

# Rockchip Sysutil 开发指南

---

文件标识: RK-KF-YF-929

发布版本: V1.0.2

日期: 2023-08-04

文件密级: ☐绝密 ☐秘密 ☐内部资料 ☒公开

## 免责声明

本文档按“现状”提供, 瑞芯微电子股份有限公司(“本公司”, 下同)不对本文档的任何陈述、信息和内容的准确性、可靠性、完整性、适销性、特定目的性和非侵权性提供任何明示或暗示的声明或保证。本文档仅作为使用指导的参考。

由于产品版本升级或其他原因, 本文档将可能在未经任何通知的情况下, 不定期进行更新或修改。

## 商标声明

“Rockchip”、“瑞芯微”、“瑞芯”均为本公司的注册商标, 归本公司所有。

本文档可能提及的其他所有注册商标或商标, 由其各自拥有者所有。

版权所有 © 2023 瑞芯微电子股份有限公司

超越合理使用范畴, 非经本公司书面许可, 任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部, 并不得以任何形式传播。

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址: 福建省福州市铜盘路软件园A区18号

网址: [www.rock-chips.com](http://www.rock-chips.com)

客户服务电话: +86-4007-700-590

客户服务传真: +86-591-83951833

客户服务邮箱: [fae@rock-chips.com](mailto:fae@rock-chips.com)

前言

概述

本文主要描述了Sysutil 组件开发参考。

产品版本

芯片名称	内核版本
RV1106	Linux 5.10

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

版本号	作者	修改日期	修改说明
V1.0.0	ZZW	2022-04-08	初始版本
V1.0.1	LZW	2022-11-28	修正引脚输入输出数据类型
V1.0.2	LZW	2023.08.04	更新ADC接口名称

# 目录

## Rockchip Sysutil 开发指南

1. 系统概述
2. GPIO
  - 2.1 概述
  - 2.2 API参考
    - 2.2.1 rk\_gpio\_export
    - 2.2.2 rk\_gpio\_unexport
    - 2.2.3 rk\_gpio\_set\_direction
    - 2.2.4 rk\_gpio\_get\_direction
    - 2.2.5 rk\_gpio\_export\_direction
    - 2.2.6 rk\_gpio\_set\_value
    - 2.2.7 rk\_gpio\_get\_value
    - 2.2.8 数据类型
      - 2.2.8.1 enum gpio\_direction
3. ADC
  - 3.1 概述
  - 3.2 API 参考
    - 3.2.1 rk\_adc\_get\_devnum
    - 3.2.2 rk\_adc\_get\_value
4. EVENT
  - 4.1 概述
  - 4.2 API参考
    - 4.2.1 rk\_event\_register
    - 4.2.2 rk\_event\_unregister
    - 4.2.3 rk\_event\_listen\_start
    - 4.2.4 rk\_event\_listen\_stop
5. PWM
  - 5.1 概述
  - 5.2 API 参考
    - 5.2.1 rk\_pwm\_export
    - 5.2.2 rk\_pwm\_unexport
    - 5.2.3 rk\_pwm\_set\_period
    - 5.2.4 rk\_pwm\_get\_period
    - 5.2.5 rk\_pwm\_set\_duty
    - 5.2.6 rk\_pwm\_get\_duty
    - 5.2.7 rk\_pwm\_set\_polarity
    - 5.2.8 rk\_pwm\_get\_polarity
    - 5.2.9 rk\_pwm\_set\_enable
    - 5.2.10 rk\_pwm\_get\_enable
    - 5.2.11 rk\_pwm\_init
    - 5.2.12 rk\_pwm\_deinit
    - 5.2.13 数据类型
      - 5.2.13.1 enum pwm\_polarity
6. TIME
  - 6.1 概述
  - 6.2 API参考
    - 6.2.1 rk\_system\_get\_time
    - 6.2.2 rk\_system\_set\_time
    - 6.2.3 rk\_system\_set\_alarm
    - 6.2.4 rk\_system\_get\_alarm
    - 6.2.5 rk\_system\_enable\_alarm
    - 6.2.6 rk\_system\_disable\_alarm
    - 6.2.7 rk\_system\_wait\_alarm

