ESP8266
Low Power Solutions

Version 1.1

Copyright © 2016

About This Guide

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

Chapter	Title	Subject
Chapter 1	Overview	An overview of ESP8266 sleep mode.
Chapter 2	Modem-sleep	Introduction to the features, interface and applications of Modem-sleep.
Chapter 3	Light-sleep	Introduction to the features, interface, external wake-up and applications of Light-sleep.
Chapter 4	Deep-sleep	Introduction to the features, interface, external wake-up, applications of and low power solutions for Deep-sleep.

Release Notes

Date	Version	Release notes
2015.06	V1.0	First release.
2016.04	V1.1	Added Section 4.5 Low Power Solutions.

Table of Contents

1. Overview 1

2. Modem-sleep 2
   - 2.1. Features 2
   - 2.2. Interface 2
     - 2.2.1. Auto Modem-sleep 2
     - 2.2.2. Forced Modem-sleep 2
   - 2.3. Application 3

3. Light-sleep 4
   - 3.1. Features 4
   - 3.2. Interface 4
     - 3.2.1. Auto Light-sleep 4
     - 3.2.2. Forced Light-sleep 4
   - 3.3. External Wake-up 4
   - 3.4. Application 5

4. Deep-sleep 6
   - 4.1. Features 6
   - 4.2. Interface 6
     - 4.2.1. Enable Deep-sleep 6
     - 4.2.2. Configure Deep-sleep 6
   - 4.3. External Wake-up 7
   - 4.4. Application 7
   - 4.5. Low Power Solutions 7

1. Overview

ESP8266 series chip provides 3 configurable sleep modes. We also provide some low power solutions related to these sleep modes. Users can choose and configure the sleep mode as required. The 3 sleep modes are:

- 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:***

- *SDK provides interfaces to enable Modem-sleep and Light-sleep modes, and the system bottom layer decides when to go into sleep. For details, please refer to* ***Chapter 2. Modem-sleep*** *and* ***Chapter 3. Light-sleep****.*
- *Deep-sleep mode is controlled by users. Users can call the function to enable Deep-sleep instantly. For details, please refer to* ***Chapter 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 connects to the router in station mode. ESP8266 stays connected to the router through the DTIM beacon mechanism.

> 📖 ***Note:***
>
> *The DTIM beacon interval of the router is usually 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 woken up before the next Beacon arrival. The sleep time is decided by the DTIM Beacon interval time 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

2.2.1. Auto Modem-sleep

The system goes into Modem-sleep via the following interface.

```
wifi_set_sleep_type(MODEM_SLEEP_T)
```

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

2.2.2. Forced Modem-sleep

Users can mandatorily enable Modem-sleep by calling force sleep APIs and turning off RF. For details on force sleep APIs, please refer to ***Section 3.7 Force Sleep APIs*** in ***ESP8266 Non-OS SDK API Reference*** and ***Section 4.12 Force Sleep APIs*** in ***ESP8266 RTOS SDK API Reference***. You can download at: http://bbs.espressif.com/viewtopic.php?f=51&t=1023.

> ⚠️ ***Notice:***
>
> *The system will not enter sleep mode instantly upon force sleep APIs are called, but only until executing idle task.*

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, resulting in less power than in Modem-sleep mode.

3.2. Interface

3.2.1. Auto Light-sleep

The system goes 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.*

3.2.2. Forced Light-sleep

Users can mandatorily enable Light-sleep by calling force sleep APIs and turning off RF. For details on force sleep APIs, please refer to ***Section 3.7 Force Sleep APIs*** in ***ESP8266 Non-OS SDK API Reference*** and ***Section 4.12 Force Sleep APIs*** in ***ESP8266 RTOS SDK API Reference***. You can download at: http://bbs.espressif.com/viewtopic.php?f=51&t=1023.

