒IO接口（须在hi_lpc_init后使用）。 |

## 2.3 应用示例

超深睡模式一般用于按键唤醒、主从设备长时间待机唤醒等极低功耗要求的场景。



## 说明

- 如果无高电平唤醒源，设备会一直保持在超深睡状态。
- 在此期间设备内存下电，超深睡唤醒后可通过接口获取唤醒源，其他均与重新上电相同。
- 唤醒管脚高电平建议至少维持100μs。

代码示例：

```
/* 设置GPIO5和GPIO7为唤醒源 */  
hi_lpc_enable_udsleep(HI_UDS_GPIO5 | HI_UDS_GPIO7);  
/* 系统休眠... */  
  
/* 唤醒后重新执行初始化流程，打印唤醒源，具体值参考hi_udsleep_src */  
(hi_void)hi_lpc_init();  
ret = hi_lpc_get_udsleep_wakeup_src(&src);  
if (ret == HI_ERR_SUCCESS) {  
    printf("udsleep wakeup src: %x\r\n", src);  
} else {  
    /* 异常处理略 */  
}
```



# 3 深睡模式

## 3.1 概述

## 3.2 接口说明

## 3.3 应用示例

## 3.1 概述

Hi3861/Hi3861L的低功耗利用idle空闲任务进行系统管理，idle任务优先级最低，该任务仅在系统空闲时执行，系统判断没有即将执行的业务则可以进入深睡模式，即CPU等下电。该模式可以通过深睡唤醒源唤醒，深睡唤醒源包括GPIO、SDIO、系统tick（RTC）等。系统休眠时间采用tickless机制，避免每个系统tick都被唤醒。

模组作为STA，设置深睡模式后，系统空闲时会自动进入休眠，默认休眠时间取决于所关联AP的DTIM及Beacon周期。

### 说明

DTIM (Delivery Traffic Indication Message)：AP发送广播/组播数据包的频率。

## 3.2 接口说明

### 说明

- 用户通过hi\_lpc\_set\_type设置模式，通过hi\_lpc\_add\_veto和hi\_lpc\_remove\_veto实现否决投票，采用一票否决机制，只要有一个模块禁止休眠，则系统不能进入系统低功耗模式。
- 具体API描述请参见《Hi3861V100 / Hi3861LV100 API 开发参考》。

表 3-1 深睡模式接口说明

| 接口名称            | 描述  |
|-----------------|---|
| hi_lpc_get_type | 获取当前的系统低功耗模式。   |
| hi_lpc_set_type | 设置系统低功耗模式（不包含超深睡模式，超深睡通过hi_lpc_enable_udsleep接口配置进入）。 |



| 接口名称                           | 描述  |
|--------------------------------|---|
| hi_lpc_add_veto                | 用户投票使用，否决进入休眠状态。用户可通过增加hi_lpc_id枚举实现否决投票，每个ID仅对应一个否决票。  |
| hi_lpc_remove_veto             | 用户投票使用，解除否决进入休眠状态。  |
| hi_lpc_register_wakeup_entry   | 注册深睡唤醒入口，仅支持注册一组。作为外设重新初始化入口函数。   |
| hi_lpc_register_check_handler  | 注册入睡检查接口，支持注册多组。<br>用户注册接口在入睡前被调用，通过返回值反馈是否允许休眠。一般用于需要实时检查的状态，如UART是否处于接收或发送数据状态的判断。其他场景建议使用执行效率更高的hi_lpc_add_veto/ hi_lpc_remove_veto接口。 |
| hi_lpc_register_hw_handler     | 注册硬件相关处理接口，仅支持注册一组，以最后一次注册为准。<br>入睡前和唤醒后调用相应处理函数，例如：深睡时IO设置为高阻态，进一步降低漏电流，醒来时恢复配置。   |
| hi_lpc_register_sw_handler     | 注册软件相关处理接口，仅支持注册一组，以最后一次注册为准。<br>进入idle任务和退出idle任务中执行，处于idle任务的关中断阶段，一般用于调试。  |
| hi_lpc_config_dsleep_wakeup_io | 配置深睡模式唤醒对应的GPIO。<br>如果有IO中断处理需求，需要与GPIO外部中断接口配合使用。  |
| hi_wifi_set_pm_switch          | 打开/关闭Wi-Fi子系统低功耗。   |

