参考网页：

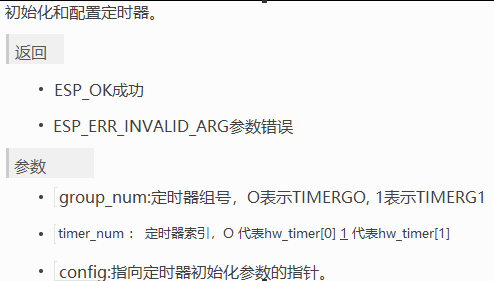
[通用定时器 - ESP32 - — ESP-IDF 编程指南 v4.3 文档 (espressif.com)](https://docs.espressif.com/projects/esp-idf/zh_CN/v4.3/esp32/api-reference/peripherals/timer.html)

[ESP32学习笔记（42）——硬件定时器接口使用\_Leung的博客-CSDN博客\_esp32硬件定时器](https://blog.csdn.net/qq_36347513/article/details/119537681)

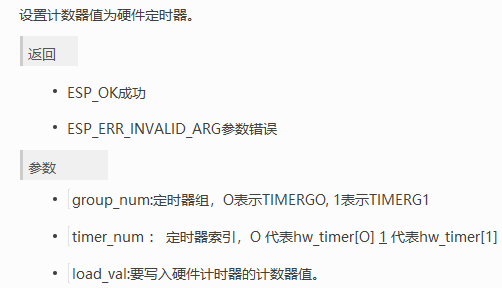
[FreeRTOS学习笔记（3）——消息队列\_Leung的博客-CSDN博客](https://blog.csdn.net/qq_36347513/article/details/109488543)

