第22章 软件定时器

主讲内容

22.1 软件定时器的基本概念

22.2 软件定时器应用场景

22.3 软件定时器的精度

22.4 软件定时器控制块

22.5 软件定时器函数接口讲解

22.6 软件定时器任务

22.7 软件定时器实验

22.8 总结

参考资料:《μCOS-III内核实现与应用开发实战指南》

软件定时器的基本概念

定时器，是指从指定的时刻开始，经过一个指定时间，然后触发一个超时事件，用户可以自定义定时器的周期与频率。

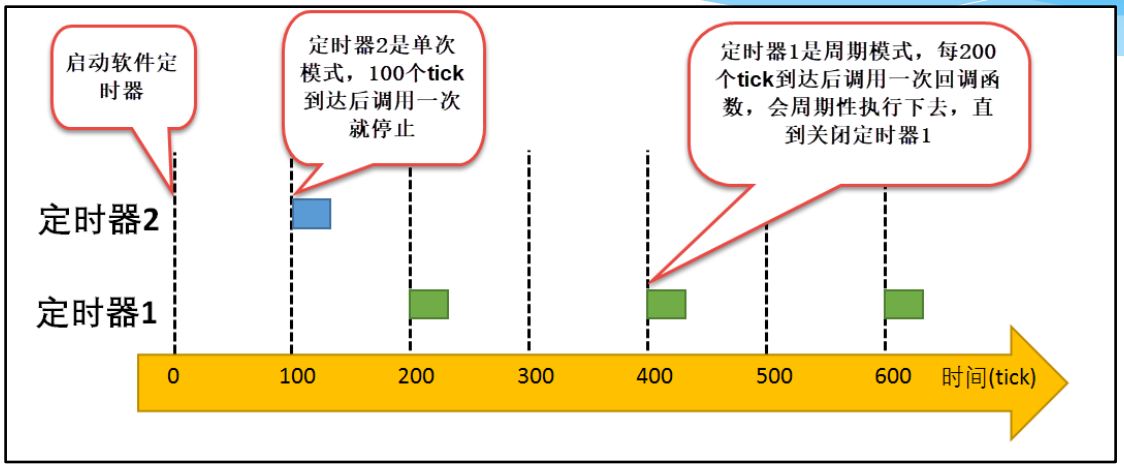
硬件定时器是芯片本身提供的定时功能。一般是由外部晶振提供给芯片输入时钟，芯片向软件模块提供一组配置寄存器，接受控制输入，到达设定时间值后芯片中断控制器产生时钟中断。硬件定时器的精度一般很高，可以达到纳秒级别，并且是中断触发方式。

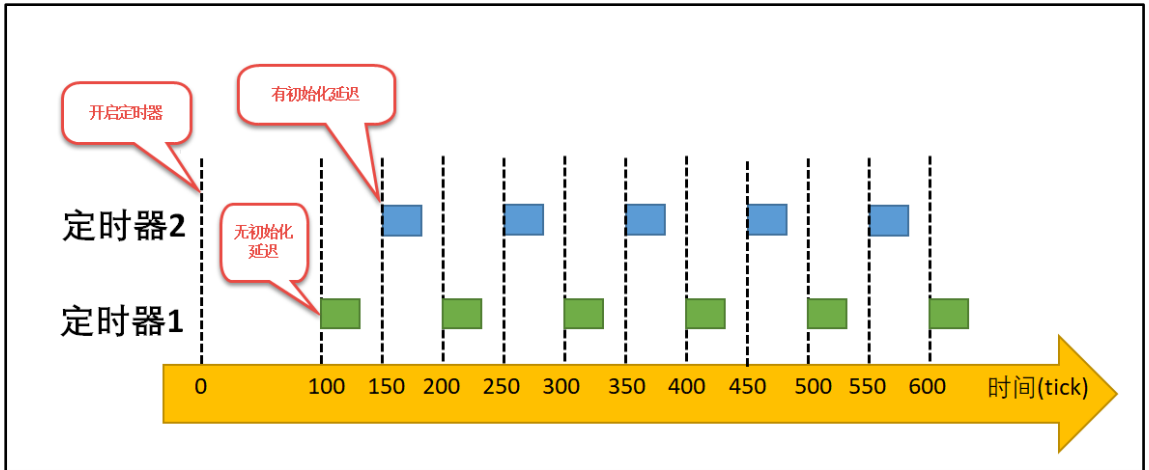
软件定时器，软件定时器是由操作系统提供的一类系统接口，它构建在硬件定时器基础之上，使系统能够提供不受硬件定时器资源限制的定时器服务，它实现的功能与硬件定时器也是类似的。

