乐鑫定时器指导页面：[High Resolution Timer - ESP32 - — ESP-IDF 编程指南 latest 文档 (espressif.com)](https://docs.espressif.com/projects/esp-idf/zh_CN/latest/esp32/api-reference/system/esp_timer.html" \l "high-resolution-timer)

CSDN 例程：[ESP32学习笔记（3）——高分辨率定时器接口使用\_Leung的博客-CSDN博客\_esp32定时器](https://blog.csdn.net/qq_36347513/article/details/115715977)

ESP\_ERROR\_CHECK参照：[ESP32 官方文档（四）错误处理\_InfiniteYuan-CSDN博客\_esp32 错误处理](https://blog.csdn.net/qq_27114397/article/details/82286592)

esp\_timer\_init;初始化esp\_timer库

esp\_timer\_deinit();取消初始化esp\_timer库

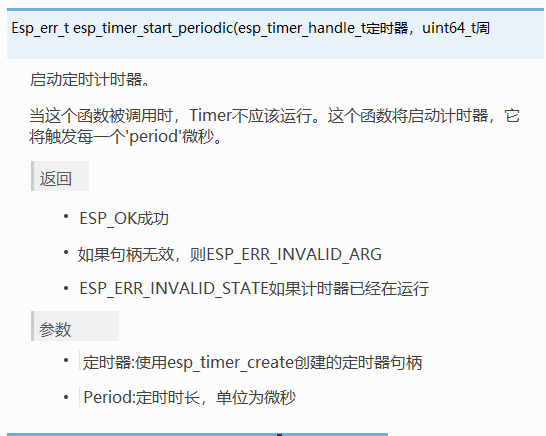
esp\_timer\_create();创建一个esp\_timer 实例



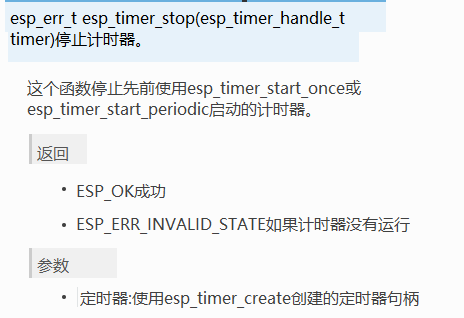
esp\_timer\_statr\_once();启动一次性定时器



esp\_timer\_start\_periodic();启动一个周期定时器



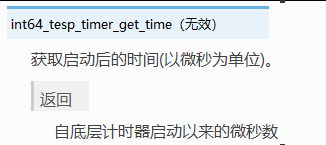
esp\_timer\_stop();暂停定时器，在删除定时器之前，需要将其停止（一次性定时器除外）



esp\_timer\_delete();删除定时器



esp\_timer\_get\_time();获取本次定时器启动以后的时间



ESP\_ERR\_CHECK();



定时器创建流程;

* 添加头文件：#include "esp\_timer.h"
* 定义定时器的回调函数

static void **timer\_callback\_name**(void\* arg)

{

    /\*code segment\*/ //编写回调函数代码

}

Lg：

static void **oneshot\_timer\_callback**(void\* arg)

{

    /\*code segment\*/

}//一次性定时器回调函数

static void **periodic\_timer\_callback**(void\* arg)

{

    /\*code segment\*/

}//周期定时器回调函数

* 主函数中定义与创建定时器

1. 定义定时器句柄（可以理解为定时器的身份证号）

esp\_timer\_handle\_t handle\_timer;//定义一个定时器句柄

Lg：

esp\_timer\_handle\_t oneshot\_timer;//一次性定时器句柄

esp\_timer\_handle\_t periodic\_timer;//周期定时器句柄

1. 定义定时器结构体：回调函数(Callback)、参数(arg)、定时器名字(name)

Lg:

const esp\_timer\_create\_args\_t oneshot\_timer\_args = {

            .callback = &oneshot\_timer\_callback,

            /\* argument specified here will be passed to timer callback function \*/

            .arg = (void\*) periodic\_timer,

            .name = "one-shot"

    };//一次性定时器结构体

const esp\_timer\_create\_args\_t periodic\_timer\_args = {

            .callback = &periodic\_timer\_callback,

            /\* name is optional, but may help identify the timer when debugging \*/

            .name = "periodic"

