Rockchip Sysutil Developer Guide

ID: RK-KF-YF-929

Release Version: V1.0.2

Release Date: 2023-08-04

Security Level: □Top-Secret □Secret □Internal ■Public

Rockchip Electronics Co., Ltd.

No.18 Building, A District, No.89, software Boulevard Fuzhou, Fujian,PRC

Website: www.rock-chips.com

Customer service Tel: +86-4007-700-590

Customer service Fax: +86-591-83951833

Customer service e-Mail: fae@rock-chips.com

Preface

Overview

This document mainly presents the reference for Sysutil component development.

Product Version

Chipset	**Kernel Version**
RV1106	Linux 5.10

Intended Audience

This document (this guide) is mainly intended for:

Technical support engineers

Software development engineers

Revision History

Version	**Author**	**Date**	**Change Description**
V1.0.0	ZZW	2022-04-08	Initial version
V1.0.1	LZW	2022-11-28	Update input and output data type of pins
V1.0.2	LZW	2023-08-04	Update the name of ADC interface

Contents

1. Overview

Sysutil is a group of user mode interfaces package based on sysfs, including peripheral interface and system functional interfaces, which is convenient for application controlling peripheral device and systems. It simplifies the difficulty of application development and facilitates customers for application development based on these hardware interfaces.

2. GPIO

2.1 Overview

This chapter will provide basic GPIO user-mode interfaces.

2.2 API Reference

2.2.1 rk_gpio_export

【Description】

To export GPIO pins that required to be controlled.

【Grammar】

int rk_gpio_export(uint32_t gpio);

【Parameters】

Parameter name	Description	Input/Output
GPIO	GPIO pin number	Input

【Return Value】

Return value	Description
0	Success
Other values	Failure

【Requirements】

Header file: rk_gpio.h

【Notice】

- Check whether there is /SYS/Class/GPIO node in the file system. If it didn't exist, Device Drivers-> GPIO Support->/SYS/Class/GPIO/… (sysfs interface) should be selected when building, the corresponding CONFIG name is GPIO_SYSFS.
- gpio pins should be in a state of idle.

【Example】

rk_gpio_test

【Related Topic】

rk_gpio_unexport

2.2.2 rk_gpio_unexport

【Description】

Notify system to cancel the export.

【Grammar】

int rk_gpio_unexport(uint32_t gpio);

【Parameters】

Parameter name	**Description**	**Input/Output**
gpio	gpio pin number	Input

【Return Value】

Return value	**Description**
0	Success
Other values	Failure

【Requirement】

Header file: rk_gpio.h

【Notice】

- Used to be called after rk_gpio_export is successful.

【Example】

rk_gpio_test

【Related Topic】

rk_gpio_export

2.2.3 rk_gpio_set_direction

【Description】

Define the input and output directions.

【Grammar】

int rk_gpio_set_direction(uint32_t gpio, enum gpio_direction input);

【Parameters】

Parameter Name	Description	Input/Output
gpio	gpio pin number	Input
input	GPIO_DIRECTION_INPUT: in GPIO_DIRECTION_OUTPUT: out	Input

【Return Value】

Return value	Description
0	Success
Other values	Failure

【Requirement】

Header file: rk_gpio.h

【Notice】

- The rk_gpio_set_direction interface can only be used after rk_gpio_export is successful.

【Example】

rk_gpio_test

【Related Topic】

rk_gpio_get_direction

2.2.4 rk_gpio_get_direction

【Description】

Get direction information of gpio pins.

【Grammar】

int rk_gpio_get_direction(uint32_t gpio);

【Parameters】

Parameter Name	Description	Input/Output
gpio	gpio pin number	Input

【Return Value】

Return value	Description
0 or 1	0:out, 1:in
Negative	Failure

【Requirement】

Header file: rk_gpio.h

【Notice】

- The rk_gpio_get_direction interface can only be used after the rk_gpio_export is successful.

