OPL2500

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

Low-Power Solution

OPULINKS

http://www.opulinks.com/

OPL2500-Power-Saving-Introduction-R01

Revision History

Date	Version	Contents Updated
2024-07-22	0.1	● Initial Release

TABLE OF CONTENTS

CHAPTER ONE

1. INTRODUCTION

1.1. Scope of Document Application

Low-power solution is used for power-saving function of OPL2500 chip. This document outlines a number of low-power solutions that allows users can select a suitable low-power solution, according to different usage scenarios so as to achieve their perspective power-saving objectives.

1.2. Abbreviations

Abbr.	Explanation
BLE	Bluetooth Low Energy
API	Application Programming Interface
DTIM	Delivery Traffic Indication Message
AT	Attention Terminal Instruction Command Index

1.3. Reference

[1] OPL2500-AT-instruction-set-and-examples.pdf

2. DESCRIPTION

OPL2500 SoC provides 3 types of flexible sleep modes, and regarding these sleep modes, we have provided multiple low-power solutions so that users can consolidate their respective needs to select the suitable sleep mode. The three types of sleep modes, supported by the chip, are as follows:

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

The differentiations amongst three modes are shown as Table 1,

Table 1: Comparison between three sleep modes

Items		Smart-sleep	Timer-sleep	Deep-sleep
Wi-Fi Connection		CONNECTED	DISCONNTECTED	DISCONNTECTED
GPIO Status		ON	ON	ON
Wi-Fi		ON	OFF	OFF
System Clock		ON	ON	OFF
CPU		OFF	OFF	OFF
Average Current	DTIM =1		OFF	OFF
	DTIM =3		OFF	OFF
	DTIM =10		OFF	OFF
BLE Connection	100ms		OFF	OFF
	500ms		OFF	OFF
	1000ms		OFF	OFF
BLE Adv.	100ms		OFF	OFF
	500ms		OFF	OFF
	1000ms		OFF	OFF

3. SMART-SLEEP

3.1. Characteristics

Currently, Smart-Sleep feature of OPL2500 can operate under Wi-Fi Station Mode or BLE mode, and comes into effect when connected with routers in WIFI system through Wi-Fi DTIM mechanism.

Description

he Beacon interval of average WIFI router is 100ms ~ 1,000ms, with DTIM value of "1".

In subsequent chapters, there will be operating procedure for power-saving through DTIM-skipping software function.

During the following scenarios, this function is applicable:

- Wi-Fi Connection Established
- Wi-Fi Scanning
- BLE Connection Established
- BLE Advertising

Under Smart-Sleep Mode, OPL2500 WIFI system would automatically adjust the receive interval and duration in between each Beacon, in turning off or on Wi-Fi module circuit, in order to achieve power-saving result. Through 32K RTC oscillator, auto-wake comes into effect prior to the next upcoming Beacon. During sleep the device can maintain connection with Wi-Fi router, and exchanged information from servers.

3.2. AT Commands Interface Description

3.2.1. Activate Smart-Sleep

Through the following AT command, system enters Smart-Sleep Mode.

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

Parameter Description(s):

mode	0: Turn-off smart sleep 1: Turn-on smart sleep
ext_io	Awaken I/O port number

3.3. API Description

Once Smart Sleep is activated, during the connected period in idle status, the system automatically enters sleep mode. Smart Sleep continues to operate until terminated by the activation of external interrupt to wake up the system.

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

Parameter Description(s):

int enable	Activate Smart Sleep

Set external input port number through the following API to achieve system wakeup.

```
void ps_set_wakeup_io(E_GpioIdx_t ext_io_num, int enable, E_ItrType_t type,
int invert, T_Gpio_CallBack callback);
```

Parameter Description(s):

E_GpioIdx_t ext_io_num	I/O Serial Number of Wake Function
Int enable	Set whether enable the external wake-up mechanism

E_ItrType_t type	The trigger type for external input, Can be INT_TYPE_LEVEL, INT_TYPE_RISING_EDGE, INT_TYPE_FALLING_EDGE INT_TYPE_BOTH_EDGE
int invert	Set whether invert the triggered signal
T_Gpio_CallBack callback	The ISR which be invoked at IO wake-up happened

For example:

```
/* in hal_pin_config_xxx.h to configure target IO to input IO */
ps_set_wakeup_io(GPIO_IDX_24, 1, INT_TYPE_LEVEL, 0, at_cmd_sys_gslp_io_callback);
ps_smart_sleep(1);
```

Users can setup callback function in which actions will come into effect after the system is woken by GPIO input interrupt.

```
void Gpio_CallBack(E_GpioIdx_t eIdx);
```

Parameter Description(s):

GPIO interrupt callback	Users can set up callback function.

For example:

```
ps_set_wakeup_io(GPIO_IDX_24, 1, INT_TYPE_LEVEL, 0, at_cmd_sys_gslp_io_callback);
void at_cmd_sys_gslp_io_callback(E_GpioIdx_t eIdx)
{
    Hal_Vic_GpioIntEn(eIdx, 0);
    ps_smart_sleep(0);
   msg_print_uart1("AT IO ISR invoked, io_num: %d\r\n", eIdx);
}
```

3.4. External Wake-Up

Under Smart Sleep Mode, in idle status, CPU will not respond to any signal from external hardware port, therefore OPL2500 can be only woken up by setting external GPIO signal, which takes approximately 1ms.