## 3.3 应用示例

代码示例：

```
/* 入睡判断前执行 */
hi_u32 sw_prepare(hi_void)
{
    if (hi_lpc_get_type() == HI_DEEP_SLEEP) {
        /* 用户可根据实际情况，关闭影响系统休眠的部分timer */
    }
    return HI_ERR_SUCCESS;
}
/* idle任务退出时执行 */
hi_u32 sw_resume(hi_void)
{
    if (hi_lpc_get_type() == HI_DEEP_SLEEP) {
        /* 用户根据实际情况，恢复对应timer，或增加维测信息的获取 */
    }
    return HI_ERR_SUCCESS;
}
/* 入睡前执行 */
hi_u32 hw_prepare(hi_void)
{
```



```
if (hi_lpc_get_type() == HI_DEEP_SLEEP) {
    /* 用户根据实际IO设计配置，防止深睡阶段漏电流 */
}
return HI_ERR_SUCCESS;
}
/* 唤醒后执行 */
hi_u32 hw_resume(hi_void)
{
    if (hi_lpc_get_type() == HI_DEEP_SLEEP) {
        /* 用户根据实际IO设计恢复配置 */
    }
    return HI_ERR_SUCCESS;
}
hi_u32 demo_init(hi_void)
{
    hi_u32 ret;
    hi_pvoid handle;
    /* 深睡唤醒阶段入口函数注册，掉电外设初始化，具体可参考SDK交付的Demo代码 */
    ret = hi_lpc_register_wakeup_entry(wakeup);
    if (ret != HI_ERR_SUCCESS) {
        /* 异常处理略 */
    }
    /* 注册是否可以进入休眠的检查函数，对应函数在idle入睡前被调用 */
    handle = hi_lpc_register_check_handler(check);
    if (handle == HI_NULL) {
        /* 异常处理略 */
    }
    /* 深睡阶段降低漏电流功耗 */
    ret = hi_lpc_register_hw_handler(hw_prepare, hw_resume);
    if (ret != HI_ERR_SUCCESS) {
        /* 异常处理略 */
    }
    /* 深睡阶段对timer的特殊处理和维测信息统计等，一般不需要注册 */
    ret = hi_lpc_register_sw_handler(sw_prepare, sw_resume);
    if (ret != HI_ERR_SUCCESS) {
        /* 异常处理略 */
    }
    /* 使能GPIO7上升沿中断 */
    ret = hi_gpio_init();
    if (ret != HI_ERR_SUCCESS) {
        /* 异常处理略 */
    }
    ret = hi_gpio_register_isr_func(HI_GPIO_IDX_7, HI_INT_TYPE_EDGE,
        HI_GPIO_EDGE_RISE_LEVEL_HIGH, demo_gpio7_wkup, HI_NULL);
    if (ret != HI_ERR_SUCCESS) {
        /* 异常处理略 */
    }
    /* 使能GPIO7唤醒 */
    ret = hi_lpc_config_dsleep_wakeup_io(HI_GPIO_IDX_7, HI_TRUE);
    if (ret != HI_ERR_SUCCESS) {
        /* 异常处理略 */
    }
    /* 关联AP，获取IP地址，配置ARP_OFFLOAD，代码略 */
    /* 设置系统休眠为深睡模式 */
    ret = hi_lpc_set_type(HI_DEEP_SLEEP);
    if (ret != HI_ERR_SUCCESS) {
        /* 异常处理略 */
    }
    /* 打开Wi-Fi子系统低功耗 */
    ret = hi_wifi_set_pm_switch(HI_TRUE, 0);
    if (ret != HI_ERR_SUCCESS) {
        /* 异常处理略 */
    }
}
```

```

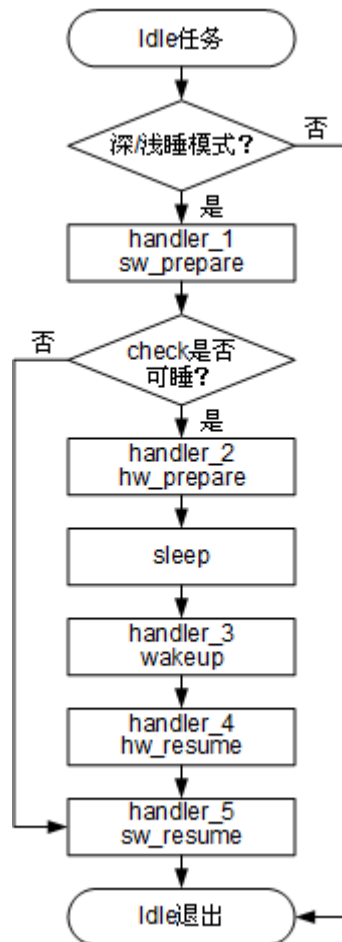
    }
    return HI_ERR_SUCCESS;
}

