[Multi-touch (MT) Protocol 小结](http://blog.csdn.net/tianruxishui/article/details/7173045)

标签： [input](http://www.csdn.net/tag/input)[report](http://www.csdn.net/tag/report)[distance](http://www.csdn.net/tag/distance)[events](http://www.csdn.net/tag/events)[forms](http://www.csdn.net/tag/forms)[palm](http://www.csdn.net/tag/palm)

2012-01-03 18:26 7782人阅读 [评论](http://blog.csdn.net/tianruxishui/article/details/7173045#comments)(2) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/tianruxishui/article/details/7173045#report)

http://static.blog.csdn.net/images/category_icon.jpg 分类：

android --- input subsystem（1） http://static.blog.csdn.net/images/arrow_triangle%20_down.jpg

版权声明：本文为博主原创文章，未经博主允许不得转载。

Android 中多点触摸协议：

                参考： <http://www.kernel.org/doc/Documentation/input/multi-touch-protocol.txt>

1， 两种多点触摸协议：

            1）A类： 处理无关联的接触： 用于直接发送原始数据；

                  B类： 处理跟踪识别类的接触： 通过事件slot发送相关联的独立接触更新。

2，  触摸协议的使用：

        A类协议：

                       A类协议在每发送完一个接触数据包后会调用 input\_mt\_sync() 声明 一次数据的结束； input\_mt\_sync() 会发出一个 SYN\_MT\_REPORT

提示接收器接收数据并准备下一次数据的接收。

       B类协议：

                     与A类协议不同的是， B类在使用input\_mt\_slot()的时候会带有一个slot的参数，在每个数据包开始时 产生一个ABS\_MT\_SLOT事件，提示接收器更新数据。

       最终A，B类协议均会调用 input\_sync();

<http://blog.csdn.net/tianruxishui/article/details/7173045>

wangcq327 20160405

A类与B类协议不同的在于： B类协议 通过B类协议 slot 协议需要是用到ABS\_MT\_TRACKING\_ID------ 可以从硬件上获取，或者从原始数据中计算。

3, B类协议：  ABS\_MT\_TRACKING\_ID  表示一次接触；  -1 代表一个不用的slot；

        使用参考例子：

                    释放事件：

                      input\_mt\_slot(data->input\_dev, i);  
                      input\_mt\_report\_slot\_state(data->input\_dev, MT\_TOOL\_FINGER, false);        ---------------释放

                    点击事件：

                      input\_mt\_slot(data->input\_dev, i);  
                      input\_mt\_report\_slot\_state(data->input\_dev, MT\_TOOL\_FINGER, true);  
                      input\_report\_abs(data->input\_dev, ABS\_MT\_TOUCH\_MAJOR, 1);  
                      input\_report\_abs(data->input\_dev, ABS\_MT\_POSITION\_X,  current\_events[i].x);  
                      input\_report\_abs(data->input\_dev, ABS\_MT\_POSITION\_Y,  current\_events[i].y)

[Linux输入子系统：多点触控协议 -- multi-touch-protocol.txt](http://blog.csdn.net/droidphone/article/details/8434768)

2012-12-25 19:30 12014人阅读 [评论](http://blog.csdn.net/droidphone/article/details/8434768#comments)(6) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/droidphone/article/details/8434768#report)

http://static.blog.csdn.net/images/category_icon.jpg 分类：

<http://blog.csdn.net/droidphone/article/details/8434768>

Linux输入子系统（3） http://static.blog.csdn.net/images/arrow_triangle%20_down.jpg

目录[(?)[+]](http://blog.csdn.net/droidphone/article/details/8434768)

Multi-touch (MT) Protocol

-------------------------

         Copyright(C) 2009-2010 Henrik Rydberg<rydberg@euromail.se>

**简介**

------------

    为了发挥新近的多点触摸和多用户设备的强大功能，为多点触摸定义一种上报详细数据的方法（比如有多个物体直接接触到设备的表面），是非常有必要的。这篇文档描述了多点触摸协议（multi-touch，MT），是的内核驱动可以对多个随意数量的触控事件上报详细的数据信息。