## 7. LED

### 7.1 概述

### 7.2 API参考

#### 7.2.1 rk\_led\_set\_mode

## 8. WATCHDOG

### 8.1 概述

### 8.2 API参考

#### 8.2.1 rk\_watchdog\_start

#### 8.2.2 rk\_watchdog\_refresh

#### 8.2.3 rk\_watchdog\_stop

## 9. SYSTEM

### 9.1 概述

### 9.2 API参考

#### 9.2.1 rk\_chip\_id\_get

#### 9.2.2 rk\_vendor\_write

#### 9.2.3 rk\_vendor\_read

#### 9.2.4 rk\_system\_reboot

#### 9.2.5 rk\_system\_shutdown

#### 9.2.6 rk\_system\_suspend

#### 9.2.7 数据类型

##### 9.2.7.1 SUSPEND\_TYPE

## 10. MOTOR

### 10.1 概述

### 10.2 API参考

#### 10.2.1 rk\_motor\_init

#### 10.2.2 rk\_motor\_deinit

#### 10.2.3 rk\_motor\_move

#### 10.2.4 rk\_motor\_stop

#### 10.2.5 rk\_motor\_get\_status

#### 10.2.6 rk\_motor\_speed

#### 10.2.7 rk\_motor\_reset

#### 10.2.8 rk\_motor\_goback

#### 10.2.9 数据类型

##### 10.2.9.1 motors\_init\_data

##### 10.2.9.2 motor\_reset\_data

# 1. 系统概述

sysutil是基于sysfs封装的一套用户态接口，包括外设接口和系统功能接口，方便应用层对外设和系统的控制，简化了应用开发难度，方便客户基于这些硬件接口进行应用开发。

## 2. GPIO

### 2.1 概述

提供gpio基本的用户态接口。

### 2.2 API参考

#### 2.2.1 rk\_gpio\_export

【描述】

导出需要控制的GPIO引脚。

【语法】

```
int rk_gpio_export(uint32_t gpio);
```

【参数】

参数名称	描述	输入/输出
gpio	gpio引脚编号	输入

【返回值】

返回值	描述
0	成功
非0	失败

【需求】

头文件：rk\_gpio.h

【注意】

- 需要查看文件系统中是否有/sys/class/gpio节点，如果没有该节点，就需要在编译内核时勾选Device Drivers-> GPIO Support ->/sys/class/gpio/... (sysfs interface)，对应的CONFIG 名字为GPIO\_SYSFS。

- gpio引脚需要在不被占用的状态下。

【举例】

rk\_gpio\_test

【相关主题】

[rk\\_gpio\\_unexport](#)。

2.2.2 rk\_gpio\_unexport

【描述】

用于通知系统取消导出。

【语法】

int rk\_gpio\_unexport(uint32\_t gpio);

【参数】

参数名称	描述	输入/输出
gpio	gpio引脚编号	输入

【返回值】

返回值	描述
0	成功
非0	失败

【需求】

头文件：rk\_gpio.h

【注意】

- 用于在rk\_gpio\_export成功后调用。

【举例】

rk\_gpio\_test

【相关主题】

[rk\\_gpio\\_export](#)

2.2.3 rk\_gpio\_set\_direction

【描述】

定义输入输出方向。

【语法】

```
int rk_gpio_set_direction(uint32_t gpio, enum gpio_direction input);
```

【参数】

参数名称	描述	输入/输出
gpio	gpio引脚编号	输入
input	GPIO_DIRECTION_INPUT: in GPIO_DIRECTION_OUTPUT: out	输入

【返回值】

返回值	描述
0	成功
非0	失败

【需求】

头文件: rk\_gpio.h

【注意】

- rk\_gpio\_export导出成功后，才能使用rk\_gpio\_set\_direction接口。

【举例】

rk\_gpio\_test

【相关主题】

[rk\\_gpio\\_get\\_direction](#)

2.2.4 rk\_gpio\_get\_direction

【描述】

获取gpio引脚的方向信息。

【语法】

```
int rk_gpio_get_direction(uint32_t gpio);
```

【参数】

参数名称	描述	输入/输出
gpio	gpio引脚编号	输入

【返回值】

返回值	描述
0或1	0:out,1:in
负	失败

【需求】

头文件：rk\_gpio.h

【注意】

- rk\_gpio\_export导出成功后，才能使用rk\_gpio\_get\_direction接口。

【举例】

rk\_gpio\_test

【相关主题】

[rk\\_gpio\\_set\\_direction](#)

### 2.2.5 rk\_gpio\_export\_direction

【描述】

gpio初始化，导出gpio的同时指定gpio的方向。

【语法】

```
int rk_gpio_export_direction(uint32_t gpio, enum gpio_direction input);
```

【参数】

参数名称	描述	输入/输出
gpio	gpio引脚编号	输入
input	GPIO_DIRECTION_INPUT: in GPIO_DIRECTION_OUTPUT: out	输入

【返回值】

返回值	描述
0	成功
非0	失败

【需求】

头文件：rk\_gpio.h

【举例】

rk\_gpio\_test

【相关主题】



无

### 2.2.6 rk\_gpio\_set\_value

**【描述】**

设置gpio引脚的值。

**【语法】**

```
int rk_gpio_set_value(uint32_t gpio, int value);
```

**【参数】**

参数名称	描述	输入/输出
gpio	gpio引脚编号	输入
value	要设置的值1/0	输入

**【返回值】**

返回值	描述
0	成功
非0	失败

**【需求】**

头文件：rk\_gpio.h

**【注意】**

- rk\_gpio\_export导出成功后，才能使用rk\_gpio\_set\_value接口。

**【举例】**

rk\_gpio\_test

**【相关主题】**

[rk\\_gpio\\_get\\_value](#)

