```
/*
 * Simple gptimers example
 *
http://docs.blackfin.uclinux.org/doku.php?id=linux-kernel:drivers:gptimers
 *
 * Copyright 2007-2009 Analog Devices Inc.
 *
 * Licensed under the GPL-2 or later.
 */

#include <linux/interrupt.h>
#include <linux/module.h>

#include <asm/gptimers.h>
#include <asm/portmux.h>

/* ... random driver includes ... */

#define DRIVER_NAME "gptimer_example"

struct gptimer_data {
        uint32_t period, width;
};
static struct gptimer_data data;

/* ... random driver state ... */

static irqreturn_t gptimer_example_irq(int irq, void *dev_id)
{
        struct gptimer_data *data = dev_id;

        /* make sure it was our timer which caused the interrupt */
        if (!get_gptimer_intr(TIMER5_id))
                return IRQ_NONE;

        /* read the width/period values that were captured for the waveform */
        data->width = get_gptimer_pwidth(TIMER5_id);
        data->period = get_gptimer_period(TIMER5_id);

        /* acknowledge the interrupt */
        clear_gptimer_intr(TIMER5_id);

        /* tell the upper layers we took care of things */
        return IRQ_HANDLED;
}

/* ... random driver code ... */

static int __init gptimer_example_init(void)
{
        int ret;

        /* grab the peripheral pins */
        ret = peripheral_request(P_TMR5, DRIVER_NAME);
        if (ret) {
                printk(KERN_NOTICE DRIVER_NAME ": peripheral request failed\n");
                return ret;
        }

        /* grab the IRQ for the timer */
        ret = request_irq(IRQ_TIMER5, gptimer_example_irq, IRQF_SHARED,
DRIVER_NAME, &data);
        if (ret) {
                printk(KERN_NOTICE DRIVER_NAME ": IRQ request failed\n");
                peripheral_free(P_TMR5);
                return ret;
        }

        /* setup the timer and enable it */
        set_gptimer_config(TIMER5_id, WDTH_CAP | PULSE_HI | PERIOD_CNT |
IRQ_ENA);
        enable_gptimers(TIMER5bit);

        return 0;
}
module_init(gptimer_example_init);

static void __exit gptimer_example_exit(void)
{
        disable_gptimers(TIMER5bit);
        free_irq(IRQ_TIMER5, &data);
        peripheral_free(P_TMR5);
}
module_exit(gptimer_example_exit);

MODULE_LICENSE("BSD");
```