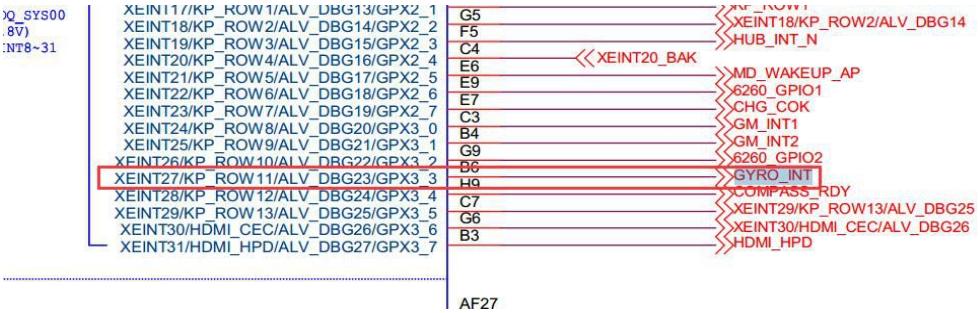
iTOP-4412 实现中断驱动例程之二（添加按键）

在上一节我们已经详细讲解了“BACK ”、“HOME”按键控制 led，实现其亮灭，这次我们将在上一次的基础上再添加一个按键“ SLEEP”，用它实现控制蜂鸣器的响与不响。

首先我们先要查看“SLEEP”所对应的 GPIO 口，在 scp 核心板我们可以发现它所对应的 io 口，如下图：



接下来我们在查看“BEEP”引脚所对应的端口，在 scp 核心板我们可以发现它所对应的 io 口，如下图：

我们只是在例程一的基础上添加代码，首先在“tatic int irq_probe(struct platform_device *pdev)”函数下添加：

```
static int irq_probe(struct platform_device *pdev)
{
        int ret, i;
        char *banner = "irq_test Initialize\n";
        printk(banner);
        gpio_free(EXYNOS4_GPL2(0));
```

```
        gpio_free(EXYNOS4_GPK1(1));
        gpio_free(EXYNOS4_GPD0(0));
        for(i=0; i<LED_NUM; i++)
        {
                ret = gpio_request(led_gpios[i], "LED");
                if (ret) {
                        printk("%s: request GPIO %d for LED failed, ret = %d\n", DRIVER_NAME,

led_gpios[i], ret);
                        return ret;
                }
                s3c_gpio_cfgpin(led_gpios[i], S3C_GPIO_OUTPUT);
                gpio_set_value(led_gpios[i], 0);
         }
         ret = gpio_request(EXYNOS4_GPX1(1), "EINT9");
         if (ret) {
                printk("%s: request GPIO %d for EINT9 failed, ret = %d\n", DRIVER_NAME,

EXYNOS4_GPX1(1), ret);
                return ret;
         }
         s3c_gpio_cfgpin(EXYNOS4_GPX1(1), S3C_GPIO_SFN(0xF));
         s3c_gpio_setpull(EXYNOS4_GPX1(1), S3C_GPIO_PULL_UP);
         gpio_free(EXYNOS4_GPX1(1));



         ret = gpio_request(EXYNOS4_GPX1(2), "EINT10");
         if (ret) {
                printk("%s: request GPIO %d for EINT10 failed, ret = %d\n", DRIVER_NAME,

EXYNOS4_GPX1(2), ret);
                return ret;
         }
         s3c_gpio_cfgpin(EXYNOS4_GPX1(2), S3C_GPIO_SFN(0xF));
         s3c_gpio_setpull(EXYNOS4_GPX1(2), S3C_GPIO_PULL_UP);
      gpio_free(EXYNOS4_GPX1(2));



         ret = gpio_request(EXYNOS4_GPX3(3), "EINT27");
         if (ret) {
```

```
                printk("%s: request GPIO %d for EINT27 failed, ret = %d\n", DRIVER_NAME,

EXYNOS4_GPX3(3), ret);
                return ret;
        }
        s3c_gpio_cfgpin(EXYNOS4_GPX3(3), S3C_GPIO_SFN(0xF));
        s3c_gpio_setpull(EXYNOS4_GPX3(3), S3C_GPIO_PULL_UP);
        gpio_free(EXYNOS4_GPX3(3));

    ret = gpio_request(EXYNOS4_GPX2(1), "EINT17");
        if (ret) {
                printk("%s: request GPIO %d for EINT17 failed, ret = %d\n", DRIVER_NAME,

EXYNOS4_GPX2(1), ret);
                return ret;
        }
        s3c_gpio_cfgpin(EXYNOS4_GPX2(1), S3C_GPIO_SFN(0xF));
        s3c_gpio_setpull(EXYNOS4_GPX2(1), S3C_GPIO_PULL_UP);
        gpio_free(EXYNOS4_GPX2(1));

#if 1
        ret = request_irq(IRQ_EINT(9), eint9_interrupt,
        IRQ_TYPE_EDGE_FALLING /*IRQF_TRIGGER_FALLING*/, "eint9", pdev);
        if (ret < 0) {
                printk("Request IRQ %d failed, %d\n", IRQ_EINT(9), ret);
                goto exit;
        }

        ret = request_irq(IRQ_EINT(10), eint10_interrupt,
        IRQ_TYPE_EDGE_FALLING /*IRQF_TRIGGER_FALLING*/, "eint10", pdev);
        if (ret < 0) {
                printk("Request IRQ %d failed, %d\n", IRQ_EINT(10), ret);
                goto exit;
        }

        ret = request_irq(IRQ_EINT(27), eint27_interrupt,
        IRQ_TYPE_EDGE_FALLING /*IRQF_TRIGGER_FALLING*/, "eint27", pdev);
        if (ret < 0) {
                printk("Request IRQ %d failed, %d\n", IRQ_EINT(27), ret);
                goto exit;
```

```
        }

    ret = request_irq(IRQ_EINT(17), eint17_interrupt,
        IRQ_TYPE_EDGE_FALLING /*IRQF_TRIGGER_FALLING*/, "eint17", pdev);
        if (ret < 0) {
                printk("Request IRQ %d failed, %d\n", IRQ_EINT(17), ret);
                goto exit;
        }

#endif
        return 0;
exit:
        return ret;
}
```

