

# [PATCH 8/11] LED: Add LED device support for ixp4xx devices

---

*Source:* <http://linux.derkeiler.com/Mailing-Lists/Kernel/2006-01/msg11425.html>

---

- *From:* Richard Purdie <[rpurdie@xxxxxxxx](mailto:rpurdie@xxxxxxxx)>
  - *Date:* Tue, 31 Jan 2006 13:41:49 +0000
- 

From: John Bowler <[jbowler@xxxxxxx](mailto:jbowler@xxxxxxx)>

NEW\_LEDS support for ixp4xx boards where LEDs are connected to the GPIO lines.

This includes a new generic ixp4xx driver (leds-ixp4xx-gpio.c name "IXP4XX-GPIO-LED")

Signed-off-by: John Bowler <[jbowler@xxxxxxx](mailto:jbowler@xxxxxxx)>

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

```
--- linux-2.6.15/drivers/leds/Kconfig 1970-01-01 00:00:00.000000000 +0000
+++ linux-2.6.15/drivers/leds/Kconfig 1970-01-01 00:00:00.000000000 +0000
@@ -43,6 +43,15 @@ config LEDS_SPITZ
```

This option enables support for the LEDs on Sharp Zaurus SL-Cxx00 series (C1000, C3000, C3100).

```
+config LEDS_IXP4XX
+ tristate "LED Support for GPIO connected LEDs on IXP4XX processors"
+ depends LEDS_CLASS && ARCH_IXP4XX
+ help
+ This option enables support for the LEDs connected to GPIO
+ outputs of the Intel IXP4XX processors. To be useful the
+ particular board must have LEDs and they must be connected
+ to the GPIO lines. If unsure, say Y.
```

```
+
+config LEDS_TRIGGER_TIMER
+ tristate "LED Timer Trigger"
+ depends LEDS_TRIGGERS
```

```
--- linux-2.6.15/drivers/leds/Makefile 1970-01-01 00:00:00.000000000 +0000
+++ linux-2.6.15/drivers/leds/Makefile 1970-01-01 00:00:00.000000000 +0000
@@ -8,6 +8,7 @@ obj-$(CONFIG_LEDS_TRIGGERS) += led-trig
obj-$(CONFIG_LEDS_CORGI) += leds-corgi.o
obj-$(CONFIG_LEDS_LOCOMO) += leds-locomo.o
obj-$(CONFIG_LEDS_SPITZ) += leds-spitz.o
+obj-$(CONFIG_LEDS_IXP4XX) += leds-ixp4xx-gpio.o
```

# LED Triggers

[PATCH 8/11] LED: Add LED device support for ixp4xx devices

```
obj-$(CONFIG_LEDS_TRIGGER_TIMER) += ledtrig-timer.o
--- linux-2.6.15/drivers/leds/leds-ixp4xx-gpio.c 1970-01-01 00:00:00.000000000 +0000
+++ linux-2.6.15/drivers/leds/leds-ixp4xx-gpio.c 1970-01-01 00:00:00.000000000 +0000
@@ -0,0 +1,209 @@
+/*
+ * IXP4XX GPIO driver LED driver
+ *
+ * Author: John Bowler <jbowler@xxxxxxx>
+ *
+ * Copyright (c) 2006 John Bowler
+ *
+ * Permission is hereby granted, free of charge, to any
+ * person obtaining a copy of this software and associated
+ * documentation files (the "Software"), to deal in the
+ * Software without restriction, including without
+ * limitation the rights to use, copy, modify, merge,
+ * publish, distribute, sublicense, and/or sell copies of
+ * the Software, and to permit persons to whom the
+ * Software is furnished to do so, subject to the
+ * following conditions:
+ *
+ * The above copyright notice and this permission notice
+ * shall be included in all copies or substantial portions
+ * of the Software.
+ *
+ * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF
+ * ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED
+ * TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
+ * PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT
+ * SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR
+ * ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
+ * ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
+ * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
+ * OTHER DEALINGS IN THE SOFTWARE.
+ */
+
+#include <linux/config.h>
+#include <linux/kernel.h>
+#include <linux/init.h>
+#include <linux/platform_device.h>
+#include <linux/spinlock.h>
+#include <linux/leds.h>
+#include <asm/arch/hardware.h>
+
+extern spinlock_t gpio_lock;
+
+/* Up to 16 gpio lines are possible. */
+#define GPIO_MAX 16
+static struct ixp4xxgpioled_device {
+ struct led_device ancestor;
```

