XRADIO WLAN Low Power Developer Guide

Revision 1.2

Nov 24, 2019

Declaration

THIS DOCUMENTATION IS THE ORIGINAL WORK AND COPYRIGHTED PROPERTY OF XRADIO TECHNOLOGY ("XRADIO"). REPRODUCTION IN WHOLE OR IN PART MUST OBTAIN THE WRITTEN APPROVAL OF XRADIO AND GIVE CLEAR ACKNOWLEDGEMENT TO THE COPYRIGHT OWNER.

THE PURCHASED PRODUCTS, SERVICES AND FEATURES ARE STIPULATED BY THE CONTRACT MADE BETWEEN XRADIO AND THE CUSTOMER. PLEASE READ THE TERMS AND CONDITIONS OF THE CONTRACT AND RELEVANT INSTRUCTIONS CAREFULLY BEFORE USING, AND FOLLOW THE INSTRUCTIONS IN THIS DOCUMENTATION STRICTLY. XRADIO ASSUMES NO RESPONSIBILITY FOR THE CONSEQUENCES OF IMPROPER USE (INCLUDING BUT NOT LIMITED TO OVERVOLTAGE, OVERCLOCK, OR EXCESSIVE TEMPERATURE).

THE INFORMATION FURNISHED BY XRADIO IS PROVIDED JUST AS A REFERENCE OR TYPICAL APPLICATIONS, ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS DOCUMENT DO NOT CONSTITUTE A WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. XRADIO RESERVES THE RIGHT TO MAKE CHANGES IN CIRCUIT DESIGN AND/OR SPECIFICATIONS AT ANY TIME WITHOUT NOTICE.

NOR FOR ANY INFRINGEMENTS OF PATENTS OR OTHER RIGHTS OF THE THIRD PARTIES WHICH MAY RESULT FROM ITS USE. NO LICENSE IS GRANTED BY IMPLICATION OR OTHERWISE UNDER ANY PATENT OR PATENT RIGHTS OF XRADIO.THIRD PARTY LICENCES MAY BE REQUIRED TO IMPLEMENT THE SOLUTION/PRODUCT. CUSTOMERS SHALL BE SOLELY RESPONSIBLE TO OBTAIN ALL APPROPRIATELY REQUIRED THIRD PARTY LICENCES. XRADIO SHALL NOT BE LIABLE FOR ANY LICENCE FEE OR ROYALTY DUE IN RESPECT OF ANY REQUIRED THIRD PARTY LICENCE. XRADIO SHALL HAVE NO WARRANTY, INDEMNITY OR OTHER OBLIGATIONS WITH RESPECT TO MATTERS COVERED UNDER ANY REQUIRED THIRD PARTY LICENCE.

Revision History

Version	Date	Summary of Changes
1.0	2019-9-13	Initial Version
1.1	2019-11-12	Format Optimization
1.2	2019-11-24	Format Optimization

