

OPL1000

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

低功耗解决方案



OPULINKS

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

Copyright © 2018~2020, Opulinks. All Rights Reserved.

OPL1000-Power-Saving-Introduction | Version 1.0

Date	Version	Contents Updated
2018-07-02	0.1	<ul style="list-style-type: none">Initial Release
2018-08-23	0.2	<ul style="list-style-type: none">Refine document typesetting
2019-12-13	0.3	<ul style="list-style-type: none">Update wake-up time and section 4.1
2020-09-24	1.0	<ul style="list-style-type: none">Modified Section 5 & 5.2

TABLE OF CONTENTS

1. 介绍 1

1.1. 文档应用范围 1

1.2. 缩略语 1

1.3. 参考文献 1

2. 概述 2

3. SMART-SLEEP 3

3.1. 特性 3

3.2. AT 命令接口说明 4

3.2.1. 启用 Smart-Sleep 4

3.3. API 接口说明 4

3.4. 外部唤醒 5

3.5. 应用 5

4. TIMER-SLEEP 6

4.1. 特性 6

4.2. AT 命令接口说明 6

4.2.1. 自动休眠 6

4.3. API 接口说明 7

4.4. 外部唤醒 7

4.5. 应用 8

5. DEEP-SLEEP 9

5.1. AT 命令接口说明 9

5.1.1. 使能 Deep-Sleep 9

5.2. API 接口说明 9

5.3. 外部唤醒 10

5.4. 应用 10

1. 介绍

1.1. 文档应用范围

低功耗解决方案用于 OPL1000 芯片的省电功能。本文介绍了低功耗的一些解决方案，让用户可以根据不同的情境之下，来选择一个适合用户的低功耗方案，进而达到省电的效果。

1.2. 缩略语

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

1.3. 参考文献

[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 1 所示。

表 1: 三种睡眠模式比较

项目	Smart-sleep	Timer-sleep	Deep-sleep
Wi-Fi 连接	保持	断连	断连
GPIO 状态	保持	保持	保持
Wi-Fi	开启	关闭	关闭
系统时钟	开启	开启	关闭
CPU	关闭	关闭	关闭
衬底电流			
平均电流	DTIM =1	关闭	关闭
	DTIM =3	关闭	关闭
	DTIM =10	关闭	关闭
BLE 联机	100ms	关闭	关闭
	500ms	关闭	关闭
	1000ms	关闭	关闭
BLE 广播	100ms	关闭	关闭
	500ms	关闭	关闭
	1000ms	关闭	关闭

3. SMART-SLEEP

3.1. 特性

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



说明

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

后续章节将说明可经由软件提供的跳过 DTIM (skip DTIM) 功能达到更省电的操作。

当有下列情况时，可以使用此功能

- Wi-Fi 已联机
- Wi-Fi 扫描中
- BLE 已联机
- BLE 广播

在 Smart-Sleep 模式下，OPL1000 WIFI 系统本身会自动调整两次 DTIM Beacon 间隔时间的接收长短，关闭或开启 Wi-Fi 模块电路，达到省电效果。在时间快到达的下次 Beacon 到来前自动唤醒，是透过 32K RTC 的振荡器来实现。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 命令接口说明

3.2.1. 启用 Smart-Sleep

系统通过以下 AT 指令进入 Smart-Sleep 模式。

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

参数说明:

mode	0: 关闭 smart sleep 1: 启用 smart sleep
ext_io	唤醒的 IO 埠号

3.3. API 接口说明

启用 Smart Sleep，系统在联机期间, 并且在闲置的状态时，系统会自动的进入睡眠模式。Smart Sleep 持续会运作，直到外部的触发唤醒而中止。

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

参数说明:

int enable	启用 Smart Sleep。
------------	-----------------

可以通过以下 API 接口，设定外部的输入埠号，来达到唤醒。

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

参数说明:

E_GpioIdx_t ext_io_num	唤醒功能的 IO 序号
E_ItrType_t ext_io_type	唤醒的触发模式

用户可以自行定义，当系统被唤醒之后，会做那些动作。

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

参数说明:

PS_WAKEUP_CALLBACK
callback

用户可以自行定义的 callback 函式。

3.4. 外部唤醒

在 Smart-Sleep 模式下，CPU 在暂停状态下不会响应来自外围硬件接口的信号与中断，因此需要通过外部 GPIO 信号将 OPL1000 唤醒，硬件唤醒过程大约为 3 ms。

