

# [PATCH 4/11] LED: Add LED Timer Trigger

---

*Source:* <http://linux.derkeiler.com/Mailing-Lists/Kernel/2006-01/msg11422.html>

---

- *From:* Richard Purdie <[rpurdie@xxxxxxxx](mailto:rpurdie@xxxxxxxx)>
  - *Date:* Tue, 31 Jan 2006 13:41:37 +0000
- 

Add an example of a complex LED trigger in the form of a generic timer which triggers the LED its attached to at a user specified frequency and duty cycle.

Signed-off-by: Richard Purdie <[rpurdie@xxxxxxxx](mailto:rpurdie@xxxxxxxx)>

Index: linux-2.6.15/drivers/leds/Kconfig

```
----- linux-2.6.15.orig/drivers/leds/Kconfig 2006-01-29 16:13:48.000000000 +0000
+++ linux-2.6.15/drivers/leds/Kconfig 2006-01-29 20:32:16.000000000 +0000
@@ -22,5 +22,12 @@
```

These triggers allow kernel events to drive the LEDs and can be configured via sysfs. If unsure, say Y.

```
+config LEDS_TRIGGER_TIMER
+ tristate "LED Timer Trigger"
+ depends LEDS_TRIGGERS
+ help
+ This allows LEDs to be controlled by a programmable timer
+ via sysfs. If unsure, say Y.
+
+endmenu
```

Index: linux-2.6.15/drivers/leds/Makefile

```
----- linux-2.6.15.orig/drivers/leds/Makefile 2006-01-29 16:13:48.000000000 +0000
+++ linux-2.6.15/drivers/leds/Makefile 2006-01-29 20:32:16.000000000 +0000
@@ -3,3 +3,6 @@
```

```
obj-$(CONFIG_NEW_LEDS) += led-core.o
obj-$(CONFIG_LEDS_CLASS) += led-class.o
obj-$(CONFIG_LEDS_TRIGGERS) += led-triggers.o
+
+# LED Triggers
+obj-$(CONFIG_LEDS_TRIGGER_TIMER) += ledtrig-timer.o
```

Index: linux-2.6.15/drivers/leds/ledtrig-timer.c

```
----- /dev/null 1970-01-01 00:00:00.000000000 +0000
+++ linux-2.6.15/drivers/leds/ledtrig-timer.c 2006-01-29 17:40:11.000000000 +0000
@@ -0,0 +1,204 @@
```

## [PATCH 4/11] LED: Add LED Timer Trigger

```
+/*
+ * LED Kernel Timer Trigger
+ *
+ * Copyright 2005–2006 Openedhand Ltd.
+ *
+ * Author: Richard Purdie <rpurdie@xxxxxxxxxxxxxxxx>
+ *
+ * This program is free software; you can redistribute it and/or modify
+ * it under the terms of the GNU General Public License version 2 as
+ * published by the Free Software Foundation.
+ *
+ */
+
+#include <linux/config.h>
+#include <linux/module.h>
+#include <linux/kernel.h>
+#include <linux/init.h>
+#include <linux/list.h>
+#include <linux/spinlock.h>
+#include <linux/device.h>
+#include <linux/sysdev.h>
+#include <linux/timer.h>
+#include <linux/leds.h>
+#include "leds.h"
+
+struct timer_trig_data {
+ unsigned long duty; /* duty cycle, as a percentage */
+ unsigned long frequency; /* frequency of blinking, in Hz */
+ unsigned long delay_on; /* milliseconds on */
+ unsigned long delay_off; /* milliseconds off */
+ struct timer_list timer;
+};
+
+static void led_timer_function(unsigned long data)
+{
+ struct led_device *led_dev = (struct led_device *) data;
+ struct timer_trig_data *timer_data = led_dev->trigger_data;
+ unsigned long brightness = LED_OFF;
+ unsigned long delay = timer_data->delay_off;
+
+ write_lock(&led_dev->lock);
+
+ if (!timer_data->frequency) {
+ led_set_brightness(led_dev, LED_OFF);
+ write_unlock(&led_dev->lock);
+ return;
+ }
+
+ if (!led_dev->brightness) {
+ brightness = LED_FULL;
+ delay = timer_data->delay_on;
```

## [PATCH 4/11] LED: Add LED Timer Trigger

```
+ }
+
+ led_set_brightness(led_dev, brightness);
+
+ mod_timer(&timer_data->timer, jiffies + msecs_to_jiffies(delay));
+ write_unlock(&led_dev->lock);
+}
+
+/* led_dev write lock needs to be held */
+static void led_timer_setdata(struct led_device *led_dev, unsigned long duty, unsigned long frequency)
+{
+ struct timer_trig_data *timer_data = led_dev->trigger_data;
+ signed long duty1;
+
+ if (frequency > 500)
+ frequency = 500;
+
+ if (duty > 100)
+ duty = 100;
+
+ duty1 = duty - 50;
+
+ timer_data->duty = duty;
+ timer_data->frequency = frequency;
+ if (frequency != 0) {
+ timer_data->delay_on = (50 - duty1) * 1000 / 50 / frequency;
+ timer_data->delay_off = (50 + duty1) * 1000 / 50 / frequency;
+ }
+
+ mod_timer(&timer_data->timer, jiffies);
+}
+
+static ssize_t led_duty_show(struct class_device *dev, char *buf)
+{
+ struct led_device *led_dev = dev->class_data;
+ struct timer_trig_data *timer_data;
+
+ read_lock(&led_dev->lock);
+ timer_data = led_dev->trigger_data;
+ sprintf(buf, "%lu\n", timer_data->duty);
+ read_unlock(&led_dev->lock);
+
+ return strlen(buf) + 1;
+}
+
+static ssize_t led_duty_store(struct class_device *dev, const char *buf, size_t size)
+{
+ struct led_device *led_dev = dev->class_data;
+ struct timer_trig_data *timer_data;
+ int ret = -EINVAL;
+ char *after;
```

