Touch Screen Porting Notes for PianoPI Based 3.10.53

i.MX FAE Nov. 2015

freescale
Agenda

- Issue: UI touch no response
- Debug: single touch and multi-touch
- Multi-touch protocol
- Issue fix: add single touch event report
Issue: UI touch no response
The touch screen IC on PianoPI board is GOODix GT9xx. In our BSP release, it is not supported. We have get the source code supporting device tree from GOODix FAE and added it into the kernel source code directory `drivers/input/touchscreen/`. We also made modifications to the Makefile and Kconfig file. Device tree configuration see below.

```
&i2c3 {
  clock-frequency = <100000>;
  pinctrl-names = "default";
  pinctrl-0 = <&pinctrl_i2c3_3>;
  status = "okay";
  goodix_ts@5d {
    compatible = "goodix,gt9xx";
    reg = <0x5d>;
    interrupt-parent = <&gpio2>;
    interrupts = <5 2>;
    goodix,rst-gpio = <&gpio2 6 0x00>;
    goodix,irq-gpio = <&gpio2 5 0x00>;
    ...
  };
```

---

**HW configuration and SW driver**

Touch Interface

![Touch Interface Diagram](image)
Issue: UI touch no response

We have met the problem that, from debug log it seems touch screen driver module has reported the correct touch coordinates. But touch on the screen UI has no response.

```bash
root@imx6qsabresd:~# evtest /dev/input/event0
...
Testing ... (interrupt to exit)
Event: time 1440752924.596007, type 1 (Key), code 330 (Touch), value 1
Event: time 1440752924.596007, type 3 (Absolute), code 53 (Position X), value 240
Event: time 1440752924.596007, type 3 (Absolute), code 54 (Position Y), value 807
Event: time 1440752924.596007, type 3 (Absolute), code 48 (Touch Major), value 28
Event: time 1440752924.596007, type 3 (Absolute), code 50 (Width Major), value 28
Event: time 1440752924.596007, type 3 (Absolute), code 57 (Tracking ID), value 0
Event: time 1440752924.596007, -------------- Config Sync --------------
Event: time 1440752924.596007, -------------- Report Sync --------------
Event: time 1440752924.610400, type 3 (Absolute), code 53 (Position X), value 240
Event: time 1440752924.610400, type 3 (Absolute), code 54 (Position Y), value 807
Event: time 1440752924.610400, type 3 (Absolute), code 48 (Touch Major), value 28
Event: time 1440752924.610400, type 3 (Absolute), code 50 (Width Major), value 28
Event: time 1440752924.610400, type 3 (Absolute), code 57 (Tracking ID), value 0
Event: time 1440752924.610400, -------------- Config Sync --------------
Event: time 1440752924.610400, -------------- Report Sync --------------
```
Debug: single touch and multi-touch
Touch events reporting comparison

We compared the evtest log of PianoPI with mx6q sabresd and find the difference as below.

```
root@imx6qsabresd:~# evtest /dev/input/event0
Input driver version is 1.0.1
Input device ID: bus 0x18 vendor 0xdead product 0xbeef version 0x28bb
Input device name: "goodix-ts"
Supported events:
    Event type 0 (Sync)
    Event type 1 (Key)
        Event code 330 (Touch)
    Event type 3 (Absolute)
        Event code 48 (Touch Major)
            Value 0
            Min 0
            Max 255
        Event code 50 (Width Major)
            Value 0
            Min 0
            Max 255
    Event code 53 (Position X)
        Value 0
        Min 0
        Max 800
    Event code 54 (Position Y)
        Value 0
        Min 0
        Max 1280
    Event code 57 (Tracking ID)
        Value 0
        Min 0
        Max 255
Testing ... (interrupt to exit)
```

```
root@imx6qdsolo:~# evtest /dev/input/event0
Input driver version is 1.0.1
Input device ID: bus 0x18 vendor 0x0 product 0x0 version 0x0
Input device name: "EETI eGalax Touch Screen"
Supported events:
    Event type 0 (Sync)
    Event type 1 (Key)
        Event code 330 (Touch)
    Event type 3 (Absolute)
        Event code 0 (X)
            Value 1472
            Min 0
            Max 32760
        Event code 1 (Y)
            Value 1824
            Min 0
            Max 32760
        Event code 47 (?)
            Value 0
            Min 0
            Max 4
    Event code 53 (Position X)
        Value 0
        Min 0
        Max 32760
    Event code 54 (Position Y)
        Value 0
        Min 0
        Max 32760
    Event code 57 (Tracking ID)
        Value 0
        Min 0
        Max 65535
Testing ... (interrupt to exit)
```
Single touch and Multi-touch events

