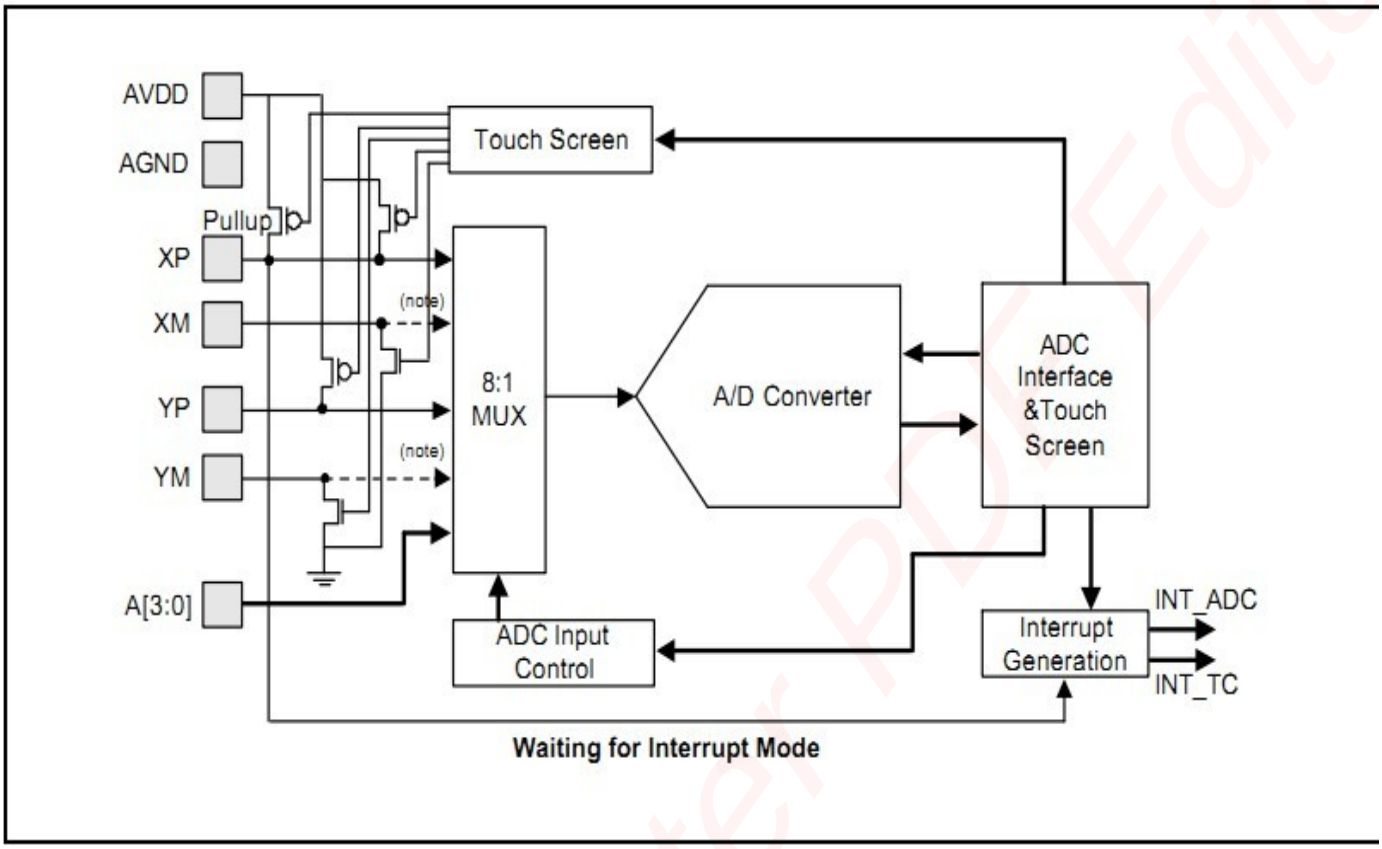
S3C2440 支持的是 4 线电阻式触摸屏，这里简单说一下触摸屏基本原理，目前的触摸屏种类有：阻型触摸屏，容性触摸屏，多点触摸。阻性触摸屏通常由三部分组成：上下两层透明的(ITO 氧化铟)导体层，两层导体之间的间隔层和电极。触摸屏工作时，上下导体层各自构成了一个电阻网络，分别称为 X 层，Y 层，X 层在左右两电极，Y 层在上下两电极分别引出信号，一共引出 4 个信号，构成所谓的 4 线电阻。当某一层加上电压时，会在该网络上形成电压梯度。如有外力使得上下两层在某一点接触，则在电极未加电压的另一层可以测得接触点的电压。得到的电压值通过 A/D 转换，就可相应的判断接触点的坐标。说白了，阻性触摸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);
```

