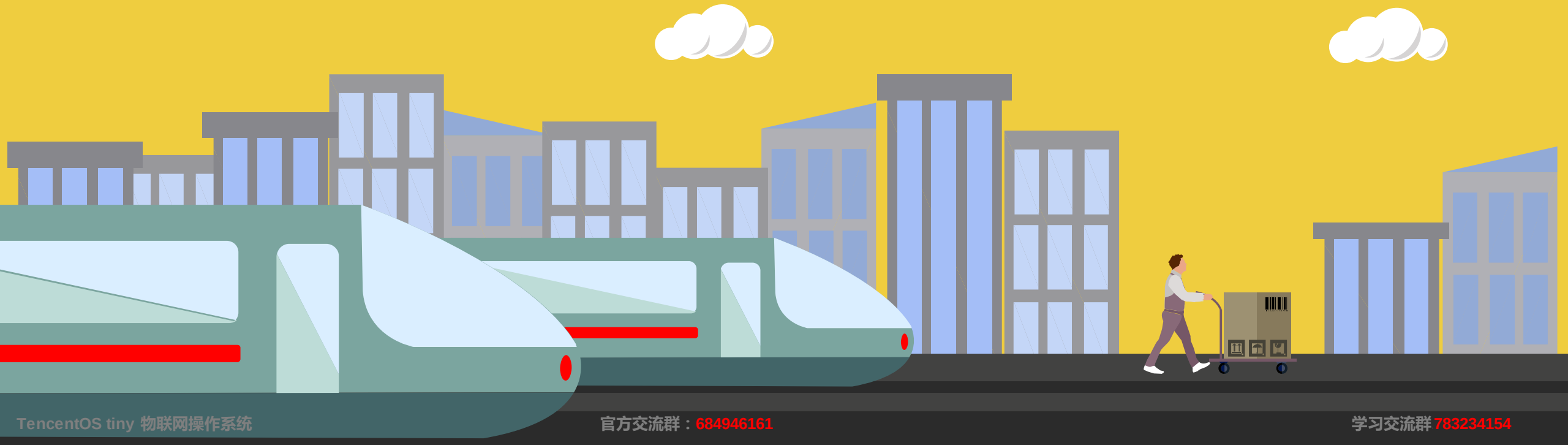
2019 TencentOS tiny 物联网操作系统

学习永无止境~

——杰杰

本讲义所有权归杰杰所有



关于我

一个走在物联网路上的小菜鸟~

博客：https://jiejietop.cn

CSDN：https://blog.csdn.net/jiejiemcu

GitHub：https://github.com/jiejieTop

个人公众号

目录

01.软件定时器的基本概念

TencentOS tiny 的软件定时器是由操作系统提供的一类系统接口，它构建在硬件定时器基础之上，使系统能够提供不受硬件定时器资源限制的定时器服务，本质上软件定时器的使用相当于扩展了定时器的数量，允许创建更多的定时业务，它实现的功能与硬件定时器也是类似的。

软件定时器的超时处理是指：在定时时间到达之后就会自动触发一个超时，然后系统跳转到对应的函数去处理这个超时，此时，调用的函数也被称回调函数。

回调函数的执行环境可以是中断，也可以是任务

TOS_CFG_TIMER_AS_PROC 为 1 ：回调函数的执行环境是中断

TOS_CFG_TIMER_AS_PROC 为 0 ：回调函数的执行环境是任务

01.软件定时器的基本概念

软件定时器在被创建之后，当经过设定的超时时间后会触发回调函数，定时精度与系统时钟的周期有关，一般可以采用SysTick作为软件定时器的时基

TencentOS tiny提供的软件定时器支持单次模式和周期模式，单次模式和周期模式的定时时间到之后都会调用软件定时器的回调函数。

单次模式：当用户创建了定时器并启动了定时器后，指定超时时间到达，只执行一次回调函数之后就将该定时器停止，不再重新执行。

周期模式：这个定时器会按照指定的定时时间循环执行回调函数，直到将定时器销毁。

02.软件定时器的数据结构

软件定时器列表

timer_ctl_t

next_expires：记录下一个到期的软件定时器时间。
list：软件定时器列表，所有的软件定时器都会被挂载到这个列表中。

软件定时器任务相关的数据结构

- k_timer_task：软件定时器任务控制块
- k_timer_task_stk：软件定时器任务栈，其大小为TOS_CFG_TIMER_TASK_STK_SIZE
- k_timer_task_prio：软件定时器任务优先级，值为TOS_CFG_TIMER_TASK_PRIO
- k_timer_task_stk_addr：软件定时器任务栈起始地址。
- k_timer_task_stk_size：软件定时器任务栈大小。

