Input with Android on i.MX

Last Update: May 26th,2009

What's in this doc

An overview about the user input (on-board keypad, USB keyboard, Touch Panel, USB mouse, etc) implementation in i.MX Android. Mainly focus on how Android framework interact with Linux kernel to grab user input and convert it to events which can be recognized by framework and application.

Some aspects related to input are not covered in this version yet:

- How does KeyCharMap file (.kcm) work (only KeyLayout file .kl is described)
- Special handling in framework code (e.g. wakeup LCD due to special key)
- BT based HID device

"Input" in Android

Android use standard Linux input event device (/dev/input/eventX) as defined in linux/input.h. There is no user space HAL for "input". When any user input occur (e.g user press or release key, touch the screen, move the mouse, or press the left button of mouse), kernel driver will report the raw input event (with Type/Code/Value) to Android via event device /dev/input/eventX. Android framework will poll on all available event devices (e.g. event0 for keypad, event1 for touchscreen, etc), read the raw event, convert the raw event in "logic" way (e.g. map the "scancode" from kernel to "keycode" used in framework/apps), do some "housekeeping" work (e.g. wakeup LCD when pressing some special key during sleep, start timer for detect long pressing), put the event in the queue, and then throw it to view/widget one by one.

Currently Android framework handle three types of input events from kernel:

1. EV_KEY (Key or Button press/release)
2. EV_REL (Relative Axes)
3. EV_ABS (Absolute Axes)

When pressing/releasing one key on keypad/keyboard, the following events (in format TYPE/CODE/VALUE) will be generated by keypad driver (or USB HID driver) and feed into Android framework:

(EV_KEY, 0x008b, 0x0001) - 0x008b is the scan code of "MENU" defined in input.h; 0x0001 means "press"
(EV_KEY, 0x008b, 0x0000) - 0x0000 means "release"

When touch and scratch on the screen, the following event sequence will be generated by touchscreen driver and feed into Android framework:

(EV_ABS, 0x0000, 0x0189) - 0x0000 means "ABS_X" (X axes), 0x0189 the absolute X value of pen point
(EV_ABS, 0x0001, 0x001c) - 0x0001 means "ABS_Y" (Y axes), 0x001c the absolute Y value of pen point
(EV_KEY, 0x014a, 0x0001) - 0x014a means "BTN_TOUCH" , 0x0001 means "pressing"
(0x0000, 0x0000, 0x0000) - sync event, no meaning, will be ignored
(EV_ABS, 0x0000, 0x0200) - pen is moving to another point with different X
(EV_ABS, 0x0001, 0x0025) - pen is moving to another point with different Y
...
(EV_ABS, 0x0000, 0x0287)
(EV_ABS, 0x0000, 0x0053)
(EV_KEY, 0x014a, 0x0000) - Here pen leave off the screen. 0x0000 means "up"
(0x0000, 0x0000, 0x0000)

When move mouse, scroll the wheel and press the left button, the following event sequence will be generated by USB HID driver and feed into Android framework:

(EV_REL, 0x0000, 0x0001) - 0x0000 means "REL_X" (move toward X axes), 0x0001 is the relative X movement

