一、触摸屏设备工作原理

1、电阻式触摸屏工作原理原理

触摸屏按其技术原理可分为五类：矢量压力传感式、电阻式、电容式、红外线式、表面声波式，其中电阻式触摸屏在嵌入式系统中用的较多。电阻触摸屏是一块 4 层的透明的复合薄膜屏，最下面是玻璃或有机玻璃构成的基层，最上面是一层外表面经过硬化处理从而光滑防刮的塑料层，中间是两层金属导电层，分别在基层之上和塑料层内表面，在两导电层之间有许多细小的透明隔离点把它们隔开。当手指触摸屏幕时，两导电层在触摸点处接触。

电阻式触摸屏中最常用和普及的触摸屏是四项式触摸屏，其结构由 X 层和 Y 层组成，中间由微小的绝缘电阻隔开。当触摸屏没有压力时，X 层和 Y 层处于断开状态。当有压力时，触摸屏 X 层和 Y 层导通。通过 X 层的探针可以侦测 Y 层接触点的电压，通过电压可以确定触摸点在 Y 层的位置。同样，通过 Y 层的探针可以侦测出 X 层接触点的电压，通过电压可以确定触摸点在 X 层的位置。这样，就可以得到触摸点在触摸屏上的位置（x，y）。X 层和 Y 层的关系如下图所示：

二、S3C2440 触摸屏接口

1、S3C2440 触摸屏接口的工作模式

s3c2440 芯片具有一个 8 通道模拟输入的 10 位 CMOS ADC（模/数转换器），其转换模拟输入信号为 10 位二进制数字编码，最大转换率为 2.5MHz A/D 转换器时钟下的 500KSPS。A/D 转换器支持片上采样-保持功能和掉电模式的操作。

触摸屏接口可以控制/选择触摸屏 X、Y 方向的引脚（XP，XM，YP，YM）的变换。触摸屏接口包括触摸屏引脚控制逻辑和带中断发生逻辑的 ADC 接口逻辑。

s3c2440 触摸屏接口有 4 中工作模式、在不同的工作模式下，触摸屏设备完成不通的功能。在某些情况下，几种工作模式需要相互配合，才能够完成一定的功能。

1. 普通转换模式

普通转换模式（AUTO_PST = 0，XY_PST = 0）用来进行一般的 ADC 转换，例如通过 ADC 测量电池电压等。此模式可以通过设置 ADCCON（ADC 控制寄存器）初始化并且通过读写 ADCDAT0（ADC 数据寄存器 0）就能够完成。

2. 独立的 X/Y 方向转换模式

独立 X/Y 轴坐标转换模式其实包含了 X 轴模式和 Y 轴模式。为获得 X、Y 坐标，需首先进行 X 轴的坐标转换（AUTO_PST = 0，XY_PST = 1），X 轴的转换资料会写到 ADCDAT0 寄存器的 XPDAT 中，等待转换完成后，触摸屏控制器会产生 INT_ADC 中断。然后，进行 Y 轴的坐标转换（AUTO_PST = 0，XY_PST = 2），Y 轴的转换资料会写到 ADCDAT1 寄存器的 YPDAT 中，等待转换完成后，触摸屏控制器也会产生 INT_ADC 中断。

3. 自动（顺序）X/Y 方向转换模式

自动（连续）X/Y 位置转换模式（AUTO_PST = 1，XY_PST = 0）运行方式是触摸屏控制自动转换 X 位置和 Y 位置。触摸屏控制器在 ADCDAT0 的 XPDATA 位写入 X 测定数据，在 ADCDAT1 的 YPADATA 位写入 Y 测定数据。自动（连续）位置转换后，触摸屏控制器产生 INT_ADC 中断。

4. 等待中断模式

当设置触摸屏接口控制器的 ADCTSC 寄存器为 0xD3 时，触摸屏就处于等待终端模式。这时触摸屏等待触摸信号的到来。当触摸信号到来时，触摸屏接口控制器通过 INT_TC 线产生中断信号，表示有触摸动作发生。当中断发生，触摸屏可以转换为其它两种状态来读取触摸点的位置（x，y）。这两种模式是独立的 X/Y 位置转换模式和自动 X/Y 位置转换模式。

触摸屏控制器产生中断信号（INT_TC）后，必须清除等待中断模式。（XY_PST 设置到无操作模式）。

编程笔记

1. A/D 转换的数据可以通过中断或查询方式访问。中断方式的总体转换时间为从 A/D 转换器开始到转换数据的读取，可能由于中断服务程序的返回时间和数据访问时间而延迟。查询方式是通过检查转换结束标志位的 ADCCON[15]，可以确定读取 ADCDAT 寄存器的时间。

2. 还提供了其它启动 A/D 转换的方法。在转换的读启动模式 ADCCON[1]设置为 1 后，A/D 转换启动同时读取数据。

5．待机模式（Standby Mode）

当 ADCCON 寄存器的 STDBM 位置 1 时，待机模式被激活。在这种模式下，A/D 转换动作被禁止，ADCDAT0 的 XPDATA 位和 ADXDATA1 的 YPDAT 保留以前被转换的数据。