Table 1- 1 Revision History

Contents

Tables

Figures

1 概述

连接 AP 时的休眠场景是指系统完成了 AP 连接，接着进入 standby 模式的场景。在此场景中，系统仅凭底层的 WLAN 硬件与 AP 保持着连接。

XRADIO SDK 的 WLAN 低功耗模式是指在上述场景的基础上，通过一系列的配置，在完成与 AP 保持连接的条件下，令整个系统的功耗降到最低。

为了达到上述目的，需要对系统的软硬件进行特别设置，这些设置项如下表

表 1-1 WLAN 低功耗模式的设置项

设置项	简述
供电模式	DC-DC 模式比 LDO 模式有较高的转换效率，因此可有较降低功耗。 XR808 系列和 XR872 系列 MCU 均支持外挂 DC-DC 转换芯片。
低频时钟	精准的低频晶振，令 WLAN 在休眠时期也可以与 AP 实现准确的时钟同步，从而降低功耗。 XR808 系列和 XR872 系列 MCU，部分型号支持外接低频晶振，具体请参阅 data sheet。
软件策略	为了达到最低的整体功耗，以些许性能或体验的牺牲为代价，调整 WLAN 的软件策略，减少 WLAN 的休眠期间进行保持 AP 连接的活动，从而降低功耗。这些软件参数有： 1、Dtim Period、Listen Interval：增大接收 Beacon 的间隔，减少接收 Beacon 的次数。 2、Beacon window：减少等待 Beacon 时间，放弃接收较长延迟的 Beacon。 3、Tx Null Period：增加发送心跳帧间隔，减少发送心跳帧次数。 4、Ps Idle Period、Ps Change Period：减少 PS 模式切换时间。

本开发指南的各设置项的操作是基于 XR808ST 评估板（XR808 底板+XR808ST 模组），其中底板原理图为 XR808MD_EVB_IO_V1_0-20190725，模组原理图为 xr808st_md01-20190708。XR808ST 评估板的电流测量点为 V-BAT 管脚，如下图所示：

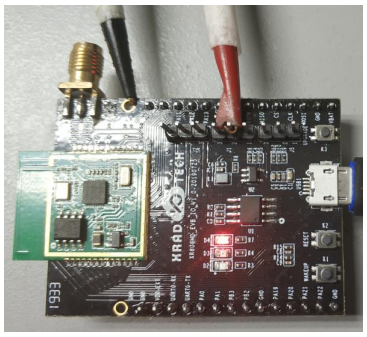

图 1-1 XR808ST 评估板电流测量连接图

关于此评估板的更多信息，请参考 XRadio Wireless MCU Develop Kit 中的产品指导目录的相关文档。

本开发指南相关的示例工程为 wlan_low_power，可在 XRadio SDK 中的 project/example/目录中获取。

2 设置操作

2.1供电模式

XR808/XR872 系列 MCU 默认使用内部的 LDO 电路完成供电，为了降低功耗，可通过外挂 DC-DC 芯片实现 DC-DC 模式供电。下面以 XR808ST 评估板为例，描述如何启用 DC-DC 模式供电。

