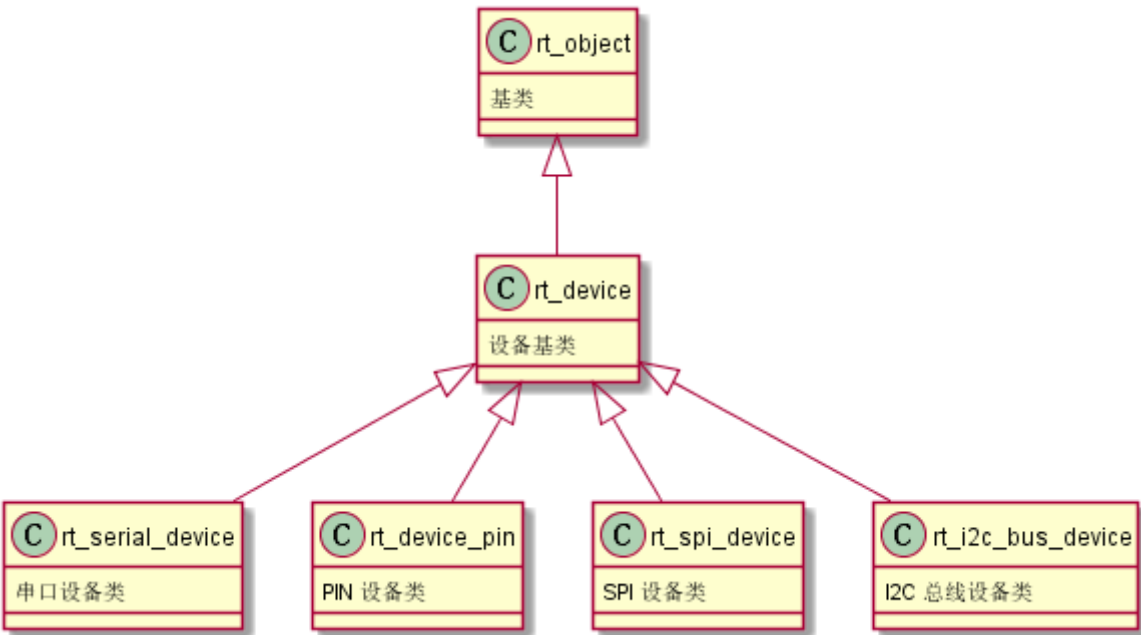
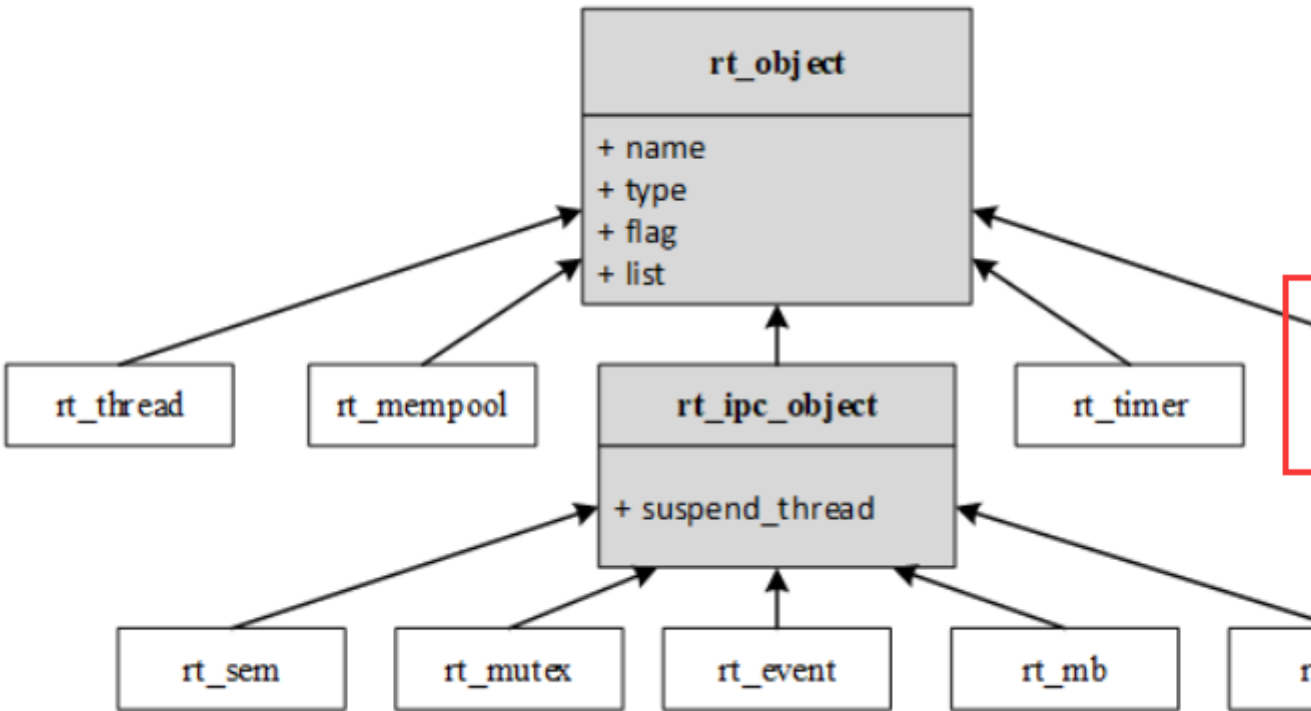