[PATCH 8/11] LED: Add LED device support for ixp4xx devices

```
+ int flags;
+} ixp4xxgpioled_devices[GPIO_MAX];
+
+void ixp4xxgpioled_brightness_set(struct led_device *pled, enum led_brightness value)
+{
+ const struct ixp4xxgpioled_device *const ixp4xx_dev =
+ container_of(pled, struct ixp4xxgpioled_device, ancestor);
+ const u32 gpio_pin = ixp4xx_dev - ixp4xxgpioled_devices;
+
+ if (gpio_pin < GPIO_MAX && ixp4xx_dev->ancestor.name != 0) {
+ /* Set or clear the 'gpio_pin' bit according to the style
+ * and the required setting (value > 0 == on)
+ */
+ const int gpio_value =
+ (value > 0) == (ixp4xx_dev->flags != IXP4XX_GPIO_LOW) ?
+ IXP4XX_GPIO_HIGH : IXP4XX_GPIO_LOW;
+
+ {
+ unsigned long flags;
+ spin_lock_irqsave(&gpio_lock, flags);
+ gpio_line_set(gpio_pin, gpio_value);
+ spin_unlock_irqrestore(&gpio_lock, flags);
+ }
+ }
+ }
+
+/* LEDs are described in resources, the following iterates over the valid
+ * LED resources.
+ */
+#define for_all_leds(i, pdev) \
+ for (i=0; i<pdev->num_resources; ++i) \
+ if (pdev->resource[i].start < GPIO_MAX && \
+ pdev->resource[i].name != 0)
+
+/* The following applies 'operation' to each LED from the given platform,
+ * the function always returns 0 to allow tail call elimination.
+ */
+static int apply_to_all_leds(struct platform_device *pdev,
+ void (*operation)(struct led_device *pled)) {
+ int i;
+ for_all_leds(i, pdev)
+ operation(&ixp4xxgpioled_devices[pdev->resource[i].start].ancestor);
+ return 0;
+ }
+
+#ifdef CONFIG_PM
+static int ixp4xxgpioled_suspend(struct platform_device *pdev, pm_message_t state)
+{
+ return apply_to_all_leds(pdev, led_device_suspend);
+ }
+
+
```

## [PATCH 8/11] LED: Add LED device support for ixp4xx devices

```
+static int ixp4xxgpioled_resume(struct platform_device *pdev)
+{
+ return apply_to_all_leds(pdev, led_device_resume);
+}
+endif
+
+static void ixp4xxgpioled_remove_one_led(struct led_device *pled) {
+ led_device_unregister(pled);
+ pled->name = 0;
+}
+
+static int ixp4xxgpioled_remove(struct platform_device *pdev)
+{
+ return apply_to_all_leds(pdev, ixp4xxgpioled_remove_one_led);
+}
+
+static int ixp4xxgpioled_probe(struct platform_device *pdev)
+{
+ /* The board level has to tell the driver where the
+ * LEDs are connected – there is no way to find out
+ * electrically. It must also say whether the GPIO
+ * lines are active high or active low.
+ *
+ * To do this read the num_resources (the number of
+ * LEDs) and the struct resource (the data for each
+ * LED). The name comes from the resource, and it
+ * isn't copied.
+ */
+ int i;
+ for_all_leds(i, pdev) {
+ const u8 gpio_pin = pdev->resource[i].start;
+ int rc;
+
+ if (ixp4xxgpioled_devices[gpio_pin].ancestor.name == 0) {
+ unsigned long flags;
+
+ spin_lock_irqsave(&gpio_lock, flags);
+ gpio_line_config(gpio_pin, IXP4XX_GPIO_OUT);
+ /* The config can, apparently, reset the state,
+ * I suspect the gpio line may be an input and
+ * the config may cause the line to be latched,
+ * so the setting depends on how the LED is
+ * connected to the line (which affects how it
+ * floats if not driven).
+ */
+ gpio_line_set(gpio_pin, IXP4XX_GPIO_HIGH);
+ spin_unlock_irqrestore(&gpio_lock, flags);
+
+ ixp4xxgpioled_devices[gpio_pin].flags =
+ pdev->resource[i].flags & IORESOURCE_BITS;
+ }
+ }
```