```

#### 说明

注册接口在代码闭源情况下可以增加用户的灵活度，与idle任务的对应关系如图3-1所示。

图 3-1 注册接口与 idle 任务关系图





# 4 浅睡模式

## 4.1 概述

## 4.2 接口说明

## 4.3 应用示例

## 4.1 概述

实现原理与深睡模式类似，区别为CPU和外设不下电。浅睡模式下CPU本身不会降低功耗，主要采用控制子模块功耗来降低功耗。这里主要通过打开Wi-Fi子系统低功耗来实现降低系统功耗（不需要设定浅睡模式）。如果需要深度定制（例如：开关外设时钟），需使能浅睡模式后注册接口，在入睡前和唤醒后时对模块时钟进行开关操作。

## 4.2 接口说明

### 说明

具体API描述请参见《Hi3861V100 / Hi3861LV100 API 开发参考》。

表 4-1 浅睡模式接口说明

| 接口名称                  | 描述                |
|-----------------------|-------------------|
| hi_wifi_set_pm_switch | 打开/关闭Wi-Fi子系统低功耗。 |

## 4.3 应用示例

代码示例：

```
/* 打开Wi-Fi子系统低功耗，休眠时间跟随AP侧配置 */  
hi_wifi_set_pm_switch(HI_TRUE, 0);
```



# 5 常见问题

## 不能进入休眠模式，可能是什么原因？

不能进入休眠模式原因比较多，常见以下几种原因：

- 定时器使用错误
  - 举例：启动10ms周期定时器。
  - 分析：系统在入睡前会检查是否有定时器即将到期，如果定时器即将到期，系统不允许进入系统低功耗模式。
  - 建议：根据实际业务启动定时器，在业务空闲时关闭对应定时器，特别是周期性定时器。
- 任务使用错误
  - 举例：任务体中循环操作，无主动释放动作，如调用阻塞接口或hi\_sleep接口。
  - 分析：某一任务无主动释放动作，则其他低优先级任务得不到执行，严重时会引起看门狗复位。系统在入睡前会检查是否有任务即将被调度，如果存在任务主动释放过少，则系统不允许进入系统低功耗模式。
  - 建议：业务设计上尽量调用阻塞接口，超时时间设置为无穷大或合理超时时间。
- hi\_sleep使用错误
  - 举例：调用hi\_sleep接口，传参为10ms。
  - 分析：该行为效果类似于启动10ms周期定时器，该任务会以10ms周期被调度，入睡前检查有任务即将到期，不允许进入系统低功耗模式。
  - 建议：尽量采用阻塞接口，阻塞时间设置为无穷大或合理超时时间；或者采用定时器实现，并在业务认为空闲时关闭定时器。
- hi\_lpc\_add\_veto/hi\_lpc\_remove\_veto使用错误
  - 举例：hi\_lpc\_add\_veto/hi\_lpc\_remove\_veto未成对使用。
  - 分析：系统在入睡前会检查“否决”休眠的状态，如有某个模块否决休眠，则一票即否决进入休眠模式，如果不解除对应的否决状态，系统将永远不能进入休眠模式。
  - 建议：严格检查是否成对使用，并保证否决休眠的状态持续时间尽量短，以保证尽快进入休眠模式。
- 未打开Wi-Fi子系统低功耗