使用硬件定时器时，每次在定时时间到达之后就会自动触发一个中断，用户在中断中处理信息；而使用软件定时器时，需要我们在创建软件定时器时指定时间到达后要调用的函数（也称超时函数/回调函数，为了统一，下文均用回调函数描述），在回调函数中处理信息。

μC/OS操作系统提供软件定时器功能，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。μC/OS软件定时器功能上支持：
裁剪：能通过宏关闭软件定时器功能。
软件定时器创建。
软件定时器启动。
软件定时器停止。
软件定时器删除。

支持单次模式和周期模式


启动软件定时器
定时器2是单次模式，100个tick到达后调用一次就停止
定时器1是周期模式，每200个tick到达后调用一次回调函数，会周期性执行下去，直到关闭定时器1
定时器2
定时器1
0
100
200
300
400
500
600
时间(tick)


开启定时器
有初始化延迟
无初始化延迟
定时器2
定时器1
0
100
150
200
250
300
350
400
450
500
550
600
时间(tick)

软件定时器应用场景

在很多应用中，我们需要一些定时器任务，硬件定时器受硬件的限制，数量上不足以满足用户的实际需求，无法提供更多的定时器，那么可以采用软件定时器来完成，由软件定时器代替硬件定时器任务。但需要注意的是软件定时器的精度是无法和硬件定时器相比的，因为在软件定时器的定时过程中是极有可能被其他中断所打断，因为软件定时器的执行上下文环境是任务。所以，软件定时器更适用于对时间精度要求不高的任务，一些辅助型的任务。

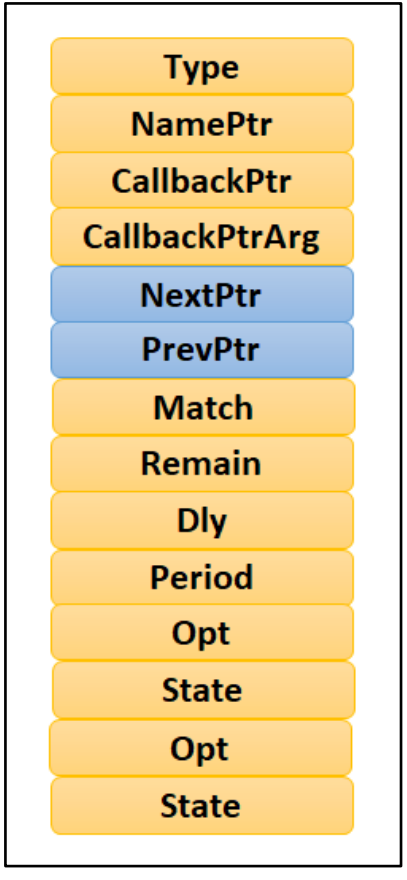
软件定时器的精度

μC/OS软件定时器的精度（分辨率）决定于系统时基频率，也就是变量OS_CFG_TMR_TASK_RATE_HZ的值，它是以Hz为单位的。

注意：定时器任务的频率OS_CFG_TMR_TASK_RATE_HZ的值不能大于系统时基频率OS_CFG_TMR_TASK_RATE_HZ的值。

软件定时器控制块

μC/OS的软件定时器也属于内核对象，是一个可以裁剪的功能模块，同样在系统中由一个控制块管理其相关信息，软件定时器的控制块中包含创建的软件定时器基本信息



创建软件定时器函数OSTmrCreate()