3.5. Applications

Smart-Sleep can be used in scenarios of low-power sensor application. For example, when BLE (Bluetooth Low Energy) is in advertisement, and if it is required to enter sleep mode later, Smart-Sleep AT Command or API serves to control its realization, while pairing is also done during sleep mode, and when it is time to wake up by sensors, GPIO pin can be used to control wake-up function.

4. TIMER-SLEEP

4.1. Characteristics

During the following scenarios, this feature is not applicable:

- Wi-Fi Connection Established
- Wi-Fi Scanning
- BLE Connection Established
- BLE Advertising

Users can implement AT Command or API code to control the system to enter Timer-Sleep mode, and under this mode, chip will terminate all Wi-Fi connections and data connections to enter sleep mode, as the system clock remains operational to facilitate timed wake-up for the chip.

4.2. AT Commands Interface Description

4.2.1. Activate Timer-Sleep

Through the following AT command, system enters Timer-Sleep Mode.

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

Parameter Description(s):

mode	2: Use of Timer Sleep
Sleep_duration	Sleep Cycle, with unit of millisecond
ext_io	Awaken I/O port number

Desctiption

Under Timer-Sleep Mode, system can be automatically woken up.

4.3. API Description

Once Timer-Sleep is activated, the system will enter sleep mode, until woken up by external activation, or when the Timer function expires.

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

Parameter Description(s):

uint32_t sleep_duration_ms	The time duration of sleep to woken-up is based on unit of millisecond.

Set external input port number through the following API ports to achieve wake.

```
void ps_set_wakeup_io(E_GpioIdx_t ext_io_num, int enable, E_ItrType_t type,
int invert, T_Gpio_CallBack callback);
```

Parameter Description(s):

	The same definition as Smart Sleep

For example:

```
/* in hal_pin_config_xxx.h to configure target IO to input IO */
ps_set_wakeup_io(GPIO_IDX_24, 1, INT_TYPE_LEVEL, 0, at_cmd_sys_tslp_io_callback);
ps_set_wakeup_cb(at_cmd_sys_tslp_wakeup_callback);
ps_timer_sleep(1000);            // Sleep 1000ms

void at_cmd_sys_tslp_io_callback(E_GpioIdx_t eIdx)
{
    Hal_Vic_GpioIntEn(eIdx, 0);
    msg_print_uart1("AT IO ISR invoked, io_num: %d\r\n", eIdx);
}
```

Users can setup callback function in which actions will come into effect after the system is woken.

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

Parameter Description(s):

PS_WAKEUP_CALLBACK callback	Users can set up callback function.

For example:

```
ps_set_wakeup_cb(at_cmd_sys_tslp_wakeup_callback);

void at_cmd_sys_tslp_wakeup_callback(PS_WAKEUP_TYPE type)
{
	msg_print_uart1("\r\nAT Wakeup, Type: %s\r\n", type == PS_WAKEUP_TYPE_IO ?
"IO" : "TIMEOUT");
}
```

4.4. External Wake-Up

Under Timer Sleep Mode, in idle status, CPU will not respond to any signal from external hardware port, therefore OPL2500 can be only woken up by setting external GPIO signal, which takes approximately 1ms.

4.5. Applications

When users clearly know the exact time interval for any given application, they can use Timer-Sleep to realize Sleep-Mode.

For example, as the sensor needs to transmit data every five minutes, it can be facilitated by using the Timer-Sleep feature. By activating Timer-Sleep feature. It will wake up sensor every five minutes, detect information before transmitting it to cloud service, before entering sleep mode again.

5. DEEP-SLEEP

As opposed to the Timer-Sleep mode of IC, the system is unable to enter into Deep-Sleep automatically, but through users using functional API to control. Under this mode, the chip will terminate all Wi-Fi connections and data connections to enter sleep mode, while RTC module is idle, therefore the chip can only be woken up through external GPIO.

5.1. AT command Interface Description

5.1.1. Activate Deep-Sleep

The system enters Deep-Sleep Mode through the following AT Commands.

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

Parameter Description(s):

Mode	3: Deep-Sleep is activated
ext_io	Awaken I/O port number

5.2. API Description

Once Deep-Sleep is activated, the system will enter sleep mode, until woken up by external activation.

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

Set external input port number through the following API to achieve wake.

```
void ps_set_wakeup_io(E_GpioIdx_t ext_io_num, int enable, E_ItrType_t type,
int invert, T_Gpio_CallBack callback);
```

Parameter Description(s):

	IO definition is the same as Smart Sleep

For Example:

```
/* in hal_pin_config_xxx.h to configure target IO to input IO */
ps_set_wakeup_io(GPIO_IDX_24, 1, INT_TYPE_LEVEL, 0, NULL);
ps_deep_sleep();          // System Enters Deep Sleep mode
```

5.3. External Wake-Up

Under Timer Sleep Mode, in idle status, CPU will not respond to any signal from external hardware port, therefore OPL2500 can be only woken up by setting external GPIO signal, which takes approximately 1ms. When awakened, the whole procedure will go into execution from cold-boot initialization.

5.4. Applications

When customers clearly know that the application will sleep most of the time, and only wake up by external event, therefore such application will need Deep-Sleep to realize sleep mode.

CONTACT

sales@Opulinks.com