gpio中断响应实验

殷　　威

October 6, 2015

1　实验目的

本实验为beaglebone开发板的gpio中断响应测试实验。实验的目的是通过简单的gpio测试实验了解android操作系统对外部中断的响应原理以及内核模块的编写方法，为以后的产品开发做铺垫。

2　实验设备

本实验所需的设备有：beaglebone开发板、led灯、一个小电阻、4根细电线、u盘。

3　实验步骤

步骤：

1. 在beaglebone上安装android操作系统(这里不多说)；
2. 将led灯和小电阻串联，引出两极线；
3. 将beaglebone用usb数据线和电脑连接，开启串口调试工具minicom；
4. android操作系统启动成功；
5. 这里采用p9扩展上的15、23针脚，即gpio1_16、gpio_17引脚，将led灯连接到15针脚和GND脚(注意极性)；
6. 在自己的操作系统上编译内核模块，将编译得到的gpio_ts.ko拷贝到u盘上；
7. 将u盘插到beaglebone上，使用insmod命令装载上步的模块；
8. 取一根电线，连接到23针脚上；
9. 上步中的电线的另一端作为测试端，触摸p9扩展的1、7号引脚来给定低电平、高电平或者直接用手触摸即可；

4　实验现象

当触摸1、7号引脚或者用手触摸测试端的时候，从minicom界面可以看到响应后打印的信息，而且led闪烁，证明gpio中断响应成功。

5 实验代码

```
/*************************************************************************
        > File Name: gpio_ts.c
        > Author:
        > Mail:
        > Created Time: Tue 29 Sep 2015 08:08:48 AM CST
 ************************************************************************/

#include  <linux/module.h>           //提供模块创建所需的所有
函数
#include  <linux/init.h>            //提供__init,__exit
#include  <linux/kernel.h>       //提供
#include  <linux/gpio.h>       //驱动头文件gpio
#include  <linux/interrupt.h>    //中断头
#include  <linux/irq.h>

#include  <asm/gpio.h>    //包含了mach/gpio.h            mach/gpio.又
包含了hplat/gpio.h
#include  <plat/am33xx.h>       //包含了架构的针脚基地址 暂时没
用

#define     DEVICE_NAME  "gpio_ts"

#define     GPIO_NUM(gpio,offset)   (gpio*32+offset)

unsigned gpio_out_num;   //用于中断响应

static irqreturn_t gpio_handler(int irq,void* dev_id) {
    int  value;
    value = gpio_get_value(gpio_out_num);
    if( !value )
    {
        gpio_set_value(gpio_out_num,1);
    }else{
        gpio_set_value(gpio_out_num,0);
    }

    printk(KERN_INFO "gpio%引脚的值为d%d",gpio_out_num, \
    gpio_get_value(gpio_out_num));

    return  IRQ_HANDLED;
}


/* 模块退出的时候执行 */
static void  __exit gpio_ts_exit(void)
{
```

```
    printk(KERN_INFO "the␣module␣exits");
}

/* 载入模块时，创建设备文件，并初始化引脚，检测中断信号 */
static int __init gpio_ts_init(void)
{
    int rq_status,out_status;
    unsigned gpio_num;
    unsigned irq_num;

    gpio_num=GPIO_NUM(1,17);
    gpio_out_num = GPIO_NUM(1,16);

    rq_status = gpio_request(gpio_num,"中断触发口");
    out_status = gpio_request(gpio_out_num,"中断响应口");


    if( rq_status || out_status )
    {
        printk(KERN_INFO"申请失败gpio..");
        return -EFAULT;
    }

    rq_status = gpio_direction_input(gpio_num);
    out_status = gpio_direction_output(gpio_out_num,1);
    if( rq_status || out_status )
    {
        printk(KERN_INFO"设置方向失败..");
        return -EFAULT;
    }

    irq_num = gpio_to_irq(gpio_num);

    if( !irq_num )
    {
        printk(KERN_INFO"转失败gpioirq...");
        return -EFAULT;
    }else{
        irq_set_irq_type(irq_num,IRQ_TYPE_EDGE_FALLING);
    }



    printk(KERN_INFO "下面开始中断申请%d引脚值,%d",irq_num,gpio_
\
    get_value(gpio_out_num));

     enable_irq(irq_num);
```

```
    rq_status=request_irq(irq_num,gpio_handler,  \
    IRQF_TRIGGER_FALLING,DEVICE_NAME,NULL);

    if( rq_status )
    {
        printk(KERN_INFO"中断申请失败...");
        return -EFAULT;
    }
    printk(KERN_INFO "中断申请成功!");
    return 0;
}



module_init(gpio_ts_init);
module_exit(gpio_ts_exit);

MODULE_LICENSE("GPL");
MODULE_AUTHOR("yinwei");
MODULE_DESCRIPTION("gpio␣irq␣test");
```