AC695N 功耗调试说明书

珠海市杰理科技股份有限公司
Zhuhai Jieli Technologyco.,LTD

地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

修改记录

版本	更新日期	描述
V1.0	2020/11/20	撰写调试文档

地址：珠海市吉大石花西路 107 号 9 栋综合楼　　邮编：519015
电话：0756-6313088　　传真：0756-6313081
网站：www.zh-jieli.com

目录

地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

1. 引 言

1.1. 编写目的

该文档为基于 AC695N 平台开发应用的人员提供相应的功耗调试文档。也可以为测试应用的测试人员提供参考。

文档中详细介绍了 AC695N 应用的蓝牙功耗、关机功耗、功耗调试方法以及部分低功耗流程，以便于应用的详细设计。

1.2. 参考资料

[1]

1.3. 术语和缩写词

缩写和术语	解 释

2. power_down

2.1. 概述

在 SDK 中，影响系统功耗的因素最主要是

（1）系统是否能够进入 power_down；

（2）系统进入 power_down 后唤醒的频率；

确保系统能进入 power_down 以及尽量减小 power_down 后唤醒的频率会对系统功耗有很大的改善。

2.2. 调试

影响系统功耗的最主要原因就是系统是否能够进入 power_down，以及进入后被唤醒的频率。我们应该先确保是否能够进入 power_down，判断是否有进入 power_down 可通过下面的打印来进行。

地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

```
void sleep_exit_callback(u32 usec)
{
    /* putchar('>'); */
    if(TCFG_LOWPOWER_POWER_SEL == PWR_DCDC15){
        gpio_set_die(power_param.dcdc_port, 1);
        gpio_direction_output(power_param.dcdc_port, 1);
        power_set_mode(TCFG_LOWPOWER_POWER_SEL);
    }
    APP_IO_DEBUG_1(A, 6);
}

void sleep_enter_callback(u8  step)
{
    /* 此函数禁止添加打印 */
    if (step == 1) {
        /* putchar('<'); */
        if(TCFG_LOWPOWER_POWER_SEL == PWR_DCDC15){
            power_set_mode(PWR_LDO15);
            gpio_set_die(power_param.dcdc_port, 1);
            gpio_direction_output(power_param.dcdc_port, 0);
        }
```

(1)当系统一直没有出现图示打印时，首先先定位是哪里导致系统无法进入低功耗。通过打开下面打印，我们可以大致找到问题所在：

```
1lib_driver_config.c  2app_config.h+
 7
 8
 9 #ifdef CONFIG_RELEASE_ENABLE
10 #define LIB_DEBUG    1
11 #else
12 #define LIB_DEBUG    1
13 #endif
14
```

```
1lib_driver_config.c+  2app_config.h
75
76 const char log_tag_const_v_PMU AT(.LOG_TAG_CONST) = CONFIG_DEBUG_LIB(FALSE);
77 const char log_tag_const_i_PMU AT(.LOG_TAG_CONST) = CONFIG_DEBUG_LIB(TRUE);
78 const char log_tag_const_d_PMU AT(.LOG_TAG_CONST) = CONFIG_DEBUG_LIB(TRUE);
79 const char log_tag_const_w_PMU AT(.LOG_TAG_CONST) = CONFIG_DEBUG_LIB(TRUE);
80 const char log_tag_const_e_PMU AT(.LOG_TAG_CONST) = CONFIG_DEBUG_LIB(TRUE);
81 const char log_tag_const_c_PMU AT(.LOG_TAG_CONST) = CONFIG_DEBUG_LIB(TRUE);
```

当打开这两处打印后，系统会出现比较多的"Name : %s busy"打印，我们系统进入 power_down 前都会去查询每个模块是否有不能被打断的任务。如果有，系统就不会进入 power_down，并一直打印正在 busy 的模块，等到所有模块查询都为空闲才会进入。系统很多地方都注册了这样的查询函数，当函数返回 1 的时候，系统才会判断这个模块为空闲；如果返回 0，则判断为忙碌。因此可以通过该打印来判断是哪个模块占用了系统。

地址：珠海市吉大石花西路 107 号 9 栋综合楼　　邮编：519015
电话：0756-6313088　　传真：0756-6313081
网站：www.zh-jieli.com

```
static u8 hrs3300_idle_query(void)
{
    return hrs3300_is_idle;
}

REGISTER_LP_TARGET(hrs3300_lp_target) = {
    .name = "hrs3300",
    .is_idle = hrs3300_idle_query,
};
```

