中断回调函数

1.中断函数列表

函数	描述
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

2.各个中断函数的作用

1. rt_pin_attch_irq():

函数原型:

```
/*
 * @brief 绑定引脚中断回调函数
 *
 * @param pin    引脚编号
 * @param mode   中断触发模式（例如上升沿、下降沿等）
 * @param hdr    中断回调函数指针
 * @param args   传递给回调函数的参数
 *
 * @return 成功返回 RT_EOK，失败返回错误码（如 -RT_ENOSYS 表示不支持该功能）
 *
 * @note 该函数用于将指定引脚的中断事件与用户定义的回调函数绑定。
 *       如果硬件抽象层（HAL）中定义了 `pin_attach_irq` 操作，则调用该操作完成绑定；
 *       否则返回 -RT_ENOSYS，表示当前硬件不支持此功能。
 */
rt_err_t rt_pin_attach_irq(rt_int32_t pin, rt_uint32_t mode,
                           void (*hdr)(void *args), void *args)
{
    RT_ASSERT(_hw_pin.ops != RT_NULL); // 确保硬件操作接口不为空
    if (_hw_pin.ops->pin_attach_irq)  // 检查是否支持 pin_attach_irq 操作
    {
        return _hw_pin.ops->pin_attach_irq(&_hw_pin.parent, pin, mode, hdr, args);
    }
    return -RT_ENOSYS; // 不支持该功能，返回错误码
}
```

2. rt_pin_irq_enable():

函数原型:

```
/*
 * @brief 使能或禁用引脚中断
 *
 * @param pin       引脚编号
 * @param enabled   使能标志（1 表示使能，0 表示禁用）
 *
 * @return 成功返回 RT_EOK，失败返回错误码（如 -RT_ENOSYS 表示不支持该功能）
```

```
 *
 * @note 该函数用于控制指定引脚的中断使能状态。
 *       如果硬件抽象层（HAL）中定义了 `pin_irq_enable` 操作，则调用该操作完成使能或禁用；
 *       否则返回 -RT_ENOSYS，表示当前硬件不支持此功能。
 */
rt_err_t rt_pin_irq_enable(rt_base_t pin, rt_uint32_t enabled)
{
    RT_ASSERT(_hw_pin.ops != RT_NULL); // 确保硬件操作接口不为空
    if (_hw_pin.ops->pin_irq_enable)  // 检查是否支持 pin_irq_enable 操作
    {
        return _hw_pin.ops->pin_irq_enable(&_hw_pin.parent, pin, enabled);
    }
    return -RT_ENOSYS; // 不支持该功能，返回错误码
}
```

3. rt_pin_detach_irq():

函数原型:

```
/*
 * @brief 脱离引脚中断回调函数
 *
 * @param pin    引脚编号
 *
 * @return 成功返回 RT_EOK，失败返回错误码（如 -RT_ENOSYS 表示不支持该功能）
 *
 * @note 该函数用于解除指定引脚的中断回调函数绑定。
 *       如果硬件抽象层（HAL）中定义了 `pin_detach_irq` 操作，则调用该操作完成解绑；
 *       否则返回 -RT_ENOSYS，表示当前硬件不支持此功能。
 */
rt_err_t rt_pin_detach_irq(rt_int32_t pin)
{
    RT_ASSERT(_hw_pin.ops != RT_NULL); // 确保硬件操作接口不为空
    if (_hw_pin.ops->pin_detach_irq)  // 检查是否支持 pin_detach_irq 操作
    {
        return _hw_pin.ops->pin_detach_irq(&_hw_pin.parent, pin);
    }
    return -RT_ENOSYS; // 不支持该功能，返回错误码
}
```

3. 将中断回调函数用于实际

```
#include <rtthread.h>
#include <rtdevice.h>
#include <board.h>

#define DBG_TAG "main"
#define DBG_LVL DBG_LOG
#include <rtdbg.h>

/* 配置 KEY 输入引脚  */
#define PIN_KEY1        GET_PIN(C, 1)       // PC1:  KEY1         --> KEY
#define PIN_WK_UP       GET_PIN(C, 5)       // PC5:  WK_UP        --> KEY

/* 配置蜂鸣器引脚 */
```

```c
#define PIN_BEEP        GET_PIN(B, 0)      // PA1:  BEEP         --> BEEP (PB1)

/* 中断回调 */
void irq_callback(void *args)
{
    rt_uint32_t sign = (rt_uint32_t) args;
    switch (sign)
    {
    case PIN_WK_UP :
        rt_pin_write(PIN_BEEP,PIN_HIGH);
        LOG_D("WK_UP interrupt. beep on.");
        break;
    case PIN_KEY1 :
        rt_pin_write(PIN_BEEP,PIN_LOW);
        LOG_D("KEY1 interrupt. beep off.");
        break;
    default:
        LOG_E("error sign= %d !", sign);
        break;
    }
}

int main(void)
{
    /* 设置按键引脚为输入模式 */
    rt_pin_mode(PIN_KEY1, PIN_MODE_INPUT_PULLUP);
    rt_pin_mode(PIN_WK_UP, PIN_MODE_INPUT_PULLUP);

    /* 设置蜂鸣器引脚为输出模式 */
    rt_pin_mode(PIN_BEEP, PIN_MODE_OUTPUT);

    /* 设置按键中断模式与中断回调函数 */
    rt_pin_attach_irq(PIN_KEY1, PIN_IRQ_MODE_FALLING, irq_callback, (void *)
PIN_KEY1);//将中断设置为低电平触发,当对应引脚为低电平(按键按下)时,会执行中断回调函数
    rt_pin_attach_irq(PIN_WK_UP, PIN_IRQ_MODE_FALLING, irq_callback, (void *)
PIN_WK_UP);

    /* 使能中断 */
    rt_pin_irq_enable(PIN_KEY1, PIN_IRQ_ENABLE);
    rt_pin_irq_enable(PIN_WK_UP, PIN_IRQ_ENABLE);

    while (1)
    {

    }
    return 0;
}
```