02.软件定时器的数据结构

软件定时器的回调函数

typedef void (*k_timer_callback_t)(void *arg);

软件定时器控制块

k_timer_t

软件定时器的工作模式

- TOS_OPT_TIMER_ONESHOT：**单次**模式，软件定时器在超时后，只会执行一次回调函数，它的状态将被设置为TIMER_STATE_COMPLETED，不再重新执行它的回调函数，
- TOS_OPT_TIMER_PERIODIC ：**周期**模式 ，软件定时器在超时后，会执行对应的回调函数，同时根据软件定时器控制块中的period成员变量的值再重新插入软件定时器列表中，这个定时器会按照指定的定时时间循环执行（周期性执行）回调函数，直到用户将定时器销毁。

01.软件定时器的基本概念

软件定时器的状态

- TIMER_STATE_UNUSED：未使用状态。
- TIMER_STATE_STOPPED：创建了软件定时器，但此时软件定时器未启动或者处于停止状态，调用tos_timer_create()函数接口或者在软件定时器启动后调用tos_timer_stop()函数接口后，定时器将变成该状态。
- TIMER_STATE_RUNNING：软件定时器处于运行状态，在定时器被创建后调用tos_timer_start()函数接口，定时器将变成该状态，表示定时器运行时的状态。
- TIMER_STATE_COMPLETED：软件定时器已到期，只有在软件定时器的模式选择为TOS_OPT_TIMER_ONESHOT时才可能发生，表示软件定时器已经完成了。

03.创建软件定时器

```
__API__ k_err_t tos_timer_create(k_timer_t *tmr,
                                 k_tick_t delay,
                                 k_tick_t period,
                                 k_timer_callback_t callback,
                                 void *cb_arg,
                                 k_opt_t opt)
```

参数	说明（杰杰）
tmr	软件定时器控制块指针
delay	软件定时器第一次运行的延迟时间间隔
period	软件定时器的周期
callback	软件定时器的回调函数，在超时时调用（由用户自己定义）
cb_arg	用于回调函数传入的形参（void指针类型）
opt	软件定时器的工作模式（单次 / 周期）

03.创建软件定时器

过程：

1. 判断传入的参数是否正确：软件定时器控制块不为null，回调函数不为null，如果是创建周期模式的软件定时器，那么 period 参数则不可以为0，而如果是单次模式的软件定时器，参数delay则不可以为0，无论是何种模式的软件定时器，delay 参数与 period 参数都不可以为K_ERR_TIMER_PERIOD_FOREVER。
2. 根据传入的参数将软件定时器控制块的成员变量赋初值，软件定时器状态state被设置为TIMER_STATE_STOPPED，expires 则被设置为0，因为还尚未启动软件定时器。
3. 调用tos_list_init()函数将软件定时器控制块中可挂载到k_tick_list列表的节点初始化。

04.销毁软件定时器

__API__ k_err_t tos_timer_destroy(k_timer_t *tmr)

软件定时器销毁函数是根据软件定时器控制块直接销毁的，销毁之后软件定时器的所有信息都会被清除，而且不能再次使用这个软件定时器，如果软件定时器处于运行状态，那么就需要将被销毁的软件定时器停止，然后再进行销毁操作。

1. 判断软件定时器是否有效，然后根据软件定时器状态判断软件定时器是否创建，如果是未使用状态TIMER_STATE_UNUSED，则直接返回错误代码K_ERR_TIMER_INACTIVE。

2. 如果软件定时器状态是 运行状态 TIMER_STATE_RUNNING，那么调用timer_takeoff()函数将软件定时器停止。

3. 最后调用timer_reset()函数将软件定时器控制块的内容重置，主要是将软件定时器的状态设置为未使用状态TIMER_STATE_UNUSED，将对应的回调函数设置为null。

