

OPL1000

ULTRA-LOW POWER 2.4GHZ WI-FI + BLUETOOTH SMART SOC

低功耗解决方案



OPULINKS

<http://www.opulinks.com/>

Copyright © 2017-2018, Opulinks. All Rights Reserved.

OPL1000-PowerSave | Version 0.1

Date	Version	Contents Updated
2018-07-02	0.1	<ul style="list-style-type: none">Initial Release

TABLE OF CONTENTS

1. 介紹 1

1.1. 文檔應用範圍 1

1.2. 縮略語 1

1.3. 參考文獻 1

2. 概述 2

3. Smart-SLEEP 4

3.1. 特性 4

3.2. AT 命令接口說明 4

3.2.1. 啟用 Smart-Sleep 4

3.3. API 接口說明 5

3.4. 外部喚醒 6

3.5. 应用 6

4. Timer-Sleep 7

4.1. 特性 7

4.2. AT 命令接口說明 7

4.2.1.1. 自動休眠 7

4.3. API 接口說明 8

4.4. 外部喚醒 9

4.5. 应用 9

5. Deep-sleep 10

5.1. AT 命令接口說明 10

5.1.1. 使能 Deep-Sleep 10

5.2. API 接口說明 10

5.3. 外部喚醒 11

5.4. 应用 11

1. 介紹

1.1. 文檔應用範圍

低功耗解決方案用於 OPL1000 芯片的省電功能。本文介紹了低功耗的一些解決方案，讓用戶可以根據不同的情境之下，來選擇一個適合用戶的低功耗方案，進而達到省電的效果。

1.2. 縮略語

Abbr.	Explanation
BLE	Bluetooth Low Energy
API	Application Programming Interface
DTIM	Delivery Traffic Indication Message
AT	Attention 終端命令指令集

1.3. 參考文獻

[1] [OPL1000-AT-instruction-set-and-examples.pdf](#)

2. 概述

OPL1000 系列芯片提供三种可配置的睡眠模式，针对这些睡眠模式，我们提供了多种低功耗解决方案，用户可以结合具体需求选择睡眠模式并进行配置。芯片支持的三种睡眠模式如下：

- Smart-sleep
- Timer-sleep
- Deep-sleep

三种模式的区别如表 2-1 所示。

表 2-1 三种睡眠模式比较

項目		Smart-sleep		Timer-sleep		Deep-sleep
		自動	強制	自動	強制	強制
Wi-Fi 连接		保持	断连	保持	断连	断连
GPIO 状态		O	O	O	O	O
Wi-Fi		V	X	X	X	X
系统时钟		V	V	V	V	X
RTC		V	V	V	V	X
CPU		X	X	X	X	X
衬底电流						
平均電流	DTIM =1			X	X	X
	DTIM =3			X	X	X
	DTIM =10			X	X	X
BLE 連線	100ms			X	X	X
	500ms			X	X	X
	1000ms			X	X	X



	4000ms			X	X	X
BLE 廣播	20ms			X	X	X
	10s			X	X	X

3. SMART-SLEEP

3.1. 特性

目前 OPL1000 的 Smart-Sleep 仅工作在 Station 模式下，於 WIFI 系統中是由连接路由器后生效，OPL1000 並通过 Wi-Fi 的 DTIM 机制与路由器保持连接。

說明

一般 WIFI 路由器的 Beacon 间隔为 100 ms ~ 1,000 ms，DTIM 為 1。

後續章節將說明可經由軟件提供的跳過 DTIM (skip DTIM) 功能達到更省電的操作。

當有下列情況時，可以使用此功能

- Wi-Fi 已連線
- Wi-Fi 掃描中
- BLE 已連線
- BLE 廣播

在 Smart-Sleep 模式下，OPL1000 WIFI 系統本身會自動調整兩次 DTIM Beacon 间隔时间的接收長短，关闭或開啟 Wi-Fi 模块电路，达到省电效果。在時間快到達的下次 Beacon 到来前自动唤醒，是透過 32K RTC 的振盪器來實現。睡眠同时可以保持与路由器的 Wi-Fi 连接，并通过路由器接受来自手机或者服务器的交互信息。

3.2. AT 命令接口說明

3.2.1. 啟用 Smart-Sleep

系統通過以下 AT 指令進入 Smart-Sleep 模式。

```
AT+SLEEP <mode>,<ext_io>
```

参数说明:

mode	0: 關閉 smart sleep 1: 啟用 smart sleep
ext_io	喚醒的 IO 埠號

3.3. API 接口說明

啟用 **Smart Sleep**，系統在連線期間，並且在閒置的狀態時，系統會自動的進入睡眠模式。

Smart Sleep 持續會運作，直到外部的觸發喚醒而中止。

```
void ps_smart_sleep(int enable);
```

参数说明:

int enable	啟用 Smart Sleep。
------------	-----------------

可以通過以下 **API** 接口，設定外部的輸入埠號，來達到喚醒。

```
void ps_set_wakeup_io(E_GpiIdx_t ext_io_num, E_ItrType_t ext_io_type);
```

参数说明:

E_GpiIdx_t ext_io_num	喚醒功能的 IO 序號
E_ItrType_t ext_io_type	喚醒的觸發模式

用戶可以自行定義，當系統被喚醒之後，會做那些動作。

```
void ps_set_wakeup_cb(PS_WAKEUP_CALLBACK callback);
```


参数说明:

PS_WAKEUP_CALLBACK
callback

用戶可以自行定義的 callback 函式。

3.4. 外部喚醒

在 **Smart-Sleep** 模式下，CPU 在暫停狀態下不會響應來自外圍硬件接口的信號與中斷，因此需要通過外部 **GPIO** 信號將 **OPL1000** 喚醒，硬件喚醒過程大約為 **1 ms**。

