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

Formatting:

1. Host (desktop) commands starting with $ are Linux console commands:

```
$ echo "Hello world"
```

2. Target (board) commands start with #:

```
# echo "On embedded board"
```

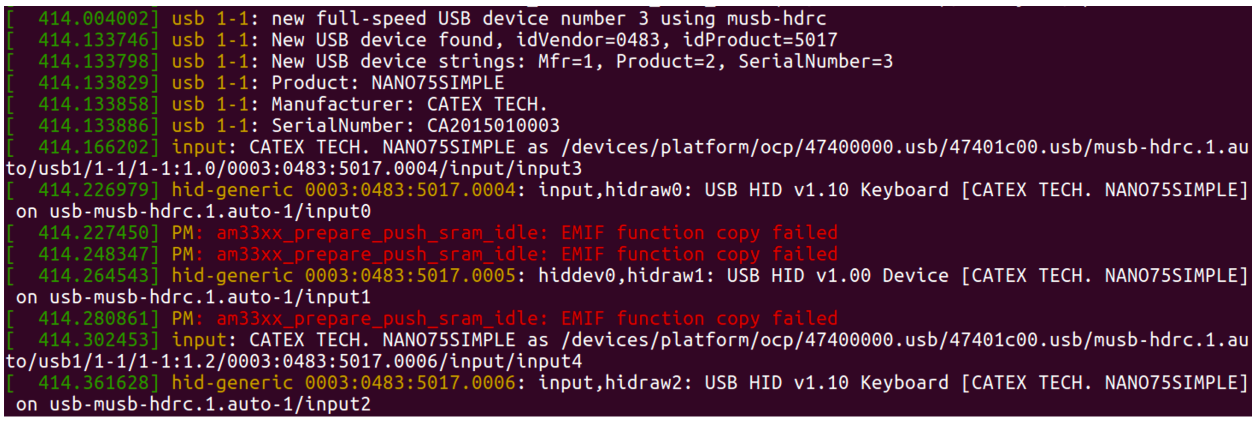
3. All commands are case sensitive.

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:



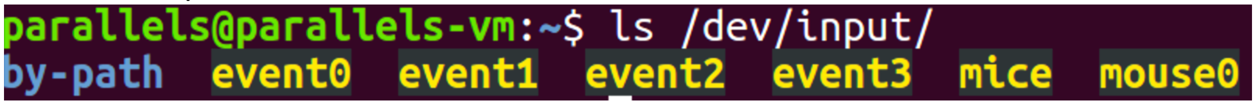
```
[  414.004002] usb 1-1: new full-speed USB device number 3 using musb-hdrc
[  414.133746] usb 1-1: New USB device found, idVendor=0483, idProduct=5017
[  414.133798] usb 1-1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[  414.133829] usb 1-1: Product: NANO75SIMPLE
[  414.133858] usb 1-1: Manufacturer: CATEX TECH.
[  414.133886] usb 1-1: SerialNumber: CA2015010003
[  414.166202] input: CATEX TECH. NANO75SIMPLE as /devices/platform/ocp/47400000.usb/47401c00.usb/musb-hdrc.1.au
to/usb1/1-1/1-1:1.0/0003:0483:5017.0004/input/input3
[  414.226979] hid-generic 0003:0483:5017.0004: input,hidraw0: USB HID v1.10 Keyboard [CATEX TECH. NANO75SIMPLE]
 on usb-musb-hdrc.1.auto-1/input0
[  414.227450] PM: am33xx_prepare_push_sram_idle: EMIF function copy failed
[  414.248347] PM: am33xx_prepare_push_sram_idle: EMIF function copy failed
[  414.264543] hid-generic 0003:0483:5017.0005: hiddev0,hidraw1: USB HID v1.00 Device [CATEX TECH. NANO75SIMPLE]
 on usb-musb-hdrc.1.auto-1/input1
[  414.280861] PM: am33xx_prepare_push_sram_idle: EMIF function copy failed
[  414.302453] input: CATEX TECH. NANO75SIMPLE as /devices/platform/ocp/47400000.usb/47401c00.usb/musb-hdrc.1.au
to/usb1/1-1/1-1:1.2/0003:0483:5017.0006/input/input4
[  414.361628] hid-generic 0003:0483:5017.0006: input,hidraw2: USB HID v1.10 Keyboard [CATEX TECH. NANO75SIMPLE]
 on usb-musb-hdrc.1.auto-1/input2
```