首先，请参考 XR808ST_MD01（原理图为《xr808st_md01-20190708.pdf》）的电路设计，完成 DC-DC 芯片连接设计，如下图所示：

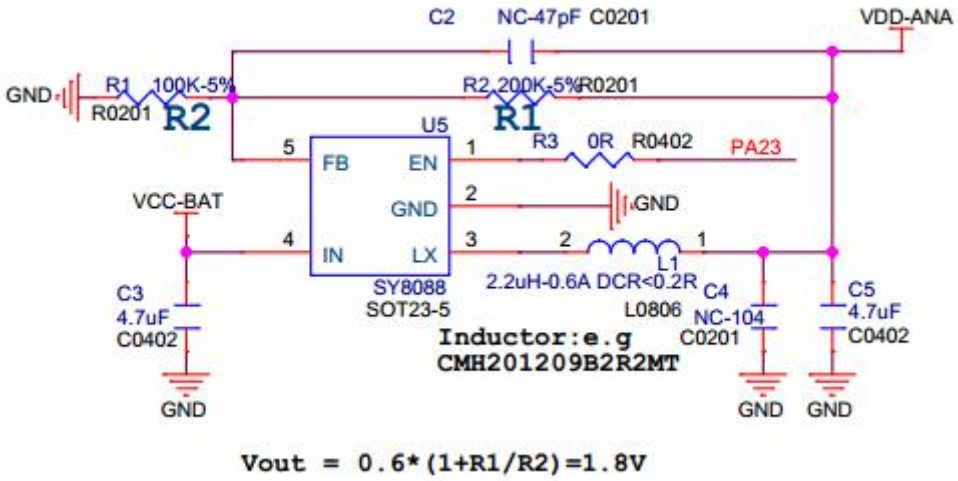

图 2-1 XR808ST MD01 模组中外挂 DCDC 转换芯片 SY8088 的原理图

在实际的 PCB 上如下图所示：

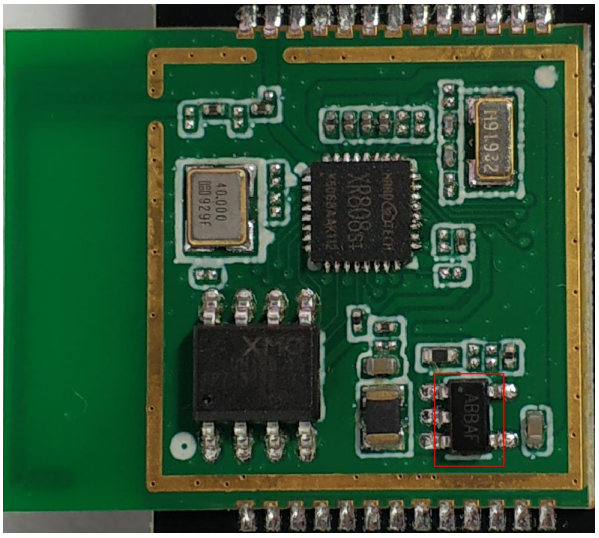

图 2-2 XR808ST MD01 模组中 DCDC 转换芯片 SY8088（红框）的实物图

其中建议使用 pin 脚 PA23 完成 DC-DC 转换芯片的控制，因为 PA23 管脚针对外挂 DC-DC 模式进行了优化，

如 XR808 系列芯片的 PinMux 表所示：

表 2-1 XR808 系列 MCU 的 PinMux

Function	IO Name	Func0	Func1	Func2	Func3	Func4	Func5	Func6	Func7	XR808ST QFN32，4x4-0.4	XR808CT QFN32，4x4-0.4
GPIOA	PA00	I	O	/	/	TWI1_SCL	/	EINTA0	Hi	1	
	PA01	I	O	/	/	TWI1_SDA	/	EINTA1	Hi	1	
	PA10	I	O	ADC_CH0	PWM2/ECT2	/	/	EINTA10	Hi	1	
	PA11	I	O	ADC_CH1	PWM3/ECT3	/	/	EINTA11	Hi	1	
	PA12	I	O	ADC_CH2	PWM4/ECT4	/		EINTA12	Hi	1	
	PA13	I	O	ADC_CH3	PWM5/ECT5	/	UART1_TX	EINTA13	Hi	1	
	PA14	I	O	ADC_CH4	PWM6/ECT6	/	UART1_RX	EINTA14	Hi	1	
	PA19/WUPIO5	I	O	UART2_RTS	TWI0_SCL	PWM0/ECT0	SPI1_MOSI	EINTA19	Hi	1	1
	PA20/WUPIO6	I	O	UART2_CTS	TWI0_SDA	PWM1/ECT1	SPI1_MISO	EINTA20	Hi	1	1
	PA21/WUPIO7	I	O	UART2_RX	/	PWM2/ECT2	SPI1_CLK	EINTA21	Hi	1	1
	PA22/WUPIO8	I	O	UART2_TX	/	PWM3/ECT3	SPI1_CS0	EINTA22	Hi	1	1
	PA23/WUPIO9/TEST	I	O	EXT_DCDC_PUP	/	/	/	EINTA23	Hi	1	1
GPIOB	PB00	I	O	UART0_TX	JTAG_TMS	PWM4/ECT4	SWD_TMS	EINTB0	Hi	1	1
	PB01	I	O	UART0_RX	JTAG_TCK	PWM5/ECT5	SWD_TCK	EINTB1	Hi	1	1
	PB02	I	O	SWD_TMS	JTAG_TD0	PWM6/ECT6	/	EINTB2	Hi	1	1
	PB03	I	O	SWD_TCK	JTAG_TDI	PWM7/ECT7	/	EINTB3	Hi	1	1
	PB04	I	O	/	/	UART1_TX	FLASH_MOSI/IO0	EINTB4	Hi	1	1
	PB05	I	O	/	/	UART1_RX	FLASH_MISO/IO1	EINTB5	Hi	1	1
	PB06	I	O	/	/	UART1_CTS	FLASH_CS	EINTB6	Hi	1	1
	PB07	I	O	/	/	UART1_RTS	FLASH_CLK	EINTB7	Hi	1	1
RESET#	CHIP_PWD									1	1
CLOCK	HXTAL1									1	1
	HXTAL2									1	1
	LXTAL1									1	
	LXTAL2									1	
ANT	ANT									1	1
POWER	VBAT									1	1
	VDD_ANA									1	1
	VDD_DIG									1	1
	VDD_IO									2	2
	VDD_EXT									1	1
GND	GND(N+1)									1	1