RTT

RT-Thread device



device

```
struct rt_device
{
    struct rt_object      parent;           /* 内核对象基类 */
    enum rt_device_class_type type;         /* 设备类型 */
    rt_uint16_t           flag;             /* 设备参数 */
    rt_uint16_t           open_flag;        /* 设备打开标志 */
    rt_uint8_t            ref_count;        /* 设备被引用次数 */
    rt_uint8_t            device_id;        /* 设备 ID, 0 - 255 */

    /* 数据收发回调函数 */
    rt_err_t (*rx_indicate)(rt_device_t dev, rt_size_t size);
    rt_err_t (*tx_complete)(rt_device_t dev, void *buffer);

    const struct rt_device_ops *ops;        /* 设备操作方法 */

    /* 设备的私有数据 */
    void *user_data;
};
typedef struct rt_device *rt_device_t;
```

type

```
enum rt_device_class_type
{
    RT_Device_Class_Char = 0,              /**< character device */
    RT_Device_Class_Block,                  /**< block device */
    RT_Device_Class_NetIf,                  /**< net interface */
    RT_Device_Class_MTD,                    /**< memory device */
    RT_Device_Class_CAN,                    /**< CAN device */
    RT_Device_Class_RTC,                    /**< RTC device */
    RT_Device_Class_Sound,                  /**< Sound device */
    RT_Device_Class_Graphic,                /**< Graphic device */
    RT_Device_Class_I2CBUS,                 /**< I2C bus device */
    RT_Device_Class_USBDevice,              /**< USB slave device */
    RT_Device_Class_USBHost,               /**< USB host bus */
    RT_Device_Class_SPIBUS,                 /**< SPI bus device */
    RT_Device_Class_SPIDevice,              /**< SPI device */
    RT_Device_Class_SDIO,                   /**< SDIO bus device */
    RT_Device_Class_PM,                     /**< PM pseudo device */
    RT_Device_Class_Pipe,                   /**< Pipe device */
    RT_Device_Class_Portal,                 /**< Portal device */
    RT_Device_Class_Timer,                  /**< Timer device */
    RT_Device_Class_Miscellaneous,          /**< Miscellaneous device */
    RT_Device_Class_Unknown                 /**< unknown device */
} « end rt_device_class_type » ;
```

```

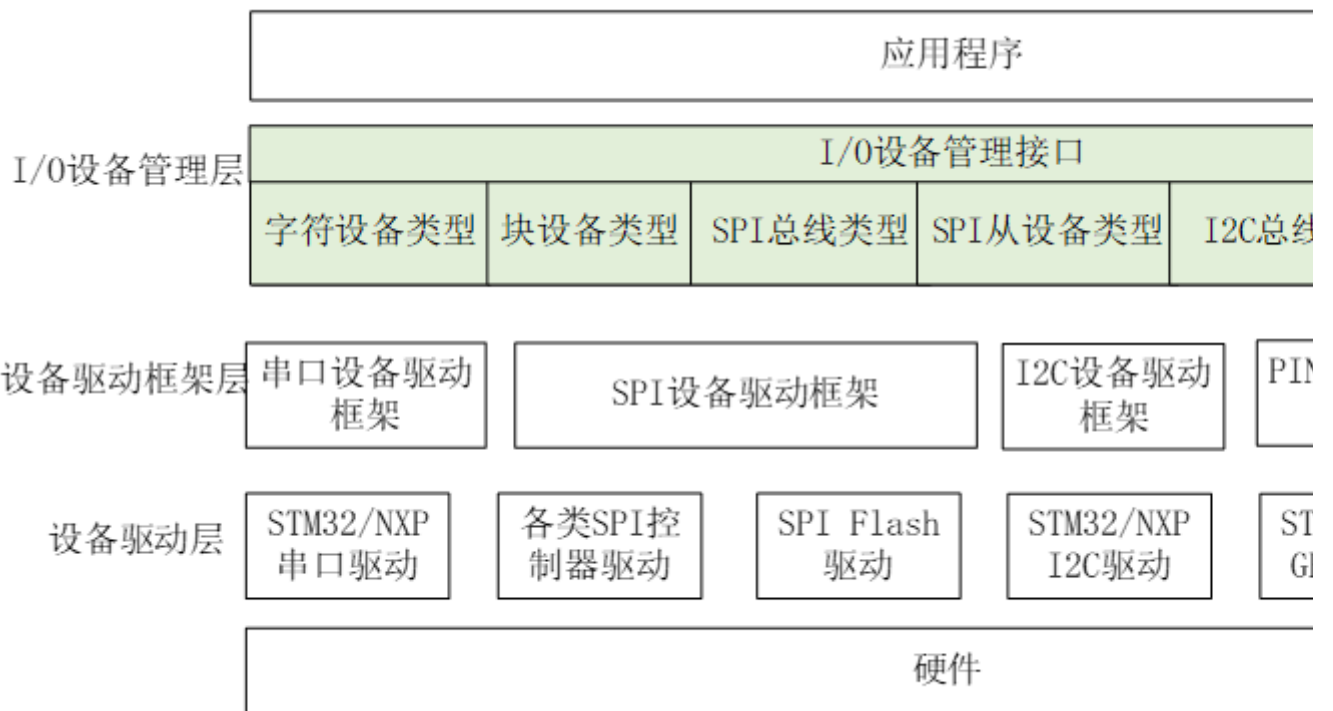
struct rt_device_ops
{
    /* common device interface */
    rt_err_t (*init) (rt_device_t dev);
    rt_err_t (*open) (rt_device_t dev, rt_uint16_t oflag);
    rt_err_t (*close) (rt_device_t dev);
    rt_size_t (*read) (rt_device_t dev, rt_off_t pos, void *buffer, rt_size_t size);
    rt_size_t (*write) (rt_device_t dev, rt_off_t pos, const void *buffer, rt_size_t size);
    rt_err_t (*control)(rt_device_t dev, int cmd, void *args);
};

