WiFi保持连接状态下低功耗的实现

——适用于ESP32/ESP32C3/ESP32S3系列模组

版本1.0.0

启明云端

一、介绍

ESP32/ESP32C3/ESP32S3 系列模组都有三种低功耗模式：

- Modem-sleep 模式：CPU 可运行，时钟频率可配置。Wi-Fi 及 Bluetooth LE 的基带和射频关闭，但 Wi-Fi 或 Bluetooth LE 可保持连接。
- Light-sleep 模式：CPU 暂停运行。任何唤醒事件（MAC、主机、RTC 定时器或外部中断）都会唤醒芯片。Wi-Fi 或 Bluetooth LE 可保持连接。
- Deep-sleep 模式：CPU 和大部分外设都会掉电，只有 RTC 存储器处于工作状态。Wi-Fi 连接数据存储在 RTC 中。RTC 时钟定时器或 RTC GPIO 可以将芯片从 Deep-sleep 模式中唤醒。

总结：Modem-sleep 和 Light-sleep 两种模式下可以保持 Wi-Fi 或 Bluetooth LE 。（详细功耗数据见规格书）

不同功耗模式下的功耗有所不同，下图以 ESP32 为例，其他型号请见技术规格书：

功耗模式	描述			功耗
Modem-sleep	CPU 处于工作状态	240 MHz *	双核芯片	30 mA ~ 68 mA
			单核芯片	N/A
		160 MHz *	双核芯片	27 mA ~ 44 mA
			单核芯片	27 mA ~ 34 mA
		正常速度：80 MHz	双核芯片	20 mA ~ 31 mA
			单核芯片	20 mA ~ 25 mA
Light-sleep	-			0.8 mA
Deep-sleep	ULP 协处理器处于工作状态			150 μA
	超低功耗传感器监测方式			100 μA @1% duty
	RTC 定时器 + RTC 存储器			10 μA

二、硬件准备

ESP32/ESP32C3/ESP32S3 系列模组既支持外置 32.768 kHz 的时钟振荡器作为 RTC 睡眠时钟，也支持外 部激励信号（如有源晶振）作为 RTC 睡眠时钟。外置 32.768 kHz 晶振的电路如图 6 所示。

32.768 kHz 晶振选择要求：

- 等效内阻 (ESR) ⩽ 70 KΩ。
- 两端负载电容值根据晶振的规格要求进行配置。
- 并联电阻 R18 用于偏置晶振电路，电阻值要求 5 MΩ < R18 ⩽ 10 MΩ。
- ESP32-D0WD-V3 外接 32.768 kHz 晶振时，并联的电阻必须上件；ESP32 系列其他芯片建议预留。

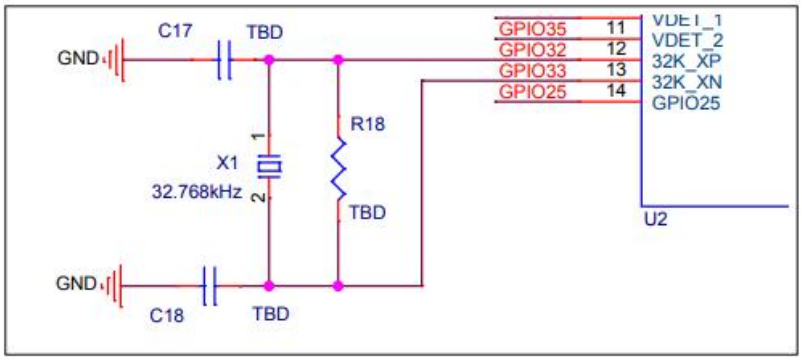


图 6: 外置晶振电路图

三、软件部分

完整示例代码（基于 IDF4.4 版本）： https://github.com/espressif/esp-idf/tree/master/examples/wifi/power_save