确认外部电路正确设计和实现后，需要通过软件设置完成 DC-DC 芯片的启用。软件设置有两项：1、使能 PA23 管脚的 EXT_DCDC_PUP 功能。2、设置内部 LDO 输出电压，令其小于外部 DC-DC 芯片的输出，此场景下内部电路会将供电由内部 LDO 切换到外部 DC-DC。

设置 PA23 管脚的示例代码如下，可在 wlan_low_power 示例工程中查看此代码的完整调用。

```
#include "driver/chip/hal_gpio.h"

void pa23_pup_example(void)
{
    char port;
    uint32_t pin, mode, pull, level;
    GPIO_InitParam param;

    port = GPIO_PORT_A;
    pin = GPIO_PIN_23;
    mode = GPIOA_P23_F2_DCXO_PUP_OUT;
    pull = GPIO_PULL_NONE;
    level = GPIO_DRIVING_LEVEL_1;
```

```
    param.mode = mode;
    param.driving = level;
    param.pull = pull;

    HAL_GPIO_Init(port, pin, &param);
    return;
}
```

设置内部 LDO 电路输出电压的示例代码如下，其中 XR808ST 评估板中的 DC-DC 转换芯片 SY8088 的输出电压为 1.8v，因此将内部 LDO 电路的输出电压设置为 1.4v 即可，可在 wlan_low_power 示例工程中查看此代码的完整调用。

```
#include "driver/chip/hal_prcm.h"

void ldo_set_low_volt_example(void)
{
    HAL_PRCM_SetTOPLDOVoltage(PRCM_TOPLDO_VOLT_1V4);
    return;
}
```

2.2低频时钟

为获得精准的低频时钟用于 WLAN 休眠时也可以实现和 AP 间的准确同步，XR808/XR872 的部分 MCU 型号支持外接低频晶振。具体信息请参考 XRadio Wireless MCU Develop Kit 中的芯片资料目录的相关 data sheet。下面以 XR808ST 评估板为例，描述如何启用外接的低频晶振。