    基于硬件的能力，该协议被分为两种类型。对于只能处理匿名接触（type A）的设备，该协议描述了如何把所有的原始触摸数据发送给接收者。对于那些有能力跟踪并识别每个触摸点的设备（type B），该协议描述了如何把每个触摸点的单独更新通过事件slots发送给接受者。

 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

本文由DroidPhone 翻译：<http://blog.csdn.net/droidphone>  
Kernel版本：V3.7  
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**协议的使用**

--------------

    详细的触控信息被按顺序地分割为多个ABS\_MT事件数据包进行发送。只有ABS\_MT事件信息被识别为触控数据包的一部分，因为这些事件在当前的单点触控（single-touch，ST）应用中是被忽略掉的，我们可以在现有的驱动中基于ST协议之上来实现MT协议。

    对于type A设备的驱动，在每个数据包的结尾用input\_mt\_sync()对多个触控包进行分割，这将会产生一个SYN\_MT\_REPORT事件，它通知接收者接受当前的触控信息并准备接收下一个信息。

    对于type B设备的驱动，在每个数据包的开始，通过调用input\_mt\_slot()进行分割，同时带入一个参数：slot。这会产生一个ABS\_MT\_SLOT事件，它通知接收者准备更新给定slot的信息。.

    两种类型的驱动通常都通过调用input\_sync()函数来标记一个多点触摸数据传送的结束，这通知接收者对从上一个EV\_SYN/SYN\_REPORT以来的所有累加事件作出响应，并准备接收新的一组事件/数据包。

    无状态的type A协议和有状态的type B slot协议之间的主要区别是通过识别相同接触点来减低发送到用户空间的数据量。slot协议需要使用到ABS\_MT\_TRACKING\_ID，它要不由硬件来提供，或者通过原始数据进行计算【5】。

    对于type A设备，内核驱动应该根据设备表面上全部有效触控进行列举并生成事件。每个触控点数据包在这次事件流中的顺序并不重要。事件过滤和手指跟踪的工作留给用户空间来实现【3】。

    对于type B设备，内核驱动应该把每一个识别出的触控和一个slot相关联，并使用该slot来传播触摸状态的改变。通过修改关联slot的ABS\_MT\_TRACKING\_ID来达到对触摸点的创建，替换和销毁。一个非负数的跟踪id被解释为有效的触摸，-1则代表一个不再使用的slot。一个之前没有出现过的跟踪id被认为是一个新的接触点，当一个跟踪id不再出现时则认为该接触点已经被移除。因为只有变化的部分被传播，每个被启动的接触点的状态信息必须驻留在接收端。每当接收到一个MT事件，只需对当前slot的相关属性进行一次简单的更新即可。

    有些设备可以识别和/或跟踪比它能报告给驱动更多的接触点，对于这种设备的驱动应该使得硬件上报的每一个接触点关联一个type B的slot。一旦识别到一个关联了slot的接触点发生了变化，驱动应该通过改变他的ABS\_MT\_TRACKING\_ID使得该slot无效。如果硬件发出通知它跟踪到了比目前上报的还要多的接触点，驱动应该使用BTN\_TOOL\_\*TAP事件知会用户空间此刻硬件跟踪的总的接触点数目已经改变。要完成此工作，驱动应该显式地发送BTN\_TOOL\_\*TAP事件，并在调用input\_mt\_report\_pointer\_emulation()时把use\_count参数设为false。驱动应该只通告硬件所能报告的slots数量。用户空间可以通过注意到最大支持的BTN\_TOOL\_\*TAP事件大于在ABS\_MT\_SLOT轴的absinfo中报告的type B的slots总数，来检测驱动是否能报告比slots数还多的触控点。

**Protocol Example A**

------------------

    对于一个两点触控的触摸信息，type A设备的最小的事件序列看起来就像下面这样：

    ABS\_MT\_POSITION\_X x[0]

    ABS\_MT\_POSITION\_Y y[0]

    SYN\_MT\_REPORT

    ABS\_MT\_POSITION\_X x[1]

    ABS\_MT\_POSITION\_Y y[1]