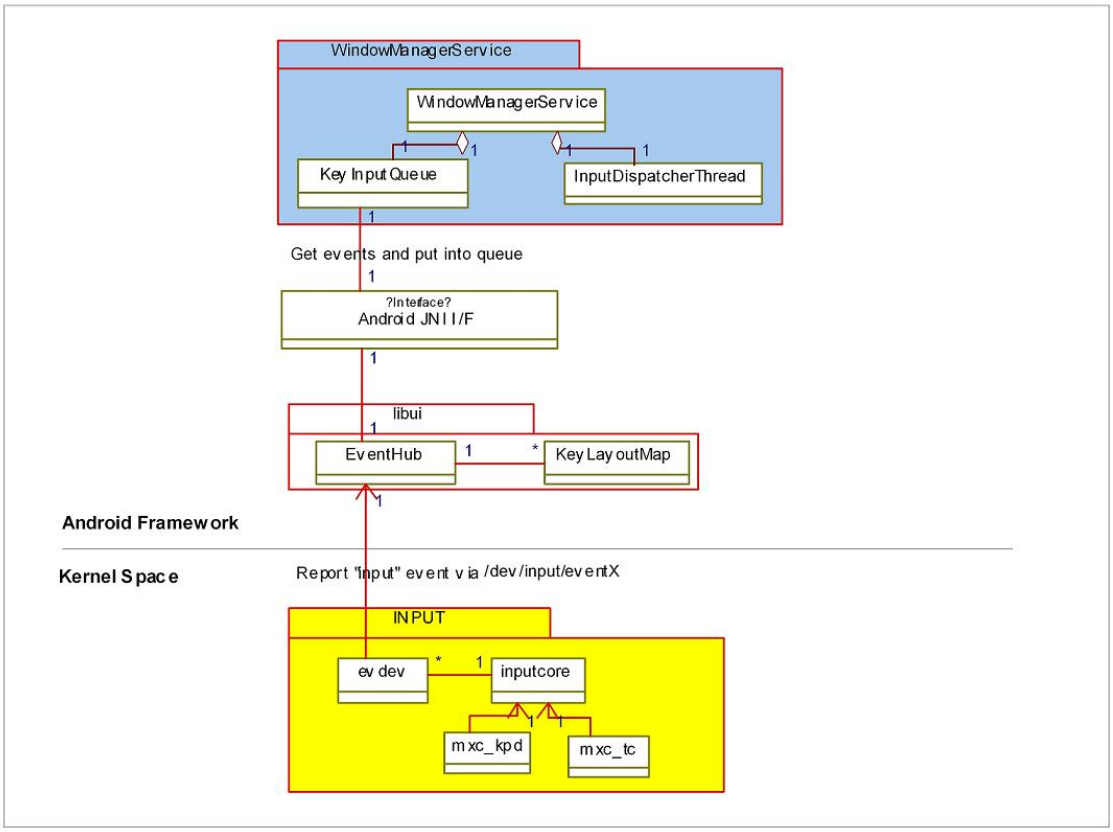
(EV_REL, 0x0001, 0xffff) - 0x0001 means "REL_Y" (move toward Y axes), 0xffff is the relative Y movement
...
(EV_REL, 0x0008, 0xffff) - 0x0008 means "REL_WHEEL" (scroll wheel)
...
(EV_KEY, 0x0110, 0x0001) - 0x0110 means "BTN_LEFT" (left button on mouse), 0x0001 means "pressing"
(EV_KEY, 0x0110, 0x0000) - 0x0000 means "up"

You can watch all events reported by kernel in Android shell by:

```
# getevent -v &
```

Components involved in "input"

Below chart demonstrate all components (both Android framework and kernel) involved in "input" event generation/grab/handling:



Kernel driver:

- inputcore/evdev: Input core for various input devices. Each input driver need register into input core during initialization.
- mxc_kpd, mxc_tc: FSL keypad and touch panel driver
- USB HID device is not showed in above chart

The following kernel config need be enabled for "input":

- CONFIG_INPUT
- CONFIG_INPUT_EVDEV
- CONFIG_INPUT_KEYBOARD
- CONFIG_KEYBOARD_MXC (in case you want to use FSL keypad driver)
- CONFIG_INPUT_TOUCHSCREEN
- CONFIG_TOUCHSCREEN_MXC (in case you want to use FSL touchscreen driver)
- CONFIG_HID, USB_HID (in case you have USB keyboard/mouse)

Framework component:

- libui (frameworks/base/libs/ui/)
  library for create instance for EventHub which will poll on all input devices, read raw event, map "scancode" to "keycode" according to key map file, and return the event (with both scan/key code) to runtime service
  For each "keyboard" device, EventHub will try search corresponding key layout file (*.kl) under /system/usr/keylayout/. For example, if you have a keypad device with name "mxckpd

- Android JNI I/F (KeyInputQueue, frameworks/base/services/jni/)
  Java->C bridge between runtime service (i.e. WindowManagerService here) and native library (i.e. libui here)