We could find "USB HID v1.10 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

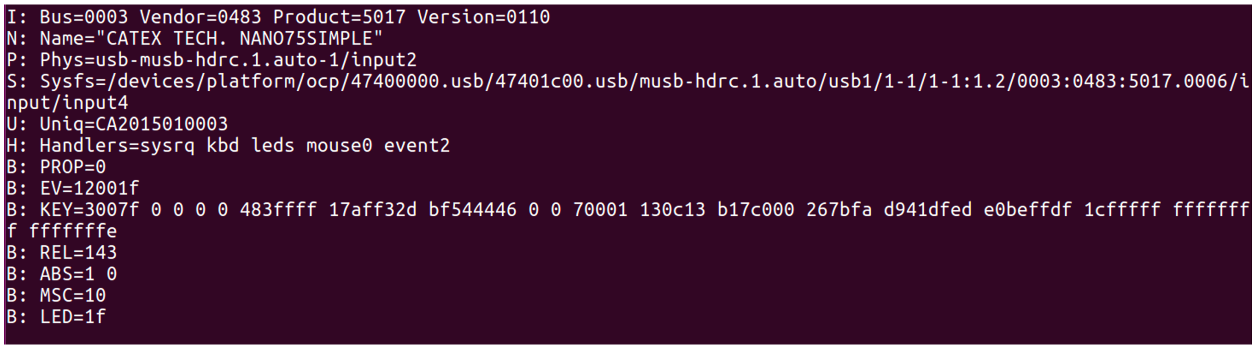


```
parallels@parallels-vm:~$ ls /dev/input/
by-path  event0  event1  event2  event3  mice  mouse0
```

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



```
I: Bus=0003 Vendor=0483 Product=5017 Version=0110
N: Name="CATEX TECH. NANO75SIMPLE"
P: Phys=usb-musb-hdrc.1.auto-1/input2
S: Sysfs=/devices/platform/ocp/47400000.usb/47401c00.usb/musb-hdrc.1.auto/usb1/1-1/1-1:1.2/0003:0483:5017.0006/i
nput/input4
U: Uniq=CA2015010003
H: Handlers=sysrq kbd leds mouse0 event2
B: PROP=0
B: EV=12001f
B: KEY=3007f 0 0 0 0 483ffff 17aff32d bf544446 0 0 70001 130c13 b17c000 267bfa d941dfed e0beffdf 1cfffff fffffff
f ffffffe
B: REL=143
B: ABS=1 0
B: MSC=10
B: LED=1f
```

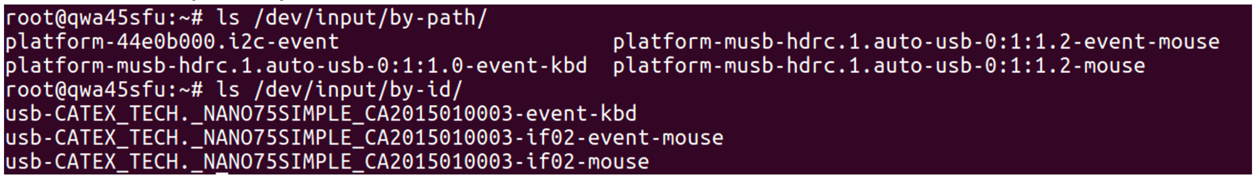
This command would display relevant information about all the input devices connected to your system. Among the devices, you could find the “Handlers” for your usb-keyboard by combining the information you obtained from dmesg and “kbd” keyword. As shown on the screenshot above, my keyboard is event2.

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.

