

OpenHarmonyOS内核开发 软件定时器



目 录

CONTENTS

- [01] 什么是软件定时器
- [02] 软件定时器运行机制
- [03] 软件定时器的相应接口
- [04] 如何使用软件定时器

01

什么是软件定时器

软件定时器，是基于系统Tick时钟中断且由软件来模拟的定时器，当经过设定的Tick时钟计数值后会触发用户定义的回调函数。定时精度与系统Tick时钟的周期有关。



02

软件定时器运行机制

一般而言，OpenHarmonyOS内核的软件定时器以Tick为基本计时单位。

- (1) 用户创建并启动一个软件定时器时；
- (2) 当Tick中断到来时，在Tick中断处理函数中扫描软件定时器的计时全局链表，看是否有定时器超时，若有则将超时的定时器记录下来；
- (3) Tick中断处理函数结束后，软件定时器任务（优先级为最高）被唤醒，在该任务中调用之前记录下来的定时器的超时回调函数。

03

软件定时器的接口

任务接口的头文件

/kernel/liteos_m/kernel/include/los_swtmr.h

OpenHarmonyOS内核开发中，任务接口有很多，主要分为几大类：

- (1) 创建、删除软件定时器；
- (2) 启动、停止软件定时器；
- (3) 获得软件定时器信息。

03

任务的接口

功能分类	接口名	功能描述
创建、删除软件定时器	LOS_SwtmrCreate	创建软件定时器
	LOS_SwtmrDelete	删除软件定时器
启动、停止软件定时器	LOS_SwtmrStart	启动软件定时器
	LOS_SwtmrStop	停止软件定时器
获得软件定时器信息	LOS_SwtmrTimeGet	获得软件定时器剩余Tick数

03

任务的接口

```
UINT32 LOS_SwtmrCreate(UINT32 interval,  
                        UINT8 mode,  
                        SWTMR_PROC_FUNC handler,  
                        UINT32 *swtmrID,  
                        UINT32 arg);
```

```
UINT32 LOS_SwtmrDelete(UINT32 swtmrID);
```

该函数主要功能是创建软件定时器。

- interval: 定时器超时时间。
- mode: 定时器模式。
- handler: 定时器回调函数。
- swtmrID: 定时器ID指针。
- arg: 定时器回调函数参数。

返回LOS_OK为成功，其余为失败。

该函数主要功能是删除软件定时器。

- swtmrID: 定时器ID指针。

返回LOS_OK为成功，其余为失败。

03

软件定时器的接口

```
UINT32 LOS_SwtmrStart(UINT32 swtmrID);
```

该函数主要功能是启动软件定时器。

参数swtmrID为软件定时器的句柄。

返回LOS_OK为成功，其余为失败。

```
UINT32 LOS_SwtmrStop(UINT32 swtmrID);
```

该函数主要功能是停止软件定时器。

参数swtmrID为软件定时器的句柄。

返回LOS_OK为成功，其余为失败。

04

如何使用软件定时器

1、打开sdk下面路径的文件

```
vendor/lockzhiner/rk2206/samples/a3_kernel_timer/kernel_timer_example.c
```

2、创建任务

在timer_example函数中，通过LOS_SwtmrCreate函数创建定时器1和定时器2，并通过LOS_SwtmrStart函数启动定时器1和定时器2。1S超时后会触发Timer1_Timeout函数并打印日志；2S超时后会触发Timer2_Timeout函数并打印日志。

如何使用软件定时器

```
void timer_example()
```

```
{
```

```
    unsigned int timer_id1, timer_id2;
```

```
    unsigned int ret;
```

```
    ret = LOS_SwtmrCreate(1000, LOS_SWTMR_MODE_PERIOD,  
timer1_timeout, &timer_id1, NULL);
```

```
    if (ret == LOS_OK)
```

```
    {
```

```
        ret = LOS_SwtmrStart(timer_id1);
```

```
        if (ret != LOS_OK)
```

```
        {
```

```
            printf("start timer1 fail ret:0x%x\n", ret);
```

```
            return;
```

```
        }
```

```
    }
```

```
    else
```

```
    {
```

```
        printf("create timer1 fail ret:0x%x\n", ret);
```

```
        return;
```

```
    }
```

```
    ret = LOS_SwtmrCreate(2000,  
LOS_SWTMR_MODE_PERIOD, timer2_timeout, &timer_id2,  
NULL);
```

如何使用软件定时器

```
if (ret == LOS_OK)
```

```
{
```

```
    ret = LOS_SwtmrStart(timer_id2);
```

```
    if (ret != LOS_OK)
```

```
    {
```

```
        printf("start timer2 fail ret:0x%x\n", ret);
```

```
        return;
```

```
    }
```

```
}
```

```
else
```

```
{
```

```
    printf("create timer2 fail ret:0x%x\n"), ret;
```

```
    return;
```

```
}
```

```
}
```



如何使用软件定时器

timer1_timeout函数执行一次打印日志。

```
void timer1_timeout(void *arg)
{
    printf("This is Timer1 Timeout function\n");
}
```

timer2_timeout函数执行一次打印日志。

```
void timer2_timeout(void *arg)
{
    printf("This is Timer2 Timeout function\n");
}
```

3、修改编译脚本

修改 `vendor/lockzhiner/rk2206/sample` 路径下 `BUILD.gn` 文件，指定 `timer_example` 参与编译。

```
"/a3_kernel_timer:timer_example",
```

修改 `device/lockzhiner/rk2206/sdk_liteos` 路径下 `Makefile` 文件，添加 `-ltimer_example` 参与编译。

```
hardware_LIBS = -lhal_iohardware -lhardware -ltimer_example
```

4、编译固件

```
hb set -root .
```

```
hb set
```

```
hb build -f
```

5、烧写固件

6、通过串口查看结果

运行结果

This is Timer1 Timeout function

This is Timer1 Timeout function

This is Timer2 Timeout function

This is Timer1 Timeout function

This is Timer1 Timeout function

This is Timer2 Timeout function

谢谢聆听

单击此处添加副标题内容