```

I/O

RT-Thread

I/O



I/O

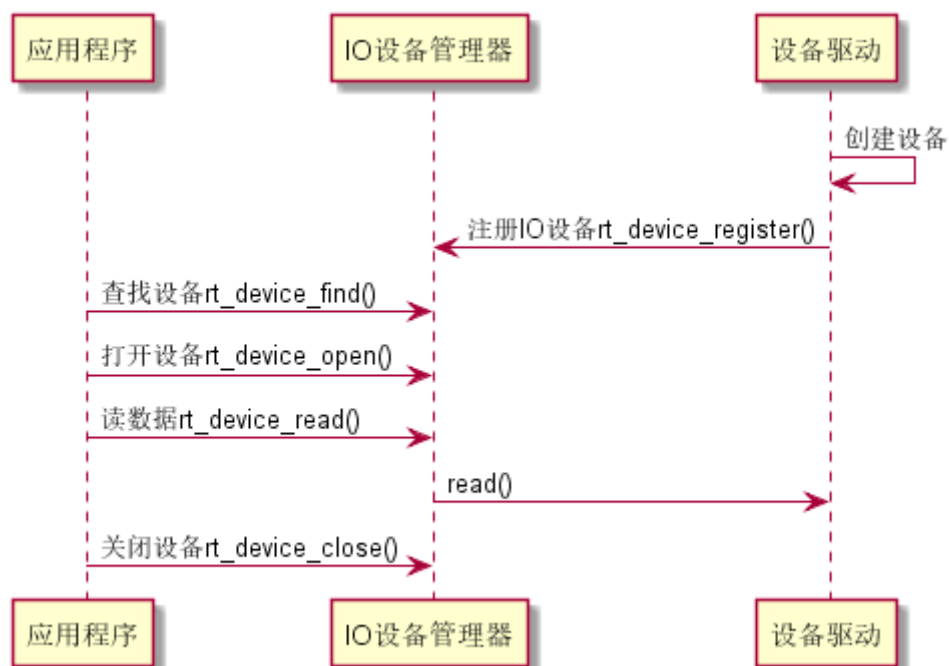
I/O

I/O

STM32 GPIO NXP GPIO