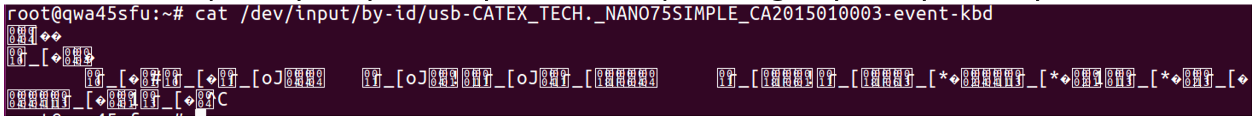
3. ls /dev/input/by-path
   ls /dev/input/by-id



```
root@qwa45sfu:~# ls /dev/input/by-path/
platform-44e0b000.i2c-event                 platform-musb-hdrc.1.auto-usb-0:1:1.2-event-mouse
platform-musb-hdrc.1.auto-usb-0:1:1.0-event-kbd  platform-musb-hdrc.1.auto-usb-0:1:1.2-mouse
root@qwa45sfu:~# ls /dev/input/by-id/
usb-CATEX_TECH._NANO75SIMPLE_CA2015010003-event-kbd
usb-CATEX_TECH._NANO75SIMPLE_CA2015010003-if02-event-mouse
usb-CATEX_TECH._NANO75SIMPLE_CA2015010003-if02-mouse
```

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



```
root@qwa45sfu:~# cat /dev/input/by-id/usb-CATEX_TECH._NANO75SIMPLE_CA2015010003-event-kbd
[illegible]��
[illegible]_[�[illegible]
        [illegible]_[�[illegible]#[illegible]_[�[illegible]_[oJ[illegible]    [illegible]_[oJ[illegible]_[oJ[illegible]_[[illegible]      [illegible]_[[illegible]_[[illegible]_[*�[illegible]_[*�[illegible]_[*�[illegible]_[�
[illegible]_[�[illegible]_[�[illegible]C
```

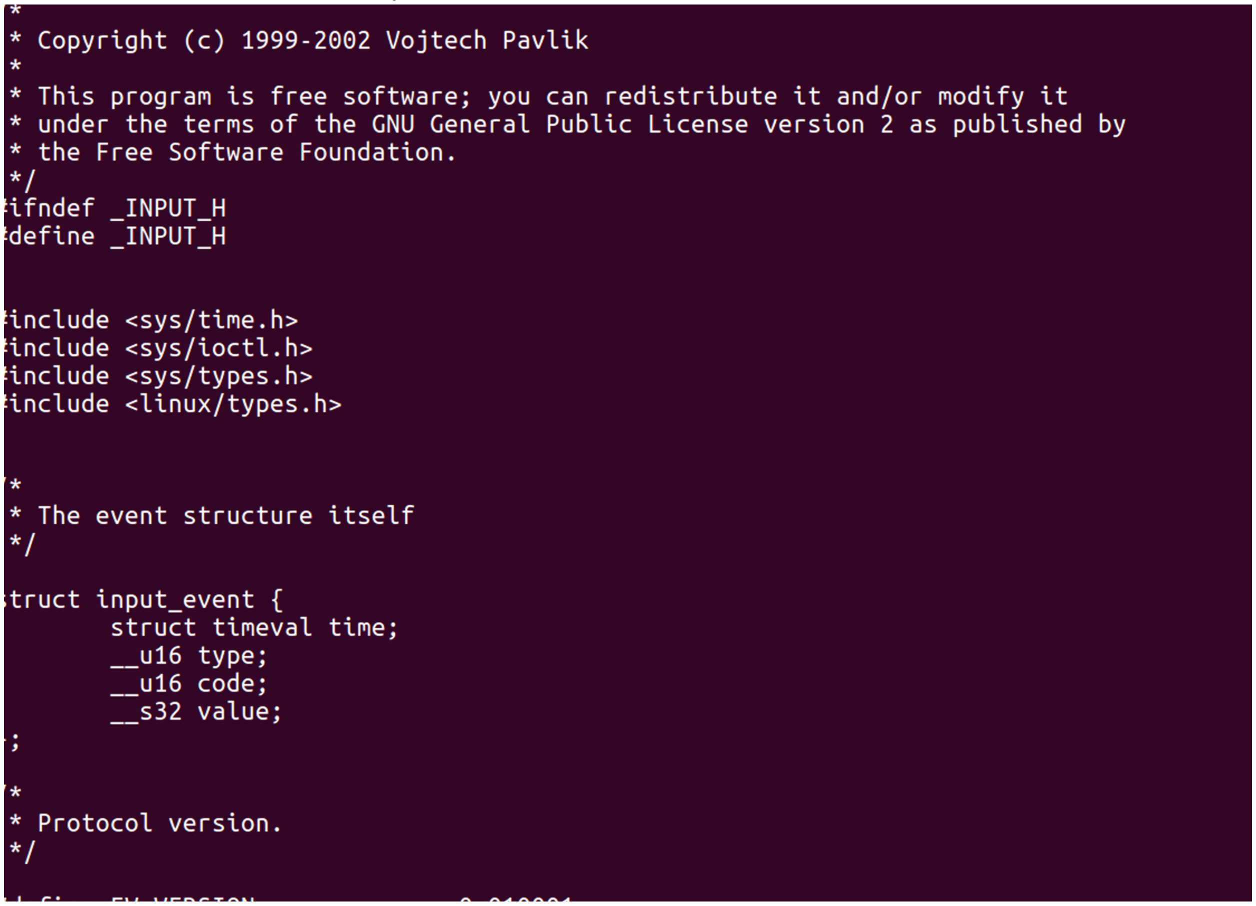
We could observe that each key will generate a corresponding data stream, which is sensitive enough for us to capture.