【Example】

rk_gpio_test

【Related Topic】

rk_gpio_set_direction

2.2.5 rk_gpio_export_direction

【Description】

Initialize gpio, specify the direction of gpio while exporting gpio.

【Grammar】

int rk_gpio_export_direction(uint32_t gpio, enum gpio_direction input);

【Parameters】

Parameter Name	Description	Input/Output
gpio	gpio pin number	Input
input	GPIO_DIRECTION_INPUT: in GPIO_DIRECTION_OUTPUT: out	Input

【Return Value】

Return value	Description
0	Success
Other values	failure

【Requirement】

Header file: rk_gpio.h

【Example】

rk_gpio_test

【Related Topic】

None

2.2.6 rk_gpio_set_value

【Description】

Set the value of a gpio pin.

【Grammar】

int rk_gpio_set_value(uint32_t gpio, int value);

【Parameters】

Parameter Name	**Description**	**Input/Output**
gpio	gpio pin number	Input
value	value to be set to 1/0	Input

【Return Value】

Return value	**Description**
0	Success
Other values	Failure

【Requirement】

Header file: rk_gpio.h

【Notice】

- The rk_gpio_set_value interface can only be used after the rk_gpio_export is successful.

【Example】

rk_gpio_test

【Related Topic】

rk_gpio_get_value

2.2.7 rk_gpio_get_value

【Description】

Get the value of a gpio pin.

【Grammar】

int rk_gpio_get_value(uint32_t gpio);

【Parameters】

Parameter Name	Description	Input/Output
gpio	gpio pin number	Input

【Return Value】

Return value	Description
0 or 1	Success
Negative	Failure

【Requirement】

Header file: rk_gpio.h

【Example】

rk_gpio_test

【Related Topic】

None

2.2.8 Data Type

The following data types can be used for GPIO input and output.

2.2.8.1 enum gpio_direction

【Introduction】

Direction of GPIO input and output

【Definition】

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

【Members】

Member Name	Description
GPIO_DIRECTION_OUTPUT	Output
GPIO_DIRECTION_INPUT	Input

rk_gpio_set_value

3. ADC

3.1 Overview

Provide basic ADC user mode interface

3.2 API Reference

3.2.1 rk_adc_get_devnum

【Description】

Obtain the device number from the device name.

【Grammars】

int rk_adc_get_devnum(const char *name);

【Parameters】

Parameter Name	Description	Input/Output
Name	The name of the device	Input

【Return Value】

Return value	Description
non-negative	Device number
Negative	Failure

【Requirement】

Header file: rk_adc.h

【Notice】

- Should be perform after saradc initialization.

【Example】

rk_adc_test

【Related Topic】

None

3.2.2 rk_adc_get_value

【Description】

Read the raw data collected by the channel AD.

【Grammar】

int rk_adc_get_value(uint32_t dev_num, uint32_t chn_num);

【Parameters】

Parameter Name	**Description**	**Input/Output**
dev_num	Device number	Input
chn_num	IO channel used by this device	Input

【Return Value】

Return value	**Description**
Non-negative	Value
Negative	Failure

【Requirement】

Header file: rk_adc.h

【Example】

rk_adc_test

【Related Topic】

None

4. EVENT

4.1 Overview

Monitor key events and other events according to the standard way of kernel input event.

4.2 API Reference

4.2.1 rk_event_register

【Description】

Register the nodes that require be monitored.

【Grammar】

int rk_event_register(char *dev_path);

【Parameters】

Parameter Name	Description	Input/Output
dev_path	Path of node	Input

【Return Value】

Return value	Description
0	Success
Other values	failure

【Requirement】

Header file: rk_event.h

【Notice】

- Does not support repeated registration
- Should ensure that the node exists

【Example】

rk_event_test

【Related Topic】

rk_event_unregister

4.2.2 rk_event_unregister

【Description】

Unregister the nodes that require to be monitored.