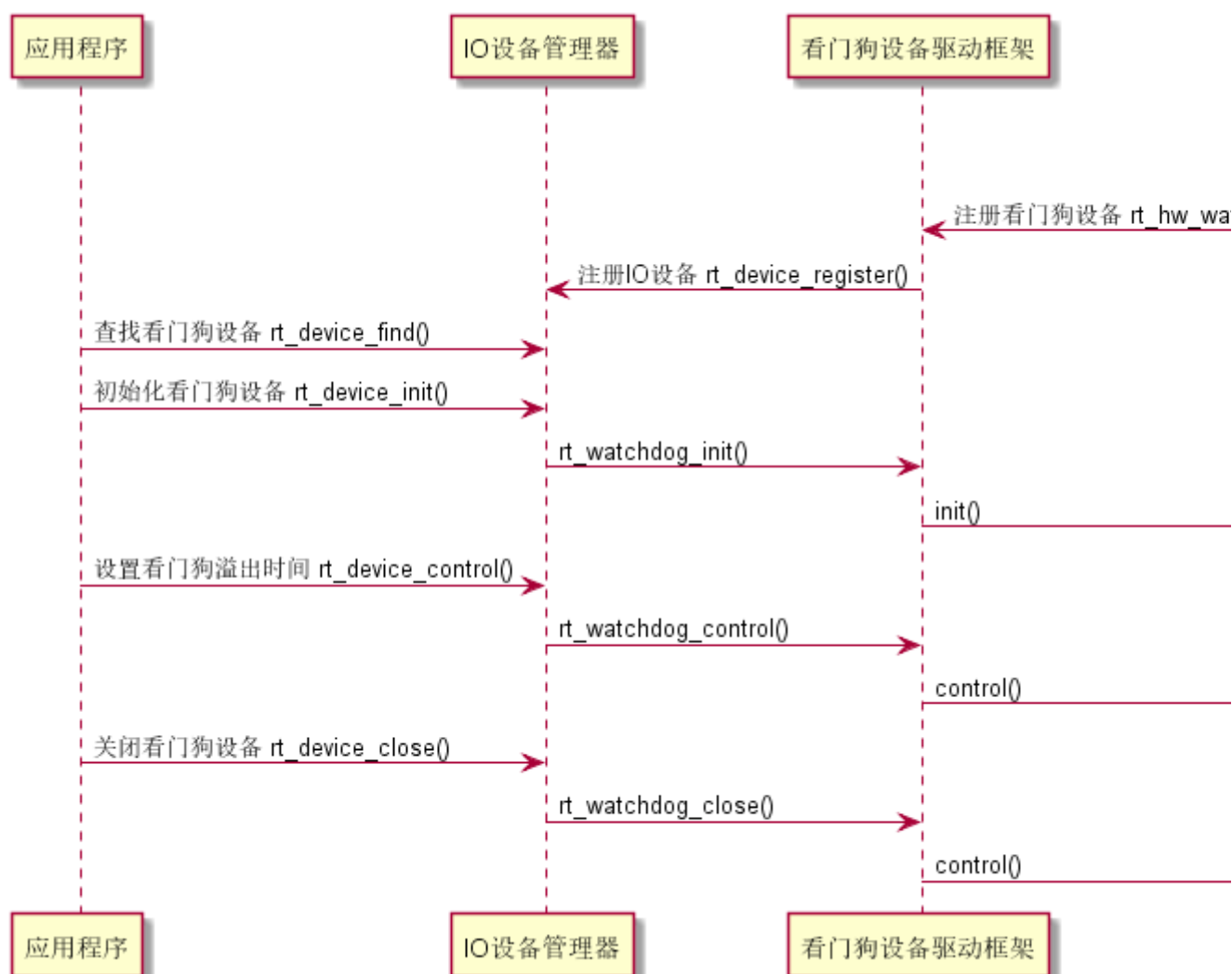
1

I/O



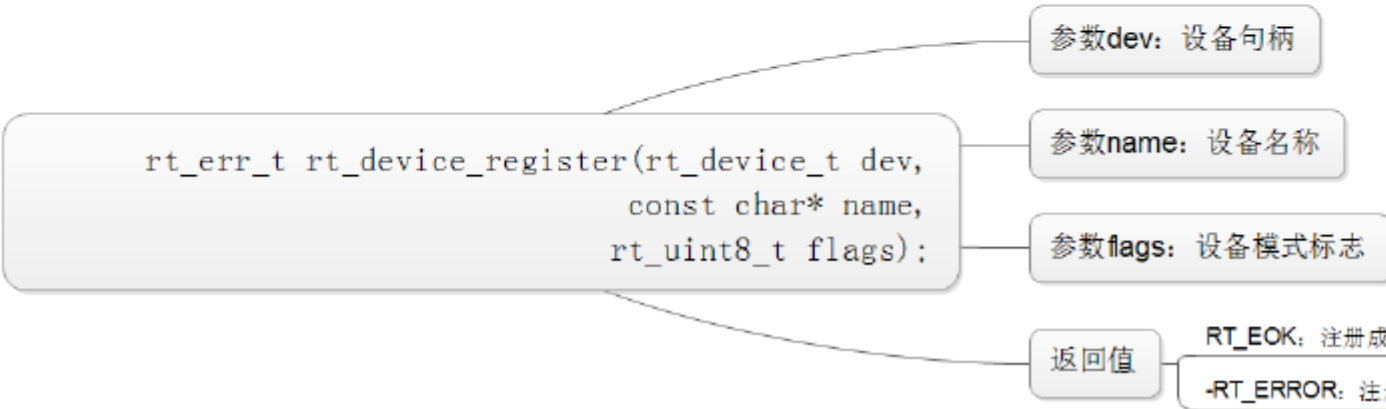
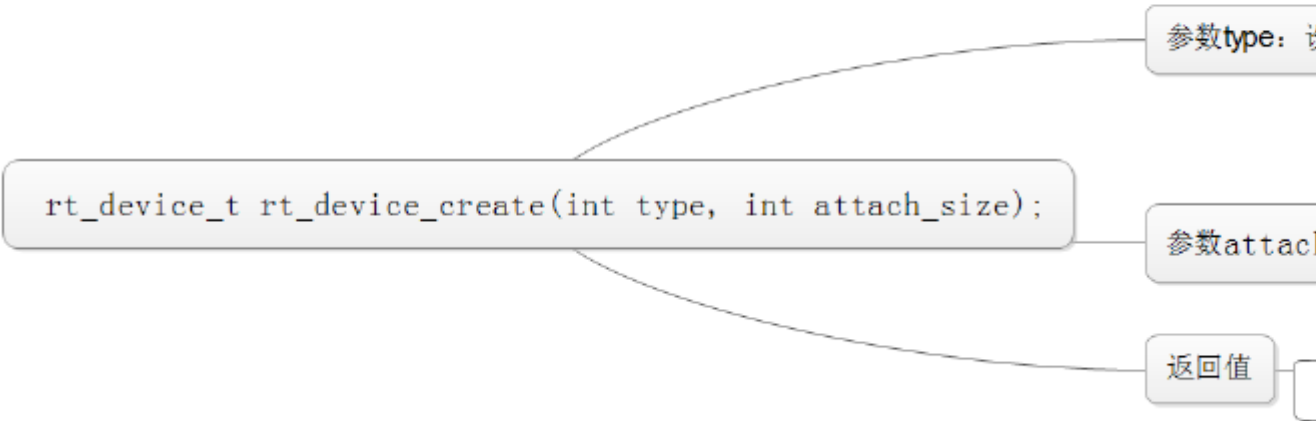
2