### 2.2.7 rk\_gpio\_get\_value

**【描述】**

获取gpio引脚的值。

**【语法】**

```
int rk_gpio_get_value(uint32_t gpio);
```

**【参数】**

参数名称	描述	输入/输出
gpio	gpio引脚编号	输入

【返回值】

返回值	描述
0或1	成功
负	失败

【需求】

头文件：rk\_gpio.h

【举例】

rk\_gpio\_test

【相关主题】

2.2.8 数据类型

GPIO输入输出使用以下数据类型

2.2.8.1 enum gpio\_direction

【说明】

GPIO输入输出方向

【定义】

```
enum gpio_direction
{
    GPIO_DIRECTION_OUTPUT = 0,
    GPIO_DIRECTION_INPUT,
};
```

【成员】

成员名称	描述
GPIO_DIRECTION_OUTPUT	输出
GPIO_DIRECTION_INPUT	输入

## 3. ADC

---

### 3.1 概述

提供基本的ADC用户态接口

### 3.2 API 参考

#### 3.2.1 rk\_adc\_get\_devnum

**【描述】**

通过设备名获取设备编号。

**【语法】**

```
int rk_adc_get_devnum(const char *name);
```

**【参数】**

参数名称	描述	输入/输出
name	设备名	输入

**【返回值】**

返回值	描述
非负	设备编号
负	失败

**【需求】**

头文件：rk\_adc.h

**【注意】**

- 需要保证在saradc初始化之后。

**【举例】**

rk\_adc\_test

**【相关主题】**

无

### 3.2.2 rk\_adc\_get\_value

**【描述】**

读取采集通道 AD 采集的原始数据。

**【语法】**

int rk\_adc\_get\_value(uint32\_t dev\_num, uint32\_t chn\_num);

**【参数】**

参数名称	描述	输入/输出
dev_num	设备编号	输入
chn_num	该设备所使用的 IIO 通道	输入

**【返回值】**

返回值	描述
非负	值
负	失败

**【需求】**

头文件：rk\_adc.h

**【举例】**

rk\_adc\_test

**【相关主题】**

无

# 4. EVENT

---

## 4.1 概述

按照内核input event标准方式，监听按键等事件

## 4.2 API参考

### 4.2.1 rk\_event\_register

**【描述】**

注册需要监听的节点。

**【语法】**

int rk\_event\_register(char \*dev\_path);

**【参数】**

参数名称	描述	输入/输出
dev_path	节点路径	输入

**【返回值】**

返回值	描述
0	成功
非0	失败

**【需求】**

头文件：rk\_event.h

**【注意】**

- 不支持重复注册
- 需要保证节点存在

**【举例】**

rk\_event\_test

**【相关主题】**

[rk\\_event\\_unregister](#)

### 4.2.2 rk\_event\_unregister

**【描述】**

注销需要监听的节点。

**【语法】**

int rk\_event\_unregister(char \*dev\_path);

**【参数】**

参数名称	描述	输入/输出
dev_path	节点路径	输入

**【返回值】**

返回值	描述
0	成功
非0	失败

**【需求】**

头文件：rk\_event.h

**【注意】**

- rk\_event\_register注册后才能调用rk\_event\_unregister

**【举例】**

rk\_event\_test

**【相关主题】**

[rk\\_event\\_register](#)

### 4.2.3 rk\_event\_listen\_start

**【描述】**

启动事件监听。

**【语法】**

int rk\_event\_listen\_start(event\_handler\_t handler);

**【参数】**

参数名称	描述	输入/输出
handler	监听事件回调函数	输入

**【返回值】**

返回值	描述
0	成功
非0	失败

**【需求】**

头文件：rk\_event.h

**【注意】**

- rk\_event\_register后才能调用rk\_event\_listen\_start。
- 不支持重复rk\_event\_listen\_start
- rk\_event\_listen\_start后不支持再注册监听节点，但是可以注销监听节点
- 调用rk\_event\_listen\_start后，触发事件回调函数开始接收数据。

**【举例】**

rk\_event\_test

**【相关主题】**

[rk\\_event\\_listen\\_stop](#)

## 4.2.4 rk\_event\_listen\_stop

**【描述】**

停止事件监听。

**【语法】**

int rk\_event\_listen\_stop(void);

**【返回值】**

返回值	描述
0	成功
非0	失败

**【需求】**

头文件：rk\_event.h

**【注意】**

- 停止监听后会清空所有的监听节点，需要重新注册

**【举例】**

rk\_event\_test

**【相关主题】**

[rk\\_event\\_listen\\_start](#)