```
65.   if(err)
66.   {
67.      printk(KERN_ERR "failed to register input device/n");
68.   }
69.   return 0;
70.}
```

下面来看一下 IRQ_TC 的中断处理函数：

```
1.  static irqreturn_t stylus_updown(int irq, void *dev_id)
2.  {
3.     unsigned long data0;
4.     unsigned long data1;
5.     int updown;
6.     /*获得锁，可能有其他的设备会用到 AD 模块*/
7.     if (down_trylock(&ADC_LOCK) == 0)
8.        {
9.        /*标识对触摸屏进行了操作*/
10.       OwnADC = 1;
11.       /*读状态寄存器的值*/
12.       data0 = ioread32(base_addr+S3C2410_ADCDAT0);
13.       data1 = ioread32(base_addr+S3C2410_ADCDAT1);
14.       /*如果 updown 为 1 表示按下，为 0 表示抬起*/
15.          updown = (!(data0 & S3C2410_ADCDAT0_UPDOWN)) && (!(data1 & S3C2410_ADCDAT0_U
    PDOWN));
16.       if (updown)
17.          {
18.          /*如果被按下，touch_timer_fire 进行实际的处理*/
19.          touch_timer_fire(0);
20.         }
21.          else
22.          {
23.          OwnADC = 0;
24.          up(&ADC_LOCK);
25.       }
26.    }
27.    return IRQ_HANDLED;
28.}
```

下面看一下这个实际进行中断处理的函数：

```
1.  static void touch_timer_fire(unsigned long data)
2.  {
3.     unsigned long data0;
4.     unsigned long data1;
5.     int updown;
6.     /*读状态，看是被按下，还是被弹起*/
```

```
7.    data0 = ioread32(base_addr+S3C2410_ADCDAT0);
8.    data1 = ioread32(base_addr+S3C2410_ADCDAT1);
9.    updown = (!(data0 & S3C2410_ADCDAT0_UPDOWN)) && (!(data1 & S3C2410_ADCDAT0_UPDOWN));
10.   if(updown)
11.   {
12.       if(count != 0)
13.         {
14.         long tmp;
15.         tmp = xp;
16.         xp = yp;
17.         yp = tmp;
18.         /*这是一种算术平均滤波法，*/
19.             xp >>= 2;
20.           yp >>= 2;
21.         input_report_abs(dev, ABS_X, xp);
22.         input_report_abs(dev, ABS_Y, yp);
23.         input_report_key(dev, BTN_TOUCH, 1);
24.         input_report_abs(dev, ABS_PRESSURE, 1);
25.         input_sync(dev);
26.       }
27.       /*如果是被按下，并且没有进行过 AD 转换，则开始 AD 转化*/
28.         xp = 0;
29.       yp = 0;
30.       count = 0;
31.       iowrite32(S3C2410_ADCTSC_PULL_UP_DISABLE | AUTOPST, base_addr+S3C2410_ADCTSC);
32.       iowrite32(ioread32(base_addr+S3C2410_ADCCON) | S3C2410_ADCCON_ENABLE_START, base_addr+S3C2410_ADCCON);
33.   }
34.   else
35.   {
36.       count = 0;
37.       input_report_key(dev, BTN_TOUCH, 0);
38.       input_report_abs(dev, ABS_PRESSURE, 0);
39.       input_sync(dev);
40.       iowrite32(WAIT4INT(0), base_addr+S3C2410_ADCTSC);
41.       if (OwnADC)
42.       {
43.         OwnADC = 0;
44.         up(&ADC_LOCK);
45.       }
46.   }
47.}
```

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.}
```

嵌入式资源免费下载

总线协议：

1. 基于 PCIe 驱动程序的数据传输卡 DMA 传输
2. 基于 PCIe 总线协议的设备驱动开发
3. CANopen 协议介绍
4. 基于 PXI 总线 RS422 数据通信卡 WDM 驱动程序设计
5. FPGA 实现 PCIe 总线 DMA 设计
6. PCI Express 协议实现与验证
7. VPX 总线技术及其实现
8. 基于 Xilinx FPGA 的 PCIE 接口实现
9. 基于 PCI 总线的 GPS 授时卡设计
10. 基于 CPCI 标准的 6U 信号处理平台的设计
11. USB30 电路保护
12. USB30 协议分析与框架设计
13. USB 30 中的 CRC 校验原理及实现
14. 基于 CPLD 的 UART 设计
15. IPMI 在 VPX 系统中的应用与设计
16. 基于 CPCI 总线的 PMC 载板设计
17. 基于 VPX 总线的工件台运动控制系统研究与开发