2、s3c2440 数模屏设备寄存器

(1)、 ADCCON(ADC CONTROL REGISTER)，ADC 控制寄存器，用于控制 AD 转换、是否使用分频、设置分频系数、读取 AD 转换器的状态等。

(2)、ADCTSC (ADC TOUCH SCREEN CONTROL REGISTER)，ADC 触摸屏控制寄存器。用于存储触摸屏的 YMON、nYPON、XMON、nXPON 等的状态。

(3)、ADCDLY (ADC START DELAY REGISTER)，ADC 延时寄存器，用于正常模式下和等待终端模式下的延时操作。

(4)、ADCDAT0 (ADC CONVERSION DATA REGISTER)，ADC 转换寄存器 0，用于存储触摸屏的点击装态、工作模式、X 坐标等。

(5)、ADCDAT1 (ADC CONVERSION DATA REGISTER)，ADC 数据寄存器 1，用于存储触摸屏的点击装态、工作模式、Y 坐标等。

使用 ADCDAT0 和 ADCDAT1 寄存器是，需要注意一下问题：

ADCDAT0 和 ADCDAT1 寄存器的第 15 位，表示 X 和 Y 方向上检测到的触摸屏是否被按下。只有当 ADCDAT0 和 ADCDAT1 寄存器的第 15 位，即两个寄存器的 UPDOWN 都等于 0 时，才表示触摸屏被按下，或者有触笔点击触摸屏。

触摸屏驱动的工作流程：

1. 用户按下触摸屏进入中断处理程序 stylus_updown，这个函数处理如下
（1）判断是否按下，如果是调用 touch_timer_fire，如果不是那么说明现在松开了
（2）touch_timer_fire 处理如下：判断是否按下
<1>确实按下了，那先判断 count 是否为 4，如果为 4 那么说明 ad 转换完成了，进行 x/y 坐标的事件报告，并报告触摸屏事件。这个才是一次正确的按下

<2>确实按下了，判断的 count 不是 4，说明还没有完成 4 次 AD 转换，启动 ad 转换，转换完成进入中断处理程序 stylus_action
<3>触摸屏这时松开了，那么只报告触摸屏事件，并设置触摸屏为等待按下中断的模式
2. stylus_action 函数处理过程，首先读取转换数据，然后判断 count 是否小于 4
（1）count 小于 4 那么再次启动转换
（2）count 不小于 4，然后重新设置定时器，延时时间为一个时钟滴答，设置触摸屏为等待中断模式，等待松开中断
3 . 在定时器事件到达之前松开触摸屏，会进入 stylus_updown，这个函数判断触摸屏松开了，释放信号量不会报告任何事件
如果定时器时间到执行 touch_timer_fire，就会判断真正按下，而这时 count 为 4，报告触摸屏事件，报告 x/y 坐标

4. 这里的定时器是为了防止抖动而设置的，所以按下操作至少要保持一个时钟滴答加四次 ad 转换的时间。

打开 linux-2.6.32.2/drivers/input/touchscreen/Makefile，定位到文件末尾，添加该源代码的目标模块，如下红色部分：

obj-$(CONFIG_TOUCHSCREEN_W90X900) +=w90p910_ts.o
obj-$(CONFIG_TOUCHSCREEN_PCAP) += pcap_ts.o
obj-$(CONFIG_TOUCHSCREEN_MINI2440) += s3c2410_ts.o

再打开 linux-2.6.32.2/drivers/input/touchscreen/Kconfig，定位到 14 行附近，加入如下红色部分，这样就在内核配置中添加了 mini2440 的触摸屏驱动选项：

if INPUT_TOUCHSCREEN

config TOUCHSCREEN_MINI2440
tristate "Samsung S3C2440 touchscreeninput driver"
depends on MACH_MINI2440 && INPUT&& INPUT_TOUCHSCREEN && MINI2440_ADC
default y
help
Say Y here if you have the s3c2440 touchscreen.
If unsure, say N.
To compile this driver as a module, choose Mhere: the module will be called s3c2410_ts.

config TOUCHSCREEN_MY2440_DEBUG
boolean "S3C2440 touchscreens input driverdebug messages"
depends on TOUCHSCREEN_MINI2440
help
Select this if you want debug messages

config TOUCHSCREEN_ADS7846
tristate "ADS7846/TSC2046 and ADS7843 based touchscreens"
depends on SPI_MASTER
depends on HWMON = n || HWMON
help

至此，我们就已经在内核中添加完了触摸屏驱动。

2、配置编译并启动内核

在命令行执行：make menuconfig，然后依次选择如下子菜单，找到刚刚添加的触摸屏驱动选项：
Device Drivers --->

Input device support --->
[*] Touchscreens --->
按空格键选中“S3C2440 touchscreens input driver debug messages (NEW)”