Below are the event definitions for touch screen.

```c
#include/uapi/linux/input.h

#define ABS_X                   0x00
#define ABS_Y                   0x01
#define ABS_Z                   0x02
#define ABS_RX                  0x03
#define ABS_RY                  0x04
#define ABS_RZ                  0x05
...
#define ABS_MT_SLOT             0x2f    /* MT slot being modified */
#define ABS_MT_TOUCH_MAJOR      0x30    /* Major axis of touching ellipse */
#define ABS_MT_TOUCH_MINOR      0x31    /* Minor axis (omit if circular) */
#define ABS_MT_WIDTH_MAJOR      0x32    /* Major axis of approaching ellipse */
#define ABS_MT_WIDTH_MINOR      0x33    /* Minor axis (omit if circular) */
#define ABS_MT_ORIENTATION      0x34    /* Ellipse orientation */
#define ABS_MT_POSITION_X       0x35    /* Center X touch position */
#define ABS_MT_POSITION_Y       0x36    /* Center Y touch position */
#define ABS_MT_TOOL_TYPE        0x37    /* Type of touching device */
#define ABS_MT_BLOB_ID          0x38    /* Group a set of packets as a blob */
#define ABS_MT_TRACKING_ID      0x39    /* Unique ID of initiated contact */
#define ABS_MT_PRESSURE         0x3a    /* Pressure on contact area */
#define ABS_MT_DISTANCE         0x3b    /* Contact hover distance */
```

The ABS_X, ABS_Y are for **single touch** while the definitions with prefix ABS_MT_ are for **multi-touch**.

We could learn from last slide that GOODix provided driver reports **multi-touch** events. The mx6q sabresd used e-galax driver reports **single touch** events together with **multi-touch** events.
Multi-touch protocol
Multi-touch Protocol

Multi-touch (MT) protocol please refer to:
Linux/Documentation/input/multi-touch-protocol.txt

The protocol is divided into two types, depending on the capabilities of the hardware.

• type A:
  For devices handling **anonymous contacts**, the protocol describes how to send the raw data for all contacts to the receiver.

• type B:
  For devices **capable of tracking identifiable contacts**, the protocol describes how to send updates for individual contacts via event slots.

Key difference: **tracking the touch point or not**
Multi-touch Protocol – ABS_MT events

Multi-touch details are sent sequentially as separate packets of ABS_MT events. Only the ABS_MT events are recognized as part of a multi-touch packet.

```c
#include/uapi/linux/input.h

#define ABS_MT_SLOT 0x2f /* MT slot being modified */
#define ABS_MT_TOUCH_MAJOR 0x30 /* Major axis of touching ellipse */
#define ABS_MT_TOUCH_MINOR 0x31 /* Minor axis (omit if circular) */
#define ABS_MT_WIDTH_MAJOR 0x32 /* Major axis of approaching ellipse */
#define ABS_MT_WIDTH_MINOR 0x33 /* Minor axis (omit if circular) */
#define ABS_MT_ORIENTATION 0x34 /* Ellipse orientation */
#define ABS_MT_POSITION_X 0x35 /* Center X ellipse position */
#define ABS_MT_POSITION_Y 0x36 /* Center Y ellipse position */
#define ABS_MT_TOOL_TYPE 0x37 /* Type of touching device */
#define ABS_MT_BLOB_ID 0x38 /* Group a set of packets as a blob */
#define ABS_MT_TRACKING_ID 0x39 /* Unique ID of initiated contact */
#define ABS_MT_PRESSURE 0x3a /* Pressure on contact area */
#define ABS_MT_DISTANCE 0x3b /* Contact hover distance */
```

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.
Multi-touch Protocol – contact packets separation

Type A devices drivers call `input_mt_sync()` at the end of each packet, which generates a `SYN_MT_REPORT` event telling the receiver current contact packet finish.