【Grammar】

int rk_event_unregister(char *dev_path);

【Parameters】

Parameter Name	Description	Input/Output
dev_path	Path of node	Input

【Return Value】

return value	description
0	Success
Other values	Failure

【Requirement】

Header file: rk_event.h

【Notice】

- rk_event_unregister can only be called after rk_event_register is registered

【Example】

rk_event_test

【Related Topic】

rk_event_register

4.2.3 rk_event_listen_start

【Description】

Start event monitoring.

【Grammar】

int rk_event_listen_start(event_handler_t handler);

【Parameters】

Parameter Name	Description	Input/Output
handler	Monitor event callback function	Input

【Return Value】

Return value	**Description**
0	Success
Other values	Failure

【Requirement】

Header file: rk_event.h

【Notice】

- rk_event_listen_start can only be called after rk_event_register.
- Does not support repeated rk_event_listen_start
- After rk_event_listen_start, it is not supported to register the monitor node, but the monitor node can be unregistered
- After calling rk_event_listen_start, the event callback function is triggered to start receiving data.

【Example】

rk_event_test

【Related Topic】

rk_event_listen_stop

4.2.4 rk_event_listen_stop

【Description】

Stop event monitoring.

【Grammar】

int rk_event_listen_stop(void);

【Return Value】

Return value	**Description**
0	Success
Other values	Failure

【Requirement】

Header file: rk_event.h

【Notice】

- After stopping monitoring, all monitoring nodes will be cleared, and re-registration is required

【Example】

rk_event_test

【Related Topic】

rk_event_listen_start

5. PWM

5.1 Overview

Provide basic pwm user mode interface.

5.2 API Reference

5.2.1 rk_pwm_export

【Description】

Export the PWM pins that require to be controlled.

【Grammar】

int rk_pwm_export(uint32_t pwm);

【Parameters】

Parameter Name	Description	Input/Output
pwm	pwm number	Input

【Return Value】

Return value	Description
0	Success
Other values	failure

【Requirement】

Header file: rk_pwm.h

【Notice】

- You should check whether there is a /sys/class/pwm node in the file system. If there is no such node, you should check Device Drivers->Pulse-Width Modulation (PWM) Support when building the kernel, and the corresponding CONFIG name is PWM.

【Example】

rk_pwm_test

【Related Topic】

rk_pwm_unexport

5.2.2 rk_pwm_unexport

【Description】

Notify the system to cancel exporting pwm.

【Grammar】

int rk_pwm_unexport(uint32_t pwm);

【Parameters】

Parameter Name	Description	Input/Output
pwm	pwm number	Input

【Return Value】

Return value	Description
0	Success
Other values	Failure

【Requirement】

Header file: rk_pwm.h

【Notice】

- Used to call after rk_pwm_export is successful.

【Example】

rk_pwm_test

【Related Topic】

rk_pwm_export

5.2.3 rk_pwm_set_period

【Description】

Set the pwm period.

【Grammar】

int rk_pwm_set_period(uint32_t pwm, uint32_t period);

【Parameters】

Parameter Name	Description	Input/Output
pwm	pwm number	Input
period	Period duration (nanoseconds)	Input

【Return Value】

Return value	Description
0	Success
Other values	Failure

【Requirement】

Header file: rk_pwm.h

【Notice】

- The period duration does not exceed the 9th power of 10

【Example】

rk_pwm_test

【Related Topic】

rk_pwm_get_period

5.2.4 rk_pwm_get_period

【Description】

Get the pwm period.

【Grammar】

int rk_pwm_get_period(uint32_t pwm);

【Parameters】

Parameter Name	Description	Input/Output
pwm	pwm number	Input

【Return Value】

Return value	Description
non-negative	Period duration (nanoseconds)
Negative	Failure

【Requirement】

Header file: rk_pwm.h

【Example】

rk_pwm_test

【Related Topic】

rk_pwm_set_period

5.2.5 rk_pwm_set_duty

【Description】

Configure the duty ratio of pwm.

