ESP8266 Sleep Mode & Low-Power Solutions

Version 1.1

About This Guide

This document introduces ESP8266 sleep mode and low-power solutions with the following topics:

Chapter	Title	Subject
Chapter 1	Overview	Provides an overview of ESP8266 sleep mode.
Chapter 2	Modem-sleep	Introduces the features, interface and applications of Modem-sleep.
Chapter 3	Light-sleep	Introduces the features, interface, external wake-up and applications of Light-sleep.
Chapter 4	Deep-sleep	Introduces the features, interface, external wake-up and applications of Deep-sleep.
Chapter 5	Low-power Solutions	Provides some low-power solutions to further lower power consumption.

Release Notes

Date	Version	Release notes
2015.06	V1.0	First release.
2016.04	V1.1	Added Chapter 5.

Table of Contents

1. Overview

ESP8266 series chip provides the following 3 configurable sleep modes. Users can choose from and configure them according to specific needs.

- Modem-sleep
- Light-sleep
- Deep-sleep

Table 1-1 shows the differences between the 3 sleep modes.

Table1-1. Differences between 3 Sleep Modes

Item		Modem-sleep	Light-sleep	Deep-sleep
Wi-Fi		OFF	OFF	OFF
System Clock		ON	OFF	OFF
RTC		ON	ON	ON
CPU		ON	Pending	OFF
Substrate current		15 mA	0.4 mA	~ 20 μA
Average current	DTIM = 1	16.2 mA	1.8 mA	-
	DTIM = 3	15.4 mA	0.9 mA	
	DTIM = 10	15.2 mA	0.55 mA	

Notes:

- For Modem-sleep and Light-sleep modes, SDK provides interfaces to enable sleep mode, and the system decides when to go into sleep. For details, please refer to 2. Modem-sleep and 3. Light-sleep.
- In Deep-sleep mode, you can call the function to control when to go into sleep. For the details, please refer to 4. Deep-sleep.
- RTC (Real-Time Clock).
- DTIM (Delivery Traffic Indication Message).

2. Modem-sleep

2.1. Features

Modem-sleep mode is enabled only when ESP8266 works in station mode and connects to the router. ESP8266 stays connected to the router through the DTIM beacon mechanism.

> **Note:**
>
> The DTIM beacon interval of the router is usually from 100ms to 1000ms.

In Modem-sleep mode, ESP8266 will close the Wi-Fi module circuit between the two DTIM Beacon intervals in order to save power., ESP8266 will be automatically waken up before the next Beacon arrival. The sleep time is decided by the DTIM Beacon of the router. During sleep, ESP8266 can stay connected to the Wi-Fi and receive the interactive information from a mobile phone or server.

2.2. Interface

The system will go into Modem-sleep via the following interface.

```
wifi_set_sleep_type(MODEM_SLEEP_T)
```

> **Note:**
>
> In Modem-sleep, the system can be waken up automatically. Users don't need to configure the interface.

2.3. Application

Modem-sleep is generally used in the applications that need the CPU powered on. An example of the applications is Pulse Width Modulation (PWM) light that needs real-time CPU control.

3. Light-sleep

3.1. Features

The working mode of Light-sleep is similar to that of Modem-sleep. The difference is that, during light-sleep mode, except from Wi-Fi circuit, ESP8266 also powers off clock and suspends internal CPU, as such it costs less power than in Modem-sleep mode.

3.2. Interface

The system will go into Light-sleep mode via the following interface.

```
wifi_set_sleep_type(LIGHT_SLEEP_T)
```

> **Note:**
>
> ESP8266 automatically enters Light-sleep mode when connected to Wi-Fi with the CPU idle.

3.3. External Wake-up

During Light-sleep, the CPU will not respond to the signals and interrupts from the peripheral hardware interfaces. Therefore, ESP8266 needs to be waken up via external GPIO. The waking process is less than 3 ms.

The GPIO wakeup function can only be configured as level triggered. The interface is as follows.

```
void gpio_pin_wakeup_enable(uint32 i, GPIO_INT_TYPE intr_state);
```

uint32 i	The IO serial number of the wake-up function.
GPIO_INT_TYPE intr_state	The trigger mode of wake-up. • GPIO_PIN_INTR_LOLEVEL • GPIO_PIN_INTR_HILEVEL

3.4. Application

Light-sleep mode can be used in the applications that can respond to the sending data from the router in real time. And the CPU can be idle before receiving command. An example is the Wi-Fi switch whose CPU is idle for most of the time and carries out GPIO operations until receiving the control commands.

> **Note:**
>
> If a task interval is shorter than the DTIM beacon interval, the system can not go into Light-sleep mode.

4. Deep-sleep

4.1. Features

Compared to the other two modes, Deep-sleep is controlled by users. Users can call the interface function to immediately enable Deep-sleep. In this mode, the chip will turn off Wi-Fi connectivity and data connection. Only the RTC module is still working, responsible for periodic wake-ups.

To enable Deep-sleep, users need to connect GPIO16 to the EXT_RSTB pin of ESP8266.