停止软件定时器（内部函数）

在销毁软件定时器的时候提到了timer_takeoff()函数，那么就来看看这个函数具体是怎么样停止软件定时器的，其实本质上就是将软件定时器从软件定时器列表中移除。

1. 首先通过TOS_LIST_FIRST_ENTRY宏定义将软件定时器列表k_timer_ctl.list中的第一个软件定时器取出，因为防止软件定时器列表中的第一个软件定时器被移除了，而没有重置软件定时器列表中的相关的信息，因此此时要记录一下第一个软件定时器。

2. 调用tos_list_del()将软件定时器从软件定时器列表中移除，表示中国软件定时器就被停止了，因为不知软件定时器列表中，中国软件定时器也就不会被处理。

3. 判断一下移除的软件定时器是不是第一个软件定时器，如果是，则重置相关信息。如果软件定时器列表中不存在其他软件定时器，则将软件定时器列表的下一个到期时间设置为TOS_TIME_FOREVER，反正则让软件定时器列表的下一个到期时间为第二个软件定时器。

05. 启动软件定时器

__API__ k_err_t tos_timer_start(k_timer_t *tmr)

在创建成功软件定时器的时候，软件定时器的状态从TIMER_STATE_UNUSED（未使用状态）变成TIMER_STATE_STOPPED（创建但未启动 / 停止状态），创建完成的软件定时器是未运行的，用户在需要的时候可以启动它，TencentOS tiny提供了软件定时器启动函数tos_timer_start()。启动软件定时器的本质就是将软件定时器插入软件定时器列表k_timer_ctl.list 中，既然是这样子，那么很显然需要根据软件定时器的不同状态进行不同的处理。

其实现过程如下：判断软件定时器控制块是否为null，然后判断软件定时器状态，如果为未使用状态TIMER_STATE_UNUSED则直接返回错误代码K_ERR_TIMER_INACTIVE；如果为已经运行状态TIMER_STATE_RUNNING，那么将软件定时器停止，然重新插入软件定时器列表k_timer_ctl.list中；如果是TIMER_STATE_STOPPED或者TIMER_STATE_COMPLETED状态，则将软件定时器的状态重新设置为运行状态TIMER_STATE_RUNNING，并且插入软件定时器列表k_timer_ctl.list中。

插入软件定时器列表

插入软件定时器列表的函数是timer_place()，这个函数会根据软件定时器的到期时间升序排序，然后再插入

1. 根据软件定时器的到期时间expires（相对值）与系统当前时间k_tick_count计算得出到期时间expires（绝对值）。

2. 通过for循环TOS_LIST_FOR_EACH找到合适的位置插入软件定时器列表，此时插入软件定时器列表安装到期时间升序插入。

3. 找到合适的位置后，调用tos_list_add_tail()函数将软件定时器插入软件定时器列表。

4. 如果插入的软件定时器是唯一定时器列表中的第一个，那么相应的，下一个到期时间就是这个软件定时器的到期时间，将到期时间更新： k_timer_ctl.next_expires = tmr->expires。如果TOS_CFG_TIMER_AS_PROC宏定义为0，则判断一下软件定时器任务是否处于睡眠状态，如果是则调用tos_task_delay_abort()函数恢复软件定时器任务运行，以便于更新它休眠的时间，因为此时是需要更新软件定时器任务睡眠的时间的，毕竟第一个软件定时器到期时间已经改变了。

5. 如果软件定时器任务处于挂起状态，表示并没有软件定时器在工作，现在插入了软件定时器，需要调用tos_task_resume()函数将软件定时器任务唤醒。

06.停止软件定时器（外部函数）

__API__ k_err_t tos_timer_stop(k_timer_t *tmr)

停止软件定时器的本质也是调用timer_takeoff()函数将软件定时器从软件定时器列表中移除，但是在调用这个函数之前还好做一些相关的判断，这样能保证系统的稳定性。

