Multi-touch (MT) Protocol

Introduction

In order to utilize the full power of the new multi-touch and multi-user devices, a way to report detailed data from multiple contacts, i.e., objects in direct contact with the device surface, is needed. This document describes the multi-touch (MT) protocol which allows kernel drivers to report details for an arbitrary number of contacts.

The protocol is divided into two types, depending on the capabilities of the hardware. For devices handling anonymous contacts (type A), the protocol describes how to send the raw data for all contacts to the receiver. For devices capable of tracking identifiable contacts (type B), the protocol describes how to send updates for individual contacts via event slots.

> **Note**
>
> MT protocol type A is obsolete, all kernel drivers have been converted to use type B.

Protocol Usage

Contact details are sent sequentially as separate packets of ABS_MT events. Only the ABS_MT events are recognized as part of a contact packet. Since these events are ignored by current single-touch (ST) applications, the MT protocol can be implemented on top of the ST protocol in an existing driver.

Drivers for type A devices separate contact packets by calling input_mt_sync() at the end of each packet. This generates a SYN_MT_REPORT event, which instructs the receiver to accept the data for the current contact and prepare to receive another.

Drivers for type B devices separate contact packets by calling input_mt_slot(), with a slot as argument, at the beginning of each packet. This generates an ABS_MT_SLOT event, which instructs the receiver to prepare for updates of the given slot.

All drivers mark the end of a multi-touch transfer by calling the usual input_sync() function. This instructs the receiver to act upon events accumulated since last EV_SYN/SYN_REPORT and prepare to receive a new set of events/packets.

The main difference between the stateless type A protocol and the stateful type B slot protocol lies in the usage of identifiable contacts to reduce the amount of data sent to userspace. The slot protocol requires the use of the ABS_MT_TRACKING_ID, either provided by the hardware or computed from the raw data [5].

For type A devices, the kernel driver should generate an arbitrary enumeration of the full set of anonymous contacts currently on the surface. The order in which the packets appear in the event stream is not important. Event filtering and finger tracking is left to user space [3].

For type B devices, the kernel driver should associate a slot with each identified contact, and use that slot to propagate changes for the contact. Creation, replacement and destruction of contacts is achieved by modifying the ABS_MT_TRACKING_ID of the associated slot. A non-negative tracking id is interpreted as a contact, and the value -1 denotes an unused slot. A tracking id not previously present is considered new, and a tracking id no longer present is considered removed. Since only changes are propagated, the full state of each initiated contact has to reside in the receiving end. Upon receiving an MT event, one simply updates the appropriate attribute of the current slot.

Some devices identify and/or track more contacts than they can report to the driver. A driver for such a device should associate one type B slot with each contact that is reported by the hardware. Whenever the identity of the contact associated with a slot changes, the driver should invalidate that slot by changing its ABS_MT_TRACKING_ID. If the hardware signals that it is tracking more contacts than it is currently reporting, the driver should use a BTN_TOOL_*TAP event to inform userspace of the total number of contacts being tracked by the hardware at that moment. The driver should do this by explicitly sending the corresponding BTN_TOOL_*TAP event and setting use_count to false when calling input_mt_report_pointer_emulation(). The driver should only advertise as many slots as the hardware can report. Userspace can detect that a driver can report more total contacts than slots by noting that the largest supported BTN_TOOL_*TAP event is larger than the total number of type B slots reported in the absinfo for the ABS_MT_SLOT axis.

The minimum value of the ABS_MT_SLOT axis must be 0.

Protocol Example A

Here is what a minimal event sequence for a two-contact touch would look like for a type A device:

```
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
```

The sequence after moving one of the contacts looks exactly the same; the raw data for all present contacts are sent between every synchronization with SYN_REPORT.

Here is the sequence after lifting the first contact:

```
ABS_MT_POSITION_X x[1]
ABS_MT_POSITION_Y y[1]
SYN_MT_REPORT
SYN_REPORT
```

And here is the sequence after lifting the second contact:

```
SYN_MT_REPORT
SYN_REPORT
```

If the driver reports one of BTN_TOUCH or ABS_PRESSURE in addition to the ABS_MT events, the last SYN_MT_REPORT event may be omitted. Otherwise, the last SYN_REPORT will be dropped by the input core, resulting in no zero-contact event reaching userland.

Protocol Example B

Here is what a minimal event sequence for a two-contact touch would look like for a type B device:

```
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
```

Here is the sequence after moving contact 45 in the x direction:

```
ABS_MT_SLOT 0
ABS_MT_POSITION_X x[0]
SYN_REPORT
```

Here is the sequence after lifting the contact in slot 0:

```
ABS_MT_TRACKING_ID -1
SYN_REPORT
```

The slot being modified is already 0, so the ABS_MT_SLOT is omitted. The message removes the association of slot 0 with contact 45, thereby destroying contact 45 and freeing slot 0 to be reused for another contact.