API参考：

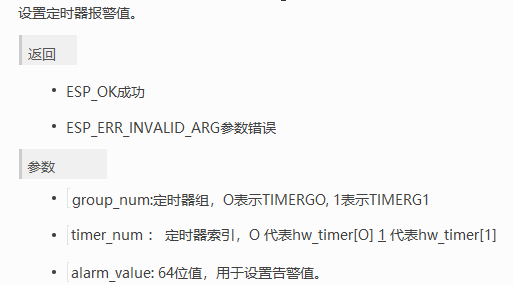




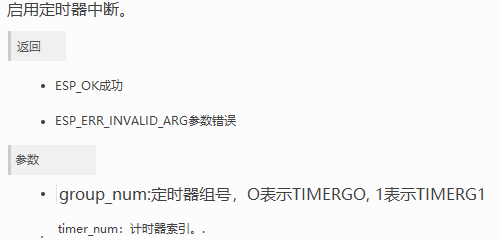


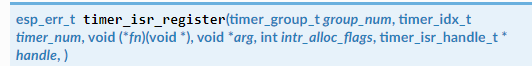


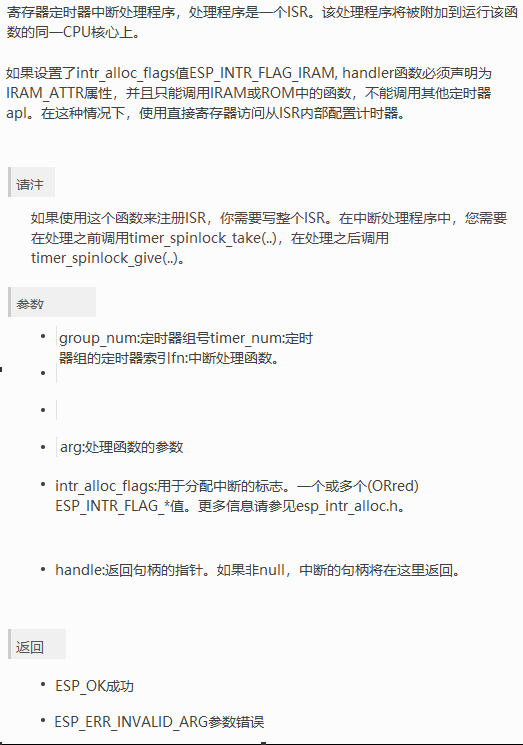




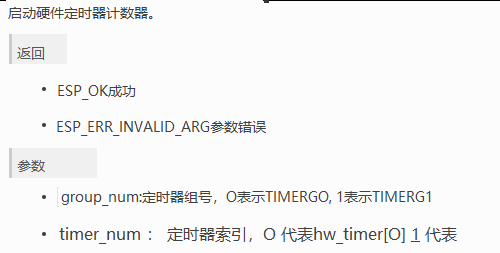




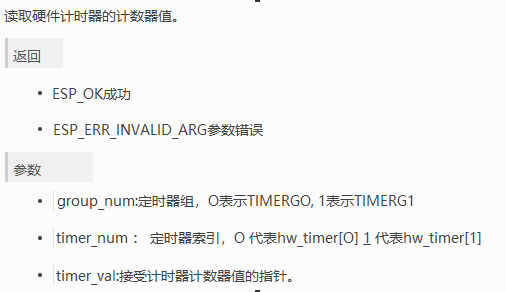




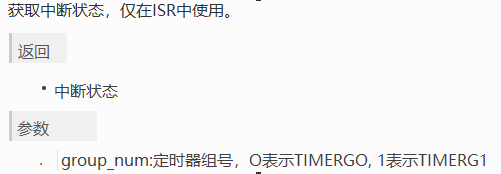




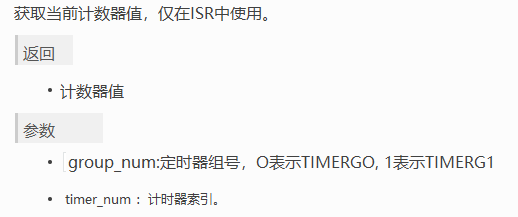




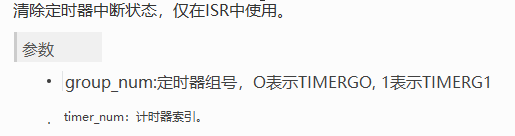




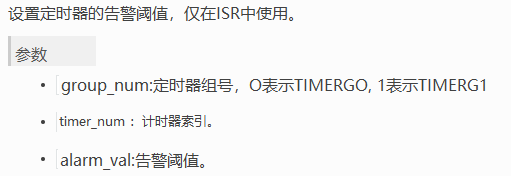




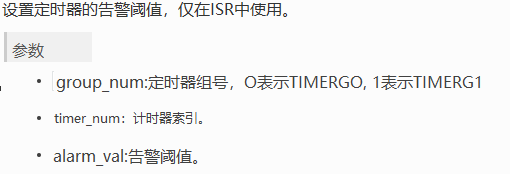












1. 定时器初始化，即相关设置

* 定时器参数结构体

typedef struct {

    int timer\_group;//定时器组号

    int timer\_idx;//定时器索引

    int alarm\_interval;//定时器预警间隔

    bool auto\_reload;//定时器是否自动重载

} example\_timer\_info\_t;

/\*\*

 \* @brief A sample structure to pass events from the timer ISR to task

 \*

 \*/

typedef struct {

    example\_timer\_info\_t info;

    uint64\_t timer\_counter\_value;//定时器计数值

} example\_timer\_event\_t;

* 初始化

定时器初始化时是调用函数timer\_init()，并给其传递一个包含定时器各种参数的结构体config（结构体名由编者自定）来达到初始化定时器的目的。

static void **example\_tg\_timer\_init**(int group, int timer, bool auto\_reload, int timer\_interval\_sec)

{

timer\_config\_t config = {

        .divider = TIMER\_DIVIDER,//分频倍数

        .counter\_dir = TIMER\_COUNT\_UP,//向上计数(1)/向下计数(0)

        .counter\_en = TIMER\_PAUSE,//定时器开(1)关(0)

        .alarm\_en = TIMER\_ALARM\_EN,//定时器中断开(1)关(0)

