**参考linux多点触控协议修改板卡测试**

注：

程序修改文件路径为/external/pcba-test/minuitwrp/event.cpp

修改函数名称为 static int vk\_modify(struct ev \*e, struct input\_event \*ev)

1. **板卡测试产生漂移**

**修改原因：**

由于现有的MT-207S板卡测试程序的编写，未考虑到linux多点触控B协议的协议内容，导致在第一个触点点击屏幕获得的X,Y坐标，会被第二个触点点击所修改（只说明多点触控中两个点点击的情况，其他更多点的点击类似），导致有可能上报的点X,Y的值会有三种情况，如下：

假设第一个触点获得的坐标为X1，Y1；第二个触点获得的坐标为X2，Y2；则上报的可能为：

1. 上报 X1，Y1; (X2，Y2的值完全与X1，Y1相同；)
2. 上报 X1，Y2; (X2与X1的值相同，Y2与Y1的值不同；)
3. 上报 X2，Y1; (Y2与Y1的值相同，X2与X1的值不同；)

以上的三种上报情况，在第一个触点点击后，再点击第二个触点时，屏幕会出现触摸的位置不在两个触摸点的任意一个点上，而偏向了X1，Y2或X2,Y1的位置，产生漂移。

**修改目的：**

为了解决MT-207S板卡测试程序产生的漂移，与多点触摸获得坐标的二义性。

**修改依据：**

根据MT-207S机型所使用的是linux多点触控B协议。上报的协议如下：（只说明X,Y坐标）

ABS\_MT\_SLOT 0 //第一个触摸的SLOT标识

ABS\_MT\_TRACKING\_ID 0 //第一个触摸的TRACKING标识

ABS\_MT\_POSITION\_X x[0] //第一个触摸的X坐标

ABS\_MT\_POSITION\_Y y[0] //第一个触摸的Y坐标

SYN\_REPORT //若两个点同时按下时，不产生此同步信号，两个点按下时有先后顺序才会产生此同步信号。

ABS\_MT\_SLOT 1 //第二个触摸的SLOT标识

ABS\_MT\_TRACKING\_ID 1 //第二个触摸的TRACKING标识

ABS\_MT\_POSITION\_X x[1] //第二个触摸的X坐标

ABS\_MT\_POSITION\_Y y[1] //第二个触摸的Y坐标

SYN\_REPORT //对按下动作上报

已有的MT-207S板卡测试程序处理如下：（对X,Y坐标处理的程序片段）

case ABS\_MT\_POSITION\_X:

e->mt\_p.x = ev->value;

break;

case ABS\_MT\_POSITION\_Y:

e->mt\_p.y = ev->value;

break;

从程序中看，MT-207S板卡测试，只要有获得X,Y坐标的动作，就将已有的X或Y的值覆盖。导致了漂移现象的产生。

**修改位置与修改内容：**

1. 在变量定义处，新建一个标识limit\_modify,作用是标识谁能改X,Y坐标。

static int limit\_modify = 0;

1. 在ABS\_MT\_TRACKING\_ID动作的判断中，设定只有TRACKING ID的值为0时标识limit\_modify才有效。

case ABS\_MT\_TRACKING\_ID:

if (ev-value == 0)

{

limit\_modify = 1;

}

else if (ev-value > 0)

{

limit\_modify = 0;

}

1. 在case ABS\_MT\_POSITION\_X 处，判断只有limit\_modify标识有效时，才允许更新X坐标的值。

case ABS\_MT\_POSITION\_X:

if (limit\_modify == 1)

{

e->mt\_p.x = ev->value;

}

break;

1. 在case ABS\_MT\_POSITION\_Y 处，判断只有limit\_modify标识有效时，才允许更新Y坐标的值。

case ABS\_MT\_POSITION\_Y:

if (limit\_modify == 1)

{

e->mt\_p.y = ev->value;

}

break;

1. **上报动作的屏蔽**

**修改原因：**

第一个触点点击后，驻留在屏幕中不动，产生一次同步上报信号。但由于第二个触点的点击或其他动作也会产生同步上报信号，由于MT-207S板卡测试程序的设计原因，它只认同步信号，不管是谁发出的同步信号，都会对其进行处理，这就造成了第一个触点所获的的数据，被第二个触点的点击或滑动的动作所上报了。