# 5. PWM

## 5.1 概述

提供基本的pwm用户态接口。

## 5.2 API 参考

### 5.2.1 rk\_pwm\_export

**【描述】**

导出需要控制的PWM引脚。

**【语法】**

```
int rk_pwm_export(uint32_t pwm);
```

**【参数】**

参数名称	描述	输入/输出
pwm	pwm编号	输入

**【返回值】**

返回值	描述
0	成功
非0	失败

**【需求】**

头文件：rk\_pwm.h

**【注意】**

- 需要查看文件系统中是否有/sys/class/pwm节点，如果没有该节点，就需要在编译内核时勾选Device Drivers-> Pulse-Width Modulation (PWM) Support，对应的CONFIG 名字为PWM。

**【举例】**

```
rk_pwm_test
```

**【相关主题】**

[rk\\_pwm\\_unexport](#)



### 5.2.2 rk\_pwm\_unexport

**【描述】**

用于通知系统取消导出pwm。

**【语法】**

int rk\_pwm\_unexport(uint32\_t pwm);

**【参数】**

参数名称	描述	输入/输出
pwm	pwm编号	输入

**【返回值】**

返回值	描述
0	成功
非0	失败

**【需求】**

头文件：rk\_pwm.h

**【注意】**

- 用于在rk\_pwm\_export成功后调用

**【举例】**

rk\_pwm\_test

**【相关主题】**

[rk\\_pwm\\_export](#)

### 5.2.3 rk\_pwm\_set\_period

**【描述】**

设置pwm周期。

**【语法】**

int rk\_pwm\_set\_period(uint32\_t pwm, uint32\_t period);

**【参数】**

参数名称	描述	输入/输出
pwm	pwm编号	输入
period	周期时长（纳秒）	输入

【返回值】

返回值	描述
0	成功
非0	失败

【需求】

头文件：rk\_pwm.h

【注意】

- 周期时长不超过10的9次方

【举例】

rk\_pwm\_test

【相关主题】

[rk\\_pwm\\_get\\_period](#)

5.2.4 rk\_pwm\_get\_period

【描述】

获取pwm周期。

【语法】

int rk\_pwm\_get\_period(uint32\_t pwm);

【参数】

参数名称	描述	输入/输出
pwm	pwm编号	输入

【返回值】

返回值	描述
非负	周期时长（纳秒）
负	失败

【需求】

头文件：rk\_pwm.h

【举例】

rk\_pwm\_test

【相关主题】

[rk\\_pwm\\_set\\_period](#)

### 5.2.5 rk\_pwm\_set\_duty

**【描述】**

配置pwm的占空比。

**【语法】**

```
int rk_pwm_set_duty(uint32_t pwm, uint32_t duty);
```

**【参数】**

参数名称	描述	输入/输出
pwm	pwm编号	输入
duty	占空比时长（纳秒）	输入

**【返回值】**

返回值	描述
0	成功
非0	失败

**【需求】**

头文件：rk\_pwm.h

**【注意】**

- 占空比时长不超过10的9次方，且不超过当前设置的周期时长

**【举例】**

```
rk_pwm_test
```

**【相关主题】**

[rk\\_pwm\\_get\\_duty](#)

### 5.2.6 rk\_pwm\_get\_duty

**【描述】**

获取pwm占空比时长。

**【语法】**

```
int rk_pwm_get_duty(uint32_t pwm);
```

**【参数】**

参数名称	描述	输入/输出
pwm	pwm编号	输入

#### 【返回值】

返回值	描述
非负	占空比时长（纳秒）
负	失败

#### 【需求】

头文件：rk\_pwm.h

#### 【举例】

rk\_pwm\_test

#### 【相关主题】

[rk\\_pwm\\_set\\_duty](#)

### 5.2.7 rk\_pwm\_set\_polarity

#### 【描述】

设置pwm极性。

#### 【语法】

int rk\_pwm\_set\_polarity(uint32\_t pwm, [enum\\_pwm\\_polarity](#) polarity);

#### 【参数】

参数名称	描述	输入/输出
pwm	pwm引脚编号	输入
polarity	极性	输入

#### 【返回值】

返回值	描述
0	成功
非0	失败

#### 【需求】

头文件：rk\_pwm.h

#### 【举例】

rk\_pwm\_test

【相关主题】

[rk\\_pwm\\_get\\_period](#)

### 5.2.8 rk\_pwm\_get\_polarity

【描述】

获取pwm引脚的极性。

【语法】

```
int rk_pwm_get_polarity(uint32_t pwm);
```

【参数】

参数名称	描述	输入/输出
pwm	pwm引脚编号	输入

【返回值】

返回值	描述
0或1	0：正极性，1：负极性
负	失败

【需求】

头文件：rk\_pwm.h

【举例】

rk\_pwm\_test

【相关主题】

[rk\\_pwm\\_set\\_polarity](#)