Finally, here is the sequence after lifting the second contact:

```
ABS_MT_SLOT 1
ABS_MT_TRACKING_ID -1
SYN_REPORT
```

Event Usage

A set of ABS_MT events with the desired properties is defined. The events are divided into categories, to allow for partial implementation. The minimum set consists of ABS_MT_POSITION_X and ABS_MT_POSITION_Y, which allows for multiple contacts to be tracked. If the device supports it, the ABS_MT_TOUCH_MAJOR and ABS_MT_WIDTH_MAJOR may be used to provide the size of the contact area and approaching tool, respectively.

The TOUCH and WIDTH parameters have a geometrical interpretation; imagine looking through a window at someone gently holding a finger against the glass. You will see two regions, one inner region consisting of the part of the finger actually touching the glass, and one outer region formed by the perimeter of the finger. The center of the touching region (a) is ABS_MT_POSITION_X/Y and the center of the approaching finger (b) is ABS_MT_TOOL_X/Y. The touch diameter is ABS_MT_TOUCH_MAJOR and the finger diameter is ABS_MT_WIDTH_MAJOR. Now imagine the person pressing the finger harder against the glass. The touch region will increase, and in general, the ratio ABS_MT_TOUCH_MAJOR / ABS_MT_WIDTH_MAJOR, which is always smaller than unity, is related to the contact pressure. For pressure-based devices, ABS_MT_PRESSURE may be used to provide the pressure on the contact area instead. Devices capable of contact hovering can use ABS_MT_DISTANCE to indicate the distance between the contact and the surface.

```
     Linux MT                               Win8

    __________                     _______________________
   /          \                   |                       |
  /            \                  |                       |
 /     ____     \                 |                       |
    /     /    \     \                |                       |
    \     \  a  \     \               |       a               |
     \     \____/      \              |                       |
      \                 \             |                       |
       \        b        \            |           b           |
        \                 \           |                       |
         \                 \          |                       |
          \                 \         |                       |
           \                /         |                       |
            \              /          |                       |
             \            /           |                       |
              \__________/            |_______________________|
```

In addition to the MAJOR parameters, the oval shape of the touch and finger regions can be described by adding the MINOR parameters, such that MAJOR and MINOR are the major and minor axis of an ellipse. The orientation of the touch ellipse can be described with the ORIENTATION parameter, and the direction of the finger ellipse is given by the vector (a - b).

For type A devices, further specification of the touch shape is possible via ABS_MT_BLOB_ID.

The ABS_MT_TOOL_TYPE may be used to specify whether the touching tool is a finger or a pen or something else. Finally, the ABS_MT_TRACKING_ID event may be used to track identified contacts over time [5].

In the type B protocol, ABS_MT_TOOL_TYPE and ABS_MT_TRACKING_ID are implicitly handled by input core; drivers should instead call input_mt_report_slot_state().

Event Semantics

ABS_MT_TOUCH_MAJOR

The length of the major axis of the contact. The length should be given in surface units. If the surface has an X times Y resolution, the largest possible value of ABS_MT_TOUCH_MAJOR is sqrt(X^2 + Y^2), the diagonal [4].

ABS_MT_TOUCH_MINOR

The length, in surface units, of the minor axis of the contact. If the contact is circular, this event can be omitted [4].

ABS_MT_WIDTH_MAJOR

The length, in surface units, of the major axis of the approaching tool. This should be understood as the size of the tool itself. The orientation of the contact and the approaching tool are assumed to be the same [4].

ABS_MT_WIDTH_MINOR

The length, in surface units, of the minor axis of the approaching tool. Omit if circular [4].

The above four values can be used to derive additional information about the contact. The ratio ABS_MT_TOUCH_MAJOR / ABS_MT_WIDTH_MAJOR approximates the notion of pressure. The fingers of the hand and the palm all have different characteristic widths.

ABS_MT_PRESSURE

The pressure, in arbitrary units, on the contact area. May be used instead of TOUCH and WIDTH for pressure-based devices or any device with a spatial signal intensity distribution.

If the resolution is zero, the pressure data is in arbitrary units. If the resolution is non-zero, the pressure data is in units/gram. See :ref:`input-event-codes` for details.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\input\[linux-master][Documentation][input]multi-touch-protocol.rst`, **line 263**); ***backlink***

Unknown interpreted text role "ref".

ABS_MT_DISTANCE

The distance, in surface units, between the contact and the surface. Zero distance means the contact is touching the surface. A positive number means the contact is hovering above the surface.

ABS_MT_ORIENTATION

The orientation of the touching ellipse. The value should describe a signed quarter of a revolution clockwise around the touch center. The signed value range is arbitrary, but zero should be returned for an ellipse aligned with the Y axis (north) of the surface, a negative value when the ellipse is turned to the left, and a positive value when the ellipse is turned to the right. When aligned with the X axis in the positive direction, the range max should be returned; when aligned with the X axis in the negative direction, the range -max should be returned.

