Linux Input Device Drivers

Bill Gatliff
bgat@billgatliff.com

Freelance Embedded Systems Developer

Overview

Roadmap:

- What is an “input device”?
- `struct input_dev`
- `input_report_*()`, `input_sync()`
- EV_ABS, ABS_X, etc.
- Examples

What is an “input device”?

Traditional devices:

- Keyboards, mice, joysticks, touch screens, ...
- LEDs, force-feedback devices

Other devices:

- A/D converters
- Gyroscopes, accelerometers, ...

What is an “input device”?

Motivation:

- Consistent protocol for input event handling
- Uniform support for “device grabbing”
- Data synchronization

Protocols:

- Absolute-axis positions
- Relative-axis positions
- Key codes, switches, LEDs, ...
- Auto-repeating

What is an “input device”?

Data “synchronization”:

- Key-down/repeat/key-up event streams
- Pen-down/movement/pen-up event streams
- X+Y+Z axis change event streams

More than keyboards and mice!

The General Idea

```
do_input_event()
{
  ...
  get_data(&x, &y, &z);
  input_report_abs(idev, ABS_X, x);
  input_report_abs(idev, ABS_Y, y);
  input_report_abs(idev, ABS_Z, z);
  input_sync(idev);
  ...
}
```

The General Idea

The fine print:

- Receiver must understand the event!

How?

- Convention, or definition
- Using `ioctl(EVIOCGBIT)`

The General Idea

Largely a “push” interface:

- Originally defined for event-driven devices
- Android HAL (ab)uses this interface somewhat
- `/sys/.../<device>/delay_ms`

`struct input_dev`

Captures driver-side details:

- Which `struct device` owns the input device
- Device protocol and capabilities
- Device `open()`, etc. methods

```
struct input_dev *idev
    = input_alloc_device();
```

struct input_dev

```
struct input_dev {
  const char *name;
  const char *phys;
  ...
  unsigned long evbit[BITS_TO_LONGS(EV_CNT)];
  unsigned long absbit[BITS_TO_LONGS(ABS_CNT)];
  ...
```

struct input_dev

```
  ...
  int  (*open  )(struct input_dev *dev);
  void (*close )(struct input_dev *dev);
  int  (*event )(struct input_dev *dev,
                 unsigned int type,
                 unsigned int code, int value);
  ...
};
```

struct input_dev

`.evbit[]`

- Bitmap describing device capabilities to users

`.absbit[]`

- Bitmap describing EV_ABS capabilities to users

`struct input_dev`

`.open()`

- Informs driver when input device is opened by users
- Not a reliable indicator of data demand

Often invoked at system startup:

- ... and not ever closed thereafter
- (This has implications for runtime-pm)

struct input_dev

`.event()`

- Invoked when users push data to the device
- Not generally useful for sensor-oriented devices

struct input_dev

```
int foo_probe(struct device *this)
{
  struct input_dev *idev
    = input_allocate_device();

  idev->name = "foo";
  idev->id.bustype = BUS_I2C;
  idev->dev.parent = this;
  idev->open = foo_input_open;
  idev->close = foo_input_close;
  idev->event = foo_input_event;
  ...
```

struct input_dev

```
  ...
  idev->evbit = BIT_MASK(EV_ABS);
  input_set_abs_params(idev, ABS_X, X_MIN, X_MAX, 0, 0);
  input_set_abs_params(idev, ABS_Y, Y_MIN, Y_MAX, 0, 0);
  input_set_abs_params(idev, ABS_Z, Z_MIN, Z_MAX, 0, 0);

  input_set_drvdata(idev, foo_data);
  input_register_device(idev);
  ...
}
```

Input Events

```
...
input_report_abs(idev, ABS_X, x);
input_report_abs(idev, ABS_Y, y);
input_report_abs(idev, ABS_Z, z);
input_report_abs(idev, ABS_MISC, t);
input_sync(idev);
...
```

Multitouch Input Events

```
...
input_report_abs(idev, ABS_MT_TRACKING_ID, id);
input_report_abs(idev, ABS_MT_POSITION_X, x);
input_report_abs(idev, ABS_MT_POSITION_Y, y);
input_report_abs(idev, ABS_MT_TOUCH_MAJOR, pressure);
input_mt_sync(idev);
...
```

Linux Input Device Drivers

Bill Gatliff
bgat@billgatliff.com

Freelance Embedded Systems Developer