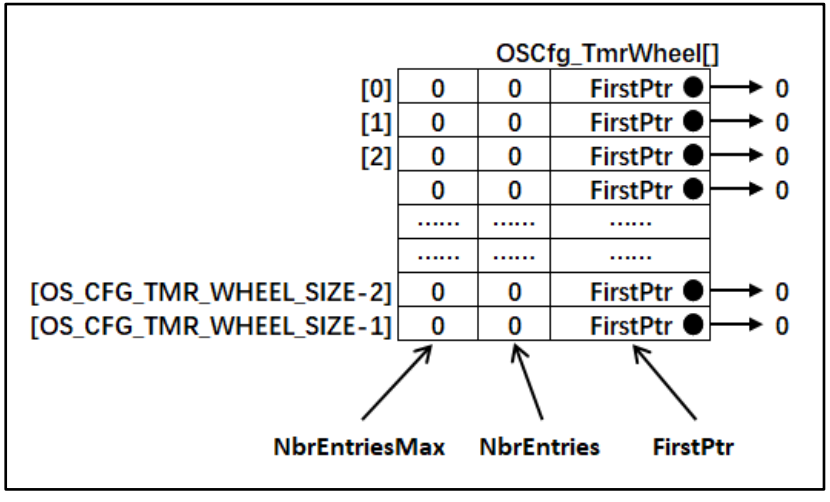
在创建的时候需要指定定时器延时初始值dly、定时器周期、定时器工作模式、回调函数等。每个软件定时器只需少许的RAM空间，理论上μC/OS支持无限多个软件定时器，只要RAM足够即可。

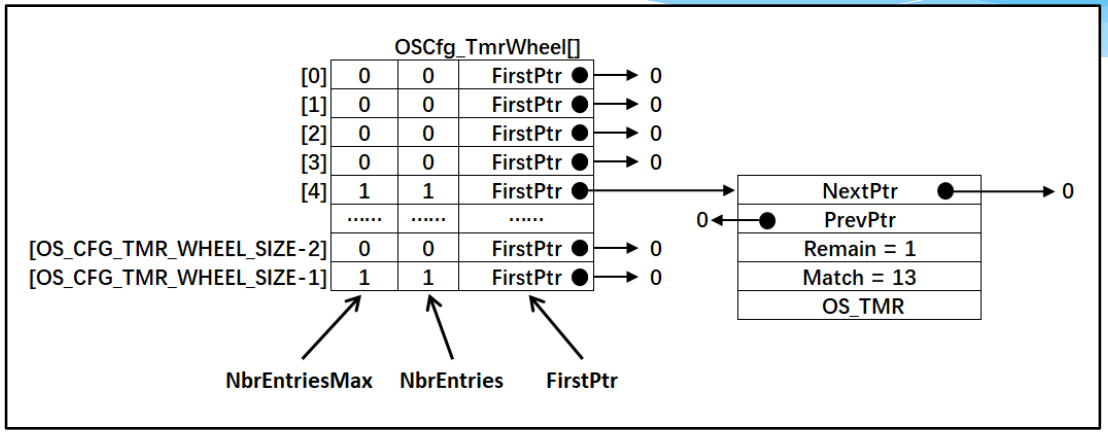
启动软件定时器函数OSTmrStart()