Touch ellipses are symmetrical by default. For devices capable of true 360 degree orientation, the reported orientation must exceed the range max to indicate more than a quarter of a revolution. For an upside-down finger, range max * 2 should be returned.

Orientation can be omitted if the touch area is circular, or if the information is not available in the kernel driver. Partial orientation support is possible if the device can distinguish between the two axes, but not (uniquely) any values in between. In such cases, the range of ABS_MT_ORIENTATION should be [0, 1] [4].

ABS_MT_POSITION_X

The surface X coordinate of the center of the touching ellipse.

ABS_MT_POSITION_Y

The surface Y coordinate of the center of the touching ellipse.

ABS_MT_TOOL_X

The surface X coordinate of the center of the approaching tool. Omit if the device cannot distinguish between the intended touch point and the tool itself.

ABS_MT_TOOL_Y

The surface Y coordinate of the center of the approaching tool. Omit if the device cannot distinguish between the intended touch point and the tool itself.

The four position values can be used to separate the position of the touch from the position of the tool. If both positions are present, the major tool axis points towards the touch point [1]. Otherwise, the tool axes are aligned with the touch axes.

ABS_MT_TOOL_TYPE

The type of approaching tool. A lot of kernel drivers cannot distinguish between different tool types, such as a finger or a pen. In such cases, the event should be omitted. The protocol currently mainly supports MT_TOOL_FINGER, MT_TOOL_PEN, and MT_TOOL_PALM [2]. For type B devices, this event is handled by input core; drivers should instead use input_mt_report_slot_state(). A contact's ABS_MT_TOOL_TYPE may change over time while still touching the device, because the firmware may not be able to determine which tool is being used when it first appears.

ABS_MT_BLOB_ID

The BLOB_ID groups several packets together into one arbitrarily shaped contact. The sequence of points forms a polygon which defines the shape of the contact. This is a low-level anonymous grouping for type A devices, and should not be confused with the high-level trackingID [5]. Most type A devices do not have blob capability, so drivers can safely omit this event.

ABS_MT_TRACKING_ID

The TRACKING_ID identifies an initiated contact throughout its life cycle [5]. The value range of the TRACKING_ID should be large enough to ensure unique identification of a contact maintained over an extended period of time. For type B devices, this event is handled by input core; drivers should instead use input_mt_report_slot_state().

Event Computation

The flora of different hardware unavoidably leads to some devices fitting better to the MT protocol than others. To simplify and unify the mapping, this section gives recipes for how to compute certain events.

For devices reporting contacts as rectangular shapes, signed orientation cannot be obtained. Assuming X and Y are the lengths of the sides of the touching rectangle, here is a simple formula that retains the most information possible:

```
ABS_MT_TOUCH_MAJOR := max(X, Y)
ABS_MT_TOUCH_MINOR := min(X, Y)
ABS_MT_ORIENTATION := bool(X > Y)
```

The range of ABS_MT_ORIENTATION should be set to [0, 1], to indicate that the device can distinguish between a finger along the Y axis (0) and a finger along the X axis (1).

For Win8 devices with both T and C coordinates, the position mapping is:

```
ABS_MT_POSITION_X := T_X
ABS_MT_POSITION_Y := T_Y
ABS_MT_TOOL_X := C_X
ABS_MT_TOOL_Y := C_Y
```

Unfortunately, there is not enough information to specify both the touching ellipse and the tool ellipse, so one has to resort to approximations. One simple scheme, which is compatible with earlier usage, is:

```
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)
```

Rationale: We have no information about the orientation of the touching ellipse, so approximate it with an inscribed circle instead. The tool ellipse should align with the vector (T - C), so the diameter must increase with distance(T, C). Finally, assume that the touch diameter is equal to the tool thickness, and we arrive at the formulas above.

Finger Tracking

The process of finger tracking, i.e., to assign a unique trackingID to each initiated contact on the surface, is a Euclidian Bipartite Matching problem. At each event synchronization, the set of actual contacts is matched to the set of contacts from the previous synchronization. A full implementation can be found in [3].

Gestures

In the specific application of creating gesture events, the TOUCH and WIDTH parameters can be used to, e.g., approximate finger pressure or distinguish between index finger and thumb. With the addition of the MINOR parameters, one can also distinguish between a sweeping finger and a pointing finger, and with ORIENTATION, one can detect twisting of fingers.

Notes

In order to stay compatible with existing applications, the data reported in a finger packet must not be recognized as single-touch events.

For type A devices, all finger data bypasses input filtering, since subsequent events of the same type refer to different fingers.

[1] Also, the difference (TOOL_X - POSITION_X) can be used to model tilt.

[2] The list can of course be extended.

[3] (*1*,*2*) The mtdev project: http://bitmath.org/code/mtdev/.

[4] (*1*,*2*,*3*,*4*,*5*) See the section on event computation.

[5] (*1*,*2*,*3*,*4*) See the section on finger tracking.