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



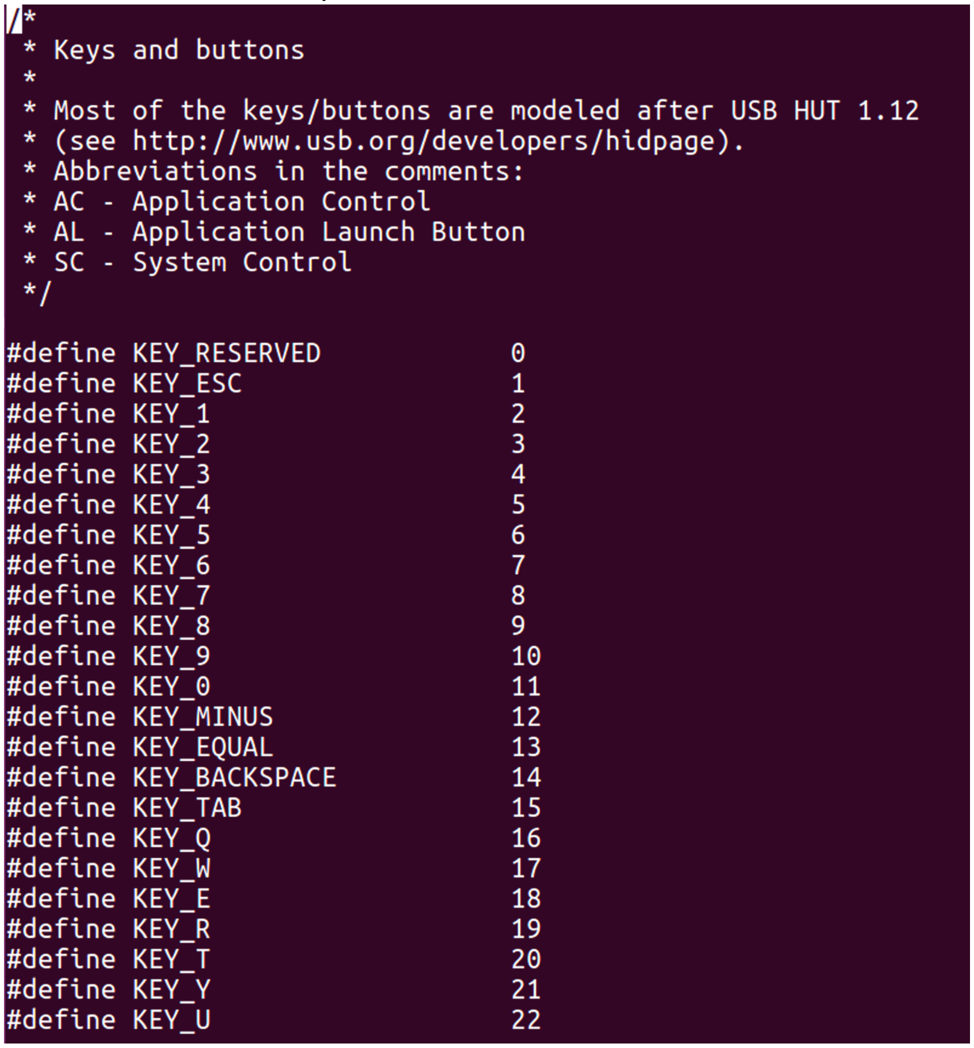
```
 *
 * Copyright (c) 1999-2002 Vojtech Pavlik
 *
 * This program is free software; you can redistribute it and/or modify it
 * under the terms of the GNU General Public License version 2 as published by
 * the Free Software Foundation.
 */
#ifndef _INPUT_H
#define _INPUT_H


#include <sys/time.h>
#include <sys/ioctl.h>
#include <sys/types.h>
#include <linux/types.h>


/*
 * The event structure itself
 */

struct input_event {
       struct timeval time;
       __u16 type;
       __u16 code;
       __s32 value;
};

/*
 * Protocol version.
 */
```