I/O



I/O

1 I/O

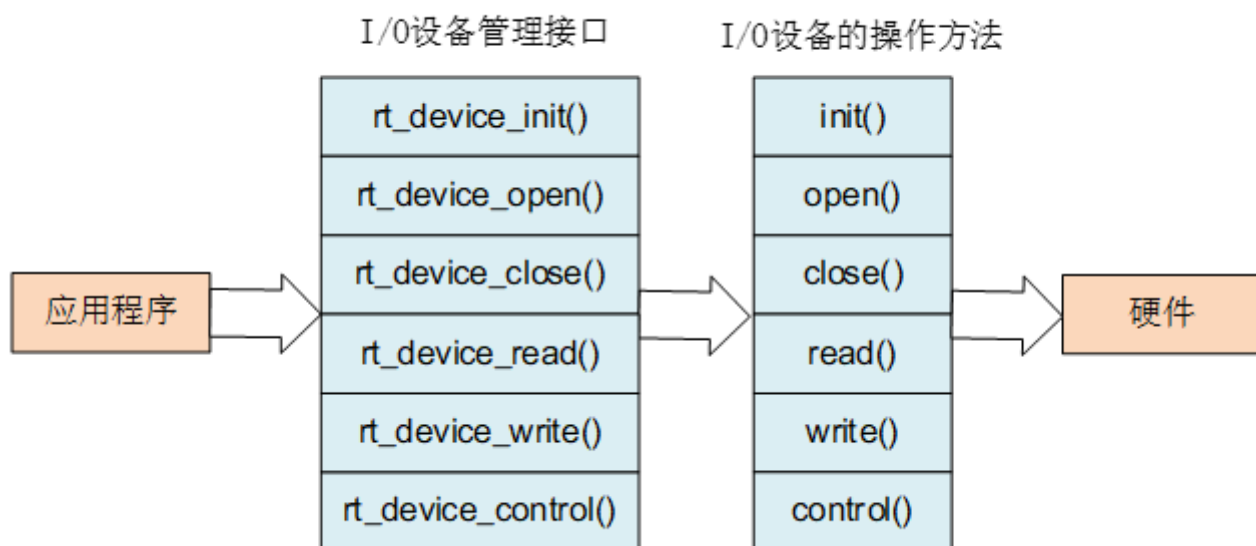


2 I/O

I/O

I/O

I/O



```
rt_device_t rt_device_find(const char* name);
```

参数name: 设备名称

返回值

```
rt_err_t rt_device_init(rt_device_t dev);
```

参数dev: 设备句柄

返回值

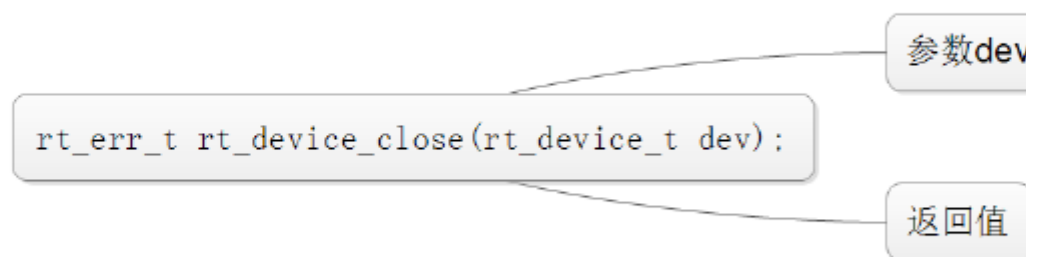
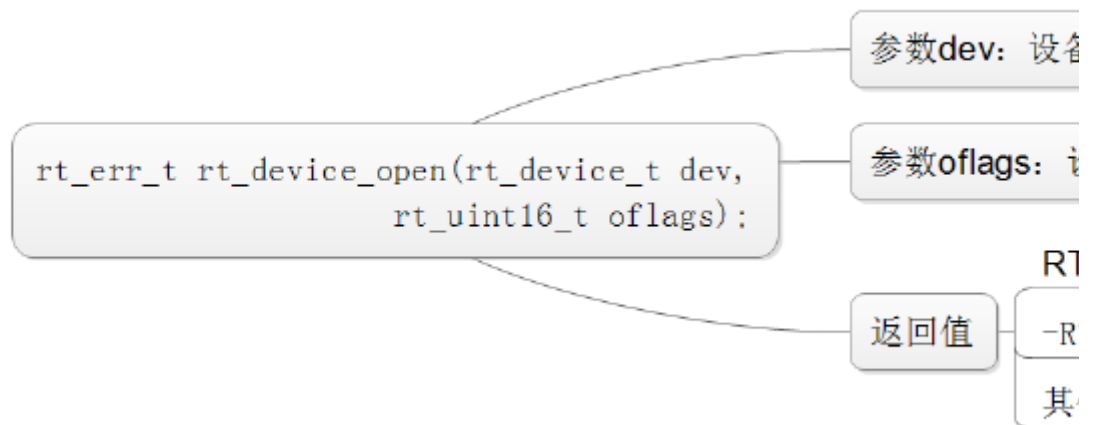
RT_EC

错误码

```

#define RT_DEVICE_OFLAG_CLOSE (
#define RT_DEVICE_OFLAG_RDONLY
#define RT_DEVICE_OFLAG_WRONLY
#define RT_DEVICE_OFLAG_RDWR 0
#define RT_DEVICE_OFLAG_OPEN 0
#define RT_DEVICE_FLAG_STREAM (
#define RT_DEVICE_FLAG_INT_RX (
#define RT_DEVICE_FLAG_DMA_RX (
#define RT_DEVICE_FLAG_INT_TX (
#define RT_DEVICE_FLAG_DMA_TX (

```



```
#define RT_DEVICE_CTRL_RESUME
#define RT_DEVICE_CTRL_SUSPEND
#define RT_DEVICE_CTRL_CONFIG
#define RT_DEVICE_CTRL_SET_INT
#define RT_DEVICE_CTRL_CLR_INT
#define RT_DEVICE_CTRL_GET_INT
```

```
rt_err_t rt_device_control(rt_device_t dev,
                           rt_uint8_t cmd,
                           void* arg);
```

参数dev: 设备句柄

参数cmd: 命令控制字

参数arg: 控制的参数

返回值

RT_EOK: 成功

-RT_ENOSYS: 不支持

其他错误码:

```
rt_size_t rt_device_read(rt_device_t dev,
                          rt_off_t pos,
                          void* buffer,
                          rt_size_t size);
```