首先，请参考 XR808ST_MD01（原理图为《xr808st_md01-20190708.pdf》）的电路设计，完成低频晶振芯片连接设计，如下图所示：

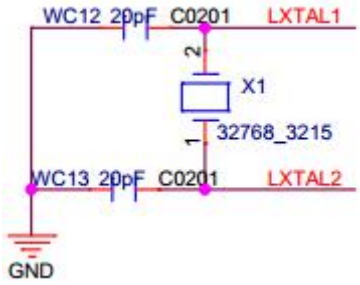

图 2-3 XR808ST MD01 模组中外挂 32K 低频晶振的原理图

在实际的 PCB 上如下图所示：

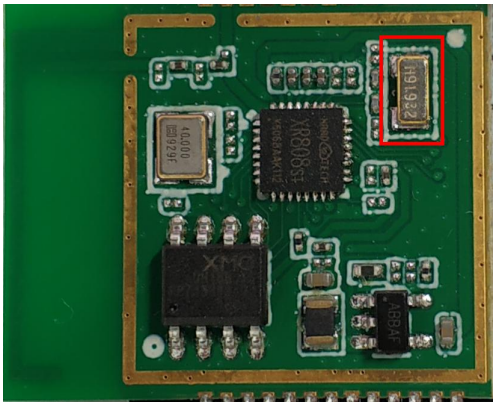

图 2-4 XR808ST MD01 模组中外挂 32K 低频晶振（红框）的实物图

确认外部电路正确设计和实现后，需要通过软件设置完成外部低频晶振芯片的启用。软件设置有两项：1、启用外部低频晶振。2、关闭内部低频时钟校准。

上述软件操作在 XRadio SDK 中的实现如下所示：

```
#if (defined(BOARD_LOSC_EXTERNAL) && BOARD_LOSC_EXTERNAL)
    HAL_PRCM_SetLFCLKSource(PRCM_LFCLK_SRC_EXT32K);
    HAL_PRCM_DisableInter32KCalib();
#else
    HAL_PRCM_SetLFCLKSource(PRCM_LFCLK_SRC_INTER32K);
    HAL_PRCM_EnableInter32KCalib();
#endif
```

在实际使用上，通过修改工程相关的板级配置文件 board_config.h 完成设置。设置外部低频晶振的示例代码

如下，可在 wlan_low_power 示例工程中查看此代码的完整调用。

```
/*
 * board configuration
 */

/* chip clock */
#define BOARD_LOSC_EXTERNAL    1   /* 0: inter 32k, 1: external 32k */
```

此外，为了降低功耗，可同时在 board_config.h 中将相关芯片的 MCU 工作频率调低，如设置为 160MHz。设置 MCU 工作频率的示例代码如下，可在 wlan_low_power 示例工程中查看此代码的完整调用。

```
/*
 * board configuration
 */

/* chip clock */
#if defined(__CONFIG_CHIP_XR872)
#define BOARD_CPU_CLK_FACTOR    PRCM_SYS_CLK_FACTOR_160M
#elif defined(__CONFIG_CHIP_XR808)
#define BOARD_CPU_CLK_FACTOR    PRCM_SYS_CLK_FACTOR_160M
#endif
```

2.3软件策略

2.3.1 减少接收 Beacon 次数

通过设置 XRadio SDK 中 Dtim Period、Listen Interval 参数，可增大接收 Beacon 的间隔，减少接收 Beacon 的次数，单位为 Beacon 周期。

在 XRadio SDK 中的 Dtim Period 参数用于控制接收 Beacon 的最小 Beacon 周期数，Listen Interval 参数用于控制接收 Beacon 的最大 Beacon 周期数，若 Listen Interval 的值为 0，则接收 Beacon 的周期数等于 Dtim Period 参数。假设 Beacon 周期为 100ms，举例如下：

1、Dtim Period 参数设置为 1，Listen Interval 参数设置 8：WLAN 根据负载，唤醒时间间隔在 100ms 到 800ms 之间动态调整。

2、Dtim Period 参数设置为 3，Listen Interval 参数设置 0：WLAN 的唤醒时间间隔为 300ms。

Dtim Period、Listen Interval 参数仅在 standby 场景有效，在 active 场景下，WLAN 的唤醒时间会根据 WLAN 协议的 Dtim Period 域和 Listen Interval 域进行设置，具体含义请查阅 IEEE802.11 标准。