3.5. 應用

Smart-Sleep 可以用於低功耗的傳感器應用，或者大部分時間都不需要進行數據傳輸的情況之下。

例如，當 **BLE (Bluetooth Low Energy)** 正在廣播，之後想讓 **BLE** 進入休眠模式，可以使用 **Smart-Sleep** 的 **AT** 指令或 **API** 來控制實現，休眠的同時也可以做配對的動作，當有需要喚醒傳感器時可以配合 **GPIO** 腳位來控制喚醒。

4. TIMER-SLEEP

4.1. 特性

有下列情況時，皆不可以使用。

- Wi-Fi 已連線
- Wi-Fi 掃描中
- BLE 已連線
- BLE 廣播

系統無法自動進入 **Timer-Sleep**，需要由用戶調用 **AT** 指令或是於代碼中呼叫 **API** 來控制。在該模式下，芯片會斷開所有 **Wi-Fi** 連接與數據連接，進入睡眠模式，只有 **RTC** 模塊仍然工作，負責芯片的定時喚醒。

4.2. AT 命令接口說明

4.2.1.1. 自動休眠

系統通過以下 **AT** 指令進入 **Timer-Sleep** 模式。

```
AT+SLEEP <mode>,<sleep_duration>,<ext_io>
```

參數說明:

mode	2: 使用 timer sleep
Sleep_duration	睡眠週期，單位 millisecond
ext_io	喚醒的 IO 埠號



說明

在 **Timer-Sleep** 模式下，系統可以自動被喚醒。

4.3. API 接口說明

啟用 **Timer Sleep**，系統會進入睡眠模式，直到外部的觸發喚醒，或者 **Timer** 時間終止。

```
void ps_timer_sleep(uint32_t sleep_duration_ms);
```

参数说明:

uint32_t sleep_duration_ms	睡眠到喚醒的時間長度，單位為 millisecond.
-------------------------------	-----------------------------

可以通過以下 **API** 接口，設定外部的輸入埠號，來達到喚醒。

```
void ps_set_wakeup_io(E_Gpioldx_t ext_io_num, E_ltrType_t ext_io_type);
```

参数说明:

E_Gpioldx_t ext_io_num	喚醒功能的 IO 序號
E_ltrType_t ext_io_type	喚醒的觸發模式

用戶可以自行定義，當系統被喚醒之後，會做那些動作。

```
void ps_set_wakeup_cb(PS_WAKEUP_CALLBACK callback);
```

参数说明:

PS_WAKEUP_CALLBACK callback	用戶可以自行定義的 callback 函式。
--------------------------------	------------------------

4.4. 外部喚醒

在 **Timer-sleep** 模式下，**CPU** 在暫停狀態下不會響應來自外圍硬件接口的信號與中斷，因此需要通過外部 **GPIO** 信號將 **OPL1000** 喚醒，硬件喚醒過程大約為 **1 ms**。

4.5. 應用

當客戶清楚知道，應用本身會有多久的時間間隔，可以使用 **Timer-Sleep** 來實現休眠模式。

例如，傳感器需要每五分鐘傳遞資料時，可以使用 **Timer-Sleep** 來實現。使用 **Timer-Sleep** 會讓傳感器固定五分鐘喚醒，偵測資料並傳送資料到雲端，隨後又進入睡眠模式。

5. DEEP-SLEEP

相對於 IC 的 Timer-sleep 模式，系統無法自動進入 Deep-sleep，需要由用戶調用函式接口來控制。在該模式下，芯片會斷開所有 Wi-Fi 連結與數據連結，進入睡眠模式，RTC 模塊也沒有動作，只能透過外部的 GPIO 來喚醒芯片。

5.1. AT 命令接口說明

5.1.1. 使能 Deep-Sleep

系統通過以下 AT 指令進入 Deep-Sleep 模式。

```
AT+SLEEP <mode>,<ext_io>
```

参数说明:

mode	3: 啟用 deep sleep
ext_io	喚醒的 IO 埠號

5.2. API 接口說明

啟用 Deep Sleep，系統會進入睡眠模式，直到外部的觸發喚醒。

```
void ps_deep_sleep(void);
```

可以通過以下 API 接口，設定外部的輸入埠號，來達到喚醒。

```
void ps_set_wakeup_io(E_GpiIdx_t ext_io_num, E_ltrType_t ext_io_type);
```

参数说明:

E_GpiIdx_t ext_io_num	喚醒功能的 IO 序號
--------------------------	-------------

E_itrType_t
ext_io_type

喚醒的觸發模式

用戶可以自行定義，當系統被喚醒之後，會做那些動作。

```
void ps_set_wakeup_cb(PS_WAKEUP_CALLBACK callback);
```

参数说明:

PS_WAKEUP_CALLBACK
callback

用戶可以自行定義的 callback 函式。

5.3. 外部喚醒

在 **Deep-Sleep** 模式下，CPU 在暫停狀態下不會響應來自外圍硬件接口的信號與中斷，因此需要通過外部 **GPIO** 信號將 **OPL1000** 喚醒，硬件喚醒過程大約為 **1 ms**。當喚醒之後，整個流程是從 **cold-boot** 的初始流程開始進行。

5.4. 應用

當客戶清楚知道，應用本身只有在事件完成時，才會觸發。這樣的應用，即可以使用 **Deep-Sleep** 來實現休眠模式。

例如，當一台洗衣機已經洗完衣服了，之後會利用外部的 **GPIO** 來觸發傳感器，要傳感器把洗完衣服，這個事件的資訊傳送到雲端上面。隨後洗衣機會在利用外部的 **GPIO** 來觸發傳感器，讓傳感器再次進入 **Deep Sleep** 的睡眠模式。

CONTACT

sales@Opulinks.com