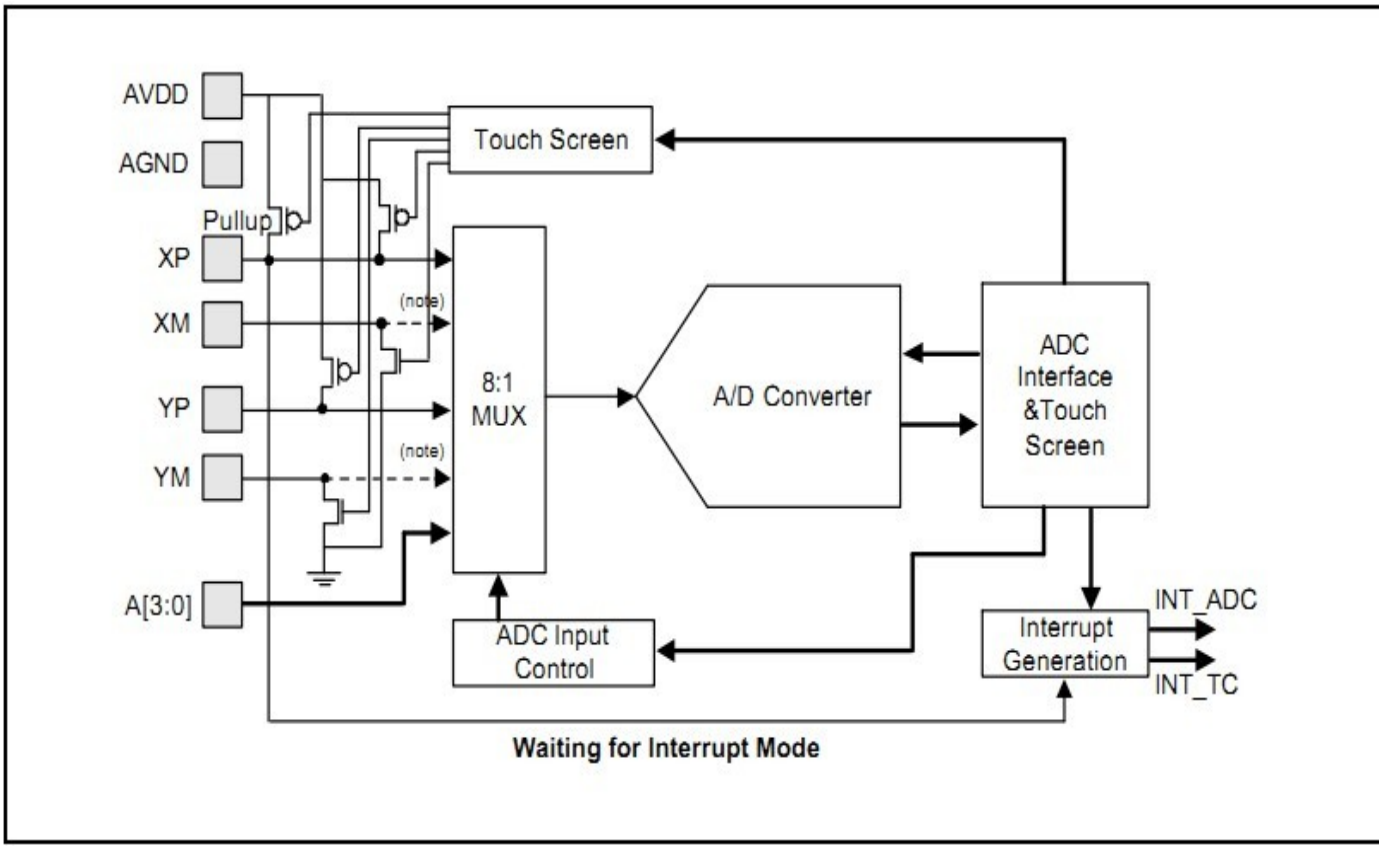
努力成为 linux kernel hacker 的人李万鹏原创作品，为梦而战。转载请标明出处
http://blog.csdn.net/woshixingaaa/archive/2011/05/20/6434725.aspx
S3C2440 支持的是 4 线电阻式触摸屏，这里简单说一下触摸屏基本原理，目前的触摸屏种类有：阻型触摸屏，容性触摸屏，多点触摸。阻性触摸屏通常由三部分组成：上下两层透明的(ITO 氧化铟)导体层，两层导体之间的间隔层和电极。触摸屏工作时，上下导体层各自构成了一个电阻网络，分别称为 X 层，Y 层，X 层在左右两电极，Y 层在上下两电极分别引出信号，一共引出 4 个信号，构成所谓的 4 线电阻。当某一层加上电压时，会在该网络上形成电压梯度。如有外力使得上下两层在某一点接触，则在电极未加电压的另一层可以测得接触点的电压。得到的电压值通过 A/D 转换，就可相应的判断接触点的坐标。说白了，阻性触摸屏可以想象成两个方向的滑动变阻，当手点上时两个接触面被接触上，接触位置不一样相当于滑动位置不一样。
再来说一下 S3C2440 使用的 ADC 控制器，是一个 10 位的 8 通道的模数转换器。在 2.5MHz 的 A/D 转换时钟下，最大的转换速率可达 500KSPS(SPS:sample per second，每秒采样的次数)。S3C2440 的 4 个控制信号的引脚与 AD 的 4 个模拟信号输入引脚复用。从下图中可以看出 ADC 和触摸屏只有一个 A/D 转换器(A/D Converter)，可以通过设置寄存器来选择对哪路模拟信号(多达 8 路)进行采样。图中有两个中断信号：INT_ADC,INT_TC，前者表示 A/D 转换器已经转换完毕，后者表示触摸屏被按下了。



在 Linux 内核的触摸屏驱动中采用了延时进行消抖和算术平均值法进行滤波，这里分析一下 s3c2410_ts 程序，首先来看模块的初始化函数：

```
1.  static int __init s3c2410ts_init(void)
2.  {
3.      struct input_dev *input_dev;
4.      int err;
5.          /*获得 ad 时钟*/
6.      adc_clock = clk_get(NULL, "adc");
7.      if (!adc_clock) {
8.          printk(KERN_ERR "failed to get adc clock source/n");
9.          return -ENOENT;
10.     }
11.     /*使能时钟*/
12.     clk_enable(adc_clock);
13.     /*获得寄存器的虚拟地址*/
14.     base_addr=ioremap(S3C2410_PA_ADC,0x20); //remap the touch panal control register.
15.     if (base_addr == NULL) {
16.         printk(KERN_ERR "Failed to remap register block/n");
17.         return -ENOMEM;
18.     }
19.     /*配置寄存器引脚*/
20.     s3c2410_ts_connect();
```

```
21.  iowrite32(S3C2410_ADCCON_PRSCEN | S3C2410_ADCCON_PRSCVL(0xFF),/
22.          base_addr+S3C2410_ADCCON);
23.
24.  iowrite32(0xffff,  base_addr+S3C2410_ADCDLY);
25.  /*进入等待中断模式*/
26.  iowrite32(WAIT4INT(0), base_addr+S3C2410_ADCTSC);
27.  /*注册输入设备*/
28.  input_dev = input_allocate_device();
29.  if (!input_dev) {
30.     printk(KERN_ERR "Unable to allocate the input device !!/n");
31.     return -ENOMEM;
32.  }
33.  dev = input_dev;
34.  /*设备支持的事件，同步事件，按键事件，绝对坐标*/
35.  dev->evbit[0] = BIT(EV_SYN) | BIT(EV_KEY) | BIT(EV_ABS ) ,
36.  /*按键事件的类型，触摸屏点击*/
37.  dev->keybit[BITS_TO_LONGS(BTN_TOUCH)] = BIT(BTN_TOUCH);