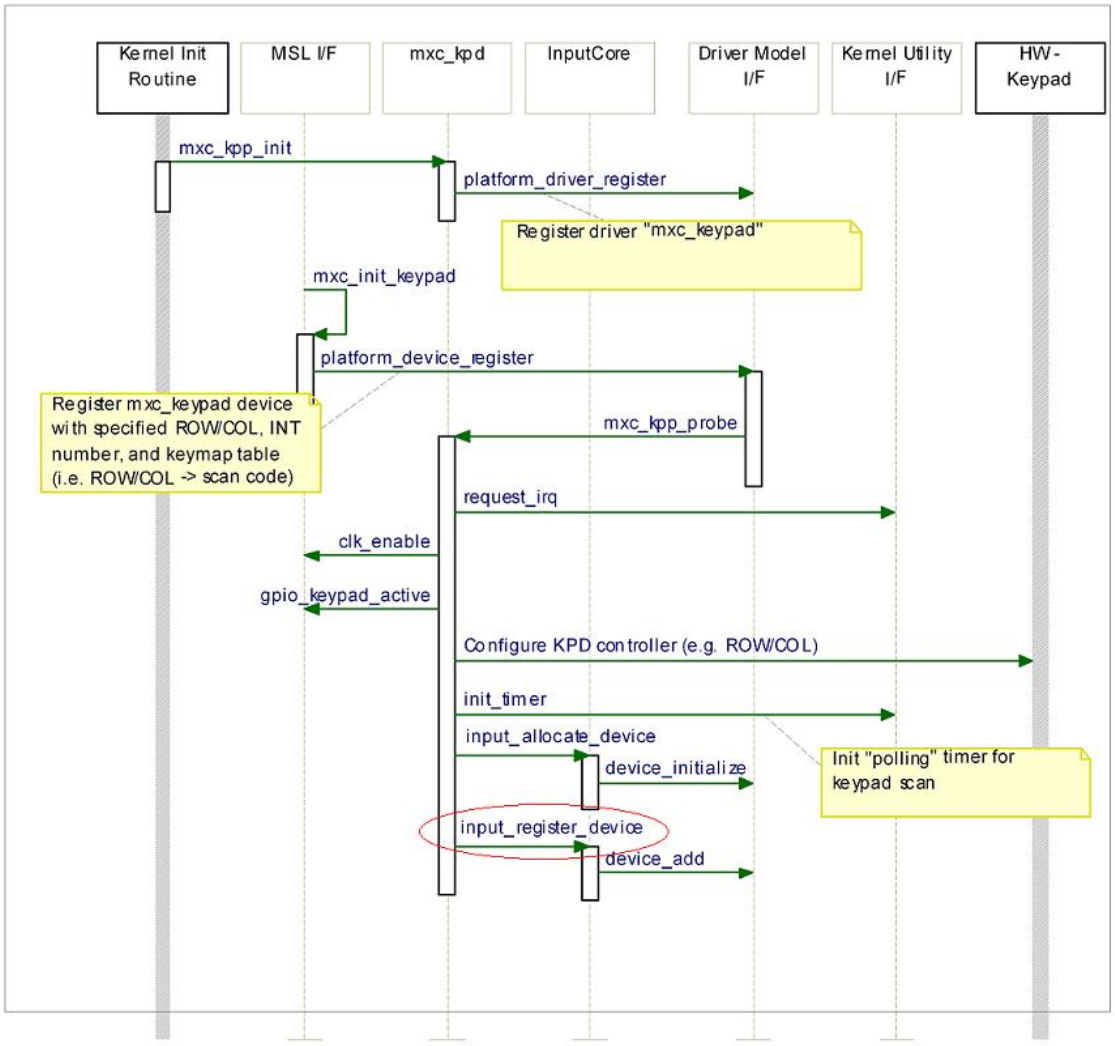
- WindowManagerService (frameworks/base/services/java/com/android/server/)
  Read input events, do some special handling (e.g. generate

"long press" event, wakeup LCD for special key, re-map UP/DOWN/LEFT/RIGHT/CENTER key when the device is rotated with 90/180/270 degree), and then put events into queue. With another thread (InputDispatcherThread), input event will be dispatched to view/widget according to it's type (KEY, ABS, REL).

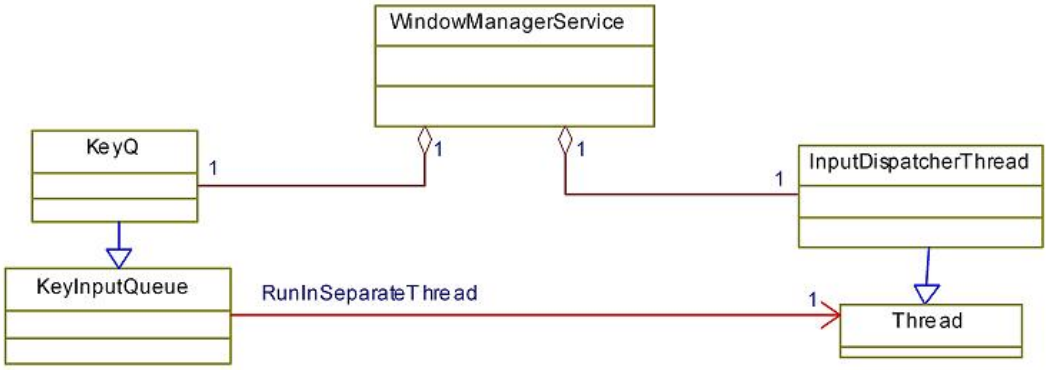
Interaction between Android framework and Linux kernel

During kernel bootup, those kernel input drivers (e.g. keypad, touchscreen) need register into input core so that "input" event can be reported to upper layer via event devices (/dev/input/eventX). USB HID driver register into input core when any USB HID device is plugged in and probed by driver.