## [PATCH 4/11] LED: Add LED Timer Trigger

```
+
+ unsigned long state = simple_strtoul(buf, &after, 10);
+ if (after - buf > 0) {
+   ret = after - buf;
+   write_lock(&led_dev->lock);
+   timer_data = led_dev->trigger_data;
+   led_timer_setdata(led_dev, state, timer_data->frequency);
+   write_unlock(&led_dev->lock);
+ }
+
+ return ret;
+}
+
+
+static ssize_t led_frequency_show(struct class_device *dev, char *buf)
+{
+   struct led_device *led_dev = dev->class_data;
+   struct timer_trig_data *timer_data;
+
+   read_lock(&led_dev->lock);
+   timer_data = led_dev->trigger_data;
+   sprintf(buf, "%lu\n", timer_data->frequency);
+   read_unlock(&led_dev->lock);
+
+   return strlen(buf) + 1;
+}
+
+static ssize_t led_frequency_store(struct class_device *dev, const char *buf, size_t size)
+{
+   struct led_device *led_dev = dev->class_data;
+   struct timer_trig_data *timer_data;
+   int ret = -EINVAL;
+   char *after;
+
+   unsigned long state = simple_strtoul(buf, &after, 10);
+   if (after - buf > 0) {
+     ret = after - buf;
+     write_lock(&led_dev->lock);
+     timer_data = led_dev->trigger_data;
+     led_timer_setdata(led_dev, timer_data->duty, state);
+     write_unlock(&led_dev->lock);
+   }
+
+   return ret;
+}
+
+static CLASS_DEVICE_ATTR(duty, 0644, led_duty_show, led_duty_store);
+static CLASS_DEVICE_ATTR(frequency, 0644, led_frequency_show, led_frequency_store);
+
+void timer_trig_activate(struct led_device *led_dev)
+{
```

## [PATCH 4/11] LED: Add LED Timer Trigger

```
+ struct timer_trig_data *timer_data;
+
+ timer_data = kzalloc(sizeof(struct timer_trig_data), GFP_KERNEL);
+ if (!timer_data)
+ return;
+
+ led_dev->trigger_data = timer_data;
+
+ init_timer(&timer_data->timer);
+ timer_data->timer.function = led_timer_function;
+ timer_data->timer.data = (unsigned long) led_dev;
+
+ timer_data->duty = 50;
+
+ class_device_create_file(led_dev->class_dev, &class_device_attr_duty);
+ class_device_create_file(led_dev->class_dev, &class_device_attr_frequency);
+}
+
+void timer_trig_deactivate(struct led_device *led_dev)
+{
+ struct timer_trig_data *timer_data = led_dev->trigger_data;
+ if (timer_data) {
+ class_device_remove_file(led_dev->class_dev, &class_device_attr_duty);
+ class_device_remove_file(led_dev->class_dev, &class_device_attr_frequency);
+ del_timer_sync(&timer_data->timer);
+ kfree(timer_data);
+ }
+}
+
+static struct led_trigger timer_led_trigger = {
+ .name = "timer",
+ .activate = timer_trig_activate,
+ .deactivate = timer_trig_deactivate,
+};
+
+static int __init timer_trig_init(void)
+{
+ return led_trigger_register(&timer_led_trigger);
+}
+
+static void __exit timer_trig_exit(void)
+{
+ led_trigger_unregister(&timer_led_trigger);
+}
+
+module_init(timer_trig_init);
+module_exit(timer_trig_exit);
+
+MODULE_AUTHOR("Richard Purdie <rpurdie@xxxxxxxxxxxxxxxx>");
+MODULE_DESCRIPTION("Timer LED trigger");
+MODULE_LICENSE("GPL");
```

—

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@xxxxxxxxxxxxxxxxxxx

More majordomo info at <http://vger.kernel.org/majordomo-info.html>

Please read the FAQ at <http://www.tux.org/lkml/>

---

- *Follow-Ups:*

- ◆ **Re: [PATCH 4/11] LED: Add LED Timer Trigger**

◇ *From:* Jan-Benedict Glaw

- Prev by Date: **Re: CD writing in future Linux (stirring up a hornets' nest)**
- Next by Date: **[patch 1/5] MMC OMAP driver**
- Previous by thread: **[PATCH 2/11] LED: Add LED Class**
- Next by thread: **Re: [PATCH 4/11] LED: Add LED Timer Trigger**
- Index(es):
  - ◆ **Date**
  - ◆ **Thread**