3.1、目标芯片选择

esp32 系列模组：

```
idf.py set-target esp32
```

esp32c3 系列模组：

```
idf.py set-target esp32c3
```

esp32s3 系列模组：

```
idf.py set-target esp32s3
```

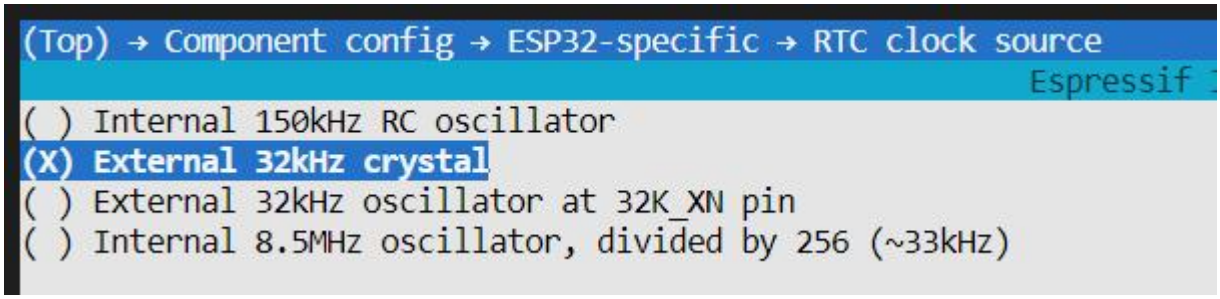
3.2、menuconfig 配置项

在工程目录下运行

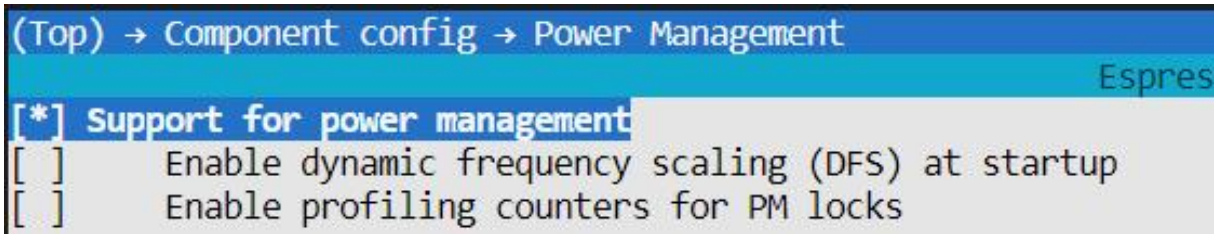
```
idf.py menuconfig
```

设置操作说明

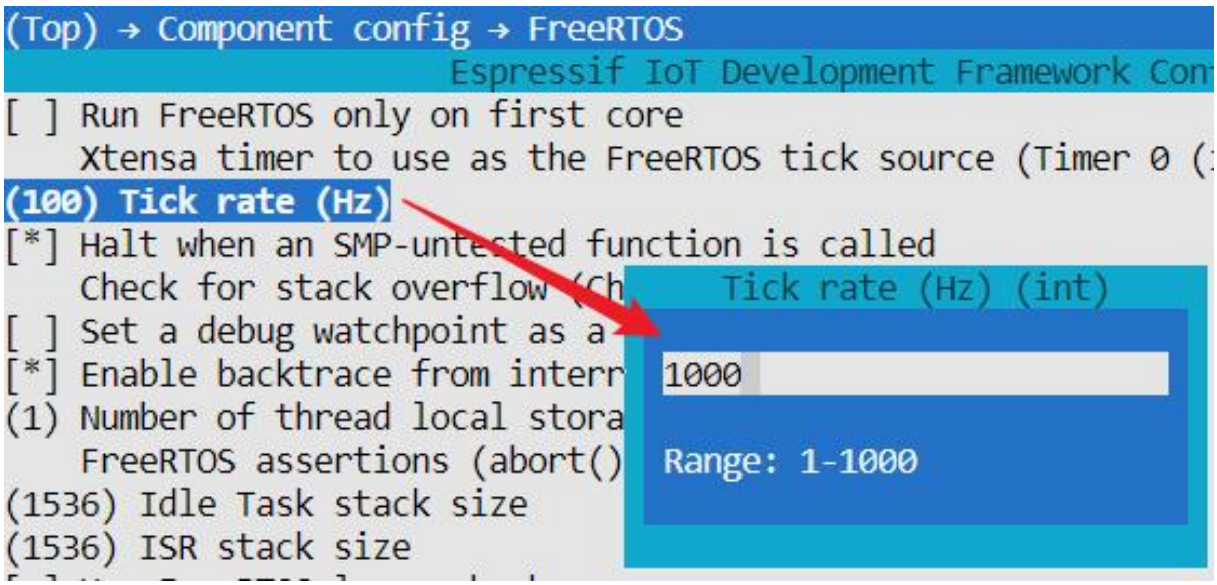
Component config → ESP32-specific/ESP32C3-specific/ESP32S3-specific → RTC clock source → External 32kHz crystal