设置 XRadio SDK 中 Dtim Period、Listen Interval 参数的示例代码如下，可在 wlan_low_power 示例工程中查看此代码的完整调用。

```
#include "common/framework/net_ctrl.h"
#include "net/wlan/wlan_ext_req.h"
#include "net/wlan/wlan_defs.h"

void wlan_set_dtim_period(void)
{
    int ret;
    uint32_t period;

    period = 1;

    ret = wlan_ext_request(g_wlan_netif,
        WLAN_EXT_CMD_SET_PM_DTIM, period);

    if (ret == -2) {
        printf("%s: invalid arg\n", __func__);
        return;
    } else if (ret == -1) {
        printf("%s: exec failed\n", __func__);
        return;
    }

    return;
}

void wlan_set_listen_interval(void)
{
    int ret;
    uint32_t listen_interval;
```

```
    listen_interval = 8;

    ret = wlan_ext_request(g_wlan_netif,
        WLAN_EXT_CMD_SET_LISTEN_INTERVAL, listen_interval);

    if (ret == -2) {
        printf("%s: invalid arg\n", __func__);
        return;
    } else if (ret == -1) {
        printf("%s: exec failed\n", __func__);
        return;
    }

    return;
}
```

2.3.2 减少等待 Beacon 时间

通过设置 XRadio SDK 中 Beacon Window 参数，可减少等待 Beacon 的时间，单位为 us。

在 XRadio SDK 中的 Beacon Window 参数用于 WLAN 在唤醒后等待接收 Beacon 帧的时长。

在实际的工作过程中，因为干扰和信道的分时复用机制，AP 的 Beacon 帧并不一定会准确地在约定的时刻点发送出来，而更多的是在约定的时刻延后若干时间进行发送。如果延后的时间较长，WLAN 就会消耗较多的能耗。

在进行 AP 的 Beacon 帧发送延迟统计后，经过综合考量，可通过 Beacon Window 参数减少等待 Beacon 时间，以减少能耗。

进行 AP 的 Beacon 帧发送延迟统计的步骤如下：

1、完成 AP 的连接。

2、设置 WLAN 为 ACTIVE 模式

3、读取一次 Beacon 延迟计数器，目的是清零。

4、等待若干时长作为统计周期。

5、读取 Beacon 延迟计数器，完成上述统计周期的 Beacon 延迟统计。

在 XRadio SDK 读取 Beacon 帧发送延迟值的示例代码如下，其中包含了 WLAN ACTIVE 模式的设置，即 PS Mode 参数需要设置为 0，可在 wlan_low_power 示例工程中查看此代码的完整调用。

```
#include "common/framework/net_ctrl.h"
#include "net/wlan/wlan_ext_req.h"
#include "net/wlan/wlan_defs.h"

void wlan_set_active_mode_example(void)
{
    int ret;
    uint32_t ps_mode, ps_ip, ps_cp;
    wlan_ext_ps_cfg_t ps_cfg;
```

```
    ps_mode = 0;
    ps_ip = 0;
    ps_cp = 0;

    memset(&ps_cfg, 0, sizeof(wlan_ext_ps_cfg_t));
    ps_cfg.ps_mode = ps_mode;
    ps_cfg.ps_idle_period = ps_ip;
    ps_cfg.ps_change_period = ps_cp;
    ret = wlan_ext_request(g_wlan_netif,
        WLAN_EXT_CMD_SET_PS_CFG, (uint32_t)&ps_cfg);

    if (ret == -2) {
        printf("%s: invalid arg\n", __func__);
        return;
    } else if (ret == -1) {
        printf("%s: exec failed\n", __func__);
        return;
    }

    return;
}

void wlan_get_beacon_stat_example(void)
{
    int ret, i;
    unsigned int sum_cnt = 0;
    int sum_avg = 0;
    wlan_ext_bcn_status_t bcn_status;

    char dly_info[][20] = {
        "(<0     )",
        "(<500us  )",
        "(<1000us )",
        "(<2000us )",
        "(<4000us )",
        "(<8000us )",
        "(<16000us)",
        "(>16000us)",
    };

    memset(&bcn_status, 0, sizeof(wlan_ext_bcn_status_t));
    ret = wlan_ext_request(g_wlan_netif,
        WLAN_EXT_CMD_GET_BCN_STATUS, (uint32_t)&bcn_status);

    if (ret == -2) {
        printf("%s: invalid arg\n", __func__);
        return;
    } else if (ret == -1) {
        printf("%s: exec failed\n", __func__);
        return;
    }

    printf("\nAPP AP Beacon Delay Stat Info================\n");
    for (i = 0; i < 8; i++) {
        printf("cnt %d %s: %d\n",
            i, dly_info[i],
```

```
            bcn_status.bcn_delay_cnt[i]);
        sum_cnt += bcn_status.bcn_delay_cnt[i];
    }

    if (sum_cnt)
        sum_avg = bcn_status.bcn_delay_sum / sum_cnt;

    printf("beacon duration: %d, rxed beacon: %d, missed beacon: %d\n",
        bcn_status.bcn_duration,
        bcn_status.bcn_rx_cnt,
        bcn_status.bcn_miss_cnt);
    printf("beacon delay stat: max %d us, avg %d us, sum %d us, cnt %d\n",
        bcn_status.bcn_delay_max,
        sum_avg, bcn_status.bcn_delay_sum, sum_cnt
        );
    printf("APP AP Beacon Delay Stat Info=================\n");

    return;
}
```