4.2. Interface

4.2.1. Go into Deep-sleep

Go into Deep-sleep mode via the following interface.

```
void system_deep_sleep(uint32 time_in_us)
```

Parameters:

uint32 time_in_us = 0	The chip won't wake up at regular intervals.
uint32 time_in_us ≠ 0	The chip will automatically wake up at regular intervals (unit: μs).

4.2.2. Configure Deep-sleep

The software workflow of the wakeup from Deep-sleep can be configured by the following interface to affect the average power consumption during long-time running.

```
bool system_deep_sleep_set_option(uint8 option)
```

deep_sleep_set_option(0)	The 108th byte of `esp_init_data_default.bin` decides whether the chip will make RF calibration after waking up from Deep-sleep.
deep_sleep_set_option(1)	The chip will make RF calibration after waking up from Deep-sleep with higher power consumption.
deep_sleep_set_option(2)	The chip won't make RF calibration after waking up from Deep-sleep with lower power consumption.
deep_sleep_set_option(4)	The chip won't turn on RF after waking up from Deep-sleep with the lowest power consumption as Modem-sleep costs.

Note:

The `init` parameter is the parameter in `esp_init_data_default.bin`. For example, to change the 108th byte of the data to 8, and call `deep_sleep_set_option (0)` means that the chip will undertake RF calibration every 8 times of wake-up from Deep-sleep. For the details please go to: http://bbs.espressif.com/viewtopic.php?f=5&t=272.

4.3. External Wake-up

In the deep-sleep mode, the chip can be waken up and initialized by a low-level pulse generated on the EXT_RSTB pin via an external IO.

⚠ **Notice:**

If the automatic wake-up and the external wake-up need to be enabled at the same time, users need to use the appropriate line logic operation circuit during the external circuit design.

4.4. Application

Deep-sleep can be used in low-power sensor applications or the applications that do not need data transmission for most of the time. The device wakes up from Deep-sleep state at intervals to measure and upload data, and then goes to Deep-sleep again. The device can also store data in the RTC memory (which can still save data in Deep-sleep mode) and then send it at a time.

5. Low-power Solutions

5.1. Low-power Solutions in Deep-sleep Mode

If the application needs to wake up and enter Deep-sleep mode frequently, users can consider the following low-power solutions to further reduce power consumption.

Users can apply the low-power solutions in this application scenario: the chip enters Deep-sleep mode without Wi-Fi connectivity and data connection. Only RTC is working, responsible for waking up the chip at intervals. The power consumption during sleep is 20μA. The chip will enable RF functionality and send information to the server after waking up and then goes into Deep-sleep again.

To reduce power consumption in the above process is to reduce wake-up time and power consumption. We provide the following 8 solutions for users to choose.

1. Set the chip to enter Deep-sleep mode instantly so as to reduce the time taken when it actually enters Deep-sleep mode.

The function `system_deep_sleep_instant` is not defined externally, but can be called directly. Definition of the function is shown as below:

```
void system_deep_sleep_instant(uint32 time_in_us)
```

Sample code:

```
// Deep-sleep for 5 seconds, and then wake up

system_deep_sleep_instant(5000*1000);
```

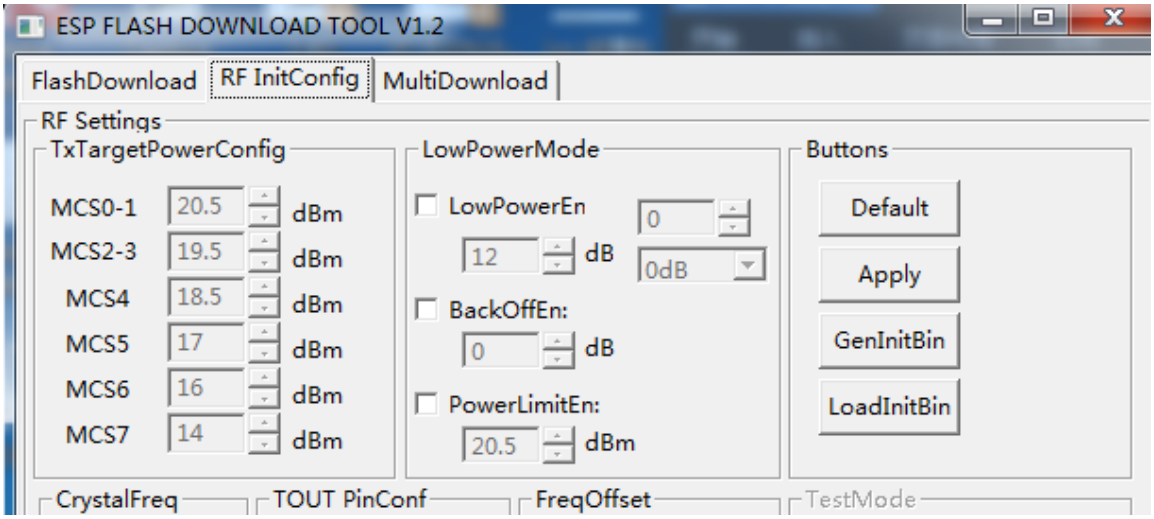
2. Turn off RF calibration when the chip is waken up from Deep-sleep so as to reduce the initialization time and current the chip consumes.

```
system_deep_sleep_set_option(2);
```

3. Reduce RF power consumption.

If the application does not require a high peak value of Tx, users can reduce the RF power consumption.