除此之外，有时候可能会发现没有出现 busy 打印，但系统仍然无法进入 power_down 的情况。这种情况很大可能性是程序里面用 sys_timer_add 注册了太多定时，或者注册了一个时间间隔很短的定时，该定时注册函数所注册的定时是会唤醒系统的。如果定时间隔小于进出 power_down 所必须消耗的恢复时间，那么系统也不会进入 power_down。因此使用 sys_timer_add 注册定时需谨慎。我们 SDK 一共有三种定时注册函数，它们的特点如下表：

定时注册函数	描述
sys_timer_add	该函数注册的定时，到时间后由 sys_timer 线程发送消息到调用注册的线程来完成注册的定时任务。另外，该函数注册的定时精度为 10ms。
sys_hi_timer_add	该函数注册的定时，到时间后会由中断来完成注册的定时任务。由于该注册函数的优先级高于 power_down，所以当系统发现存在用该函数注册定时，系统一直都不会进 power_down，只有等到该定时被删除为止。
sys_hi_timeout_add	该函数注册的定时，到时间后会由中断来完成注册的定时任务。当系统处于 power_down 时，该函数注册的定时计数节拍会被停止，系统唤醒后继续计数。这种情况会导致系统处于 power_down 时，定时的时间间隔变得不固定。

(2)当系统出现图示“<>”打印时，说明系统已经能够成功进入低功耗。在确保系统能够正常进入低功耗后，接下来就是要根据“<>”打印来判断系统是否被频繁唤醒。系统低功耗期间(power_down)，其自身是会以大概 500ms 的时间周期唤醒。因此，在正常情况下，我们会看到一个比较缓慢的“<>”打印。Power_down 除了自身能够周期性唤醒外，还能被其他系统调度和 IO 中断唤醒。因此，当我们看到“<>”打印比较频繁的时候说明系统其他地方一直存在频繁调度或者有某个唤醒 IO 一直触发中断。

系统的频繁调度一般都是由蓝牙和 sys_tiemr_add 引起的。蓝牙模块引起的系统调度后面章节再进行介绍。而 sys_timer_add，我们 SDK 自身会有部分模块使用了 sys_timer_add，例如各个设备检测（SD 卡，USB，LINEIN）等，这些设备检测注册的定时时间都相对较短，因此设备检测开得越多，也会造成系统唤醒变频繁。

IO 中断唤醒要是某些 IO 被注册成唤醒 IO。当这些 IO 满足事先设置好的电平变化后，系统就会被 IO 中断唤醒，进行中断处理。除了 IO 中断外，某些硬件中断也会唤醒系统，但最常见的一般都是 IO 中断，如果出现其他中断频繁唤醒系统，就需要根据中断的类型进一步调试。下面是给开发人员预留的设置唤

地址：珠海市吉大石花西路 107 号 9 栋综合楼　　邮编：519015
电话：0756-6313088　　传真：0756-6313081
网站：www.zh-jieli.com

醒 IO 的接口。

```
/************************ PWR config *****************************/
struct port_wakeup port0 = {
    .pullup_down_enable = ENABLE,                       //配置I/O 内部上下拉是否使能
    .edge       = FALLING_EDGE,                         //唤醒方式选择,可选：上升沿\下降沿
    .attribute  = BLUETOOTH_RESUME,                     //保留参数
    .iomap      = IO_PORTB_01,                          //唤醒口选择
};
```

```
const struct wakeup_param wk_param = {
    .port[1] = &port0,
    .port[2] = &port1,
    .sub = &sub_wkup,
    .charge = &charge_wkup,
};
```

3. 蓝牙状态切换及 sniff

3.1. 概述

影响功耗的原因除了能否进 power_down 外，蓝牙的状态也是一个关键因素。

3.2. 调试

影响系统功耗的另一个原因就是蓝牙状态的切换。蓝牙的状态切换包括了经典蓝牙和 BLE。

经典蓝牙在开启可发现(inquiry_scan)、可连接(page_scan)、回连(page)时功耗会比较高。因此，我们应该在保证不影响经典蓝牙功能的前提下尽量减少这些状态的出现。我们 SDK 上面配置了一个接口函数，主要是用于调整蓝牙空闲状态的时间。其中，1600 为时间参数，时间实际为 1600*0.625ms。因为经典蓝牙处于未连接状态时，一般都在工作状态(可发现(inquiry_scan)、可连接(page_scan)、回连(page))和空闲状态直接切换，因此加大空闲时间，就会会加大每次切花到工作状态的时间间隔，从而达到降低功耗的作用。但需要注意的是如果空闲时间加得过大，每次进工作状态所花的时间就会变长，有可能会影响蓝牙的连接。另外，经典蓝牙在连接成功后，如果没有其他操作，一段时间后会进入 sniff 状态。进入 sniff 状态后，系统的功耗也会进一步降低。

```
/*-------------------------------------------------------------
void bt_function_select_init()
{
    set_idle_period_slot(1600);
```