获取了 AP 的 Beacon 延迟统计并权衡后，设置 Beacon Window 参数。

设置 XRadio SDK 中 Beacon Window 参数的示例代码如下，可在 wlan_low_power 示例工程中查看此代码的完整调用。

```
#include "common/framework/net_ctrl.h"
#include "net/wlan/wlan_ext_req.h"
#include "net/wlan/wlan_defs.h"

void wlan_set_beacon_win(void)
{
    int ret;
    int bcn_win;

    bcn_win = 2300;

    ret = wlan_ext_request(g_wlan_netif,
        WLAN_EXT_CMD_SET_BCN_WIN_US, bcn_win);

    if (ret == -2) {
        printf("%s: invalid arg\n", __func__);
        return;
    } else if (ret == -1) {
        printf("%s: exec failed\n", __func__);
        return;
    }

    return;
}
```

2.3.3 减少发送心跳帧次数

通过设置 XRadio SDK 中 Tx Null Period 参数，可增加发送心跳帧的发送间隔，减少发送次数，单位为 s。

在 XRadio SDK 中的 Tx Null Period 参数用于设置为保持与 AP 连接，WLAN 需要进行心跳帧发送的间隔。

在实际的实践中，若在一定的时长内，STA 与 AP 之间没有帧交互，那么 AP 就会检查 STA 是否已经离开，有些 AP 甚至会直接踢掉 STA。

AP 检查 STA 是否离开，会进行较多的帧交换，若 STA 在空闲一段时间后发送空帧即 NULL 帧，则可以避免上述的检查帧交互。在 XRadio SDK 中，这种 Null 帧称为心跳帧。

不同 AP 对帧交换空闲的容忍程度是不一样的，一般没有说明，但可通过以下方式获知：

1、连接 AP 后，对 STA 设备断电，并通过空口抓包记录最后一次帧交互时刻。

2、通过空口抓包查看，AP 向此 STA 发送 deauth 帧的时刻，此时即为 AP 踢掉 STA 的时刻。

3、对比上述两个时刻，即可获取 AP 对帧交换空闲的容忍程度。

在获知容忍程度后，经过权衡，设置 Tx Null Period 参数。

设置 XRadio SDK 中 Tx Null Period 参数的示例代码如下，建议值为 48 秒，可在 wlan_low_power 示例工程中查看此代码的完整调用。

```
#include "common/framework/net_ctrl.h"
#include "net/wlan/wlan_ext_req.h"
#include "net/wlan/wlan_defs.h"

void wlan_set_tx_null_period(void)
{
    int ret;
    int period;

    period = 48;

    ret = wlan_ext_request(g_wlan_netif,
        WLAN_EXT_CMD_SET_PM_TX_NULL_PERIOD, period);

    if (ret == -2) {
        printf("%s: invalid arg\n", __func__);
        return;
    } else if (ret == -1) {
        printf("%s: exec failed\n", __func__);
        return;
    }

    return;
}
```