【Grammar】

int rk_pwm_set_duty(uint32_t pwm, uint32_t duty);

【Parameters】

Parameter Name	**Description**	**Input/Output**
pwm	pwm number	Input
duty	duty ratio duration (nanoseconds)	Input

【Return Value】

Return value	**Description**
0	Success
Other values	Failure

【Requirement】

Header file: rk_pwm.h

【Notice】

- The duty ratio duration does not exceed the 9th power of 10, and does not exceed the currently set period duration

【Example】

rk_pwm_test

【Related Topic】

rk_pwm_get_duty

5.2.6 rk_pwm_get_duty

【Description】

Get the pwm duty ratio duration.

【Grammar】

int rk_pwm_get_duty(uint32_t pwm);

【Parameters】

Parameter Name	Description	Input/Output
pwm	pwm number	Input

【Return Value】

Return value	Description
Non-negative	Duty ratio duration (nanoseconds)
Negative	Failure

【Requirement】

Header file: rk_pwm.h

【Example】

rk_pwm_test

【Related Topic】

rk_pwm_set_duty

5.2.7 rk_pwm_set_polarity

【Description】

Set the polarity of pwm.

【Grammar】

int rk_pwm_set_polarity(uint32_t pwm, enum pwm_polarity polarity);

【Parameters】

Parameter Name	Description	Input/Output
pwm	pwm pin number	Input
polarity	Polarity	Input

【Return Value】

Return value	Description
0	Success
Other values	Failure

【Requirement】

Header file: rk_pwm.h

【Example】

rk_pwm_test

【Related Topic】

rk_pwm_get_period

5.2.8 rk_pwm_get_polarity

【Description】

Get the polarity of a pwm pin.

【Grammar】

int rk_pwm_get_polarity(uint32_t pwm);

【Parameters】

Parameter Name	Description	Input/Output
pwm	pwm pin number	Input

【Return Value】

Return value	Description
0 or 1	0: positive polarity, 1: negative polarity
Negative	Failure

【Requirement】

Header file: rk_pwm.h

【Example】

rk_pwm_test

【Related Topic】

rk_pwm_set_polarity

5.2.9 rk_pwm_set_enable

【Description】

Enable pwm.

【Grammar】

int rk_pwm_set_enable(uint32_t pwm, bool enabled);

【Parameters】

Parameter Name	Description	Input/Output
pwm	pwm pin number	Input
enabled	Write 1 to enable pwm, write 0 to disable pwm	Input

【Return Value】

Return value	Description
0	Success
Other values	Failure

【Requirement】

Header file: rk_pwm.h

【Example】

rk_pwm_test

【Related Topic】

rk_pwm_get_enable

5.2.10 rk_pwm_get_enable

【Description】

Check whether the pwm pin is enabled.

【Grammar】

int rk_pwm_get_enable(uint32_t pwm);

【Parameters】

Parameter Name	Description	Input/Output
pwm	pwm pin number	Input

【Return Value】

return value	description
0 or 1	0: disable, 1: enable
Negative	Failure

【Requirement】

Header file: rk_pwm.h

【Example】

rk_pwm_test

【Related Topic】

rk_pwm_set_enable

5.2.11 rk_pwm_init

【Description】

Initialize pwm, and set the period, duty ratio and polarity of pwm pin at the same time.

【Grammar】

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

【Parameters】

Parameter Name	Description	Input/Output
pwm	pwm pin number	Input
period	period duration (nanoseconds)	Input
duty	duty ratio duration (nanoseconds)	Input
polarity	polarity	Input

【Return Value】

Return value	Description
0	Success
Other values	Failure

【Requirement】

Header file: rk_pwm.h

【Example】

rk_pwm_test

【Related Topic】

rk_pwm_deinit

5.2.12 rk_pwm_deinit

【Description】

De-initialize pwm and notify the system to cancel the export.