        .auto\_reload = auto\_reload,//是否自动重载

    }; // default clock source is APB

**timer\_init**(group, timer, &config);

**timer\_set\_counter\_value**(group, timer, 0);//定时器计数起始值

    //设置中断阈值及中断使能

**timer\_set\_alarm\_value**(group, timer, timer\_interval\_sec \* TIMER\_SCALE);

**timer\_enable\_intr**(group, timer);//中断使能

    example\_timer\_info\_t \*timer\_info = **calloc**(1, sizeof(example\_timer\_info\_t));

    timer\_info->timer\_group = group;

    timer\_info->timer\_idx = timer;

    timer\_info->auto\_reload = auto\_reload;

    timer\_info->alarm\_interval = timer\_interval\_sec;

    /\*

    timer\_isr\_callback\_add的参数中，timer\_info会传递给

timer\_group\_isr\_callback的arg

\*/

**timer\_isr\_callback\_add**(group, timer, timer\_group\_isr\_callback, timer\_info, 0);

**timer\_start**(group, timer);//启动定时器

}

1. 定时器控制

当定时器启用（即config结构体中.alarm\_en = TIMER\_ALARM\_EN,）可调用：timer\_set\_counter\_value()设置定时器的初始计数值，调用timer\_get\_counter\_value()获取当前的计数值，调用timer\_pause()暂停计数器，调用timer\_start()重启计数器

1. 定时器中断

ESP32的硬件定时器有一个ISP程序，我们可以将定时器的回调函数配置在这个ISR程序中，即让ISR为我们执行回调函数，但要求回调函数尽可能简短，不执行任何耗时操作

* 队列

static xQueueHandle s\_timer\_queue;//定义队列句柄

//main函数中创建队列

s\_timer\_queue = **xQueueCreate**(10, sizeof(example\_timer\_event\_t));

//ISR Callback函数中通过队列将定时器参数信息传给main

**xQueueSendFromISR**(s\_timer\_queue, &evt, &high\_task\_awoken);

//main函数中，接受回调函数中，通过队列传回的定时器参数信息

**xQueueReceive**(s\_timer\_queue, &evt, portMAX\_DELAY);

* 创建ISP 回调函数

static bool IRAM\_ATTR **timer\_group\_isr\_callback**(void \*args)

{

    BaseType\_t high\_task\_awoken = pdFALSE;

example\_timer\_info\_t \*info = (example\_timer\_info\_t \*) args;

uint64\_t timer\_counter\_value = **timer\_group\_get\_counter\_value\_in\_isr**(info->timer\_group, info->timer\_idx);

    /\*

    Prepare basic event data that will be then sent back to task

    将本函数中由example\_timer\_info\_t定义的结构体变量info传递给由

    example\_timer\_event\_t定义的结构体变量evt

    注：建议将info变量更名为 info\_temp 以便于区分

    如下:

    example\_timer\_info\_t \*info\_temp = (example\_timer\_info\_t \*) args;

    example\_timer\_event\_t evt = {

        .info.timer\_group = info\_temp->timer\_group,

        .info.timer\_idx = info\_temp->timer\_idx,

        .info.auto\_reload = info\_temp->auto\_reload,

        .info.alarm\_interval = info\_temp->alarm\_interval,

        .timer\_counter\_value = timer\_counter\_value

    };

     \*/

    example\_timer\_event\_t evt = {

        .info.timer\_group = info->timer\_group,

        .info.timer\_idx = info->timer\_idx,

        .info.auto\_reload = info->auto\_reload,

        .info.alarm\_interval = info->alarm\_interval,

        .timer\_counter\_value = timer\_counter\_value

};

    if (!info->auto\_reload) {

        //如果不自动重载，则设置下次预警值为此时的预警值加上预警间隔

        timer\_counter\_value += info->alarm\_interval \* TIMER\_SCALE;

        //更新预警值

**timer\_group\_set\_alarm\_value\_in\_isr**(info->timer\_group, info->timer\_idx, timer\_counter\_value);

    }

    /\* Now just send the event data back to the main program task \*/

**xQueueSendFromISR**(s\_timer\_queue, &evt, &high\_task\_awoken);

return high\_task\_awoken == pdTRUE;