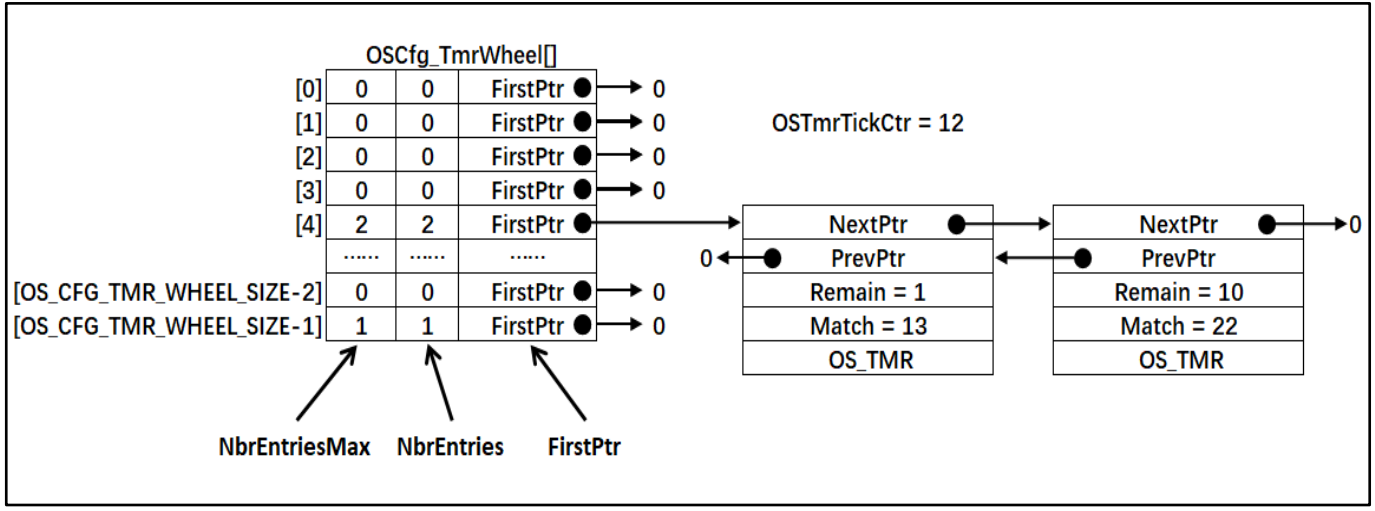
将已经创建的软件定时器添加到定时器列表中

软件定时器列表管理

μC/OS对软件定时器列表的管理就跟时间节拍一样，采用哈希算法。将不同的定时器变量根据其对 OSCfg_TmrWheelSize 余数的不同插入数组 OSCfg_TmrWheel[OS_CFG_TMR_WHEEL_SIZE]中去




OSCfg_TmrWheel[]
[0] 0 0 FirstPtr 0
[1] 0 0 FirstPtr 0
[2] 0 0 FirstPtr 0
[3] 0 0 FirstPtr 0
[4] 1 1 FirstPtr
....
[OS_CFG_TMR_WHEEL_SIZE-2] 0 0 FirstPtr 0
[OS_CFG_TMR_WHEEL_SIZE-1] 1 1 FirstPtr 0
NextPtr 0
0 PrevPtr
Remain = 1
Match = 13
OS_TMR
NbrEntriesMax
NbrEntries
FirstPtr


OSCfg_TmrWheel[]
[0] 0 0 FirstPtr 0
[1] 0 0 FirstPtr 0
[2] 0 0 FirstPtr 0
[3] 0 0 FirstPtr 0
[4] 2 2 FirstPtr
....
[OS_CFG_TMR_WHEEL_SIZE-2] 0 0 FirstPtr 0
[OS_CFG_TMR_WHEEL_SIZE-1] 1 1 FirstPtr 0
OSTmrTickCtr = 12
NextPtr
0 PrevPtr
Remain = 1
Match = 13
OS_TMR
NextPtr 0
PrevPtr
Remain = 10
Match = 22
OS_TMR
NbrEntriesMax
NbrEntries
FirstPtr

停止定时器函数OSTmrStop()

OSTmrStop()函数用于停止一个软件定时器。软件定时器被停掉之后可以调用OSTmrStart()函数重启，但是重启之后定时器是从头计时，而不是接着上次停止的时刻继续计时

删除软件定时器函数OSTmrDel()

软件定时器任务OS_TmrTask()

软件定时器的回调函数的上下文是在任务中，所有，系统中必须要一个任务来管理所有的软件定时器，等到定时时间到达后就调用定时器对应的回调函数

软件定时器实验

软件定时器实验是在μC/OS中创建了一个应用任务 AppTaskTmr，在该任务中创建一个软件定时器，周期性定时 1s，每次定时完成切换 LED1 的亮灭状态，并且打印时间戳的计时，检验定时的精准度。

总结

μC/OS允许用户建立任意数量的定时器（只限制于处理器的RAM大小）。

回调函数是在定时器任务中被调用，所以回调函数的上下文环境是在任务中，并且运行回调函数时调度器处于被锁状态。一般我们编写的回调函数越简短越好，并且不能在回调函数中等待消息队列、信号量、事件等操作，否则定时器任务会被挂起，导致定时器任务崩溃，这是绝对不允许的。

此外我们还需要注意几点：

- 回调函数是在定时器任务中被执行的，这意味着定时器任务需要有足够的栈空间供回调函数去执行。
- 回调函数是在根据定时器队列中依次存放的，所以在定时器时间到达后回调函数是依次被执行的。
- 定时器任务的执行时间决定于：有多少个定时器期满，执行定时器中的回调函数需多少时间。因为回调函数是由用户提供，它可能很大程度上影响了定时器任务的执行时间。
- 回调函数被执行时会锁调度器，所以我们必须让回调函数尽可能地短，以便其他任务能正常运行。

THANKS

淘宝：fire-stm32.taobao.com

论坛： www.firebbs.cn

扫描进入淘宝店铺