1. 对软件定时器控制块检测，如果软件定时器控制块为null，则直接返回错误代码。

2. 如果软件定时器状态为未使用状态TIMER_STATE_UNUSED，则直接返回错误代码K_ERR_TIMER_INACTIVE。

3. 如果软件定时器状态为TIMER_STATE_COMPLETED 或者是TIMER_STATE_STOPPED，则不需要停止软件定时器，因为这个软件定时器是未启动的。则直接返回错误代码K_ERR_TIMER_STOPPED。

4. 如果软件定时器状态为TIMER_STATE_RUNNING ，就将软件定时器状态设置为停止状态TIMER_STATE_STOPPED，并且调用timer_takeoff()函数将软件定时器从软件定时器列表中移除。

07.软件定时器的处理（在中断上下文环境）

TencentOS tiny的软件定时器是可以在中断上下文环境来处理回调函数的，因此当软件定时器到期后，会在tos_tick_handler()函数中调用timer_update()来处理软件定时器。这个函数在每次tick中断到来的时候都会判断一下是否有软件定时器到期，如果有则去处理它。

- 判断软件定时器的下一个到期时间k_timer_ctl.next_expires是否小于k_tick_count，如果是小于则表示还未到期，直接退出。
- 反之则表示到期，此时要遍历软件定时器列表，找到所有到期的软件定时器，并处理他们。
- 到期后的处理就是：调用timer_takeoff()函数将到期的软件定时器停止，如果是周期工作的定时器就调用timer_place()函数将它重新插入软件定时器列表中（它到期的相对时间就是软件定时器的周期值：tmr->expires = tmr->period ）；如果是单次工作模式的软件定时器，就仅将软件定时器状态设置为TIMER_STATE_COMPLETED。
- 调用软件定时器的回调函数处理相关的工作：(*tmr->cb)(tmr->cb_arg)

08.软件定时器的处理（在任务上下文环境）

创建软件定时器任务

这个任务将在timer_init()函数中被创建

软件定时器任务主体

1. 调用timer_next_expires_get()函数获取软件定时器列表中的下一个到期时间，并且更新next_expires的值。

2. 根据next_expires的值，判断一下软件定时器任务应该休眠多久，在多久后到期时才唤醒软件定时器任务并且处理回调函数。也就是说，软件定时器任务在软件定时器没有到期的时候是不会被唤醒的，都是处于休眠状态，调用tos_task_delay()函数将任务进入休眠状态，此时任务会被挂载到系统的延时（时基）列表中。

08.软件定时器的处理（在任务上下文环境）

软件定时器任务主体

1. 任务如果被唤醒了，或者被恢复运行了，则表明软件定时器到期了或者有新的软件定时器插入列表了，那么在唤醒之后就要判断一下是哪种情况，如果是到期了则处理对应的回调函数：首先调用timer_takeoff()函数将到期的软件定时器停止，如果是周期工作的定时器就调用timer_place()函数将它重新插入软件定时器列表中（它到期的相对时间就是软件定时器的周期值：tmr->expires = tmr->period）；如果是单次工作模式的软件定时器，就仅将软件定时器状态设置为TIMER_STATE_COMPLETED。（这里也是会遍历软件定时器列表以处理所有到期的软件定时器）

2. 最后将调用软件定时器的回调函数处理相关的工作：(*tmr->cb)(tmr->cb_arg)。

3. 如果定时器还未到期，并且软件定时器任务被唤醒了，那么就表示有新的软件定时器插入列表了，此时要更新一下任务的睡眠时间，因为软件定时器任务主体是一个while循环，还是会回到 timer_next_expires_get()函数中重新获取下一个唤醒任务的时间的。