}

注：其中args 用于接受在timer\_isr\_callback-add(ISR Callback注册函数)中穿进来的定时器的信息结构体指针

在ISR中可以调用：timer\_group\_get\_counter\_value\_in\_isr()来获取计数器的值，调用timer\_group\_set\_alarm\_value\_in\_isr()来设置定时器的预警值

**完整例程**

#include <stdio.h>

#include "freertos/FreeRTOS.h"

#include "freertos/task.h"

#include "freertos/queue.h"

#include "driver/timer.h"

#define **TIMER\_DIVIDER**         (16)  //硬件定时器分频倍数Hardware timer clock divider

// 定时器计数频率convert counter value to seconds

#define **TIMER\_SCALE**           (TIMER\_BASE\_CLK / TIMER\_DIVIDER)

typedef struct {

    int timer\_group;//定时器组号

    int timer\_idx;//定时器索引

    int alarm\_interval;//定时器预警间隔

    bool auto\_reload;//定时器是否自动重载

} example\_timer\_info\_t;

/\*\*

 \* @brief A sample structure to pass events from the timer ISR to task

 \*

 \*/

typedef struct {

    example\_timer\_info\_t info;

    uint64\_t timer\_counter\_value;

} example\_timer\_event\_t;

static xQueueHandle s\_timer\_queue;//队列句柄

/\*

 \* A simple helper function to print the raw timer counter value

 \* and the counter value converted to seconds

 \* inline：在定义函数的时候  将其定义为内联函数

 \*/

static void inline **print\_timer\_counter**(uint64\_t counter\_value)

{

**printf**("Counter: 0x%08x%08x\r\n", (uint32\_t) (counter\_value >> 32),

           (uint32\_t) (counter\_value));

**printf**("Time   : %.8f s\r\n", (double) counter\_value / TIMER\_SCALE);

}

static bool IRAM\_ATTR **timer\_group\_isr\_callback**(void \*args)

{

    BaseType\_t high\_task\_awoken = pdFALSE;

example\_timer\_info\_t \*info = (example\_timer\_info\_t \*) args;

uint64\_t timer\_counter\_value = **timer\_group\_get\_counter\_value\_in\_isr**(info->timer\_group, info->timer\_idx);

    /\*

    Prepare basic event data that will be then sent back to task

    将本函数中由example\_timer\_info\_t定义的结构体变量info（见第41行）传递给由

    example\_timer\_event\_t定义的结构体变量evt（见第63行）

    注：建议将info变量更名为 info\_temp 以便于区分

    如下:

    example\_timer\_info\_t \*info\_temp = (example\_timer\_info\_t \*) args;

    example\_timer\_event\_t evt = {

        .info.timer\_group = info\_temp->timer\_group,

        .info.timer\_idx = info\_temp->timer\_idx,

        .info.auto\_reload = info\_temp->auto\_reload,

        .info.alarm\_interval = info\_temp->alarm\_interval,

        .timer\_counter\_value = timer\_counter\_value

    };

     \*/

    example\_timer\_event\_t evt = {

        .info.timer\_group = info->timer\_group,

        .info.timer\_idx = info->timer\_idx,

        .info.auto\_reload = info->auto\_reload,

        .info.alarm\_interval = info->alarm\_interval,

        .timer\_counter\_value = timer\_counter\_value

};

    if (!info->auto\_reload) {

        //如果不自动重载，则设置下次预警值为此时的预警值加上预警间隔

        timer\_counter\_value += info->alarm\_interval \* TIMER\_SCALE;

        //更新预警值

**timer\_group\_set\_alarm\_value\_in\_isr**(info->timer\_group, info->timer\_idx, timer\_counter\_value);

}

    /\* Now just send the event data back to the main program task \*/

**xQueueSendFromISR**(s\_timer\_queue, &evt, &high\_task\_awoken);

    return high\_task\_awoken == pdTRUE; // return whether we need to yield at the end of ISR