38.  /*触摸屏使用的是绝对坐标系，所以设置 x,y 的范围和压力*/
39.  input_set_abs_params(dev, ABS_X, 0, 0x3FF, 0, 0);
40.  input_set_abs_params(dev, ABS_Y, 0, 0x3FF, 0, 0);
41.  input_set_abs_params(dev, ABS_PRESSURE, 0, 1, 0, 0);
42.  /*设备的身份信息，这里写死*/
43.  dev->name = s3c2410ts_name;
44.  dev->id.bustype = BUS_RS232;
45.  dev->id.vendor = 0xDEAD;
46.  dev->id.product = 0xBEEF;
47.  dev->id.version = S3C2410TSVERSION;
48.  /*注册 AD 中断处理函数*/
49.  if (request_irq(IRQ_ADC, stylus_action, IRQF_SHARED|IRQF_SAMPLE_RANDOM, "s3c2410_action
     ", dev))
50.    {
51.     printk(KERN_ERR "s3c2410_ts.c: Could not allocate ts IRQ_ADC !/n");
52.     iounmap(base_addr);
53.     return -EIO;
54.  }
55.  /*注册 irq 中断处理函数*/
56.     if (request_irq(IRQ_TC, stylus_updown, IRQF_SAMPLE_RANDOM, "s3c2410_action", dev))
57.    {
58.     printk(KERN_ERR "s3c2410_ts.c: Could not allocate ts IRQ_TC !/n");
59.     iounmap(base_addr);
60.     return -EIO;
61.  }
62.  printk(KERN_INFO "%s successfully loaded/n", s3c2410ts_name);
63.  /*注册输入设备*/
64.  err = input_register_device(dev);
```

```
    if(err)
    {
        printk(KERN_ERR "failed to register input device/n");
    }
    return 0;
}
```

下面来看一下 IRQ_TC 的中断处理函数：

```
static irqreturn_t stylus_updown(int irq, void *dev_id)
{
    unsigned long data0;
    unsigned long data1;
    int updown;
    /*获得锁，可能有其他的设备会用到 AD 模块*/
    if (down_trylock(&ADC_LOCK) == 0)
        {
        /*标识对触摸屏进行了操作*/
        OwnADC = 1;
        /*读状态寄存器的值*/
        data0 = ioread32(base_addr+S3C2410_ADCDAT0);
        data1 = ioread32(base_addr+S3C2410_ADCDAT1);
        /*如果 updown 为 1 表示按下，为 0 表示抬起*/
            updown = (!(data0 & S3C2410_ADCDAT0_UPDOWN)) && (!(data1 & S3C2410_ADCDAT0_U
PDOWN));
        if (updown)
            {
            /*如果被按下，touch_timer_fire 进行实际的处理*/
            touch_timer_fire(0);
            }
            else
            {
            OwnADC = 0;
            up(&ADC_LOCK);
        }
    }
    return IRQ_HANDLED;
}
```

下面看一下这个实际进行中断处理的函数：

```
static void touch_timer_fire(unsigned long data)
{
    unsigned long data0;
    unsigned long data1;
    int updown;
    /*读状态，看是被按下，还是被弹起*/
```

```
    data0 = ioread32(base_addr+S3C2410_ADCDAT0);