3.5. 应用

Smart-Sleep 可以用于低功耗的传感器应用，或者大部分时间都不需要进行数据传输的情况之下。

例如，当 BLE (Bluetooth Low Energy) 正在广播，之后想让 BLE 进入休眠模式，可以使用 Smart-Sleep 的 AT 指令或 API 来控制实现，休眠的同时也可以做配对的动作，当有需要唤醒传感器时可以配合 GPIO 脚位来控制唤醒。

4. TIMER-SLEEP

4.1. 特性

有下列情况时，皆不可以使用。

- Wi-Fi 已联机
- Wi-Fi 扫描中
- BLE 已联机
- BLE 广播

系统无法自动进入 Timer-Sleep，需要由用户调用 AT 指令或是于代码中呼叫 API 来控制。

系统必须转入到 Idle 状态（没有 WIFI 和 BLE 通信操作）才能进入 Timer-Sleep 模式。进入睡眠模式，只有系统时钟模块仍然工作，负责芯片的定时唤醒。

4.2. AT 命令接口说明


4.2.1. 自动休眠

系统通过以下 AT 指令进入 Timer-Sleep 模式。

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

参数说明:

mode	2: 使用 timer sleep
Sleep_duration	睡眠周期，单位 millisecond
ext_io	唤醒的 IO 埠号

说明

在 Timer-Sleep 模式下，系统可以自动被唤醒。

4.3. API 接口说明

启用 Timer Sleep，系统会进入睡眠模式，直到外部的触发唤醒，或者 Timer 时间终止。

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

参数说明:

uint32_t sleep_duration_ms	睡眠到唤醒的时间长度，单位为 millisecond.
-------------------------------	-----------------------------

可以通过以下 API 接口，设定外部的输入埠号，来达到唤醒。

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

参数说明:

E_GpioIdx_t ext_io_num	唤醒功能的 IO 序号
E_ItrType_t ext_io_type	唤醒的触发模式

用户可以自行定义当系统被唤醒之后会做哪些操作。

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

参数说明:

PS_WAKEUP_CALLBACK callback	用户可以自行定义的 callback 函数。
--------------------------------	------------------------

4.4. 外部唤醒

在 Timer-sleep 模式下，CPU 在暂停状态下不会响应来自外围硬件接口的信号与中断，因此需要通过外部 GPIO 信号将 OPL1000 唤醒，硬件唤醒过程大约为 3 ms。

4.5. 应用

当客户清楚知道，应用本身会有多久的时间间隔，可以使用 Timer-Sleep 来实现休眠模式。

例如，传感器需要每五分钟传递数据时，可以使用 Timer-Sleep 来实现。使用 Timer-Sleep 会让传感器固定五分钟唤醒，侦测数据并传送数据到云端，随后又进入睡眠模式。

5. DEEP-SLEEP

相对于 IC 的 Timer-sleep 模式，系统无法自动进入 Deep-sleep，需要由开发者设定将芯片会断开所有 Wi-Fi 链接与数据链接后，再下指令，才会进入睡眠模式。RTC 模块也没有动作，只能透过外部的 GPIO 来唤醒芯片。

5.1. AT 命令接口说明

5.1.1. 使能 Deep-Sleep

系统通过以下 AT 指令进入 Deep-Sleep 模式。

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

参数说明:

mode	3: 启用 deep sleep
ext_io	唤醒的 IO 埠号

5.2. API 接口说明

启用 Deep Sleep，系统会进入睡眠模式，直到外部的触发唤醒。

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

可以通过以下 API 接口，设定外部的输入埠号，来达到唤醒。

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

参数说明:

E_Gpioldx_t ext_io_num	唤醒功能的 IO 序号
---------------------------	-------------

E_itrType_t ext_io_type	唤醒的触发模式
----------------------------	---------

5.3. 外部唤醒

在 Deep-Sleep 模式下，CPU 在暂停状态下不会响应来自外围硬件接口的信号与中断，因此需要通过外部 GPIO 信号将 OPL1000 唤醒，硬件唤醒过程大约为 3 ms。当唤醒之后，整个流程是从 cold-boot 的初始流程开始进行。

5.4. 应用

当客户清楚知道，应用本身只有在事件完成时，才会触发。这样的应用,即可以使用 Deep-Sleep 来实现休眠模式。

例如，当一台洗衣机已经洗完衣服了，之后会利用外部的 GPIO 来触发传感器，要传感器把洗完衣服，这个事件的信息传送到云端上面。随后洗衣机会在利用外部的 GPIO 来触发传感器，让传感器再次进入 Deep Sleep 的睡眠模式。

CONTACT

sales@Opulinks.com