【Grammar】

int rk_pwm_deinit(uint32_t pwm);

【Parameters】

Parameter Name	Description	Input/Output
pwm	pwm pin number	Input

【Return Value】

Return value	Description
0	Success
Other values	Failure

【Requirement】

Header file: rk_pwm.h

【Notice】

- rk_pwm_deinit can only be called after rk_pwm_init is called successfully.

【Example】

rk_pwm_test

【Related Topic】

rk_pwm_init

5.2.13 Data Type

PWM parameters mainly provide the following data types:

5.2.13.1 enum pwm_polarity

【Introduction】

The polarity of pwm.

【Definition】

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

【Members】

Member Name	Description
PWM_POLARITY_NORMAL	Positive Polarity
PWM_POLARITY_INVERSED	Negative Polarity

6. TIME

6.1 Overview

Provides user mode interfaces for hardware time and system time.

6.2 API Reference

6.2.1 rk_system_get_time

【Description】

Get hardware time.

【Grammar】

int rk_system_get_time(struct tm *time);

【Parameters】

Parameter Name	Description	Input/Output
time	Save the time obtained	Output

【Return Value】

Return value	Description
0	Success
Other values	Failure

【Requirement】

Header file: rk_time.h

【Notice】

- You have to make sure that the /dev/rtc node is enabled. If it is not enabled, you should select Device Drivers-> Real Time Clock when building kernel, and the corresponding CONFIG name is RTC_CLASS.

【Example】

rk_time_test

【Related Topic】

rk_system_set_time

6.2.2 rk_system_set_time

【Description】

Modify the hardware time and update the system time.

【Grammar】

int rk_system_set_time(struct tm *time);

【Parameters】

Parameter Name	Description	Input/Output
time	Target modification time	Input

【Return Value】

Return value	Description
0	Success
Other values	Failure

【Requirement】

Header file: rk_time.h

【Notice】

- You have to make sure that the /dev/rtc node is enabled. If it is not enabled, you should check Device Drivers-> Real Time Clock when building kernel, and the corresponding CONFIG name is RTC_CLASS.
- Make sure to enter a valid time

【Example】

rk_time_test

【Related Topic】

rk_system_get_time

6.2.3 rk_system_set_alarm

【Description】

Set the trigger time of alarm interrupt.

【Grammar】

rk_system_set_alarm(struct tm *time);

【Parameters】

Parameter Name	Description	Input/Output
time	Target setting time	Input

【Return Value】

Return value	Description
0	Success
Other values	Failure

【Requirement】

Header file: rk_time.h

【Notice】

- The trigger time of the alarm interrupt can only be a moment within 24 hours, so only the hours, minutes, and seconds are valid, and the year, month, and day of the parameter time will be ignored.

【Example】

rk_time_test

【Related Topic】

rk_system_get_alarm

6.2.4 rk_system_get_alarm

【Description】

Read the alarm interrupt time that has been set.

【Grammar】

int rk_system_get_alarm(struct tm *time);

【Parameters】

Parameter Name	Description	Input/Output
time	Save the read time	Output

【Return Value】

Return value	Description
0	Success
Other values	Failure

【Requirement】

Header file: rk_time.h

【Notice】

- rk_system_get_alarm should be called after rk_system_set_alarm is set successful.

【Example】

rk_time_test

【Related Topic】

rk_system_set_alarm

6.2.5 rk_system_enable_alarm

【Description】

Enable the alarm that has been set.

【Grammar】

int rk_system_enable_alarm(void);

【Return Value】

Return value	**Description**
0	Success
Other values	Failure

【Requirement】

Header file: rk_time.h

【Notice】

- rk_system_enable_alarm should be called after rk_system_set_alarm is successful.
- The trigger time of alarm is not supported to set repeatedly after enabling.

【Example】

rk_time_test

【Related Topic】

rk_system_disable_alarm

6.2.6 rk_system_disable_alarm

【Description】

Disable the alarm interrupt.