**修改目的：**

不对第二个触点按下的上报动作感兴趣，只对第一个点的上报动作感兴趣，如果两个点同时按下，对其上报动作都不感兴趣。

**修改依据：**

根据MT-207S机型所使用的是linux多点触控B协议。

1、第一个触点与第二个触点有先后顺序的按下，点击上报的协议如下：（只说明X,Y坐标）

ABS\_MT\_SLOT 0 //第一个触摸的SLOT标识

ABS\_MT\_TRACKING\_ID 0 //第一个触摸的TRACKING标识

ABS\_MT\_POSITION\_X x[0] //第一个触摸的X坐标

ABS\_MT\_POSITION\_Y y[0] //第一个触摸的Y坐标

SYN\_REPORT //若两个点同时按下时，不产生此同步信号，两个点按下时有先后顺序才会产生此同步信号。

ABS\_MT\_SLOT 1 //第二个触摸的SLOT标识

ABS\_MT\_TRACKING\_ID 1 //第二个触摸的TRACKING标识

ABS\_MT\_POSITION\_X x[1] //第二个触摸的X坐标

ABS\_MT\_POSITION\_Y y[1] //第二个触摸的Y坐标

SYN\_REPORT //对按下动作上报

1. 第一个触点驻留在屏幕上，第二个触点横向滑动，上报的协议如下：

ABS\_MT\_SLOT 0 //第一个触摸的SLOT标识

ABS\_MT\_TRACKING\_ID 0 //第一个触摸的TRACKING标识

ABS\_MT\_POSITION\_X x[0] //第一个触摸的X坐标

ABS\_MT\_POSITION\_Y y[0] //第一个触摸的Y坐标

SYN\_REPORT //对按下动作上报

ABS\_MT\_SLOT 1 //第二个触摸的SLOT标识

ABS\_MT\_TRACKING\_ID 1 //第二个触摸的TRACKING标识

ABS\_MT\_POSITION\_X x[1] //第二个触摸的X坐标

ABS\_MT\_POSITION\_Y y[1] //第二个触摸的Y坐标

SYN\_REPORT //对按下动作上报

ABS\_MT\_POSITION\_X x[1] //第二个触摸的X坐标

SYN\_REPORT //对按下动作上报

ABS\_MT\_POSITION\_X x[1] //第二个触摸的X坐标

SYN\_REPORT //对按下动作上报

ABS\_MT\_POSITION\_X x[1] //第二个触摸的X坐标

SYN\_REPORT //对按下动作上报

ABS\_MT\_POSITION\_X x[1] //第二个触摸的X坐标

SYN\_REPORT //对按下动作上报

1. 第一个触点驻留在屏幕上，第二个触点纵向滑动，上报的协议如下：

ABS\_MT\_SLOT 0 //第一个触摸的SLOT标识

ABS\_MT\_TRACKING\_ID 0 //第一个触摸的TRACKING标识

ABS\_MT\_POSITION\_X x[0] //第一个触摸的X坐标

ABS\_MT\_POSITION\_Y y[0] //第一个触摸的Y坐标

SYN\_REPORT

ABS\_MT\_SLOT 1 //第二个触摸的SLOT标识

ABS\_MT\_TRACKING\_ID 1 //第二个触摸的TRACKING标识

ABS\_MT\_POSITION\_X x[1] //第二个触摸的X坐标

ABS\_MT\_POSITION\_Y y[1] //第二个触摸的Y坐标

SYN\_REPORT //对按下动作上报

ABS\_MT\_POSITION\_Y y[1] //第二个触摸的Y坐标

SYN\_REPORT //对按下动作上报

ABS\_MT\_POSITION\_Y y[1] //第二个触摸的Y坐标

SYN\_REPORT //对按下动作上报

ABS\_MT\_POSITION\_Y y[1] //第二个触摸的Y坐标

SYN\_REPORT //对按下动作上报

ABS\_MT\_POSITION\_Y y[1] //第二个触摸的Y坐标

SYN\_REPORT //对按下动作上报

1. 第一个触点驻留在屏幕上，第二个触点画圆或斜边滑动，上报的协议如下：

ABS\_MT\_SLOT 0 //第一个触摸的SLOT标识

ABS\_MT\_TRACKING\_ID 0 //第一个触摸的TRACKING标识

ABS\_MT\_POSITION\_X x[0] //第一个触摸的X坐标

ABS\_MT\_POSITION\_Y y[0] //第一个触摸的Y坐标

SYN\_REPORT