2.3.4 减少 PS 模式切换时间

通过设置 XRadio SDK 中 Ps Idle Period、Ps Change Period 参数，可减少 PS 模式切换时间，单位为 ms。

在 XRadio SDK 中的 Ps Idle Period、Ps Change Period 参数用于设置 PS 模式各状态的切换时间，在设置时 PS Mode 参数需要同时设置为 1 才生效。

IEEE802.11 标准中并没定义 STA 切换到 PS 模式后，AP 是否接受此切换的标志动作。因此，在 STA 的实际的实践中，会设置若干等待时间用于判断 AP 是否接受 STA 的 PS 模式切换。默认 100ms 的等待时间，对功耗有较大负面影响，因此可考虑以帧交互延迟的增大为成本，减少 PS 模式切换等待时间，以减少功耗。

设置 XRadio SDK 中 Ps Idle Period、Ps Change Period 参数的示例代码如下，建议设置值为 10ms 或更大，其中包含了 WLAN PS 模式的设置，即 PS Mode 参数需要设置为 1，可在 wlan_low_power 示例工程中查看此代码的完整调用。

```
#include "common/framework/net_ctrl.h"
#include "net/wlan/wlan_ext_req.h"
#include "net/wlan/wlan_defs.h"

void wlan_set_ps_exchange_time_example(void)
{
    int ret;
    uint32_t ps_mode, ps_ip, ps_cp;
    wlan_ext_ps_cfg_t ps_cfg;

    ps_mode = 1;
    ps_ip = 10;
    ps_cp = 10;

    memset(&ps_cfg, 0, sizeof(wlan_ext_ps_cfg_t));
    ps_cfg.ps_mode = ps_mode;
    ps_cfg.ps_idle_period = ps_ip;
    ps_cfg.ps_change_period = ps_cp;
    ret = wlan_ext_request(g_wlan_netif,
        WLAN_EXT_CMD_SET_PS_CFG, (uint32_t)&ps_cfg);

    if (ret == -2) {
        printf("%s: invalid arg\n", __func__);
        return;
    } else if (ret == -1) {
        printf("%s: exec failed\n", __func__);
        return;
    }

    return;
}
```

3 常见问题

问题：在 XR808ST 评估板中运行 wlan_low_power 示例，在 DC-DC 模式下的功耗与预期不符

答：按如下步骤检查：

1.在示例完成，提示并等待用户输入命令时，使用电压表确认 VDD-ANA 是否为 1.8v。

如果不是 1.8v，那么 SY8088 工作异常，需要进行 SY8088 电路完整性检查。如果为 1.8v，那么 SY8088 工作正常，此时可能为 XR808ST 工作异常，检查内部 LDO 模式是否设置输出为 1.4v。

2.确认 XR808ST 模组上的 SY8088 的相关器件完整。

如果不完整，则是 SY8088 的外围电路可能未焊好，检查原理图和 PCB 图，修复电路。如果完整，那么可能是 SY8088 的使能管脚 PA23 未设置正常，检查 PA23 的输出是否为 3.3v，若不是 3.3v，检查 GPIO 设置代码。

若以上设置均正常，请咨询 FAE

4 参考文档

以下文档位于 XRadio Wireless MCU Develop Kit 中。

1、《xr808st_md01-20190708.pdf》：XR808ST 模组原理图

2、《XR808MD_EVB_IO_V1_0-20190725.pdf》：XR808 底板原理图

3、《XR808_Datasheet》

4、《XR872_Datasheet》

5、《XR808_PIN_Multiplexing_V1_0_20190726.pdf》

6、《XR872_PIN_Multiplexing_V1_0_20190726.pdf》

7、《XRADIO_Quick_Start_Guide-CN.pdf》：XRadio SDK 快速上手指南