Type B devices drivers call `input_mt_slot()` at the beginning of each packet, which generates an `ABS_MT_SLOT` event telling 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.

```plaintext
<table>
<thead>
<tr>
<th>type A</th>
<th>type B</th>
</tr>
</thead>
<tbody>
<tr>
<td>{ ABS_MT_POSITION_X x[0] }</td>
<td>{ ABS_MT_SLOT 0 }</td>
</tr>
<tr>
<td>ABS_MT_POSITION_Y y[0]</td>
<td>ABS_MT_TRACKING_ID 45</td>
</tr>
<tr>
<td>SYN_MT_REPORT</td>
<td>ABS_MT_POSITION_X x[0]</td>
</tr>
<tr>
<td>ABS_MT_POSITION_X x[1]</td>
<td>ABS_MT_POSITION_Y y[0]</td>
</tr>
<tr>
<td>ABS_MT_POSITION_Y y[1]</td>
<td>ABS_MT_SLOT 1</td>
</tr>
<tr>
<td>SYN_MT_REPORT</td>
<td>ABS_MT_TRACKING_ID 46</td>
</tr>
<tr>
<td>SYN_REPORT</td>
<td>ABS_MT_POSITION_X x[1]</td>
</tr>
<tr>
<td></td>
<td>ABS_MT_POSITION_Y y[1]</td>
</tr>
</tbody>
</table>
```
Multi-touch Protocol – event reporting mechanism

The **main difference** between type A protocol and type B protocol:
whether tracking the contacts to reduce the amount of data sent to user space

For type A devices, just report all contacts currently on the surface. Reporting order is not important. Event filtering and finger tracking is left to user space.

For type B devices, each contact is associated with a slot. Driver use that slot to report when the contact attribute is changed. In this way data amount sent to user space is greatly reduced.

Creation, replacement and destruction of contacts is achieved by modifying the ABS_MT_TRACKING_ID of the associated slot.

```plaintext
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
```

**type B**
Multi-touch Protocol – type A report example

Event sequence for a two-contact touch 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
```

After lifting the first contact:

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

After lifting the second contact:

```
SYN_MT_REPORT
SYN_REPORT
```
Multi-touch Protocol – type B report example

Event sequence for a two-contact touch for a type B device:

<table>
<thead>
<tr>
<th>ABS_MT_SLOT 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABS_MT_TRACKING_ID 45</td>
</tr>
<tr>
<td>ABS_MT_POSITION_X x[0]</td>
</tr>
<tr>
<td>ABS_MT_POSITION_Y y[0]</td>
</tr>
<tr>
<td>ABS_MT_SLOT 1</td>
</tr>
<tr>
<td>ABS_MT_TRACKING_ID 46</td>
</tr>
<tr>
<td>ABS_MT_POSITION_X x[1]</td>
</tr>
<tr>
<td>ABS_MT_POSITION_Y y[1]</td>
</tr>
<tr>
<td>SYN_REPORT</td>
</tr>
</tbody>
</table>

After lifting the second contact:

<table>
<thead>
<tr>
<th>ABS_MT_SLOT 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABS_MT_TRACKING_ID -1</td>
</tr>
<tr>
<td>SYN_REPORT</td>
</tr>
</tbody>
</table>

The slot being modified just now is 0, so here need to sent ABS_MT_SLOT 1.

After moving contact 45 in the x direction:

<table>
<thead>
<tr>
<th>ABS_MT_SLOT 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABS_MT_TRACKING_ID -1</td>
</tr>
<tr>
<td>SYN_REPORT</td>
</tr>
</tbody>
</table>

x changes, only report x.

After lifting the contact in slot 0:

<table>
<thead>
<tr>
<th>ABS_MTTRACKING_ID -1</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYN_REPORT</td>
</tr>
</tbody>
</table>

The slot being modified is already 0, so the ABS_MT_SLOT is omitted.

Set ABS_MT_TRACKING_ID to -1 removes the association of slot 0 with contact 45, thereby destroying contact 45 and freeing slot 0 to be reused for another contact.
Issue fix: add single touch event report
Yocto UI responds to single touch

On mx6q sabresd board, the evtest log looks like this:

```
root@imx6qsabresd:~# evtest /dev/input/event0
...
Testing ... (interrupt to exit)
Event: time 1420438798.864839, type 3 (Absolute), code 57 (Tracking ID), value 27
Event: time 1420438798.864839, type 3 (Absolute), code 53 (Position X), value 5584
Event: time 1420438798.864839, type 3 (Absolute), code 54 (Position Y), value 15536
Event: time 1420438798.864839, type 1 (Key), code 330 (Touch), value 1
Event: time 1420438798.864839, type 3 (Absolute), code 0 (X), value 5584
Event: time 1420438798.864839, type 3 (Absolute), code 1 (Y), value 15536
Event: time 1420438798.864839, ------------ Report Sync ------------
Event: time 1420438798.944020, type 3 (Absolute), code 53 (Position X), value 5616
Event: time 1420438798.944020, type 3 (Absolute), code 54 (Position Y), value 15568
Event: time 1420438798.944020, type 3 (Absolute), code 0 (X), value 5616
Event: time 1420438798.944020, type 3 (Absolute), code 1 (Y), value 15568
Event: time 1420438798.944020, ------------ Report Sync ------------
Event: time 1420438798.970491, type 3 (Absolute), code 53 (Position X), value 6144
Event: time 1420438798.970491, type 3 (Absolute), code 54 (Position Y), value 15520
Event: time 1420438798.970491, type 3 (Absolute), code 0 (X), value 6144
Event: time 1420438798.970491, type 3 (Absolute), code 1 (Y), value 15520
Event: time 1420438798.970491, ------------ Report Sync ------------
```

As we see, it reports both single touch and multi-touch coordinates, and the Yocto built UI responds to the touch. While the Goodix GT9xx driver reports only the multi-touch coordinates, and the UI doesn’t respond to the touch operations.

So the Yocto built UI responds to the single touch coordinates report events only. We need to add single touch events into the driver.
Issue fix: add single touch event report

We added single touch events to the driver source codes as below:

```c
static s8 gtp_request_input_dev(struct goodix_ts_data *ts)
{
    input_set_abs_params(ts->input_dev, ABS_X, 0, ts->abs_x_max, 0, 0);
    input_set_abs_params(ts->input_dev, ABS_Y, 0, ts->abs_y_max, 0, 0);

    input_set_abs_params(ts->input_dev, ABS_MT_POSITION_X, 0, ts->abs_x_max, 0, 0);
    input_set_abs_params(ts->input_dev, ABS_MT_POSITION_Y, 0, ts->abs_y_max, 0, 0);
    input_set_abs_params(ts->input_dev, ABS_MT_WIDTH_MAJOR, 0, 255, 0, 0);
    input_set_abs_params(ts->input_dev, ABS_MT_TOUCH_MAJOR, 0, 255, 0, 0);
    input_set_abs_params(ts->input_dev, ABS_MT_TRACKING_ID, 0, 255, 0, 0);
}

static void gtp_touch_down(struct goodix_ts_data* ts,s32 id,s32 x,s32 y,s32 w)
{
    input_report_abs(ts->input_dev, ABS_X, x);
    input_report_abs(ts->input_dev, ABS_Y, y);
    input_sync(ts->input_dev);
}
```

We registered two events ABS_X and ABS_Y, and reported them when triggered. After the modification, touch on the panel UI get correct responses.