ABS\_MT\_SLOT 1 //第二个触摸的SLOT标识

ABS\_MT\_TRACKING\_ID 1 //第二个触摸的TRACKING标识

ABS\_MT\_POSITION\_X x[1] //第二个触摸的X坐标

ABS\_MT\_POSITION\_Y y[1] //第二个触摸的Y坐标

SYN\_REPORT //对按下动作上报

ABS\_MT\_POSITION\_X x[1] //第二个触摸的X坐标

ABS\_MT\_POSITION\_Y y[1] //第二个触摸的Y坐标

SYN\_REPORT //对按下动作上报

ABS\_MT\_POSITION\_X x[1] //第二个触摸的X坐标

ABS\_MT\_POSITION\_Y y[1] //第二个触摸的Y坐标

SYN\_REPORT //对按下动作上报

ABS\_MT\_POSITION\_X x[1] //第二个触摸的X坐标

ABS\_MT\_POSITION\_Y y[1] //第二个触摸的Y坐标

SYN\_REPORT //对按下动作上报

**修改位置与修改内容：**

1、在变量定义处，新建一个标识syn,作用是标识处理来自谁的按下上报，默认值为1，原因是无论怎么按都是，ABS\_MT\_SLOT 0的处理在前。

static int syn = 1;

2、添加case ABS\_MT\_SLOT动作的判断，设定只有SLOT的值为0时标识syn才有效。

case ABS\_MT\_SLOT:

if (ev-value == 0)

{

syn = 1;

}

else if (ev-value > 0)

{

syn = 0;

}

3、在switch事件处理完后的上报出理处，判断只有syn标识有效时，才处理上报事件。

if (syn == 0) return 0;

1. **限制单点点击有效**

**修改原因：**

第一个触点点击后，驻留在屏幕中不动，第二个触点点击后，驻留在屏幕中不动。因为松开触点后都会产生同步上报信号。此时若是先松开第一个触点，处理函数先将由于先将变量touchReleaseOnNextSynReport 设置为2，然后又在程序中将变量touchReleaseOnNextSynReport 重新设置为0，为下一次动作做准备。但紧接着若松开第二个触点，由于前面修改的不处理第二个触点的上报信号，但是在case ABS\_MT\_TRACKING\_ID的判断中，却将变量touchReleaseOnNextSynReport 设置为2，导致下一次，第一个触点按下时变量touchReleaseOnNextSynReport 的初始值为2，在上报时直接退出，不会去调用vk\_tp\_to\_screen函数，也就是按下的按钮不会有阴影出现。

**修改目的：**

使得无论松开的顺序是触点一在前，触点二在前还是同时松开，都不能影响下一次动作的进行。也就是下一次，第一个触点点击按钮要有阴影出现。

**修改依据：**

根据MT-207S机型所使用的是linux多点触控B协议。第一个触点与第二个触点有先后顺序的松开，松开上报的协议如下：（只说明X,Y坐标）

1. 触点一先松开，触点二后松开：

ABS\_MT\_SLOT 0

ABS\_MT\_TRACKING\_ID -1

SYN\_REPORT

ABS\_MT\_SLOT 1

ABS\_MT\_TRACKING\_ID -1

SYN\_REPORT

1. 触点二先松开，触点一后松开：

ABS\_MT\_SLOT 0

ABS\_MT\_TRACKING\_ID -1

SYN\_REPORT

ABS\_MT\_SLOT 1

ABS\_MT\_TRACKING\_ID -1

SYN\_REPORT

1. 触点一、二同时松开：

ABS\_MT\_SLOT 0

ABS\_MT\_TRACKING\_ID -1

ABS\_MT\_SLOT 1

ABS\_MT\_TRACKING\_ID -1

SYN\_REPORT

**修改位置与修改内容：**

1、在变量定义处，新建一个标识syn\_report,作用是标识处理来自谁的松开上报，默认值为0。

static int syn\_report = 0;

2、添加case ABS\_MT\_TRACKING\_ID动作的判断中，设定TRACKING ID的值为-1，且只有syn的值为1时,标识syn\_report才有效，也就是，只要有第一个触点松开，无论是有顺序的松开，还是同时松开，都要处理它的松开上报。

case ABS\_MT\_TRACKING\_ID:

if (ev-value < 0)