VxWorks：

1. 基于 VxWorks 的多任务程序设计
2. 基于 VxWorks 的数据采集存储装置设计
3. Flash 文件系统分析及其在 VxWorks 中的实现
4. VxWorks 多任务编程中的异常研究
5. VxWorks 应用技巧两例
6. 一种基于 VxWorks 的飞行仿真实时管理系统
7. 在 VxWorks 系统中使用 TrueType 字库
8. 基于 FreeType 的 VxWorks 中文显示方案
9. 基于 Tilcon 的 VxWorks 简单动画开发
10. 基于 Tilcon 的某武器显控系统界面设计
11. 基于 Tilcon 的综合导航信息处理装置界面设计

12. VxWorks 的内存配置和管理
13. 基于 VxWorks 系统的 PCI 配置与应用
14. 基于 MPC8270 的 VxWorks BSP 的移植
15. Bootrom 功能改进经验谈
16. 基于 VxWorks 嵌入式系统的中文平台研究与实现

Linux：

1. Linux 程序设计第三版及源代码
2. NAND FLASH 文件系统的设计与实现
3. 多通道串行通信设备的 Linux 驱动程序实现
4. Zsh 开发指南-数组
5. 常用 GDB 命令中文速览
6. 嵌入式 C 进阶之道
7. Linux 串口编程实例
8. 基于 Yocto Project 的嵌入式应用设计
9. Android 应用的反编译
10. 基于 Android 行为的加密应用系统研究
11. 嵌入式 Linux 系统移植步步通
12. 嵌入式 CC++语言精华文章集锦
13. 基于 Linux 的高性能服务器端的设计与研究
14. S3C6410 移植 Android 内核
15. Android 开发指南中文版
16. 图解 Linux 操作系统架构设计与实现原理（第二版）

Windows CE：

1. Windows CE.NET 下 YAFFS 文件系统 NAND Flash 驱动程序设计
2. Windows CE 的 CAN 总线驱动程序设计
3. 基于 Windows CE.NET 的 ADC 驱动程序实现与应用的研究
4. 基于 Windows CE.NET 平台的串行通信实现
5. 基于 Windows CE.NET 下的 GPRS 模块的研究与开发
6. win2k 下 NTFS 分区用 ntldr 加载进 dos 源代码
7. Windows 下的 USB 设备驱动程序开发
8. WinCE 的大容量程控数据传输解决方案设计
9. WinCE6.0 安装开发详解
10. DOS 下仿 Windows 的自带计算器程序 C 源码

11. G726 局域网语音通话程序和源代码
12. WinCE 主板加载第三方驱动程序的方法
13. WinCE 下的注册表编辑程序和源代码
14. WinCE 串口通信源代码
15. WINCE 的 SD 卡程序[可实现读写的源码]
16. 基于 WinCE 的 BootLoader 研究

PowerPC：

1. Freescale MPC8536 开发板原理图
2. 基于 MPC8548E 的固件设计
3. 基于 MPC8548E 的嵌入式数据处理系统设计
4. 基于 PowerPC 嵌入式网络通信平台的实现
5. PowerPC 在车辆显控系统中的应用
6. 基于 PowerPC 的单板计算机的设计
7. 用 PowerPC860 实现 FPGA 配置

ARM：

1. 基于 DiskOnChip 2000 的驱动程序设计及应用
2. 基于 ARM 体系的 PC-104 总线设计
3. 基于 ARM 的嵌入式系统中断处理机制研究
4. 设计 ARM 的中断处理
5. 基于 ARM 的数据采集系统并行总线的驱动设计
6. S3C2410 下的 TFT LCD 驱动源码
7. STM32 SD 卡移植 FATFS 文件系统源码
8. STM32 ADC 多通道源码
9. ARM Linux 在 EP7312 上的移植
10. ARM 经典 300 问
11. 基于 S5PV210 的频谱监测设备嵌入式系统设计与实现
12. Uboot 中 start.S 源码的指令级的详尽解析

Hardware：

1. DSP 电源的典型设计
2. 高频脉冲电源设计
3. 电源的综合保护设计
4. 任意波形电源的设计
5. 高速 PCB 信号完整性分析及应用
6. DM642 高速图像采集系统的电磁干扰设计
7. 使用 COMExpress Nano 工控板实现 IP 调度设备