BLE 影响功耗的主要原因是其广播和通讯的时间间隔。因为在广播或通讯时候，系统会被唤醒。因此要降低功耗，一般都是通过调节广播和通讯的时间间隔来解决。

为了进一步降低蓝牙功耗，在某些特殊应用下面可根据实际情况关闭经典蓝牙，只保留 BLE，当需要使用经典蓝牙的时候，有 BLE 通知设备重新开启经典蓝牙并连接，这也是一直降低功耗的方法。下面

地址：珠海市吉大石花西路 107 号 9 栋综合楼　　邮编：519015
电话：0756-6313088　　传真：0756-6313081
网站：www.zh-jieli.com

是开关经典蓝牙的接口函数。

```
void bt_close_bredr()
{
    if (__this->bt_close_bredr == 1) {
        return;
    }
    __this->bt_close_bredr = 1;
    if (bt_user_priv_var.auto_connection_timer) {
        sys_timeout_del(bt_user_priv_var.auto_connection_timer);
        bt_user_priv_var.auto_connection_timer = 0;
    }
    user_send_cmd_prepare(USER_CTRL_WRITE_SCAN_DISABLE, 0, NULL);
    user_send_cmd_prepare(USER_CTRL_WRITE_CONN_DISABLE, 0, NULL);
    user_send_cmd_prepare(USER_CTRL_PAGE_CANCEL, 0, NULL);
    user_send_cmd_prepare(USER_CTRL_CONNECTION_CANCEL, 0, NULL);
    user_send_cmd_prepare(USER_CTRL_POWER_OFF, 0, NULL);
    user_send_cmd_prepare(USER_CTRL_INQUIRY_CANCEL, 0, NULL);

#if TCFG_USER_TWS_ENABLE
    tws_cancle_all_noconn();
#endif
    sys_auto_sniff_controle(0, NULL);
    btctrler_task_close_bredr();
}

/*-------------------------------------------------------------------------*/
/**@brief   蓝牙   开启bredr
   @param
   @return
   @note
*/
/*-------------------------------------------------------------------------*/
void bt_init_bredr()
{
    if (__this->bt_close_bredr == 0) {
        return;
    }
    __this->bt_close_bredr = 0;
    btctrler_task_init_bredr();
    bt_wait_phone_connect_control(1);
    sys_auto_sniff_controle(1, NULL);
}
```

4. 其他因素

4.1. 概述

影响实际功耗除了上述两个主要因素外，还有其他的次要因素存在。

地址：珠海市吉大石花西路 107 号 9 栋综合楼　　邮编：519015
电话：0756-6313088　　传真：0756-6313081
网站：www.zh-jieli.com

4.2. 调试

影响功耗的次要因素一般都与硬件有关。以 IO 漏电为例，IO 漏电是比较常见的一种引起功耗变大的原因。一般 IO 漏电都是由于外部硬件存在上拉或者下拉，然后 IO 口设置成与之相反的输出状态，两端之间存在压差导致的电流损耗。针对这个问题，需要结合实际的硬件电路和对 IO 的应用状态进行分析和处理。除了 IO 漏电以为，也会存在其他的硬件漏电情况。开发者可先用一个比较干净的硬件环境先对功耗进行一个摸底，然后再上实际样机，对样机功能进行模块拆分，一个一个模块进行分析，从而最终定位到引起功耗变大的原因。下面附上我们测试的一些功耗情况以供参考：

	LDO			DCDC		
	MAX	MIN	AVG	MAX	MIN	AVG
未连蓝牙(bredr)	9.984mA	0.365mA	1.14mA	4.822mA	0.362mA	0.737mA
连接待机(bredr)	3.894mA	0.364mA	0.680mA	2.369mA	0.362mA	0.532mA
ble未连(广播周期1s)	3.755mA	0.187mA	0.328mA	2.121mA	0.213mA	0.285mA
ble连接(500ms通讯一次)	3.833mA	0.187mA	0.370mA	2.054mA	0.213mA	0.300mA
bredr未连+ble未连(广播周期1s)	10.300mA	0.364mA	1.202mA	4.758mA	0.362mA	0.782mA
bredr连接待机+ble未连(广播周期1s)	3.888mA	0.364mA	0.686mA	2.447mA	0.361mA	0.591mA
bredr未连+ble连接(500ms通讯一次)	10.365mA	0.365mA	1.332mA	5.223mA	0.361mA	0.821mA
bredr连接待机+ble连接(500ms通讯一次)	4.013mA	0.365mA	0.953mA	2.481mA	0.359mA	0.667mA

地址：珠海市吉大石花西路 107 号 9 栋综合楼　　邮编：519015
电话：0756-6313088　　传真：0756-6313081
网站：www.zh-jieli.com