【Grammar】

int rk_system_disable_alarm(void);

【Return Value】

Return value	Description
0	Success
Other values	Failure

【Requirement】

Header file: rk_time.h

【Notice】

- It should be called after calling rk_system_enable_alarm, the interrupt trigger time that has been set will not be changed.

【Example】

rk_time_test

【Related Topic】

rk_system_enable_alarm

6.2.7 rk_system_wait_alarm

【Description】

Wait for device node interrupt to wake up.

【Grammar】

int rk_system_wait_alarm(uint32_t wait_seconds);

【Parameters】

Parameter Name	Description	Input/Output
wait_seconds	The duration of waiting timeout	Input

【Return Value】

Return value	Description
0	Success
Other values	Failure

【Requirement】

Header file: rk_time.h

【Notice】

- It will block the current thread, and be called after rk_system_enable_alarm is called, the timeout unit is second

【Example】

rk_time_test

【Related Topic】

rk_system_enable_alarm

rk_system_set_time

7. LED

7.1 Overview

At present, led only provides the functions of adjusting brightness and starting and stopping flashing

7.2 API Reference

7.2.1 rk_led_set_mode

【Description】

LED interface for adjusting brightness and starting flashing.

【Grammar】

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

【Parameters】

Parameter Name	**Description**	**Input/Output**
dev_path	Led node path	Input
blink	Whether to flash	Input
brightness	Brightness	Input

【Return Value】

Return value	**Description**
0	Success
Other values	Failure

【Requirement】

Header file: rk_led.h

【Notice】

- You need to make sure that there is a /sys/class/leds device node. If it does not exist, you should select `Device Drivers-> LED Support -> LED Class Support` when building kernel, and the corresponding CONFIG name is LEDS_CLASS.

【Example】

rk_led_test

【Related Topic】

None

8. WATCHDOG

8.1 Overview

Provide the user mode interface of watchdog.

8.2 API Reference

8.2.1 rk_watchdog_start

【Description】

Enable watchdog, set timeout optionally.

【Grammar】

int rk_watchdog_start(int timeval);

【Parameters】

Parameter Name	Description	Input/Output
timeval	Greater than 0 will reset the timeout, otherwise use the default time	Input

【Return Value】

Return value	Description
0	Success
Other values	Failure

【Requirement】

Header file: rk_watchdog.h

【Notice】

- Repeated initialization is not supported.

【Example】

rk_watchdog_test

【Related Topic】

rk_watchdog_stop

8.2.2 rk_watchdog_refresh

【Description】

Feed the dog and refresh the watchdog time.

【Grammar】

int rk_watchdog_refresh(void);

【Return Value】

Return value	**Description**
0	Success
Other values	Failure

【Requirement】

Header file: rk_watchdog.h

【Notice】

- rk_watchdog_refresh can only be used after rk_watchdog_start is enabled.

【Example】

rk_watchdog_test

【Related Topic】

None

8.2.3 rk_watchdog_stop

【Description】

Close watchdog, dog is fed by kernel.

【Grammar】

int rk_watchdog_stop(void);

【Return Value】

Return value	Description
0	Success
Other values	Failure

【Requirement】

Header file: rk_watchdog.h

【Example】

rk_watchdog_test

【Related Topic】

rk_watchdog_start

9. SYSTEM

9.1 Overview

Provides a user-mode interface for system operations. Including device restart, sleep, shutdown and getting of chipid and SN number.

9.2 API Reference

9.2.1 rk_chip_id_get

【Description】

Get chipid.

【Grammar】

int rk_chip_id_get(char *chipid);

【Parameters】

Parameter Name	Description	Input/Output
chipid	Save chipid	Output

【Return Value】

Return value	Description
0	Success
Other values	Failure

【Requirement】

Header file: rk_system.h

【Example】

rk_system_test

【Related Topic】

None

9.2.2 rk_vendor_write

【Description】

Write data to vendor.