    SYN\_MT\_REPORT

    SYN\_REPORT

     实际上，在移动其中一个触控点后的上报序列看起来是一样的、所有存在触控点的原始数据被发送，然后在它们之间用SYN\_REPORT进行同步。

     当第一个接触点离开后，事件序列如下：

     ABS\_MT\_POSITION\_X x[1]

     ABS\_MT\_POSITION\_Y y[1]

     SYN\_MT\_REPORT

     SYN\_REPORT

    当第二个接触点离开后，事件序列如下：

    SYN\_MT\_REPORT

    SYN\_REPORT

    假如驱动在ABS\_MT事件之外上报一个BTN\_TOUCH 或ABS\_PRESSURE事件，最后一个SYN\_MT\_REPORT可以省略掉，否则，最后的SYN\_REPORT会被input核心层扔掉，结果就是一个0触控点事件被传到用户空间中。

**Protocol Example B**

------------------

    对于一个两点触控的触摸信息，type B设备的最小的事件序列看起来就像下面这样：

     ABS\_MT\_SLOT 0

     ABS\_MT\_TRACKING\_ID 45

    ABS\_MT\_POSITION\_X x[0]

    ABS\_MT\_POSITION\_Y y[0]

    ABS\_MT\_SLOT 1

    ABS\_MT\_TRACKING\_ID 46

    ABS\_MT\_POSITION\_X x[1]

    ABS\_MT\_POSITION\_Y y[1]

    SYN\_REPORT

    id 45的触控点在x方向移动后的事件序列如下：

    ABS\_MT\_SLOT 0

    ABS\_MT\_POSITION\_X x[0]

    SYN\_REPORT

    slot 0对应的接触点离开后，对应的事件序列如下：

    ABS\_MT\_TRACKING\_ID -1

   SYN\_REPORT

    上一个被修改的slot也是0，所以ABS\_MT\_SLOT被省略掉。这一消息移除了接触点45相关联的slot 0，于是接触点45被销毁，slot 0被释放后可以被另一个接触点重用。

     最后，第二个接触点离开后的时间序列如下：

     ABS\_MT\_SLOT 1

     ABS\_MT\_TRACKING\_ID -1

     SYN\_REPORT

**事件的使用**

-----------

    一组ABS\_MT事件集合按需要的特性被定义。这些事件被分成几个组，以便允许只实现其中的一部分。最小的集合由ABS\_MT\_POSITION\_X和ABS\_MT\_POSITION\_Y组成，用于跟踪多点接触。如果设备支持，ABS\_MT\_TOUCH\_MAJOR和ABS\_MT\_WIDTH\_MAJOR可以用来提供接触面积和对应的接触工具。

     TOUCH和WIDTH参数有一个几何解释。想象一下通过一个窗户观察一个人把一个手指按压在对面的玻璃上，你会看到两个区域，内圈的区域包含了手指实际和玻璃接触的部分，而外圈则是手指的外轮廓。接触区域(a)就是ABS\_MT\_POSITION\_X/Y，而手指轮廓区域(b) 的中心就是ABS\_MT\_TOOL\_X/Y。接触区域的直径是ABS\_MT\_TOUCH\_MAJOR，而手指轮廓的直径就是ABS\_MT\_WIDTH\_MAJOR。现在想象一下此人把手指向玻璃压得更紧，接触区域会增加，通常，ABS\_MT\_TOUCH\_MAJOR / ABS\_MT\_WIDTH\_MAJOR的比值，总是小于1的，它和接触的压力相关。不过对于基于压力的设备，应该改为使用ABS\_MT\_PRESSURE来提供接触的压力值。可以检测悬浮的设备可以使用ABS\_MT\_DISTANCE来指出离表面的距离。

           Linux MT                               Win8

         \_\_\_\_\_\_\_\_\_\_                    \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

       /                         \                   |                                                    |

      /                           \                  |                                                    |

     /      \_\_\_\_              \                 |                                                    |

    /     /           \               \                |                                                  |

    \     \  a        \              \               |       a                                         |

     \      \\_\_\_\_/                \              |                                                  |

      \                                 \             |                                                  |

       \        b                      \            |           b                                     |