    data1 = ioread32(base_addr+S3C2410_ADCDAT1);
    updown = (!(data0 & S3C2410_ADCDAT0_UPDOWN)) && (!(data1 & S3C2410_ADCDAT0_UPDOWN));
    if(updown)
    {
        if(count != 0)
            {
            long tmp;
            tmp = xp;
            xp = yp;
            yp = tmp;
            /*这是一种算术平均滤波法，*/
                xp >>= 2;
              yp >>= 2;
            input_report_abs(dev, ABS_X, xp);
            input_report_abs(dev, ABS_Y, yp);
            input_report_key(dev, BTN_TOUCH, 1);
            input_report_abs(dev, ABS_PRESSURE, 1);
            input_sync(dev);
        }
        /*如果是被按下，并且没有进行过 AD 转换，则开始 AD 转化*/
            xp = 0;
        yp = 0;
        count = 0;
        iowrite32(S3C2410_ADCTSC_PULL_UP_DISABLE | AUTOPST, base_addr+S3C2410_ADCTSC);
        iowrite32(ioread32(base_addr+S3C2410_ADCCON) | S3C2410_ADCCON_ENABLE_START, base_addr+S3C2410_ADCCON);
    }
    else
    {
        count = 0;
        input_report_key(dev, BTN_TOUCH, 0);
        input_report_abs(dev, ABS_PRESSURE, 0);
        input_sync(dev);
        iowrite32(WAIT4INT(0), base_addr+S3C2410_ADCTSC);
        if (OwnADC)
        {
            OwnADC = 0;
            up(&ADC_LOCK);
        }
    }
}
```

如果 AD 转换完成，会调用 AD 完成的中断处理程序：

```
1.  static irqreturn_t stylus_action(int irq, void *dev_id)
2.  {
3.     unsigned long data0;
4.     unsigned long data1;
5.     /*如果确实对触摸屏进行了操作*/
6.     if (OwnADC)
7.        {
8.        /*获得转换后的数据，并增加采样次数*/
9.        data0 = ioread32(base_addr+S3C2410_ADCDAT0);
10.       data1 = ioread32(base_addr+S3C2410_ADCDAT1);
11.       count++;
12.       /*如果采样次数少于 4 次，则继续进行 AD 采样*/
13.       if (count < (1<<2))
14.          {
15.          iowrite32(S3C2410_ADCTSC_PULL_UP_DISABLE | AUTOPST, base_addr+S3C2410_ADCTSC);

16.          iowrite32(ioread32(base_addr+S3C2410_ADCCON) | S3C2410_ADCCON_ENABLE_START, ba
     se_addr+S3C2410_ADCCON);
17.       }
18.          else
19.          {
20.          /*否则启动定时器，在 1 个滴答之后上报事件，并进入等待中断状态*/
21.          mod_timer(&touch_timer, jiffies+1);
22.          iowrite32(WAIT4INT(1), base_addr+S3C2410_ADCTSC);
23.       }
24.    }
25.    return IRQ_HANDLED;
26.}
```