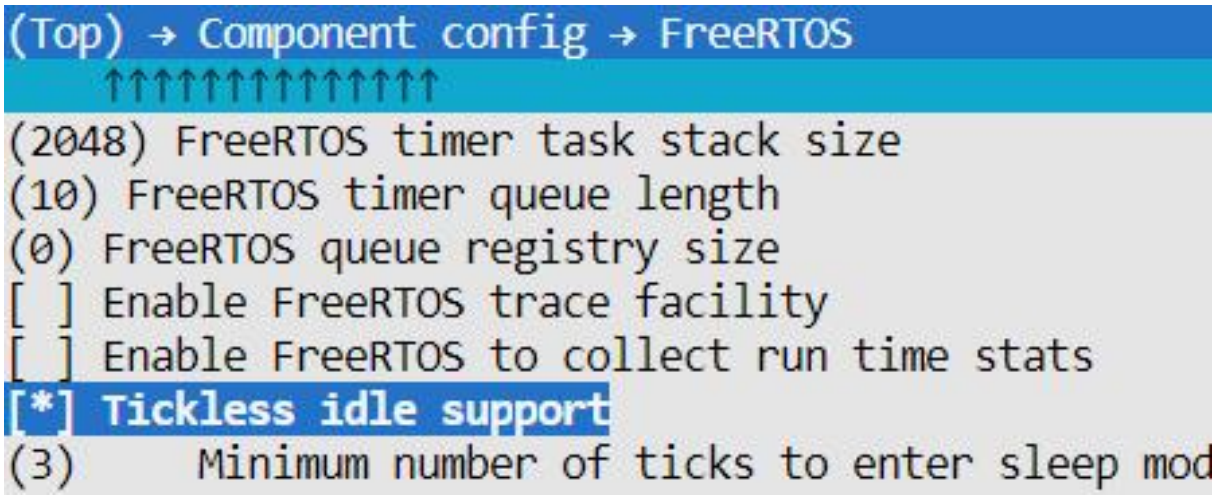
Component config → Power Management → Support for power management



Component config → FreeRTOS → Tick rate (Hz) 改为 1000



Component config → FreeRTOS → Tickless idle support



3.3、源码解析

3.3.1、电源管理初始化

```
#if CONFIG_PM_ENABLE
    // Configure dynamic frequency scaling:
    // maximum and minimum frequencies are set in sdkconfig,
    // automatic light sleep is enabled if tickless idle support is enabled.
#if CONFIG_IDF_TARGET_ESP32
    esp_pm_config_esp32_t pm_config = {
#elif CONFIG_IDF_TARGET_ESP32S2
    esp_pm_config_esp32s2_t pm_config = {
#elif CONFIG_IDF_TARGET_ESP32C3
    esp_pm_config_esp32c3_t pm_config = {
#elif CONFIG_IDF_TARGET_ESP32S3
    esp_pm_config_esp32s3_t pm_config = {
#endif
            .max_freq_mhz = CONFIG_EXAMPLE_MAX_CPU_FREQ_MHZ,
            .min_freq_mhz = CONFIG_EXAMPLE_MIN_CPU_FREQ_MHZ,
#if CONFIG_FREERTOS_USE_TICKLESS_IDLE
            .light_sleep_enable = true
#endif
    };
    ESP_ERROR_CHECK( esp_pm_configure(&pm_config) );
#endif // CONFIG_PM_ENABLE
```

最大 CPU 频率 (MHz)

最小 CPU 频率 (MHz)

使能light_sleep

3.3.2、设置 Listen_Interval

ESP32 的节能模式一共有三种类型：modem 最小节能模式、modem 最大节能模式、以及不节能模式。

- modem 最小节能模式：该模式为默认模式。在该模式下，ESP32 从 Light-sleep 中醒来收 beacon 的时间间隔由路由器端的 DTIM 决定，为 (DTIM * 102.4) ms，即假如路由器的 DTIM 为 1，则每 隔 100 ms ESP32 会醒来进行一次收包。
- modem 最大节能模式：在该模式下，ESP32 从 Light-sleep 中醒来收 beacon 的时间间隔由 wifi_sta_config_t 这个结构体中的 listen_interval 参数决定，为 (listen interval * 102.4) ms，即假如路由器的 DTIM 为 1，而 listen_interval = 10，则每隔 1 s ESP32 会醒来进行一次收 包。
- 不节能模式：不进行节能处理。

```
/*init wifi as sta and set power save mode*/
static void wifi_power_save(void)
{
    ESP_ERROR_CHECK(esp_netif_init());
    ESP_ERROR_CHECK(esp_event_loop_create_default());
    esp_netif_t *sta_netif = esp_netif_create_default_wifi_sta();
    assert(sta_netif);

    wifi_init_config_t cfg = WIFI_INIT_CONFIG_DEFAULT();
    ESP_ERROR_CHECK(esp_wifi_init(&cfg));

    ESP_ERROR_CHECK(esp_event_handler_instance_register(WIFI_EVENT, ESP_EVENT_ANY_ID, &event_handler, NULL, NULL));
    ESP_ERROR_CHECK(esp_event_handler_instance_register(IP_EVENT, IP_EVENT_STA_GOT_IP, &event_handler, NULL, NULL));

    wifi_config_t wifi_config = {
        .sta = {
            .ssid = "test",
            .password = "123456789",
            .listen_interval = 10,
        },
    };
    ESP_ERROR_CHECK(esp_wifi_set_mode(WIFI_MODE_STA));
    ESP_ERROR_CHECK(esp_wifi_set_config(WIFI_IF_STA, &wifi_config));
    ESP_ERROR_CHECK(esp_wifi_start());

    ESP_LOGI(TAG, "esp_wifi_set_ps().");
    esp_wifi_set_ps(DEFAULT_PS_MODE);
}
```