【Grammar】

int rk_vendor_write(int vendor_id, const char *data, int size);

【Parameters】

Parameter Name	**Description**	**Input/Output**
vendor_id	Number	Input
data	Data to be written	Input
size	Size of data to be written	Input

【Return Value】

Return value	**Description**
0	Success
Other values	Failure

【Requirement】

Header file: rk_system.h

【Notice】

- Make sure that /dev/vendor_storage exists.
- The size is recommended to be within 1024 bytes

【Example】

rk_system_test

【Related Topic】

rk_vendor_read

9.2.3 rk_vendor_read

【Description】

Read vendor data.

【Grammar】

int rk_vendor_write(int vendor_id, const char *data, int size);

【Parameters】

Parameter Name	Description	Input/Output
vendor_id	Number	Input
data	Save the read data	Output
size	Read data size	Input

【Return Value】

Return value	Description
0	Success
Other values	Failure

【Requirement】

Header file: rk_system.h

【Notice】

- Make sure that /dev/vendor_storage exists.

【Example】

rk_system_test

【Related Topic】

rk_vendor_write

9.2.4 rk_system_reboot

【Description】

The system restarts.

【Grammar】

int rk_system_reboot(void);

【Return Value】

Return value	Description
0	Success
Other values	Failure

【Requirement】

Header file: rkadk_storage.h

【Notice】

- Make sure all files are saved before calling this function.

【Example】

rk_system_test

【Related Topic】

None

9.2.5 rk_system_shutdown

【Description】

Shuts down the system

【Grammar】

int rk_system_shutdown(void);

【Return Value】

Return value	Description
0	Success
Other values	Failure

【Requirement】

Header file: rk_system.h

【Example】

rk_system_test

【Related Topic】

None

9.2.6 rk_system_suspend

【Description】

Suspend the system

【Grammar】

int rk_system_suspend(SUSPEND_TYPE type);

【Parameters】

Parameter Name	Description	Input/Output
type	Suspend type	Input

【Return Value】

Return value	Description
0	Success
Other values	Failure

【Requirement】

Header file: rk_system.h

【Notice】

- You should check whether the system supports the suspend mode of FREEZE and MEM

【Example】

rk_systmp_test

【Related Topic】

None

9.2.7 Type of Data

The SYSTEM parameter mainly provides the following data types:

9.2.7.1 SUSPEND_TYPE

【Introduction】

Suspend type.

【Definition】

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

【Members】

Member Name	Description
SUSPEND_FREEZE	FREEZE suspend mode
SUSPEND_MEM	MEM suspend mode

10. MOTOR

10.1 Overview

Based on the 24byj48 motor, it provides a user mode interface and controls the operation of two motors at the same time.

10.2 API Reference

10.2.1 rk_motor_init

【Description】

Motor initialization.

【Grammar】

int rk_motor_init(struct motors_init_data *data);

【Parameters】

Parameter Name	Description	Input/Output
data	Initialization parameters	Input

【Return Value】

Return value	Description
0	Success
Other values	Failure

【Requirement】

Header file: rk_motor.h

【Notice】

- It is necessary to ensure that the dts configuration is correct, the /dev/motor node exists, and the naming rules of the 8 gpios are motorA-gpios to motorH-gpios.
- The gpios in the x direction are motorA-gpios to motorD-gpios, and the gpios in the y direction are motorE-gpios to motorH-gpios.

【Example】

rk_motor_test

【Related Topic】

rk_motor_deinit

10.2.2 rk_motor_deinit

【Description】

Motor de-initialization

【Grammar】

int rk_motor_deinit(void);

【Return Value】

Return value	Description
0	Success
Other values	Failure

【Requirement】

Header file: rk_motor.h

【Notice】

- This interface can only be called after calling rk_motor_init to initialize the motor module.

【Example】

rk_motor_test

【Related Topic】

rk_motor_init

10.2.3 rk_motor_move

【Description】

Turn both motors at the same time, negative input means reverse rotation.