获取软件定时器下一个到期时间（内部函数）

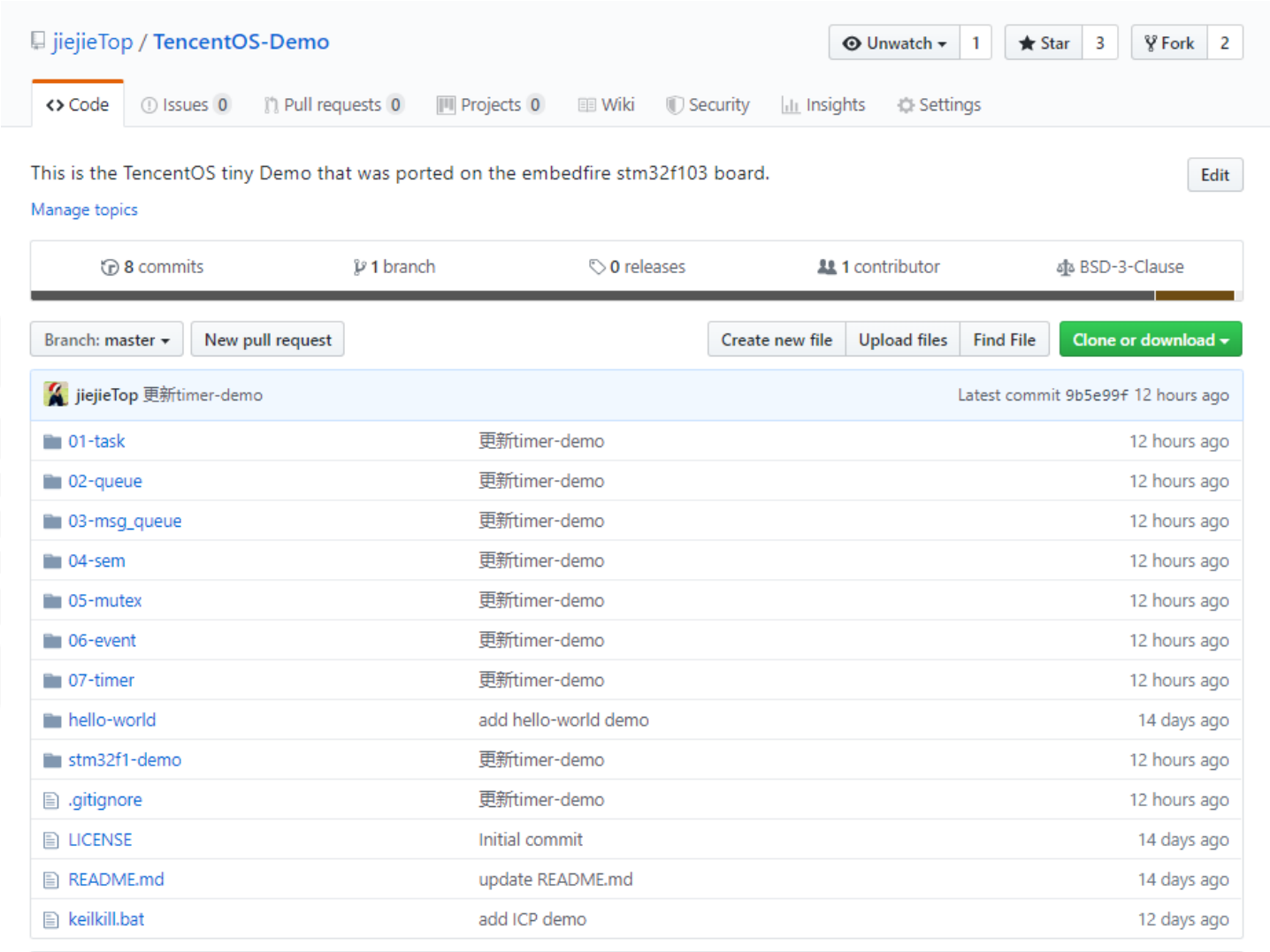
__KERNEL__ k_tick_t timer_next_expires_get(void)

timer_next_expires_get()就是用于获取软件定时器下一个到期时间，如果软件定时器到期时间是TOS_TIME_FOREVER，就返回TOS_TIME_FOREVER，如果下一个到期时间小于k_tick_count则直接返回0，表示已经到期了，可以直接处理它，而如果是其他值，则需要减去k_tick_count，将其转变为相对值，因为调用这个函数就是为了知道任务能休眠多少时间。

09. 实验

代码获取：https://github.com/jiejieTop/TencentOS-Demo

或者关注公众号，在后台回复“19”




加我微信 ▲ 进群交流
物联网|云|小程序|RTOS


设为星标
长按扫码
物联网IoT
全栈开发
关注物联网那些事，专注分享物联网知识

THANKS