- 举例：未调用hi\_wifi\_set\_pm\_switch接口打开Wi-Fi子系统低功耗。
- 分析：打开Wi-Fi子系统低功耗是系统可进入休眠模式的前提，否则即使设置了低功耗模式系统也不会进入休眠模式。
- 建议：采用低功耗策略，必须打开Wi-Fi子系统低功耗。
- 协议栈（lwIP）未配置低功耗模式
  - 举例：采用低功耗策略，必须在SDK版本menuconfig配置协议栈（lwIP）子系统低功耗，具体配置方法请参见《Hi3861V100 / Hi3861LV100 SDK开发环境搭建 用户指南》。
  - 影响：协议栈配置低功耗可能会影响WiFi业务的性能，默认为关闭协议栈低功耗模式。

## 为什么从深睡模式唤醒后系统执行异常？

- 举例：增加I2C接口的调用后，上电后系统运行正常，深睡模式唤醒后I2C工作异常。
- 分析：深睡模式后外设模块掉电，唤醒后需要进行重新初始化。
- 建议：在hi\_lpc\_register\_wakeup\_entry注册的接口中增加I2C的初始化，同时注意初始化位置，须在UART和Flash初始化之后调用对应初始化函数。

## 为什么外设对应业务概率性收发报文异常？

- 举例：SPI设备通信发现偶尔接收不到数据，或发送数据与预期不符。
- 分析：深睡后外设均掉电，会导致接收不到对端的发送数据，或者调用完异步发送接口后，数据实际并未发送出去，但系统认为没有业务即将或正在执行，从而进入休眠模式导致数据发送异常。
- 建议：对应业务上需要增加hi\_lpc\_add\_veto/hi\_lpc\_remove\_veto等接口调用，以通知系统是否可以进入休眠模式。

## 为什么有时候进入超深睡模式后立即被唤醒？

建议：查看唤醒源对应管脚电平是不是稳定的低电平状态。

## 为什么功耗数据会比预期偏高？

建议：从硬件和软件两个方面排查：

- 硬件：检查底电流是否符合预期。
- 软件：通过以下手段排查：
  - a. 通过调用维测接口hi\_lpc\_get\_info查看统计量type，确认模式正确，查看统计量sleep\_times，确认曾经进入休眠模式，隔一段时间观测其值是否持续增加，即是否在持续切换入睡唤醒状态。
  - b. 通过调用维测接口hi\_lpc\_get\_info查看统计量veto\_info，查看是否有模块投票拒绝休眠。
  - c. 通过调用维测接口hi\_lpc\_get\_info查看统计量sleep\_threshold\_refuse\_times，如果该值持续增加，进一步观测结构体对应统计量task\_xxx和timer\_xxx，确认是否是任务或定时器即将到期引起的禁止进入休眠模式。
  - d. 通过在注册回调hw\_prepare和hw\_resume中调用hi\_systick\_get\_cur\_tick获取系统时钟（基于RTC时钟）来计算休眠时间，确认每次休眠时长是否足够，



可累积计算醒睡比。但由于hi\_systick\_get\_cur\_tick获取时间会产生百微秒级别耗时，此步骤仅适合调试定位阶段使用。