【Grammar】

int rk_motor_move(struct motors_input_steps input_steps);

【Parameters】

Parameter Name	Description	Input/Output
input_steps	The steps of the two motors, limited to the maximum step	Input

【Return Value】

Return value	Description
0	Success
Other values	Failure

【Requirement】

Header file: rk_motor.h

【Notice】

- This interface can only be called after calling rk_motor_init to initialize the motor module.
- Reverse rotation can only be rotated to the origin at most.

【Example】

rk_motor_test

【Related Topic】

rk_motor_stop

10.2.4 rk_motor_stop

【Description】

Stop both motors at the same time.

【Grammar】

int rk_motor_stop(void);

【Return Value】

Return value	Description
0	Success
Other values	Failure

【Requirement】

Header file: rk_motor.h

【Notice】

- This interface can only be called after calling rk_motor_init to initialize the motor module.

【Example】

rk_motor_test

【Related Topic】

rk_motor_move

10.2.5 rk_motor_get_status

【Description】

Obtain the information of the current two motors, including the current step distance, running status, and speed.

【Grammar】

int rk_motor_get_status(struct motor_message *message);

【Parameters】

Parameter Name	**Description**	**Input/Output**
message	Save acquired motor data	Output

【Return Value】

Return value	**Description**
0	Success
Other values	Failure

【Requirement】

Header file: rk_motor.h

【Notice】

- This interface can only be called after calling rk_motor_init to initialize the motor module.

【Example】

rk_motor_test

【Related Topic】

None

10.2.6 rk_motor_speed

【Description】

Adjust the speed of both motors at the same time, which describes the time required for the motor to rotate a unit step.

【Grammar】

int rk_motor_speed(struct motors_input_speed input_speed);

【Parameters】

Parameter Name	Description	Input/Output
input_speed	Motor speed	Input

【Return Value】

return value	description
0	Success
Other values	Failure

【Requirement】

Header file: rk_motor.h

【Notice】

- This interface can only be called after calling rk_motor_init to initialize the motor module.
- Whether it is running does not affect the motor speed changing
- The speed is limited, the slowest is 9000000, the fastest is 1800000, the unit is microseconds, the smaller the value, the faster the speed

【Example】

rk_motor_test

【Related Topic】

None

10.2.7 rk_motor_reset

【Description】

Resets both motors at the same time, change to the user-entered step size.

【Grammar】

int rk_motor_reset(struct motor_reset_data reset_data);

【Parameters】

Parameter Name	Description	Input/Output
reset_data	Reset info	Input

【Return Value】

Return value	Description
0	Success
Other values	Failure

【Requirement】

Header file: rk_motor.h

【Notice】

- This interface can only be called after calling rk_motor_init to initialize the motor module.

【Example】

rk_motor_test

【Related Topic】

rk_motor_goback

10.2.8 rk_motor_goback

【Description】

Return to the position where is reset.

【Grammar】

int rk_motor_goback(void);

【Return Value】

Return value	Description
0	Success
Other values	Failure

【Requirement】

Header file: rk_motor.h

【Notice】

- This interface can only be called after calling rk_motor_init to initialize the motor module.

【Example】

rk_motor_test

【Related Topic】

rk_motor_reset

10.2.9 Type of Data

MOTOR parameters mainly provide the following data types:

10.2.9.1 motors_init_data

【Introduction】

Motor initialization data.

【Definition】

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

【Members】

Member Name	Description
struct motor_reset_data	Motor reset parameters
struct motors_input_speed	Motor speed

10.2.9.2 motor_reset_data

【Introduction】

Motor reset parameters.

【Definition】

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

【Members】

Member Name	Description
x_max_steps	The maximum step distance of the horizontal motor
y_max_steps	The maximum step size of the vertical motor
reset_x	The position where the horizontal motor stays after reset
reset_y	The position where the vertical motor stays after reset