        \                                \           |                                                   |

         \                                \          |                                                   |

          \                                \         |                                                   |

           \                               /         |                                                   |

            \                            /          |                                                     |

              \                         /           |                                                     |

               \\_\_\_\_\_\_\_\_\_\_/            |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|

    除了MAJOR参数之外，接触和手指区域的椭圆外形还可以增加一个MINOR参数，MAJOR和MINOR参数相当于椭圆的长轴和短轴。接触区域的椭圆的方向可以用ORIENTATION参数描述，而手指区域的椭圆方向可以通过向量运算（a-b）来获得。

    对于type A设备，将来的规格可能会通过ABS\_MT\_BLOB\_ID来描述接触的外形。

    ABS\_MT\_TOOL\_TYPE可以用来指出触控工具是手指还是笔或者其它物体。最后，ABS\_MT\_TRACKING\_ID应该一直用来跟踪被识别的接触点【5】。

    在type B协议里，ABS\_MT\_TOOL\_TYPE和ABS\_MT\_TRACKING\_ID被隐藏在input核心层中进行处理，驱动程序应该改为调用input\_mt\_report\_slot\_state()。

**事件的语义说明**

---------------

**ABS\_MT\_TOUCH\_MAJOR**

    接触区域的长轴的长度。该长度应该按接触表面的单位提供。如果表面的分辨率是X-Y，则ABS\_MT\_TOUCH\_MAJOR可能的最大值是sqrt(X^2 + Y^2)，它的对角线【4】。

**ABS\_MT\_TOUCH\_MINOR**

    接触区域短轴的表面单位长度。如果区域是圆形，该事件可以忽略【4】。

**ABS\_MT\_WIDTH\_MAJOR**

    工具轮廓区域长轴的表面单位长度。这应该就是工具本省的大小。接触面和轮廓面的方向被假设是一样的【4】。

**ABS\_MT\_WIDTH\_MINOR**

    工具轮廓区域短轴的表面单位长度，圆形的话可以被忽略【4】。

    上述4个值可以被用来推导出接触面的额外信息。ABS\_MT\_TOUCH\_MAJOR / ABS\_MT\_WIDTH\_MAJOR可以被用作接近的压力。手指和手掌有不用的特征宽度。

**ABS\_MT\_PRESSURE**

    接触区域的压力值，可以是任意单位。基于压力的设备应该使用该事件而不是TOUCH和 WIDTH事件。也用于可以报告空间信号强度的设备。

**ABS\_MT\_DISTANCE**

    接触物到接触表面之间的表面单位距离。距离为0表明已经接触到了表面。一个正值表示接触物悬浮在表面之上。

**ABS\_MT\_ORIENTATION**

    接触椭圆区域的方向。该值描述了围绕触摸中心做1/4顺时针转动的有符号数。没有限定该数值的范围，但是在椭圆和表面的Y方向对齐时应该返回0值，椭圆往左转时应该返回负值，往右转时应该返回正值。当和X轴完全对齐时，应该返回最大值。

    接触的椭圆默认为是对称点。对于可以360度转动的设备，需要报告超出最大范围来指出转动多于1/4圈。对于倒置的手指，应该上报最大范围的两倍。

     如果接触面是一个圆形，或者方向信息在内核驱动中不可用，此时方向参数可以被忽略。如果设备可以识别出两个轴方向，有可能只支持部分不连续的方向，在这种情况下，ABS\_MT\_ORIENTATION的范围应该是[0,1]【4】。

**ABS\_MT\_POSITION\_X**

    接触中心的X坐标。

**ABS\_MT\_POSITION\_Y**

    接触中心的Y坐标

**ABS\_MT\_TOOL\_X**

    接触工具轮廓中心的X坐标。如果设备不能区分接触面和工具本身时，可以忽略该事件。

**ABS\_MT\_TOOL\_Y**

    接触工具轮廓中心的Y坐标。如果设备不能区分接触面和工具本身时，可以忽略该事件。

    这4个位置值可以用于从触控工具位置中分离实际的接触位置。如果两种位置信息都存在，那么触控工具的长轴指向接触点，否则，工具和接触面的轴互相对其。

**ABS\_MT\_TOOL\_TYPE**

    接近工具的类型。很多内核驱动程序不能区分不同的触控工具，比如手指和笔。这种情况下，该事件可以被忽略。目前的协议支持MT\_TOOL\_FINGER 和MT\_TOOL\_PEN【2】两种类型。对于type B设备，该事件由input核心层处理，驱动应该改为使用input\_mt\_report\_slot\_state()来上报。