### 5.2.9 rk\_pwm\_set\_enable

【描述】

使能pwm。

【语法】

```
int rk_pwm_set_enable(uint32_t pwm, bool enabled);
```

【参数】

参数名称	描述	输入/输出
pwm	pwm引脚编号	输入
enabled	写入 1 使能 pwm，写入 0 关闭 pwm	输入

**【返回值】**

返回值	描述
0	成功
非0	失败

**【需求】**

头文件：rk\_pwm.h

**【举例】**

rk\_pwm\_test

**【相关主题】**

[rk\\_pwm\\_get\\_enable](#)

## 5.2.10 rk\_pwm\_get\_enable

**【描述】**

判断pwm引脚是否使能。

**【语法】**

```
int rk_pwm_get_enable(uint32_t pwm);
```

**【参数】**

参数名称	描述	输入/输出
pwm	pwm引脚编号	输入

**【返回值】**

返回值	描述
0或1	0：关闭，1：使能
负	失败

**【需求】**

头文件：rk\_pwm.h

**【举例】**

rk\_pwm\_test

【相关主题】

[rk\\_pwm\\_set\\_enable](#)

### 5.2.11 rk\_pwm\_init

【描述】

初始化pwm，同时设置pwm引脚的周期、占空比、极性。

【语法】

```
int rk_pwm_init(uint32_t pwm, uint32_t period, uint32_t duty,
                enum pwm_polarity polarity);
```

【参数】

参数名称	描述	输入/输出
pwm	pwm引脚编号	输入
period	周期时长（纳秒）	输入
duty	占空比时长（纳秒）	输入
polarity	极性	输入

【返回值】

返回值	描述
0	成功
非0	失败

【需求】

头文件：rk\_pwm.h

【举例】

rk\_pwm\_test

【相关主题】

[rk\\_pwm\\_deinit](#)

### 5.2.12 rk\_pwm\_deinit

【描述】

反初始化pwm，通知系统取消导出。

【语法】

int rk\_pwm\_deinit(uint32\_t pwm);

【参数】

参数名称	描述	输入/输出
pwm	pwm引脚编号	输入

【返回值】

返回值	描述
0	成功
非0	失败

【需求】

头文件：rk\_pwm.h

【注意】

- 在rk\_pwm\_init调用成功后才能调用rk\_pwm\_deinit。

【举例】

rk\_pwm\_test

【相关主题】

[rk\\_pwm\\_init](#)

5.2.13 数据类型

PWM参数主要提供以下数据类型：

5.2.13.1 enum pwm\_polarity

【说明】

pwm极性。

【定义】

```
enum pwm_polarity {
    PWM_POLARITY_NORMAL,
    PWM_POLARITY_INVERSED,
};
```

【成员】



成员名称	描述
PWM_POLARITY_NORMAL	正极性
PWM_POLARITY_INVERSED	负极性

# 6. TIME

---

## 6.1 概述

提供硬件时间和系统时间的用户态接口。

## 6.2 API参考

### 6.2.1 rk\_system\_get\_time

**【描述】**

获取硬件时间。

**【语法】**

int rk\_system\_get\_time(struct tm \*time);

**【参数】**

参数名称	描述	输入/输出
time	保存获取时间	输出

**【返回值】**

返回值	描述
0	成功
非0	失败

**【需求】**

头文件：rk\_time.h

**【注意】**

- 需要确保/dev/rtc节点开启，如果没有开启，就需要在编译内核时勾选Device Drivers-> Real Time Clock，对应的CONFIG 名字为RTC\_CLASS。

**【举例】**

rk\_time\_test

**【相关主题】**

[rk\\_system\\_set\\_time](#)

## 6.2.2 rk\_system\_set\_time

### 【描述】

修改硬件时间同步更新系统时间。

### 【语法】

```
int rk_system_set_time(struct tm *time);
```

### 【参数】

参数名称	描述	输入/输出
time	目标修改时间	输入

### 【返回值】

返回值	描述
0	成功
非0	失败

### 【需求】

头文件：rk\_time.h

### 【注意】

- 需要确保/dev/rtc节点开启，如果没有开启，就需要在编译内核时勾选Device Drivers-> Real Time Clock，对应的CONFIG 名字为RTC\_CLASS。
- 需要输入有效的时间

### 【举例】

```
rk_time_test
```

### 【相关主题】

[rk\\_system\\_get\\_time](#)

