USB -Keyboard Guide
by DeadPool2
Last update: August 2, 2018

This document guides the user through:
1: Figuring out how to detect the USB-Keyboard event.
2. Translate USB-Keyboard raw input (keycode) to ASCII

Table of Contents
1. Detect USB-Keyboard by event.................................................................2
   1.1 Checking dmesg content.................................................................2
   1.2 Exploring /dev directory...............................................................2
   1.3 Find out eventX ............................................................................2
2. Translate raw input to ASCII
   2.1 Understand data stream from keyboard........................................4
   2.2 Interpret to ASCII..........................................................................5

Formatting:
1. Host (desktop) commands starting with $ are Linux console commands:
   $ echo "Hello world"
2. Target (board) commands start with #:
   # echo "On embedded board"
3. All commands are case sensitive.
1. Detect USB-Keyboard by event
When we connect USB-Keyboard to the host USB port on Beagle Bone, the output of keyboard could only be seen on the monitor that is also connected to BB by hub. Hence additional work needs to be done to get the output data stream from the USB-Keyboard.

1.1 Checking dmesg content
Firstly, connect your keyboard to the host USB port. Then use dmesg command to check the information about the usb device. You will find similar content as following:

```
We could find “USB HID v1.1.0 Keyboard” message in the dmesg, which tells us that your usb keyboard has been detected and assigned to some input port. Now we know that our USB Keyboard has been successfully connected to beagle bone and let’s explore the way that linux handles usb input.
```

1.2 Exploring /dev directory
The /dev directory is the one in Linux that contains all the device files for all the devices that are on your system. /dev/input is a sub directory that holds the device files for various input devices such as mouse, keyboard, joystick and so on.

1. ls /dev/input

The output would be similar to the screenshot above. Each event is corresponding to a device. Now we need to find out the specific event that is responsible for our USB Keyboard.

1.3 Find out eventX
1. cat /proc/bus/input/devices
This command would display relevant information about all the input devices connected to your system. Among the devices, you could find the “Handlers” for your usb-keyboard by combining the information you obtained from dmesg and “kbd” keyword. As shown on the screenshot above, my keyboard is event2.

2. `cat /dev/input/event2`

By executing this command and pressing the keyboard, you will find the corresponding data stream is showing on the screen.

However, the data stream is not sensitive enough for each key pressed (at least not sensitive in my case). And you might find multiple devices shown on the “/proc/bus/input/devices” that has the same information, in which it is hard to figure out which one is your usb keyboard. There are better places to look at.

3. `ls /dev/input/by-path`

The output would be similar to the screenshot above. We could find that there are specific event names for our usb-keyboard, which are ended with event-kbd. This output is very clear and straightforward.

4. `cat /dev/input/by-id/your-keyboard` and pressing keys on your keyboard

We could observe that each key will generate a corresponding data stream, which is sensitive enough for us to capture.

**2. Translate raw input to ASCII**
Now we know how Linux handles the usb keyboard. But we still don’t know the format of data stream we seen from the /dev/input and how to translate the raw data into the human understanding characters.

2.1 Understand data stream from keyboard

1. vi /usr/include/linux/input.h

The format for the input stream is given in this file as input_event structure shown in the above screenshot.

By using this structure, we could get the raw data from the keyboard.
2.2 Interpret to ASCII

The raw data we get from the keyboard each has a keycode. And each keycode is mapped to each key on keyboard. The map could also be found in /usr/include/linux/input.h file.

```
/*
 * Keys and buttons
 *
 * Most of the keys/buttons are modeled after USB HUT 1.12
 * (see http://www.usb.org/developers/hidpage).
 * Abbreviations in the comments:
 * AC - Application Control
 * AL - Application Launch Button
 * SC - System Control
 */

#define KEY_RESERVED      0
#define KEY_ESC            1
#define KEY_1              2
#define KEY_2              3
#define KEY_3              4
#define KEY_4              5
#define KEY_5              6
#define KEY_6              7
#define KEY_7              8
#define KEY_8              9
#define KEY_9             10
#define KEY_0             11
#define KEY_MINUS          12
#define KEY_EQUAL          13
#define KEY_BACKSPACE      14
#define KEY_TAB            15
#define KEY_Q             16
#define KEY_W             17
#define KEY_E             18
#define KEY_R             19
#define KEY_T             20
#define KEY_Y             21
#define KEY_U             22
```

We could use this map to make a table that translate the keycode to ASCII. The following table completes most mappings.
Troubleshooting:

1. You’d better connect your keyboard to usb port after beagle bone is completely booted, otherwise it could lead to unrecognizable of your beagle bone to host pc.
2. If you find multiple devices with almost the same information after cat /proc/bus/input/devices, you could follow the third step in 1.3 find out evenX.
3. If you are going to use input_event struct in your c code, please include linux/input.h library.
4. If could not open the device in your code, please double check the file path to your device is correct. The path should be look like following:

```
#define DEVICE "/dev/input/by-id/usb-CATEX_TECH._NANO7SSIMPLE_CA2015010003-event-kbd"
```