3、测试

1、如果定义了程序中的测试宏，点击触摸屏将会显示对应的坐标值

2、用网上现有的测试程序进行测试

测试程序代码如下：

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <errno.h>
#include <sys/poll.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <linux/input.h>
struct sample{
char position[15];
int x;
int y;
};
struct sample sample_array[4]=
{
{"topleft",0,0},
{"topright",0,0},
{"bottonleft",0,0},
{"bottonright",0,0},
};
int X1,X2,Y1,Y2;
getsample(int fd,int position)
{
struct input_event ev[128];
int rb,sample_cnt,cntx=0,cnty=0;
rb=read(fd,ev,sizeof(struct input_event)*128);
if (rb < (int) sizeof(struct input_event)) {
perror("evtest: short read");
exit (1);
}
for (sample_cnt = 0;
sample_cnt< (int) (rb / sizeof(struct input_event));
sample_cnt++)
```

```
{
if (EV_ABS== ev[sample_cnt].type){
if( sample_cnt%20==0){
printf("%ld.%06ld ",
ev[sample_cnt].time.tv_sec,
ev[sample_cnt].time.tv_usec);
printf("type %d code %d value %d\n",
ev[sample_cnt].type,
ev[sample_cnt].code, ev[sample_cnt].value);
}
if(ABS_X==ev[sample_cnt].code){
sample_array[position].x+= ev[sample_cnt].value;
cntx++;
}
if(ABS_Y==ev[sample_cnt].code){
sample_array[position].y+= ev[sample_cnt].value;
cnty++;
}
}
}
sample_array[position].x/=cntx;
sample_array[position].y/=cnty;
}
int ts_coordinate(int value,int axes)
{
int tempX,ret;
if(ABS_X==axes)ret=240-(240*(value-X2)/(X1-X2));
if(ABS_Y==axes)ret=320-(320*(value-Y2)/(Y1-Y2));
return ret;
}

int main(int argc, char **argv)
{
struct pollfd pfd;
int n,fd,i=0;
if ((fd = open("/dev/input/event0",O_RDONLY) )< 0) {
printf("open error! \n");
exit(1);
}

for(i=0;i<4;i++){
printf("Please touch the %s for 6 second ! \n",sample_array[i].position);
sleep(6);
printf("Time is up Please release\n");
getsample(fd,i);
sleep(1);
```

```
}
for(i=0;i<4;i++){
printf("%12s x=%4d,y=%4d\n",sample_array[i].position,
sample_array[i].x,
sample_array[i].y);
}
X1=(sample_array[0].x+ sample_array[2].x )/2;
X2=(sample_array[1].x+ sample_array[3].x )/2;
Y1=(sample_array[0].y+ sample_array[1].y )/2;
Y2=(sample_array[2].y+ sample_array[3].y )/2;
printf("Coordinate complete,test it now\n");
for(i=0;i<4;i++){
printf("Please touch the %s for 6 second ! \n",sample_array[i].position);
sleep(6);
printf("Time is up Please release\n");
getsample(fd,i);
sample_array[i].x=ts_coordinate(sample_array[i].x,ABS_X);
sample_array[i].y=ts_coordinate(sample_array[i].y,ABS_Y);
sleep(1);
}
printf("the data after coordinate \n");
for(i=0;i<4;i++){
printf("%12s x=%4d,y=%4d\n",sample_array[i].position,
sample_array[i].x,
sample_array[i].y);
}
close(fd);
exit(0);
}
```

查看设备信息：

```
root@MINI2440:/# cat proc/devices
Character devices:
  1 mem
  2 pty
  3 ttyp
  4 /dev/vc/0
  4 tty
  5 /dev/tty
  5 /dev/console
  5 /dev/ptmx
  7 vcs
 10 misc
 13 input
 14 sound
```

29 fb
89 i2c
90 mtd
可以得知输入类设备的主设备号为 13，又按如下操作：
root@MINI2440:/# cat proc/bus/input/devices
I: Bus=0019 Vendor=dead Product=beef Version=0001
N: Name="mini2440_TouchScreen"
P: Phys=
S: Sysfs=/devices/virtual/input/input0
U: Uniq=
H: Handlers=mouse0 event0
B: EV=b
B: KEY=400 0 0 0 0 0 0 0 0 0 0
B: ABS=1000003
根据这些信息我们建立好设备节点:

root@MINI2440:/#mkdir /dev/input
root@MINI2440:/#mknod /dev/input/event0 c 13 64
root@MINI2440:/#echo 8 > /proc/sys/kernel/printk
root@MINI2440:/#cat /dev/input/event0(点击触摸屏将会看到一些乱码)

注：可以将这些信息加入 etc/profile 系统启动时自动创建

clx@Think:/rootfs/etc$ sudo gedit profile

#建立触摸屏设备节点
mkdir /dev/input
mknod /dev/input/event0 c 13 64
echo 8 > /proc/sys/kernel/printk

运行测试程序将看到如下信息：

root@MINI2440:/opt# ./ts_test
Please touch the topleft for 6 second !
Time is up Please release
123.800012 type 3 code 0 value 90
Please touch the topright for 6 second !
Time is up Please release
130.590014 type 3 code 1 value 903
130.725012 type 3 code 0 value 922
130.980012 type 3 code 0 value 918
Please touch the bottonleft for 6 second !