参数dev: 设备句柄

参数pos: 读取数据位置

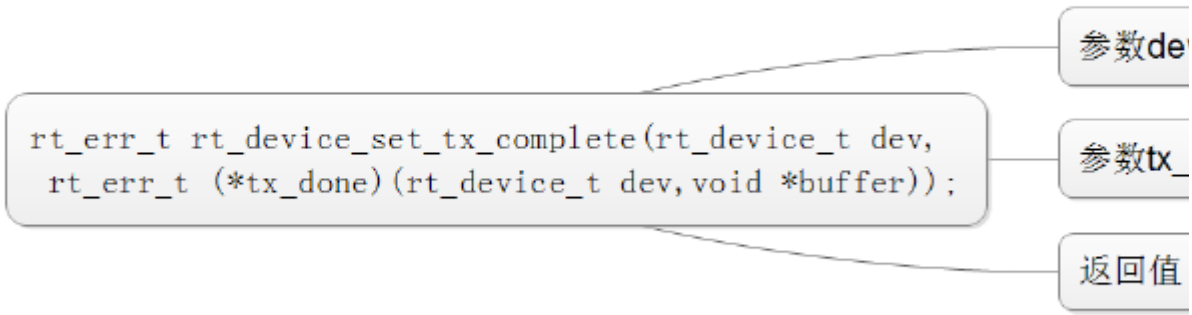
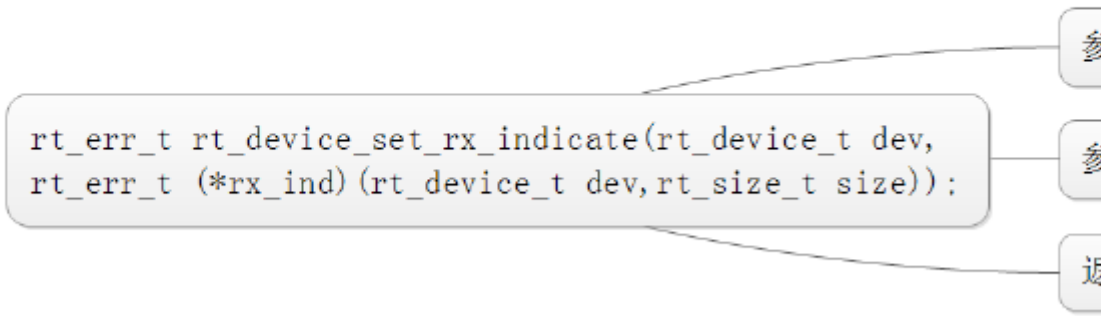
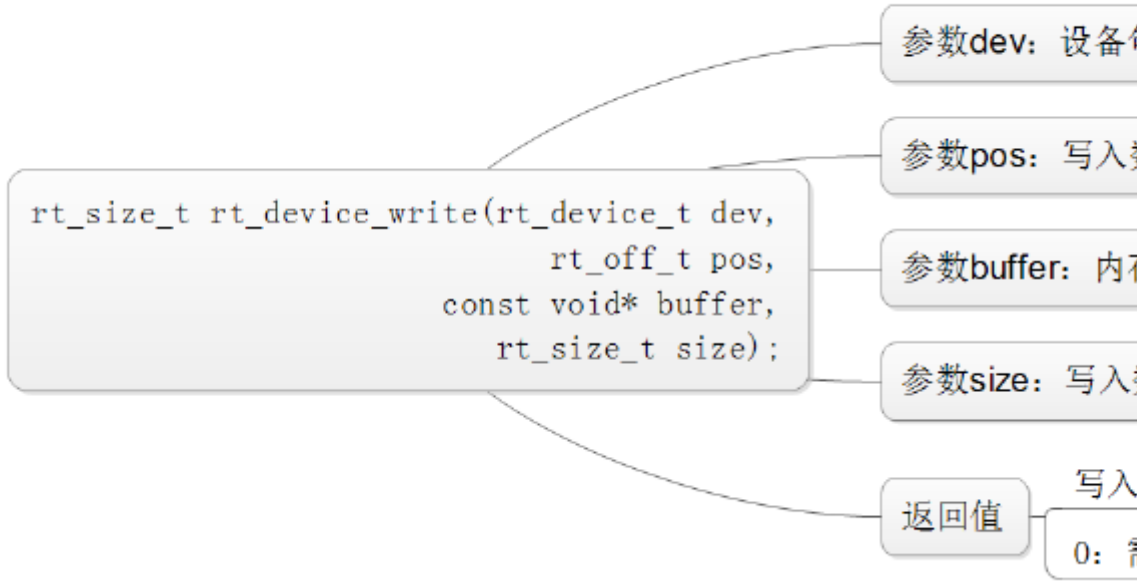
参数buffer: 内存缓冲区

参数size: 读取数据大小

返回值

读到的数据量

0: 需要继续读取



GPIO PIN





















pin.c

GPIO

RT-Thread

GPIO

rt-thread\components\drivers

 audio	2018/12/26 星期...	文件夹	
 can	2018/12/26 星期...	文件夹	
 cputime	2018/12/26 星期...	文件夹	
 hwtimer	2018/12/26 星期...	文件夹	
 i2c	2018/12/26 星期...	文件夹	
 include	2018/12/26 星期...	文件夹	
 misc	2018/12/26 星期...	文件夹	
 mtd	2018/12/26 星期...	文件夹	
 pm	2018/12/26 星期...	文件夹	
 rtc	2018/12/26 星期...	文件夹	
 sdio	2018/12/26 星期...	文件夹	
 sensors	2018/12/26 星期...	文件夹	
 serial	2018/12/26 星期...	文件夹	
 spi	2018/12/26 星期...	文件夹	
 src	2018/12/26 星期...	文件夹	
 usb	2018/12/26 星期...	文件夹	
 watchdog	2018/12/26 星期...	文件夹	
 wlan	2018/12/26 星期...	文件夹	
 Kconfig	2018/12/11 星期...	文件	15 KB
 SConscript	2018/12/11 星期...	文件	1 KB

GPIO

pin.c

misc

GPIO

pin

LED LED

```
static void pin_thread_entry(void *parameter)
{
    unsigned int count = 1;

    /* LED */
    rt_pin_mode(PIN_LED_R, PIN_MODE_OUTPUT);
}
```

```

/* KEY0 */
rt_pin_mode(PIN_KEY0, PIN_MODE_INPUT);

while (count > 0)
{
    /* KEY0 */
    if (rt_pin_read(PIN_KEY0) == PIN_LOW)
    {
        rt_thread_mdelay(50);
        if (rt_pin_read(PIN_KEY0) == PIN_LOW)
        {
            count++;
            rt_kprintf("KEY0 pressed! LED ON! count = %d\n", count);
            rt_pin_write(PIN_LED_R, PIN_LOW);
        }
    }
    else
    {
        rt_pin_write(PIN_LED_R, PIN_HIGH);
    }
    rt_thread_mdelay(10);
}

}

int main(void)
{
    /* */
    rt_thread_t tid;

    /* pin 25 5 512 */
    tid = rt_thread_create("pin_thread",
                           pin_thread_entry,
                           RT_NULL,
                           STACK_SIZE,
                           THREAD_PRIORITY,
                           TIMESLICE);

    /* */
    if (tid != RT_NULL)
    {