> ⚠️ ***Notice:***
>
> *The system will not enter sleep mode instantly upon force sleep APIs are called, but only until executing idle task.*

3.3. External Wake-up

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

The GPIO wake-up function can only be enabled by level triggers. 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 scenarios where the applications need to stay connected to the router and can respond to the sending data from the router in real time. The CPU can be idle before receiving commands. An example is the Wi-Fi switch whose CPU is idle for most of the time and only performs GPIO operations until receiving the control commands.

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

4. Deep-sleep

4.1. Features

Unlike the other two modes, the system cannot go into Deep-sleep automatically. Users can call the interface function `system_deep_sleep` 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.

> ⚠ ***Notice***:
>
> *To enable Deep-sleep, you need to connect GPIO16 to the EXT_RSTB pin of ESP8266.*

4.2. Interface

4.2.1. Enable Deep-sleep

The system goes 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 be woken up at regular intervals, i.e., won't wake up automatically.
uint32 time_in_us ≠ 0	The chip will be automatically woken up at regular intervals (unit: μs).

4.2.2. Configure Deep-sleep

Users can configure the software workflow during the wake-up from Deep-sleep via the following interface to adjust 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 `init` parameter decides whether RF calibration will be performed after the chip wakes up from Deep-sleep.
deep_sleep_set_option(1)	The chip will make RF calibration after waking up from Deep-sleep. Power consumption is high.
deep_sleep_set_option(2)	The chip won't make RF calibration after waking up from Deep-sleep. Power consumption is low.
deep_sleep_set_option(4)	The chip won't turn on RF after waking up from Deep-sleep. Power consumption is the lowest, same as in Modem-sleep.

> 📖 ***Note:***
>
> *The* `init` *parameter is the parameter value 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 ESP8266 low-power sensor example application, please refer to:* *https://github.com/EspressifSystems/low_power_voltage_measurement*.

4.3. External Wake-up

In Deep-sleep mode, the chip can be woken 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 are to be enabled at the same time, users need the right line logic operation circuit when designing the external circuit.*

4.4. Application

Deep-sleep can be used in low-power sensor applications or in the scenarios where data transmission is not required for most of the time. The device wakes up from Deep-sleep 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.

4.5. Low Power Solutions

We provide the following 8 solutions to reduce power consumption in Deep-sleep as follows:

1. Set the device to enter Deep-sleep mode instantly to reduce the time for it to actually enters Deep-sleep.

Function:

```
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);
```

> 📖 ***Note:***
>
> *The function* `system_deep_sleep_instant` *is undeclared, but can be called directly.*

2. Call the following function so that the chip will not perform RF calibration after waking up from Deep-sleep to reduce the initialization time and current consumption.

```
system_deep_sleep_set_option(2);
```

3. Reduce RF power consumption.

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

Please make sure you are using the ESP8266 Download Tool V1.2 or higher. In the ***RF InitConfig*** tab you can 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 binaries with the following command to reduce the time and current consumption during flash initialization.

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

If you are using FOTA firmware, modify ***boot_v1.5.bin*** and download the generated ***boot_v15_low_power.bin*** to the address `0x0`.

If you are using Non-FOTA firmware, modify ***eagle.flash.bin*** and download the generated ***eagle.flash_low_power.bin*** to the address `0x0`.

> 📖 ***Note:***
>
> *You can download **Add_Low-power_Cmd** at: http://www.espressif.com/en/support/download/other-tools.*

5. Select flash type and the working mode.

The right flash can greatly reduce the firmware loading time, e.g., ISSI-IS25LQ025. The appropriate working mode of flash can also reduce firmware loading time. We recommend four-line working mode.

6. Clear up UART FIFO to reduce printing time.

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 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 than the device wake-up does and low power. We recommend sending multiple data packets at a time when ESP8266 is woken up from Deep-sleep mode.

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

> 📖 ***Note:***
>
> *Due to the low time consumption (<3ms) during wake-up from Light-sleep mode, if the application sleeps less than 2 seconds, then Light-sleep mode is preferred to save power; if the application sleeps more than 2 seconds, then Deep-sleep mode is recommended.*

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.

Copyright © 2016 Espressif Inc. All rights reserved.