## 6.2.3 rk\_system\_set\_alarm

### 【描述】

设置alarm中断的触发时刻。

### 【语法】

```
rk_system_set_alarm(struct tm *time);
```

### 【参数】

参数名称	描述	输入/输出
time	目标设置时间	输入

#### 【返回值】

返回值	描述
0	成功
非0	失败

#### 【需求】

头文件：rk\_time.h

#### 【注意】

- alarm中断的触发时间只能是24小时内的一个时刻，所以只有时、分、秒的部分是有效的，参数time的年、月、日部分会被忽略。

#### 【举例】

rk\_time\_test

#### 【相关主题】

[rk\\_system\\_get\\_alarm](#)

## 6.2.4 rk\_system\_get\_alarm

#### 【描述】

读取已经设置的alarm中断时刻。

#### 【语法】

```
int rk_system_get_alarm(struct tm *time);
```

#### 【参数】

参数名称	描述	输入/输出
time	保存读取时间	输出

#### 【返回值】

返回值	描述
0	成功
非0	失败

#### 【需求】

头文件：rk\_time.h

**【注意】**

- 需要在rk\_system\_set\_alarm成功后调用rk\_system\_get\_alarm

**【举例】**

rk\_time\_test

**【相关主题】**

[rk\\_system\\_set\\_alarm](#)

**6.2.5 rk\_system\_enable\_alarm**

**【描述】**

使能已经设置的alarm。

**【语法】**

int rk\_system\_enable\_alarm(void);

**【返回值】**

返回值	描述
0	成功
非0	失败

**【需求】**

头文件：rk\_time.h

**【注意】**

- 需要在rk\_system\_set\_alarm成功后调用rk\_system\_enable\_alarm
- 不支持enable后重新设置alarm的触发时间

**【举例】**

rk\_time\_test

**【相关主题】**

[rk\\_system\\_disable\\_alarm](#)

**6.2.6 rk\_system\_disable\_alarm**

**【描述】**

关闭alarm中断。

**【语法】**

int rk\_system\_disable\_alarm(void);

**【返回值】**

返回值	描述
0	成功
非0	失败

**【需求】**

头文件：rk\_time.h

**【注意】**

- 调用rk\_system\_enable\_alarm后调用，不会改变已经设置的中断触发时刻

**【举例】**

rk\_time\_test

**【相关主题】**

[rk\\_system\\_enable\\_alarm](#)

### 6.2.7 rk\_system\_wait\_alarm

**【描述】**

等待设备节点中断唤醒。

**【语法】**

int rk\_system\_wait\_alarm(uint32\_t wait\_seconds);

**【参数】**

参数名称	描述	输入/输出
wait_seconds	等待超时时间	输入

**【返回值】**

返回值	描述
0	成功
非0	失败

**【需求】**

头文件：rk\_time.h

**【注意】**

- 会阻塞当前线程，调用rk\_system\_enable\_alarm后调用，超时时间单位为秒

**【举例】**

rk\_time\_test

【相关主题】

[rk\\_system\\_enable\\_alarm](#)

[rk\\_system\\_set\\_time](#)

## 7. LED

### 7.1 概述

目前led只提供调节亮度和启停闪烁的功能

### 7.2 API参考

#### 7.2.1 rk\_led\_set\_mode

【描述】

led调节亮度和启动闪烁接口。

【语法】

```
int rk_led_set_mode(char *dev_path, bool blink, uint32_t brightness);
```

【参数】

参数名称	描述	输入/输出
dev_path	led节点路径	输入
blink	是否闪烁	输入
brightness	亮度	输入

【返回值】

返回值	描述
0	成功
非0	失败

【需求】

头文件：rk\_led.h

【注意】

- 需要保证/sys/class/leds设备节点存在，如果不存在，就需要在编译内核时勾选Device Drivers-> LED Support ->LED Class Support ,对应的CONFIG 名字为LEDS\_CLASS。

【举例】

rk\_led\_test

【相关主题】

无

## 8. WATCHDOG

---

### 8.1 概述

提供看门狗的用户态接口。

### 8.2 API参考

#### 8.2.1 rk\_watchdog\_start

【描述】

使能watchdog，可选设置超时时间。

【语法】

int rk\_watchdog\_start(int timeval);

【参数】

参数名称	描述	输入/输出
timeval	大于0会重新设置超时时间，否则使用默认时间	输入

【返回值】

返回值	描述
0	成功
非0	失败

【需求】

头文件：rk\_watchdog.h

【注意】

- 不支持重复初始化。

【举例】

rk\_watchdog\_test



【相关主题】

[rk\\_watchdog\\_stop](#)