```

```

    rt_thread_startup( tid );
}

return 0;
}

```

list_device

The screenshot shows a terminal window titled "Serial-COM - SecureCRT". The menu bar includes "文件(F)", "编辑(E)", "查看(V)", "选项(O)", "传输(T)", "脚本(S)", "工具(L)", and "帮助(H)". The toolbar contains various icons for file operations and system functions. The terminal output is as follows:

```

\  \  \
- RT -   Thread operating system
/  /  /   4.0.0 build Sep  9 2019
2006 - 2018 Copyright by rt-thread team
msh > list_device
device      type          ref count
-----
uart1  character Device    2
pin     Miscellaneous Device 0
msh > KEY0 pressed! LED ON! count = 2
KEY0 pressed! LED ON! count = 3
KEY0 pressed! LED ON! count = 4
KEY0 pressed! LED ON! count = 5
KEY0 pressed! LED ON! count = 6
KEY0 pressed! LED ON! count = 7
KEY0 pressed! LED ON! count = 8
KEY0 pressed! LED ON! count = 9
KEY0 pressed! LED ON! count = 10
KEY0 pressed! LED ON! count = 11
KEY0 pressed! LED ON! count = 12
KEY0 pressed! LED ON! count = 13
KEY0 pressed! LED ON! count = 14
KEY0 pressed! LED ON! count = 15
KEY0 pressed! LED ON! count = 16
KEY0 pressed! LED ON! count = 17
KEY0 pressed! LED ON! count = 18
KEY0 pressed! LED ON! count = 19
KEY0 pressed! LED ON! count = 20
KEY0 pressed! LED ON! count = 21
KEY0 pressed! LED ON! count = 22
KEY0 pressed! LED ON! count = 23
KEY0 pressed! LED ON! count = 24
KEY0 pressed! LED ON! count = 25
KEY0 pressed! LED ON! count = 26
KEY0 pressed! LED ON! count = 27
KEY0 pressed! LED ON! count = 28
KEY0 pressed! LED ON! count = 29

```

device pin Miscellaneous Device GPIO

```

device.c
pin.c
drv_gpio.c

```

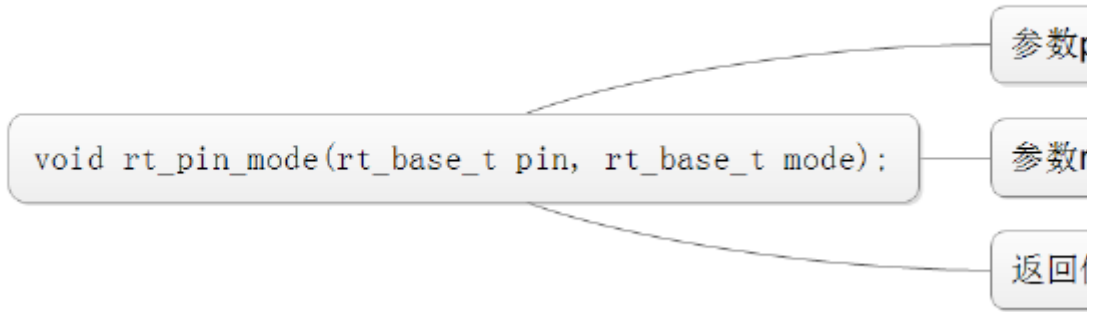
```
static void stm32_pin_write(rt_device_t dev, rt_base_t pin, rt_base_t value)
{
    const struct pin_index *index;

    index = get_pin(pin);
    if (index == RT_NULL)
    {
        return;
    }

    HAL_GPIO_WritePin(index->gpio, index->pin, (GPIO_PinState)value);
}
```

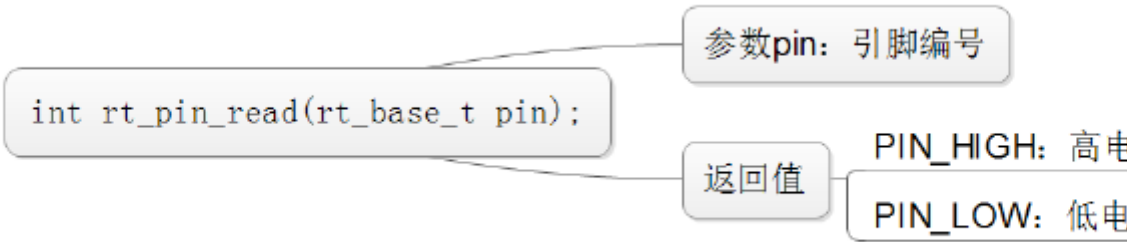
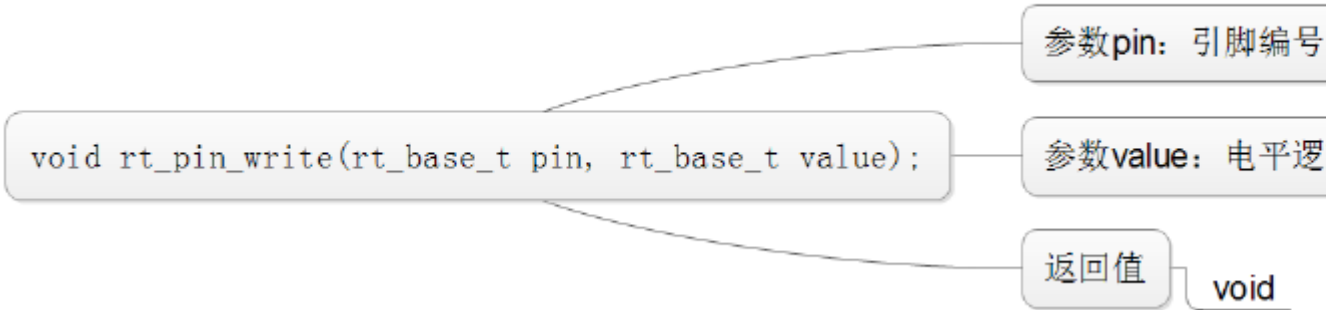
函数	描述
rt_pin_mode()	设置引脚模式
rt_pin_write()	设置引脚电平
rt_pin_read()	读取引脚电平
rt_pin_attach_irq()	绑定引脚中断回调函数
rt_pin_irq_enable()	使能引脚中断
rt_pin_detach_irq()	脱离引脚中断回调函数

```
#define PIN_MODE_
#define PIN_MODE_
#define PIN_MODE_
#define PIN_MODE_
#define PIN_MODE_
```