这里我们主要以“SLEEP”为例讲解，因为我们需要把“SLEEP”所对应的 io 口配置为复用中断功能，首先我们需要查看手册 4412datasheet，打开手册，在里面搜“GPX3CON”我们可以发现 GPX3CON[ ]=0xF 时，引脚为中断功能。

```
s3c_gpio_cfgpin(EXYNOS4_GPX3(3), S3C_GPIO_SFN(0xF));
```

中断必须有中断号，没有中断号，当中断发生时我们将不知道函数该去哪里执行，同样在手册里面搜“XEINT_27”我们将会发现它的中断号，如下图：

56.7 I/O Description

Table 56-1 describes the keypad interface I/O.

Table 56-1 Keypad Interface I/O Description

Signal	I/O	Description	Pad		Type
			Port0	Port1	
KP_ROW[13]	I	KEYPAD interface row[13] data	XEINT_29 (GPX3[5])	XEINT_29 (GPX3[5])	muxed
KP_ROW [12]	I	KEYPAD interface row[12] data	XEINT_28 (GPX3[4])	XEINT_28 (GPX3[4])	muxed
KP_ROW [11]	I	KEYPAD interface row[11] data	XEINT_27 (GPX3[3])	XEINT_27 (GPX3[3])	muxed
KP_ROW [10]	I	KEYPAD interface row[10] data	XEINT_26 (GPX3[2])	XEINT_26 (GPX3[2])	muxed
KP_ROW [9]	I	KEYPAD interface row[9] data	XEINT_25 (GPX3[1])	XEINT_25 (GPX3[1])	muxed
KP_ROW [8]	I	KEYPAD interface row[8] data	XEINT_24 (GPX3[0])	XEINT_24 (GPX3[0])	muxed
KP_ROW [7]	I	KEYPAD interface row[7] data	XEINT_23 (GPX2[7])	XEINT_23 (GPX2[7])	muxed
KP_ROW [6]	I	KEYPAD interface row[6] data	XEINT_22 (GPX2[6])	XEINT_22 (GPX2[6])	muxed

因此我们可以理解下面代码"EINT27"。

```
ret = gpio_request(EXYNOS4_GPX3(3), "EINT27");
```

将 led_gpios[ ]改为如下：

```
static int led_gpios[] = {
        EXYNOS4_GPL2(0),
        EXYNOS4_GPK1(1),
        EXYNOS4_GPD0(0),
};
```

同样上一次的中断处理函数，添加“SLEEP”中断处理函数：

```
static irqreturn_t eint27_interrupt(int irq, void *dev_id) {
        printk("%s(%d)\n", __FUNCTION__, __LINE__);
        if(gpio_get_value(led_gpios[2]))
                gpio_set_value(led_gpios[2], 0);
        else
                gpio_set_value(led_gpios[2], 1);
        return IRQ_HANDLED;
}
```

在上面的工作做完之后我们只需要在中断例程一的基础上把“beep”的驱动去掉就可以了，编译烧写到开发板，我们可以按按开发板上的“sleep”键观察现象，当我们按下“sleep”键时，可以听见“beep”响，再次按下“sleep”键时，可以发现“beep”不响了，如下图所示：

```
[root@iTOP-4412]#
[root@iTOP-4412]#
[root@iTOP-4412]#
[root@iTOP-4412]# [  102.920256] eint27_interrupt(63)
[  103.080776] eint27_interrupt(63)
```

至此，linux 下中断驱动我们就已经完成了。