## 8.2.2 rk\_watchdog\_refresh

【描述】

喂狗，刷新watchdog时间。

【语法】

```
int rk_watchdog_refresh(void);
```

【返回值】

返回值	描述
0	成功
非0	失败

【需求】

头文件：rk\_watchdog.h

【注意】

- rk\_watchdog\_start开启后才能使用rk\_watchdog\_refresh。

【举例】

```
rk_watchdog_test
```

【相关主题】

无

## 8.2.3 rk\_watchdog\_stop

【描述】

关闭watchdog，内核喂狗。

【语法】

```
int rk_watchdog_stop(void);
```

【返回值】

返回值	描述
0	成功
非0	失败

【需求】

头文件：rk\_watchdog.h

【举例】

rk\_watchdog\_test

【相关主题】

[rk\\_watchdog\\_start](#)

# 9. SYSTEM

---

## 9.1 概述

提供系统操作的用户态接口。包括设备重启，休眠，关机和chipid以及SN号的获取。

## 9.2 API参考

### 9.2.1 rk\_chip\_id\_get

【描述】

获取chipid。

【语法】

int rk\_chip\_id\_get(char \*chipid);

【参数】

参数名称	描述	输入/输出
chipid	保存chipid	输出

【返回值】

返回值	描述
0	成功
非0	失败

【需求】

头文件：rk\_system.h

【举例】

rk\_system\_test

【相关主题】

无

### 9.2.2 rk\_vendor\_write

**【描述】**

向vendor写数据。

**【语法】**

int rk\_vendor\_write(int vendor\_id, const char \*data, int size);

**【参数】**

参数名称	描述	输入/输出
vendor_id	编号	输入
data	待写入数据	输入
size	待写入数据大小	输入

**【返回值】**

返回值	描述
0	成功
非0	失败

**【需求】**

头文件：rk\_system.h

**【注意】**

- 必须保证/dev/vendor\_storage存在。
- size的大小建议在1024字节内

**【举例】**

rk\_system\_test

**【相关主题】**

[rk\\_vendor\\_read](#)

### 9.2.3 rk\_vendor\_read

**【描述】**

读取vendor的数据。

**【语法】**

int rk\_vendor\_write(int vendor\_id, const char \*data, int size);

【参数】

参数名称	描述	输入/输出
vendor_id	编号	输入
data	保存读取数据	输出
size	读取数据大小	输入

【返回值】

返回值	描述
0	成功
非0	失败

【需求】

头文件：rk\_system.h

【注意】

- 必须保证/dev/vendor\_storage存在。

【举例】

rk\_system\_test

【相关主题】

[rk\\_vendor\\_write](#)

9.2.4 rk\_system\_reboot

【描述】

系统重启。

【语法】

int rk\_system\_reboot(void);

【返回值】

返回值	描述
0	成功
非0	失败

【需求】

头文件：rkadk\_storage.h

【注意】

- 在调用此函数之前请确保已保存所有文件。

**【举例】**

rk\_system\_test

**【相关主题】**

无

## 9.2.5 rk\_system\_shutdown

**【描述】**

系统关机。

**【语法】**

```
int rk_system_shutdown(void);
```

**【返回值】**

返回值	描述
0	成功
非0	失败

**【需求】**

头文件：rk\_system.h

**【举例】**

rk\_system\_test

**【相关主题】**

无

## 9.2.6 rk\_system\_suspend

**【描述】**

系统休眠。

**【语法】**

```
int rk_system_suspend(SUSPEND\_TYPE type);
```

**【参数】**

参数名称	描述	输入/输出
type	休眠类型	输入

**【返回值】**

返回值	描述
0	成功
非0	失败

【需求】

头文件：rk\_system.h

【注意】

- FREEZE与MEM的休眠方式需要查看系统是否支持

【举例】

rk\_systmp\_test

【相关主题】

无

## 9.2.7 数据类型

SYSTEM参数主要提供以下数据类型：

### 9.2.7.1 SUSPEND\_TYPE

【说明】

休眠类型。

【定义】

```
typedef enum {
    SUSPEND_FREEZE = 0,
    SUSPEND_MEM,
} SUSPEND_TYPE;
```

【成员】

成员名称	描述
SUSPEND_FREEZE	FREEZE休眠方式
SUSPEND_MEM	MEM休眠方式

## 10. MOTOR

---

# 10.1 概述

基于24byj48型号电机，提供用户态接口，同时控制两个电机的运转。

## 10.2 API参考

### 10.2.1 rk\_motor\_init

【描述】

motor初始化。

【语法】

```
int rk_motor_init(struct motors\_init\_data *data);
```

【参数】

参数名称	描述	输入/输出
data	初始化参数	输入

【返回值】

返回值	描述
0	成功
非0	失败

【需求】

头文件：rk\_motor.h

【注意】

- 需要保证dts配置正确， /dev/motor节点存在， 8个gpio的命名规则为motorA-gpios到motorH-gpios。
- x方向的gpio为motorA-gpios到motorD-gpios， y方向的gpio为motorE-gpios到motorH-gpios。