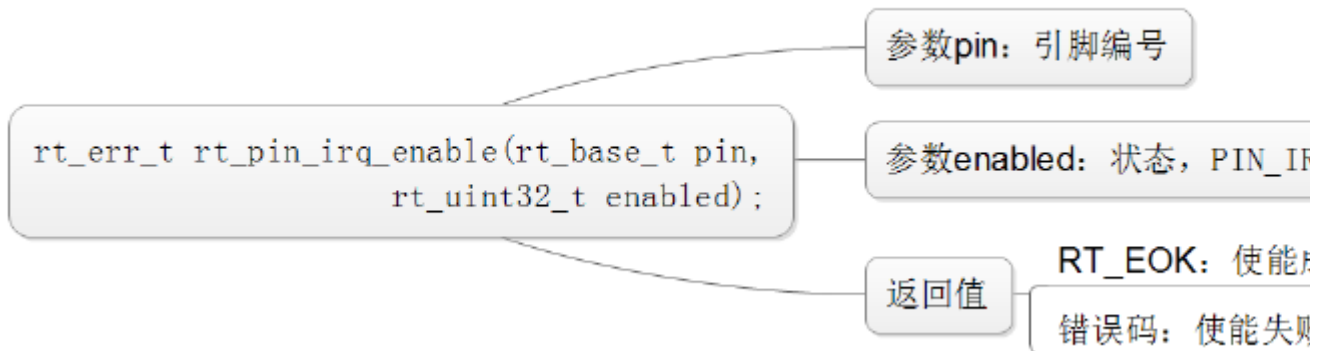
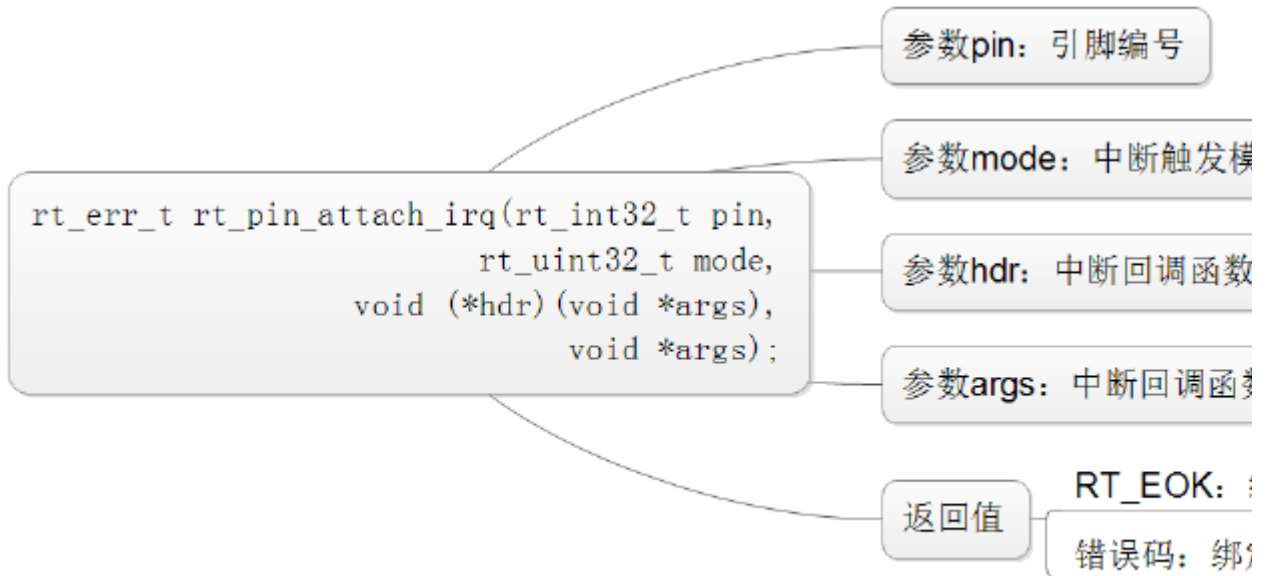


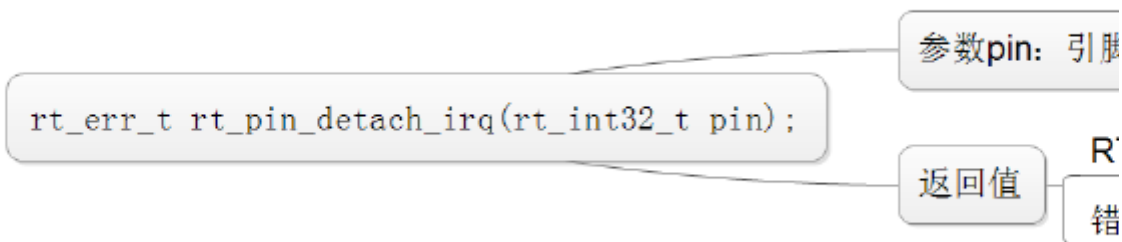
RT-Thread

PIN



```
#define PIN_IRQ_MODE_RISING 0x00
#define PIN_IRQ_MODE_FALLING 0x01
#define PIN_IRQ_MODE_RISING_FALLING 0x02
#define PIN_IRQ_MODE_HIGH_LEVEL 0x03
#define PIN_IRQ_MODE_LOW_LEVEL 0x04
```





Revision #3

Created 27 March 2022 12:30:41 by

Updated 27 March 2022 12:41:46 by