[PATCH 8/11] LED: Add LED device support for ixp4xx devices

```
+ ixp4xxgpioled_devices[gpio_pin].ancestor.name =
+ pdev->resource[i].name;
+
+ /* This is how a board manufacturer makes the LED
+ * come on on reset – the GPIO line will be high, so
+ * make the LED light when the line is low...
+ */
+ if (ixp4xxgpioled_devices[gpio_pin].flags != IXP4XX_GPIO_LOW)
+ ixp4xxgpioled_devices[gpio_pin].ancestor.brightness = 100;
+ else
+ ixp4xxgpioled_devices[gpio_pin].ancestor.brightness = 0;
+
+ ixp4xxgpioled_devices[gpio_pin].ancestor.flags = 0;
+
+ ixp4xxgpioled_devices[gpio_pin].ancestor.brightness_set =
+ ixp4xxgpioled_brightness_set;
+
+ ixp4xxgpioled_devices[gpio_pin].ancestor.default_trigger = 0;
+ }
+
+ rc = led_device_register(&pdev->dev,
+ &ixp4xxgpioled_devices[gpio_pin].ancestor);
+ if (rc < 0) {
+ ixp4xxgpioled_devices[gpio_pin].ancestor.name = 0;
+ ixp4xxgpioled_remove(pdev);
+ return rc;
+ }
+ }
+
+ return 0;
+}
+
+static struct platform_driver ixp4xxgpioled_driver = {
+ .probe = ixp4xxgpioled_probe,
+ .remove = ixp4xxgpioled_remove,
+ #ifdef CONFIG_PM
+ .suspend = ixp4xxgpioled_suspend,
+ .resume = ixp4xxgpioled_resume,
+ #endif
+ .driver = {
+ .name = "IXP4XX-GPIO-LED",
+ },
+ };
+
+static int __devinit ixp4xxgpioled_init(void)
+{
+ return platform_driver_register(&ixp4xxgpioled_driver);
+}
+
+static void ixp4xxgpioled_exit(void)
+{
```

[PATCH 8/11] LED: Add LED device support for ixp4xx devices

```
+ platform_driver_unregister(&ixp4xxgpioled_driver);
+}
+
+module_init(ixp4xxgpioled_init);
+module_exit(ixp4xxgpioled_exit);
+
+MODULE_AUTHOR("John Bowler <jbowler@xxxxxxx>");
+MODULE_DESCRIPTION("IXP4XX GPIO LED driver");
+MODULE_LICENSE("MIT");
```

—

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@xxxxxxxxxxxxxxxxx  
More majordomo info at <http://vger.kernel.org/majordomo-info.html>  
Please read the FAQ at <http://www.tux.org/lkml/>

- 
- Prev by Date: [\*\*\*\[PATCH 1/11\] LED Class Documentation\*\*\*](#)
  - Next by Date: [\*\*\*Re: Rescan SCSI Bus without /proc/scsi?\*\*\*](#)
  - Previous by thread: [\*\*\*\[PATCH 1/11\] LED Class Documentation\*\*\*](#)
  - Next by thread: [\*\*\*Re: Rescan SCSI Bus without /proc/scsi?\*\*\*](#)
  - Index(es):
    - ◆ [\*\*\*Date\*\*\*](#)
    - ◆ [\*\*\*Thread\*\*\*](#)