【举例】

```
rk_motor_test
```

【相关主题】

```
rk\_motor\_deinit
```

### 10.2.2 rk\_motor\_deinit

【描述】

motor反初始化

【语法】

```
int rk_motor_deinit(void);
```

【返回值】

返回值	描述
0	成功
非0	失败

【需求】

头文件：rk\_motor.h

【注意】

- 调用rk\_motor\_init初始化motor模块后，才能调用该接口。

【举例】

```
rk_motor_test
```

【相关主题】

[rk\\_motor\\_init](#)

10.2.3 rk\_motor\_move

【描述】

同时转动两个电机，输入为负表示反向转动。

【语法】

```
int rk_motor_move(struct motors_input_steps input_steps);
```

【参数】

参数名称	描述	输入/输出
input_steps	两个电机转动的步距，限制在最大步距之内	输入

【返回值】

返回值	描述
0	成功
非0	失败

【需求】

头文件：rk\_motor.h

【注意】

- 调用rk\_motor\_init初始化motor模块后，才能调用该接口。
- 反向转动最多转动到原点。



【举例】

rk\_motor\_test

【相关主题】

[rk\\_motor\\_stop](#)

10.2.4 rk\_motor\_stop

【描述】

同时停止两个电机。

【语法】

int rk\_motor\_stop(void);

【返回值】

返回值	描述
0	成功
非0	失败

【需求】

头文件：rk\_motor.h

【注意】

- 调用rk\_motor\_init初始化motor模块后，才能调用该接口。

【举例】

rk\_motor\_test

【相关主题】

[rk\\_motor\\_move](#)

10.2.5 rk\_motor\_get\_status

【描述】

获取当前两个电机的信息，包括当前步距、运行状态、速度。

【语法】

int rk\_motor\_get\_status(struct motor\_message \*message);

【参数】

参数名称	描述	输入/输出
message	保存获取的电机数据	输出

**【返回值】**

返回值	描述
0	成功
非0	失败

**【需求】**

头文件：rk\_motor.h

**【注意】**

- 调用rk\_motor\_init初始化motor模块后，才能调用该接口。

**【举例】**

rk\_motor\_test

**【相关主题】**

无

**10.2.6 rk\_motor\_speed**

**【描述】**

同时调整两个电机的速度，该速度描述的是电机转动单位步距所需要的时间

**【语法】**

int rk\_motor\_speed(struct motors\_input\_speed input\_speed);

**【参数】**

参数名称	描述	输入/输出
input_speed	电机速度	输入

**【返回值】**

返回值	描述
0	成功
非0	失败

**【需求】**

头文件：rk\_motor.h

**【注意】**

- 调用rk\_motor\_init初始化motor模块后，才能调用该接口。
- 是否正在转动中不影响电机变速
- 速度存在限制，最慢为9000000，最快为1800000，单位为微秒，数值越小速度越快

【举例】

rk\_motor\_test

【相关主题】

无

10.2.7 rk\_motor\_reset

【描述】

同时重置两个电机，转动到用户输入的步距。

【语法】

int rk\_motor\_reset([struct motor\\_reset\\_data](#) reset\_data);

【参数】

参数名称	描述	输入/输出
reset_data	重置信息	输入

【返回值】

返回值	描述
0	成功
非0	失败

【需求】

头文件：rk\_motor.h

【注意】

- 调用rk\_motor\_init初始化motor模块后，才能调用该接口。

【举例】

rk\_motor\_test

【相关主题】

[rk\\_motor\\_goback](#)

10.2.8 rk\_motor\_goback

【描述】

回到reset时设置的位置。

【语法】

int rk\_motor\_goback(void);

【返回值】

返回值	描述
0	成功
非0	失败

【需求】

头文件: rk\_motor.h

【注意】

- 调用rk\_motor\_init初始化motor模块后，才能调用该接口。

【举例】

rk\_motor\_test

【相关主题】

[rk\\_motor\\_reset](#)

10.2.9 数据类型

MOTOR参数主要提供以下数据类型：

10.2.9.1 motors\_init\_data

【说明】

motor初始化数据。

【定义】

```
struct motors_init_data {
    struct motor_reset_data motor_data;
    struct motors_input_speed motor_speed;
};
```

【成员】

成员名称	描述
struct motor_reset_data	motor重置参数
struct motors_input_speed	motor速度

### 10.2.9.2 motor\_reset\_data

#### 【说明】

motor重置参数。

#### 【定义】

```
struct motor_reset_data {  
    unsigned int x_max_steps;  
    unsigned int y_max_steps;  
    unsigned int reset_x;  
    unsigned int reset_y;  
};
```

#### 【成员】

成员名称	描述
x_max_steps	水平电机的最大步距
y_max_steps	垂直电机的最大步距
reset_x	水平电机重置后停留的位置
reset_y	垂直电机重置后停留的位置