描述

本库并不涉及内置的定时器，它仅仅使用了“millis()”，以一种较为粗糙的调度方式来决定什么时候该做某事。

例子

Arduino的“delay”功能是福也是祸。为初学者展示如何使LED闪烁时，它是极好的。  
但是一旦你想要更复杂而且开始放慢你的“loop”功能，你会遇到问题。  
delay方法的缺点是当“delay”正在运行时，别的任何事都不能进行。  
(外部中断相应等一些活动仍然可以执行，译者注）  
你不能更新显示屏，或者检查按键是否被按下。  
  
使用Timer库版本的例子如下所示

#include "Timer.h"

Timer t;

int pin = 13;

void setup()

{

pinMode(pin, OUTPUT);

t.pulse(pin, 10 \* 60 \* 1000, HIGH); // 10 minutes

}

void loop()

{

t.update();

}

“pulse”方法有三个参数：要改变的引脚，改变它的周期和它的初始状态。  
调用t.update()会花费几微秒来运行, 除非已经经过了合适了时间周期. 让我们来看另一个使用了两个定时事件的例子。  
一个用来闪烁LED，另一个读取A0口并把结果显示在串口监视器。

#include "Timer.h"

Timer t;

int pin = 13;

void setup()

{

Serial.begin(9600);

pinMode(pin, OUTPUT);

t.oscillate(pin, 100, LOW);

t.every(1000, takeReading);

}

void loop()

{

t.update();

}

void takeReading()

{

Serial.println(analogRead(0));

}

首先要注意的是我们使用了一个回调函数“takeReading”。  
我们使用“every”命令将它连接到定时器。  
这样的话，每秒会调用这个函数一次。  
我们也在定时器上加入了另一个事件，利用“oscillate”方法。这会使LED每100微秒切换状态。  
每个事件都有一个整型ID与它关联，因此你可以暂停一个事件，正如我们在下面这个例子所做的。  
这个例子每2秒向串口监视器写入数据；快速闪烁LED 5秒钟然后停止，再缓慢闪烁5次。

#include "Timer.h"

Timer t;

int ledEvent;

void setup()

{

Serial.begin(9600);

int tickEvent = t.every(2000, doSomething);

Serial.print("2 second tick started id=");

Serial.println(tickEvent);

pinMode(13, OUTPUT);

ledEvent = t.oscillate(13, 50, HIGH);

Serial.print("LED event started id=");

Serial.println(ledEvent);

int afterEvent = t.after(10000, doAfter);

Serial.print("After event started id=");

Serial.println(afterEvent);

}

void loop()

{

t.update();

}

void doSomething()

{

Serial.print("2 second tick: millis()=");

Serial.println(millis());

}

void doAfter()

{

Serial.println("stop the led event");

t.stop(ledEvent);

t.oscillate(13, 500, HIGH, 5);

}

你最多可以在一个定时器上附加10个事件。

安装

你可以在这下载到这个库 <https://github.com/JChristensen/Timer>

这个库适用于Arduino 1.0以及更早的版本。（译者在1.0.4版本测试可用）

用法

**int every(long period, callback)**   
每“period”个毫秒调用“callback”方法。  
返回定时事件的ID。

**int every(long period, callback, int repeatCount**)  
每“period”个毫秒调用“callback”方法，共“repeatCount”次。  
返回定时事件的ID。

**int after(long duration, callback)**  
在“period”个毫秒后调用“callback”方法一次。  
返回定时事件的ID。

**int oscillate(int pin, long period, int startingValue)**  
每“period”个毫秒切换数字输出引脚“pin”的状态。引脚的初始值由“startingValue”定义，应为HIGH或LOW。  
返回定时事件的ID。

**int oscillate(int pin, long period, int startingValue, int repeatCount)**  
每“period”个毫秒切换数字输出引脚“pin”的状态，共“repeatCount”次。引脚的初始值由“startingValue”定义，应为HIGH或LOW。  
返回定时事件的ID。

**int pulse(int pin, long period, int startingValue)**  
在“period”个毫秒后切换数字输出引脚“pin”的状态一次。引脚的初始值由“startingValue”定义，应为HIGH或LOW。  
返回定时事件的ID。

**int stop(int id)**  
停止正在运行的定时事件。  
返回定时事件的ID。

**int update()**  
必须在“loop”中调用。Must be called from 'loop'. 它会服务所有与定时器相关的事件。