wifi 省电模式下的四个基本概念：

1. **TIM**（traffic indication message）

   每一个 Beacon 的帧中都有一个 TIM 信息元素 ，它主要用来由 AP 通告它管辖下的哪个 STA 有信息现在缓存在 AP 中，而在 TIM 中包含一个 Bitmap control 字段，它允许您传输 1 个字节到 TIM 的全部 251 个字节（2008 位），每一位映射一个 STA，当为 1 时表示该位对应的 STA 有信息在 AP 中。总而言之，收到与自己关联的 TIM 就要发送 PS-POLL 帧来与 AP 取来联系并取得它 的缓存帧了。标准的 TIM 中仅仅指示 AP 缓存的单播信息。

2. **DTIM**(Delivery Traffic Indication Message）

   这个是 TIM 的特殊情况，当发送几个 TIM 之后，就要发送一个 DTIM，其除了缓存单播信息，也 同时指示 AP 缓存的组播或广播信息，一旦 AP 发送了 DTIM, STA 就必须处于清醒，因为广播或 组播无重发机制，不醒来数据就收不到了。 这个值不会影响单播的流量传递，如果没有开启 PS 的用户使用组播也不会收到影响，但是会影响开启了 PS 的用户接收多播数据的传递，如果设置的太小，起不到节电作用，太大又可能 会影响组播通讯的质量，这个过程是一个 trial-error 的调整过程，只能一个一个测试调整，以 达到最佳，即既可以达到最佳节电效果又不影响应用。

   调整方法如下： 1.设置 DTIM 为 1，然后记录应用效果，作为基线。 2.然后提高间隔，使应用效果可以接受为止。

   DTIM=1 表示每个 beacon 中都包含 DTIM， DTIM=2 表示每两个 beacon 中包含一个 DTIM，以此类推。

3. **Beacon-Interval**(信标间隔)

   Beacon-Interval 越大，有助于 client 端省电。

   Beacon-Interval 越小，有助于提高 client 端连接速度。降低基地台的 buffer frame 负载。一般预设为 100mS。以 Beacons 封包发送 SSID 的速率是 1Mbit/S。

4. **Listen-Interval**，(STA 即 Client 接收 Beacon的周期)

   AP 广播 Beacon 的周期为 Beacon-Interval，STA 可以自由选择 Beacon-Interval 的整数倍作为自己的 Listen-Interval，比如 10。

   STA 每隔 Listen-Interval 接收 Beacon 并解码其中的 TIM，如果 TIM 指示没有数据缓存，STA 就可以立刻转入 Doze 状态，如果 TIM 指示其有数据缓存，STA 就要向 AP 发一个竞选控制包Poll，AP 在收到 Poll 后就可以向该 Poll 的源 STA 发送一个为它缓存的数据包。