Please make sure you are using the Flash Download Tool V1.2 or higher. In the tool, RF InitConfig can be used to modify RF power consumption. Please replace esp_init_data_default.bin with the newly generated bin file esp_init_data_setting.bin.



4. Modify the bin file in python to reduce the initialization time and current the flash consumes.

Download add_low-power_deepsleep_cmd.py here: http://bbs.espressif.com/viewtopic.php?f=57&p=4783#p4783

Modify the bin file by executing the following command, then burn the modified bin file into the flash.

```
python add_low-power_deepsleep_cmd.py ./bin file
```

Note:

The bin file should be replaced by actual firmware such as eagle.flash.bin or user.bin.

5. Select flash type and its working mode.

Choosing the right flash can greatly reduce the firmware uploading time. ISSI-IS25LQ025 is a good choice. Besides, the appropriate working mode of flash can also reduce firmware uploading time. We recommend four-line working mode.

6. A FIFO (First In First Out) is a UART buffer that forces each byte of your serial communication to be passed on in the order received. To reduce time consumption, too much information printing should be avoided. Therefore, all UART FIFO should be erased before the chip enters Deep-sleep mode, otherwise the system will not go into Deep-sleep mode until all UART FIFO information has been printed out.

```
SET_PERI_REG_MASK(UART_CONF0(0), UART_TXFIFO_RST);//RESET FIFO
CLEAR_PERI_REG_MASK(UART_CONF0(0), UART_TXFIFO_RST);
```

7. Synchronous data transmission.

Data transmission consumes less time and power than the device wake-up. We recommend sending multiple data packets at a time when ESP8266 is waken up from Deep-sleep mode.

8. esp_iot_sdk_v1.4.0, esp_iot_rtos_sdk_v1.3.0 and the later versions of SDKs have largely optimized the power consumption capability. Please make sure that the SDK you are using is up to date.

Notes:

- By following the above-mentioned instructions, users can reduce ESP8266's power consumption in Deep-sleep mode. This can also be identified when ESP8266 enters Light-sleep mode during which the Wi-Fi Modem circuit is turned off and CPU is suspended. The initialization process consumes less time when ESP8266 is waken up from Light-sleep mode.
- The test results indicate that if the application sleeps less than 2 seconds, then Light-sleep mode is preferred; if the application sleeps more than 2 seconds, then Deep-sleep mode is recommended.

5.2. Other Low-power Solutions

Apart from the above-mentioned low-power solutions for Deep-sleep mode, we also provide other low-power solutions. For example, users can call the forced sleep interface or mandatorily close the RF circuit to lower power consumption.

⚠ Notice:

When forced sleep interface is called, the chip will not enter sleep mode instantly, but when the system is executing idle task. Please refer to the below sample codes.

Example one: Modem-sleep Mode

```
#define FPM_SLEEP_MAX_TIME    0xFFFFFFF

wifi_station_disconnect();
wifi_set_opmode(NULL_MODE);               // set WiFi mode to null
mode
wifi_fpm_set_sleep_type(MODEM_SLEEP_T);   // set modem sleep
wifi_fpm_open();                      // enable force sleep
wifi_fpm_do_sleep(FPM_SLEEP_MAX_TIME);
...
wifi_fpm_do_wakeup();                     // wake up to use WiFi again

wifi_fpm_close();                         // disable force sleep
wifi_set_opmode(STATION_MODE);            //set station mode
wifi_station_connect();                   //connect to AP
```

Example two: Light-sleep Mode

```
      void fpm_wakup_cb_func1(void)
{
   wifi_fpm_close();                      // disable force sleep
function
   wifi_set_opmode(STATION_MODE);         // set station mode
```

```
    wifi_station_connect();                 // connect to AP
}

    void user_func(...)
{
    wifi_station_disconnect();
    wifi_set_opmode(NULL_MODE);                  // set WiFi mode to
null mode.
    wifi_fpm_set_sleep_type(LIGHT_SLEEP_T); // light sleep
    wifi_fpm_open();                        // enable force sleep
    wifi_fpm_set_wakeup_cb(fpm_wakup_cb_func1);    // Set wakeup
callback
    wifi_fpm_do_sleep(10*1000);
    ...
}
```

Espressif IOT Team

www.espressif.com

Disclaimer and Copyright Notice

Information in this document, including URL references, is subject to change without notice.

THIS DOCUMENT IS PROVIDED AS IS WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE.

All liability, including liability for infringement of any proprietary rights, relating to use of information in this document is disclaimed. No licenses express or implied, by estoppel or otherwise, to any intellectual property rights are granted herein.

The Wi-Fi Alliance Member logo is a trademark of the Wi-Fi Alliance. The Bluetooth logo is a registered trademark of Bluetooth SIG.

All trade names, trademarks and registered trademarks mentioned in this document are property of their respective owners, and are hereby acknowledged.