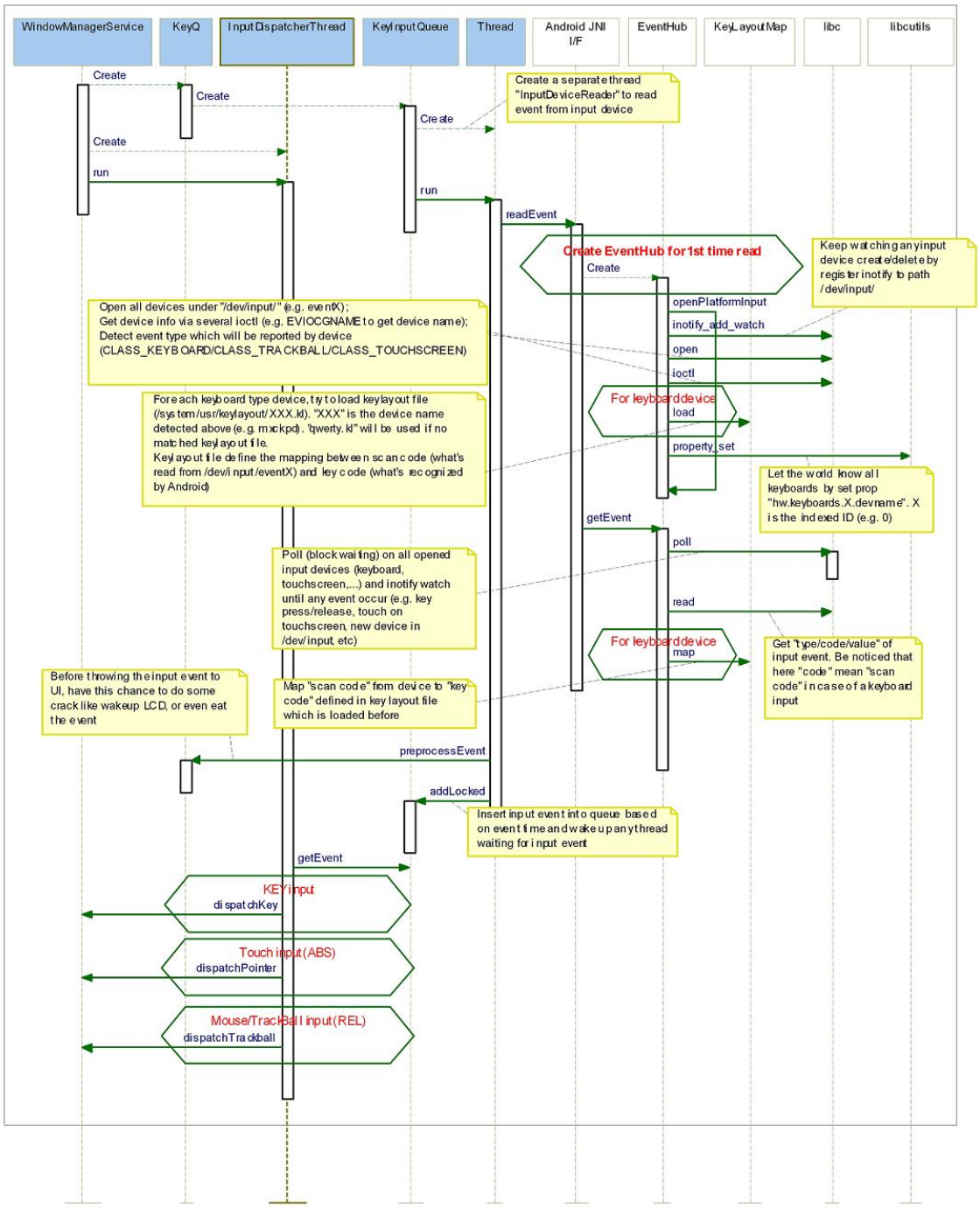
Here is an example for FSL keypad driver initializing:



Then Android bootup with WindowManager Service created. Window Manager service run two separate threads for input, one is a KeyQ object (inherited from KeyInputQueue class) running as a thread, read event from kernel and put into queue; another is a InputDispatcherThread which retrieve events from queue and dispatch them to view/widget.



Interaction between Android framework (e.g. WindowManagerService) and kernel is demonstrated in below chart:



Reference

1. "Keymaps and Keyboard Input" in Android Porting Guide under "development/pdk/docs/"
2. input.txt under kernel tree "Documentation/input/"