}

/\*\*

 \* @brief Initialize selected timer of timer group

 \*

 \* @param group Timer Group number, index from 0

 \* @param timer timer ID, index from 0

 \* @param auto\_reload whether auto-reload on alarm event

 \* @param timer\_interval\_sec interval of alarm

 \*/

static void **example\_tg\_timer\_init**(int group, int timer, bool auto\_reload, int timer\_interval\_sec)

{

    /\* Select and initialize basic parameters of the timer \*/

    timer\_config\_t config = {

        .divider = TIMER\_DIVIDER,//分频倍数

        .counter\_dir = TIMER\_COUNT\_UP,//向上计数(1)/向下计数(0)

        .counter\_en = TIMER\_PAUSE,//定时器开(1)关(0)

        .alarm\_en = TIMER\_ALARM\_EN,//定时器中断开(1)关(0)

        .auto\_reload = auto\_reload,//是否自动重载

    }; // default clock source is APB

**timer\_init**(group, timer, &config);

    /\* Timer's counter will initially start from value below.

       Also, if auto\_reload is set, this value will be automatically reload on alarm \*/

**timer\_set\_counter\_value**(group, timer, 0);//定时器计数起始值

    /\*

    Configure the alarm value and the interrupt on alarm.

    设置中断阈值及中断使能

    \*/

**timer\_set\_alarm\_value**(group, timer, timer\_interval\_sec \* TIMER\_SCALE);

**timer\_enable\_intr**(group, timer);//中断使能

    example\_timer\_info\_t \*timer\_info = **calloc**(1, sizeof(example\_timer\_info\_t));

    timer\_info->timer\_group = group;

    timer\_info->timer\_idx = timer;

    timer\_info->auto\_reload = auto\_reload;

    timer\_info->alarm\_interval = timer\_interval\_sec;

    /\*

    timer\_isr\_callback\_add的参数中，timer\_info会传递给

    timer\_group\_isr\_callback的arg(详见51行)

    \*/

**timer\_isr\_callback\_add**(group, timer, timer\_group\_isr\_callback, timer\_info, 0);

**timer\_start**(group, timer);

}

void **app\_main**(void)

{

s\_timer\_queue = **xQueueCreate**(10, sizeof(example\_timer\_event\_t));

    //初始化两个定时器

**example\_tg\_timer\_init**(TIMER\_GROUP\_0, TIMER\_0, true, 3);

**example\_tg\_timer\_init**(TIMER\_GROUP\_1, TIMER\_0, false, 5);

    //测试前两个计数器是否准时

**example\_tg\_timer\_init**(TIMER\_GROUP\_1, TIMER\_1, false, 1);

    while (1) {

        /\*

        定义一个example\_timer\_event\_t结构体变量evt来接收从回调函数中通过队列传回的数据

        (详见第84行：xQueueSendFromISR(s\_timer\_queue, &evt, &high\_task\_awoken);)

        其中包括定时器组号：timer\_group、定时器索引：timer\_idx、

        定时器是否自动重载：auto\_reload、定时器中断阈值：timer\_interval\_sec

        定时器当前计数值：timer\_counter\_value

        \*/

        example\_timer\_event\_t evt;

**xQueueReceive**(s\_timer\_queue, &evt, portMAX\_DELAY);

**printf**("\n");

**printf**("/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/\n");

        /\* Print information that the timer reported an event \*/

        if (evt.info.auto\_reload) {

**printf**("Timer Group with auto reload\n");

        } else {

**printf**("Timer Group without auto reload\n");

        }

**printf**("Group[%d], timer[%d] alarm event\n", evt.info.timer\_group, evt.info.timer\_idx);

        /\* Print the timer values passed by event \*/

**printf**("------- EVENT TIME --------\n");

**print\_timer\_counter**(evt.timer\_counter\_value);

        /\* Print the timer values as visible by this task \*/

**printf**("-------- TASK TIME --------\n");

        uint64\_t task\_counter\_value;

**timer\_get\_counter\_value**(evt.info.timer\_group, evt.info.timer\_idx, &task\_counter\_value);

**print\_timer\_counter**(task\_counter\_value);

**printf**("/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/\n");

**printf**("\n");

    }

}

23233