**ABS\_MT\_BLOB\_ID**

    BLOB\_ID事件把几个数据包组合在一起来组成一个任意的接触形状。一些点组成的序列定义了一个多边形的外形。这是一个type A设备的底层匿名数据组合，不应该和上层的跟踪ID相混淆。多数type A设备没有blog这一能力，所以驱动可以放心地忽略该事件。

**ABS\_MT\_TRACKING\_ID**

    TRACKING\_ID识别一个被启动的触控点的整个生命周期【5】。TRACKING\_ID的范围应该足够大，从而保证在足够长的时间内都可以维护一个唯一的值。对于type B设备，该事件由input核心层处理，驱动程序应该改为使用input\_mt\_report\_slot\_state()来上报该事件。

**事件的计算**

-----------------

    一堆不同的硬件不可避免地导致一些设备比另一些更适合于MT协议。为了简单和统一地进行对应，本节给出一些方法来确定如何计算某些事件。

    对于那些报告接触形状为矩形的设备，我们不能获得带符号的方向值，假设X和Y分别是接触面矩形的两个边长，以下这些公式可以最大可能地获取最多的信息：

    ABS\_MT\_TOUCH\_MAJOR := max(X, Y)

    ABS\_MT\_TOUCH\_MINOR := min(X, Y)

    ABS\_MT\_ORIENTATION := bool(X > Y)

    ABS\_MT\_ORIENTATION的范围应该被设为[0,1]，以便设备可以区分手指是沿着Y轴还是沿着X轴【1】。

    对于有T和C坐标的win8设备，位置信息的对应关系是：

    ABS\_MT\_POSITION\_X := T\_X

    ABS\_MT\_POSITION\_Y := T\_Y

    ABS\_MT\_TOOL\_X := C\_X

    ABS\_MT\_TOOL\_X := C\_Y

    很不幸的是，没有足够的信息可以确定触摸椭圆和工具椭圆，所以只能求助于近似算法。一种兼容早期使用方式的简单的方案是：

     ABS\_MT\_TOUCH\_MAJOR := min(X, Y)

     ABS\_MT\_TOUCH\_MINOR := <not used>

     ABS\_MT\_ORIENTATION := <not used>

     ABS\_MT\_WIDTH\_MAJOR := min(X, Y) + distance(T, C)

     ABS\_MT\_WIDTH\_MINOR := min(X, Y)

    基本原理是：我们没有关于触摸椭圆的相关信息，所以改用它的内切圆形来近似它。触控工具的椭圆应该与向量（T - C）对齐，所以它的直径必须增加（T, C）之间的距离。最后，假设触摸区域的直径等于触控工具的厚度，我们可以得到上述公式。

**手指跟踪**

---------------

    跟踪手指的处理过程，例如，为在表面上启动的每个触控点分配一个唯一的trackingID，是一个欧几里得偶匹配问题。在每一次的事件同步中，一组实际的触控点要和上一个同步的一组触控点进行匹配，完整的实现请参考【3】。

**手势**

--------

    在某些可以产生手势事件的特殊应用中，可以使用TOUCH和WIDTH参数，比如，近似手指的压力或者区分食指和拇指。利用额外的MINOR参数，可以区分整个手指压下还是只是手指的一个点压。利用ORIENTATION参数，我们可以检测手指的拧动。

**注意事项**

------------

    为了和现有的应用兼容，一个手指数据包中报告的数据不能被识别为单点触控事件。

    对于type A设备，所有的手指数据都要略过输入过滤算法，因为接下来的同一类型的事件指向的是不同的手指。

    使用type A协议设备的例子，请参考bcm5974的驱动。对于type B协议的例子，请参考hid-egalax的驱动。