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.

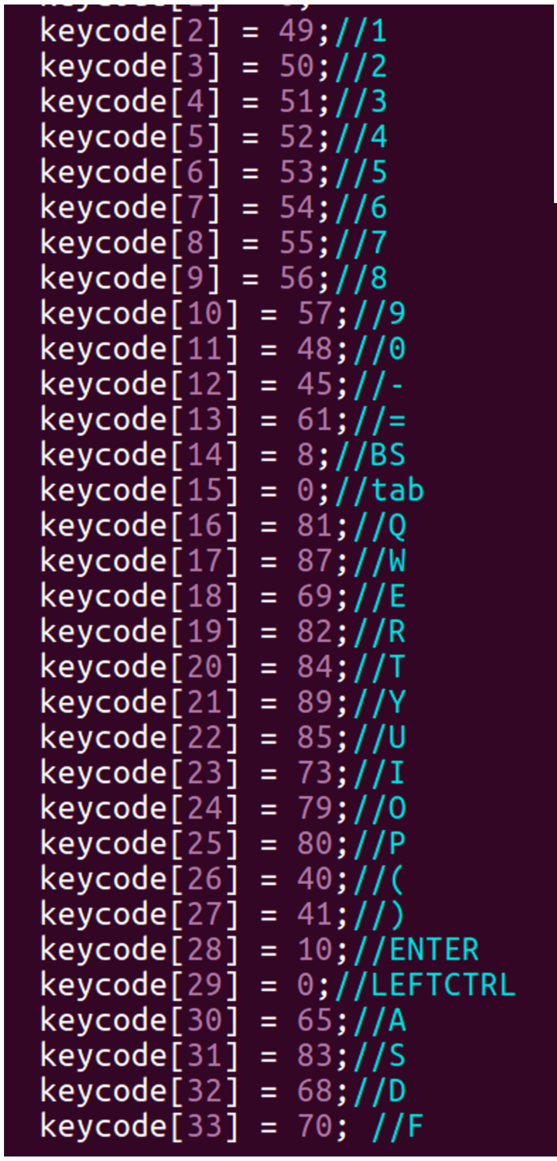
2.2 Interpret to ASCii

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.