{

if (syn == 1)

{

syn\_report = 1;

}

}

3、在switch事件处理完后的上报出理处，判断只有syn\_report标识有效时，才处理松开上报事件，为了处理第二个触点先松开，修改变量的情况，在阻止处理上报中加入touchReleaseOnNextSynReport = 0，在结合按下上报的处理，修改为：

if (syn\_report == 0 && syn == 0)

{

touchReleaseOnNextSynReport = 0; //只要阻止了上报就将变量重新置为0；

return 0;

}

syn\_report = 0; //没有阻止上报，重新将变量置为0；

1. **单点上报条件的修改**

**修改原因：**

由于MT-207S机型所使用的是linux多点触控B协议。会“保存”之前所得到的X，Y的值，如果没有新的X,Y值产生则不更新。所以在判定调用vk\_tp\_to\_screen函数的时候，只需满足获得X,或Y就可以调用，而先前的程序中需同时获得X与Y的值才能调用vk\_tp\_to\_screen函数，不满足多点触控协议B的内容。

**修改目的：**

让程序只需更新X坐标，或者Y坐标,或者X,Y坐标都同时更新，即满足判断条件，进而调用函数，而改进的板卡测试程序只允许第一个触点进行上报动作，故而在判断条件中加上触点一的标识，满足条件则调用函数。

**修改依据：**

根据多点触控协议B，这里只列出第一的触点在点击后沿着纵坐标移动的上报。

ABS\_MT\_SLOT 0 //第一个触摸的SLOT标识

ABS\_MT\_TRACKING\_ID 0 //第一个触摸的TRACKING标识

ABS\_MT\_POSITION\_X x[0] //第一个触摸的X坐标

ABS\_MT\_POSITION\_Y y[0] //第一个触摸的Y坐标

SYN\_REPORT

ABS\_MT\_POSITION\_Y y[0] //第一个触摸的Y坐标

SYN\_REPORT

ABS\_MT\_POSITION\_Y y[0] //第一个触摸的Y坐标

SYN\_REPORT

ABS\_MT\_POSITION\_Y y[0] //第一个触摸的Y坐标

SYN\_REPORT

**修改位置与修改内容：**

1、在case ABS\_MT\_POSITION\_X 处，判断只有limit\_modify标识有效时，才允许标识X坐标的更新。

case ABS\_MT\_POSITION\_X:

if (limit\_modify == 1)

{

e->mt\_p.synced |= 0x02;

}

2、在case ABS\_MT\_POSITION\_Y 处，判断只有limit\_modify标识有效时，才允许标识Y坐标的更新。

case ABS\_MT\_POSITION\_Y:

if (limit\_modify == 1)

{

e->mt\_p.synced |= 0x04;

}

3、在case ABS\_MT\_TRACKING\_ID 处，判断只有 TRACKING ID的值为0时，标识第一个触点。

case ABS\_MT\_TRACKING\_ID:

if (ev->value == 0)

{

e->mt\_p.synced |= 0x01;

}

4、在上报事件判断调用vk\_tp\_to\_screen函数处，修改判断条件。

else if (e->mt\_p.synced == 0x03 || e->mt\_p.synced == 0x05 || e->mt\_p.synced == 0x07)

{

vk\_tp\_to\_screen(&e->mt\_p, &x, &y);

}

1. **点击优化**

**修改原因：**

由于第一个触点点击测试按钮后，按着屏幕拖动，按下时产生的按钮阴影不会消失，需等到松开第一个触点才会消失。

**修改目的：**

禁止按下按钮后进行滑动行为，只要按下按钮后，X或Y坐标的值有改变，就上报取消按钮阴影，调用触点松开时的动作。

**修改依据：**

在MT-207S当前的板卡测试程序中，触点松开时的动作，只需满足能够上报，且变量touchReleaseOnNextSynReport = 2 与 变量 use\_tracking\_id\_negative\_as\_touch\_release =1即可。

**修改位置与修改内容：**

1. 在上报事件判断调用vk\_tp\_to\_screen函数处。

else if (e->mt\_p.synced == 0x03 || e->mt\_p.synced == 0x05 || e->mt\_p.synced == 0x07)

{

vk\_tp\_to\_screen(&e->mt\_p, &x, &y);

touchReleaseOnNextSynReport = 2;

use\_tracking\_id\_negative\_as\_touch\_release =1;

}