    };//周期定时器结构体

1. 创建定时器

Lg：

**esp\_timer\_create**(&oneshot\_timer\_args, &oneshot\_timer)//创建一次性定时器

**esp\_timer\_create**(&periodic\_timer\_args, &periodic\_timer)//创建周期定时器

1. 启动定时器

Lg：

**esp\_timer\_start\_once**(oneshot\_timer, 5\*1000\*1000)//启动一个计时为5s的定时器

**esp\_timer\_start\_periodic**(periodic\_timer, 500\*1000)//启动一个计时为500ms的定时器

注：在创建与启动定时器时，可以使用以下语句：

**ESP\_ERROR\_CHECK**(**esp\_timer\_create**(&oneshot\_timer\_args, &oneshot\_timer));

**ESP\_ERROR\_CHECK**(**esp\_timer\_start\_once**(oneshot\_timer, 5\*1000\*1000));

**ESP\_ERROR\_CHECK**(**esp\_timer\_create**(&periodic\_timer\_args, &periodic\_timer));

**ESP\_ERROR\_CHECK**(**esp\_timer\_start\_periodic**(periodic\_timer, 500\*1000));

当创建和启动定时器失败时，会有错误信息输出，便于检查代码。

完整创建一个定时器，并在定时器回调函数执行10次之后，停止定时器并将其删除。

#include <stdio.h>

#include "esp\_timer.h"

#include "esp\_log.h"

static const char \*TAG = "TEST";

int i = 0;

void **timer\_test\_callback**(void \*arg);

void **app\_main**(void)

{

**esp\_timer\_init**();//定时器接口初始化

    esp\_timer\_handle\_t test\_timer;//创建定时器句柄

    const esp\_timer\_create\_args\_t test\_timer\_args = {

            .callback = &timer\_test\_callback,

            /\* argument specified here will be passed to timer callback function \*/

            .arg = NULL,

            .name = "one-test\_timer"

    };//定时器参数结构体

    // esp\_timer\_create(&test\_timer\_args, &test\_timer);

    // esp\_timer\_start\_periodic(test\_timer, 500 \* 1000);

    //创建定时器

**ESP\_ERROR\_CHECK**(**esp\_timer\_create**(&test\_timer\_args, &test\_timer));

    //启动定时器

**ESP\_ERROR\_CHECK**(**esp\_timer\_start\_periodic**(test\_timer, 500 \* 1000));

    while (i);

**esp\_timer\_stop**(test\_timer);

**esp\_timer\_delete**(test\_timer);

**ESP\_LOGI**(TAG, "Stopped and deleted timers");

}

void **timer\_test\_callback**(void \*arg)

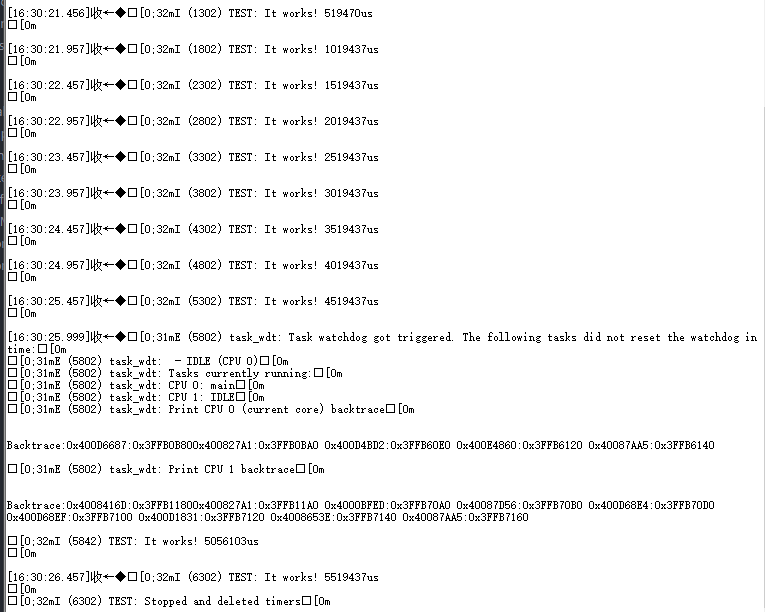
{

**ESP\_LOGI**(TAG, "It works! %lldus\n", **esp\_timer\_get\_time**());

    i++;

}

运行结果:



运行结果中显示看门狗被触发，目前原因未知。