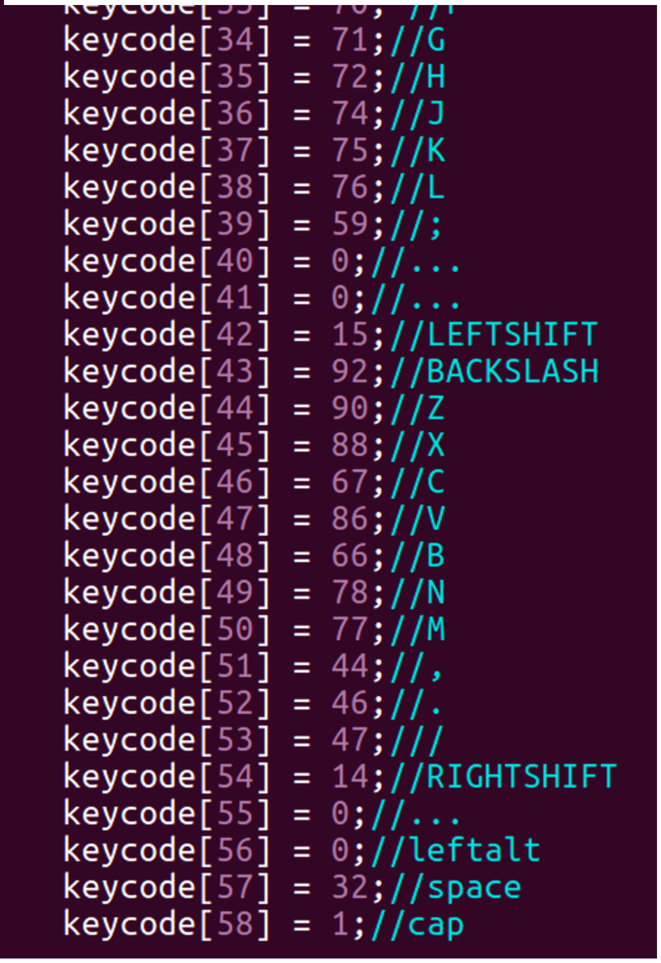


```
/*
 * Keys and buttons
 *
 * Most of the keys/buttons are modeled after USB HUT 1.12
 * (see http://www.usb.org/developers/hidpage).
 * Abbreviations in the comments:
 * AC - Application Control
 * AL - Application Launch Button
 * SC - System Control
 */

#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.



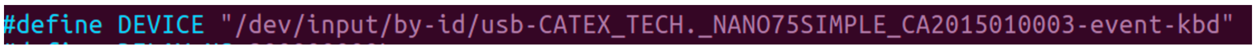
```
keycode[2] = 49;//1
keycode[3] = 50;//2
keycode[4] = 51;//3
keycode[5] = 52;//4
keycode[6] = 53;//5
keycode[7] = 54;//6
keycode[8] = 55;//7
keycode[9] = 56;//8
keycode[10] = 57;//9
keycode[11] = 48;//0
keycode[12] = 45;//-
keycode[13] = 61;//=
keycode[14] = 8;//BS
keycode[15] = 0;//tab
keycode[16] = 81;//Q
keycode[17] = 87;//W
keycode[18] = 69;//E
keycode[19] = 82;//R
keycode[20] = 84;//T
keycode[21] = 89;//Y
keycode[22] = 85;//U
keycode[23] = 73;//I
keycode[24] = 79;//O
keycode[25] = 80;//P
keycode[26] = 40;//(
keycode[27] = 41;//)
keycode[28] = 10;//ENTER
keycode[29] = 0;//LEFTCTRL
keycode[30] = 65;//A
keycode[31] = 83;//S
keycode[32] = 68;//D
keycode[33] = 70; //F
```



```
keycode[34] = 71;//G
keycode[35] = 72;//H
keycode[36] = 74;//J
keycode[37] = 75;//K
keycode[38] = 76;//L
keycode[39] = 59;//;
keycode[40] = 0;//...
keycode[41] = 0;//...
keycode[42] = 15;//LEFTSHIFT
keycode[43] = 92;//BACKSLASH
keycode[44] = 90;//Z
keycode[45] = 88;//X
keycode[46] = 67;//C
keycode[47] = 86;//V
keycode[48] = 66;//B
keycode[49] = 78;//N
keycode[50] = 77;//M
keycode[51] = 44;//,
keycode[52] = 46;//.
keycode[53] = 47;///
keycode[54] = 14;//RIGHTSHIFT
keycode[55] = 0;//...
keycode[56] = 0;//leftalt
keycode[57] = 32;//space
keycode[58] = 1;//cap
```

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._NANO75SIMPLE_CA2015010003-event-kbd"
```