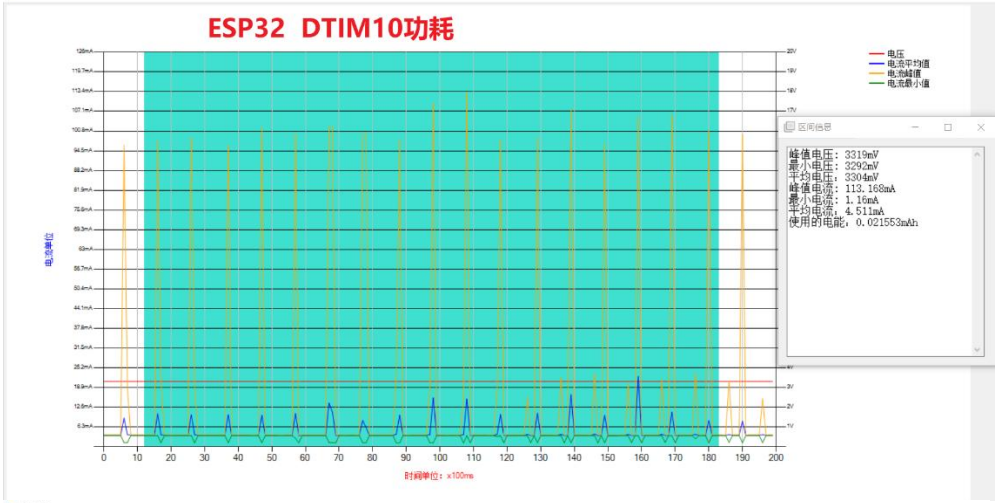
四、功耗测试

以下数据外挂 32.768K 晶振在屏蔽房测试所得

系列模组	ESP32	ESP32C3	ESP32S2	ESP32S3
DTIM10 功耗	4.5ma	1.4ma	3.6ma	4.4ma

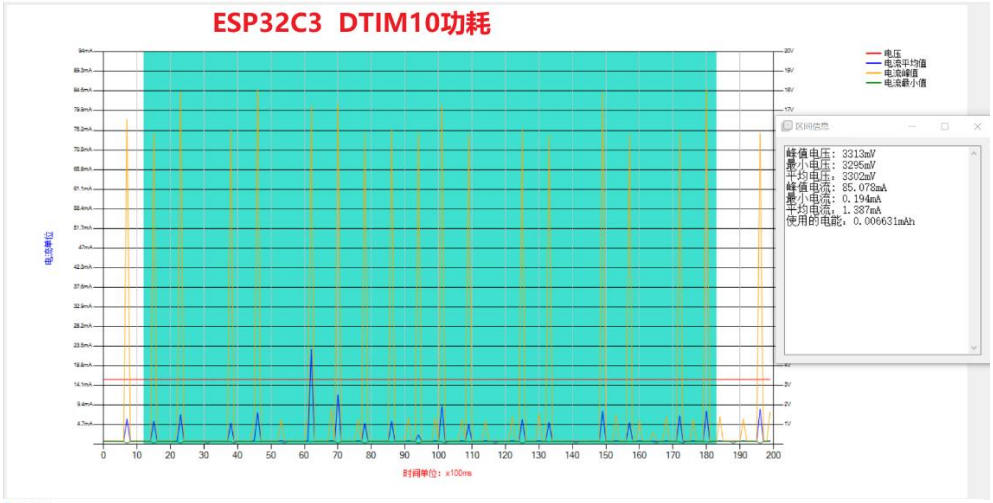
数据分析

ESP32 DTIM10功耗



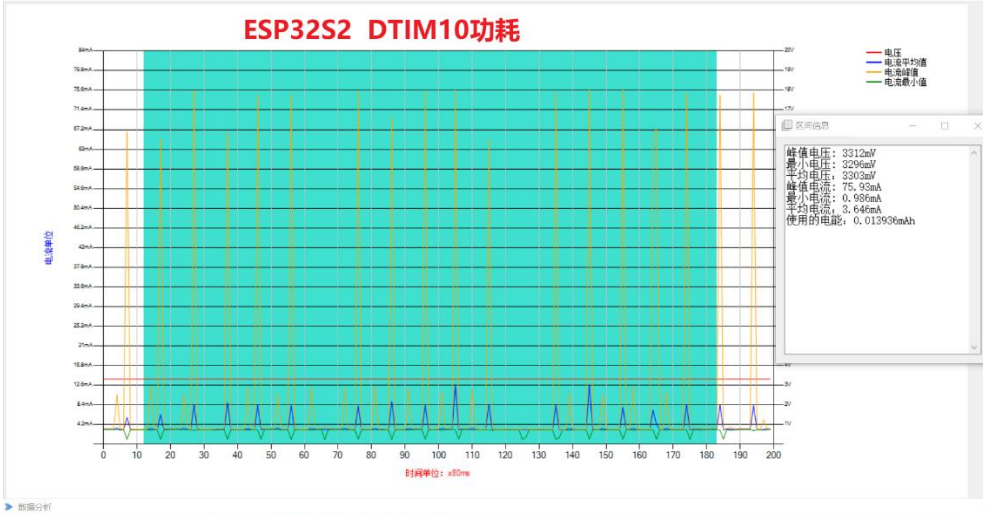
数据分析

ESP32C3 DTIM10功耗



数据分析

ESP32S2 DTIM10功耗



数据分析

ESP32S3 DTIM10功耗