

EETI eGTouch Android Guide v2.5d

TABLE OF CONTENTS

TABLE OF CONTENTS	0
Sec 1: Introduction	1
Sec 2: Before install	1
2.1 Patch kernel module	1
2.1.a make menuconfig	2
2.1.b manually modify config file.....	3
2.2 Patch kernel source code	3
2.2.a kernel 2.6.33 downwards	3
2.2.b kernel 2.6.34 upwards	6
2.2.c kernel 3.8.0 upwards.....	8
2.3 check device	9
Sec 3: Install Process	10
Sec 4: Calibration Tool	12
4-1 USB interface	12
4-2 RS232 interface	12
Sec 5: eGTouchA.ini Parameter Explanations	12
Sec 6: Technical Support	15
6-1 Environment Information	15
6-2 Register input devices	15
6-3 Driver debug log	15

Sec 1: Introduction

EETI provides all kinds of touch solution. EETI eGTouch is a touch daemon driver for EETI touch controller. Support USB & RS232 interface. Available for Android 2.1 upward and 4.0 as well. This document would assist you to install eGTouch and describe eGTouch feature in detail.

If you encounter any problem as running eGTouchD driver, please help us follow the steps described in SEC 6 to collect debug information. Send the information to us and tell us your problem. With useful information we could help you solve the problem faster.

Technical support and any question: **touch_fae@eeti.com**

Sec 2: Before install

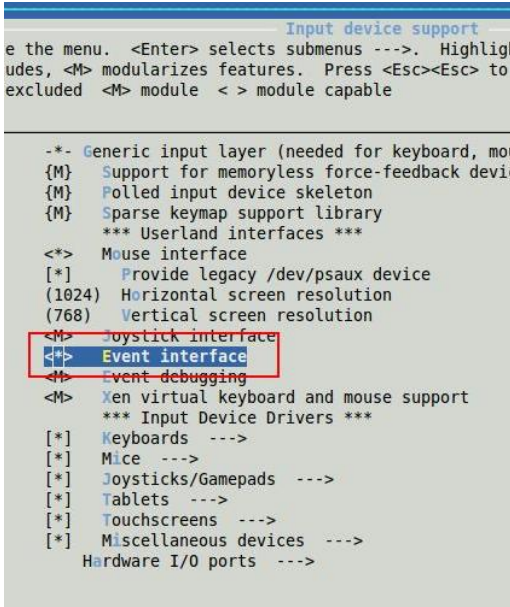
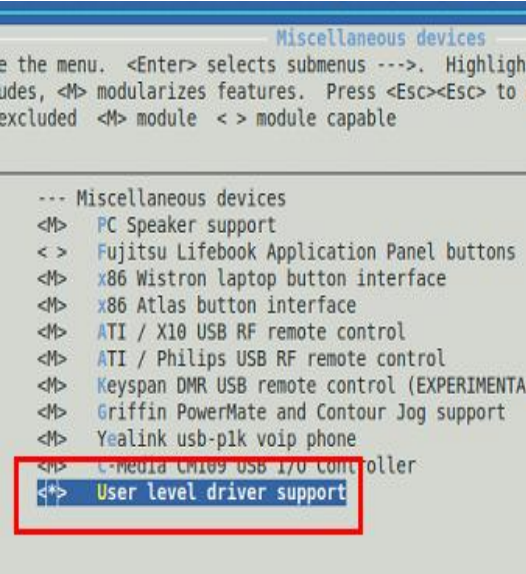
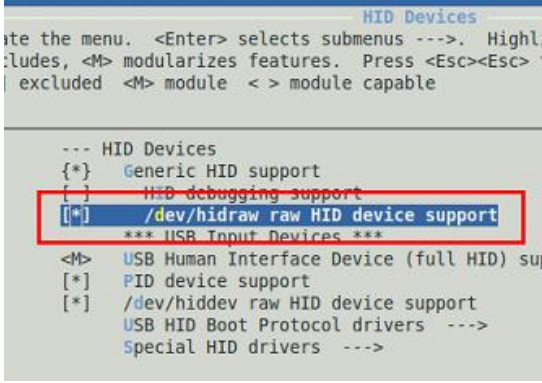
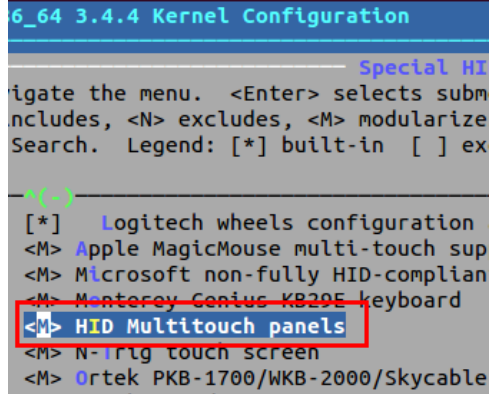
2.1 Patch kernel module

Before installing the driver, we need below kernel modules support:

1. CONFIG_INPUT_EVDEV
2. CONFIG_INPUT_UINPUT
3. CONFIG_HIDRAW (for USB Interface)
4. HID_MULTITOUCH (USB Interface & Kernel 3.0 upwards)

Please make sure to these modules are enabled. Users could check this by command `make menuconfig` or modify config file manually.

2.1.a make menuconfig

<p>[Device Drivers] / [Input device support] / [Event interface]</p>	<p>[Device Drivers] / [Input device support] / [Miscellaneous devices] / [User level driver support]</p>
 <pre> Input device support e the menu. <Enter> selects submenus --->. Highligh udes, <M> modularizes features. Press <Esc><Esc> to excluded <M> module < > module capable -* Generic input layer (needed for keyboard, mou {M} Support for memoryless force-feedback devic {M} Polled input device skeleton {M} Sparse keymap support library *** Userland interfaces *** <*> Mouse interface [*] Provide legacy /dev/psaux device (1024) Horizontal screen resolution (768) Vertical screen resolution <M> Joystick interface <*> Event interface <M> Event debugging <M> Xen virtual keyboard and mouse support *** Input Device Drivers *** [*] Keyboards ---> [*] Mice ---> [*] Joysticks/Gamepads ---> [*] Tablets ---> [*] Touchscreens ---> [*] Miscellaneous devices ---> Hardware I/O ports ---> </pre>	 <pre> Miscellaneous devices e the menu. <Enter> selects submenus --->. Highligh udes, <M> modularizes features. Press <Esc><Esc> to excluded <M> module < > module capable --- Miscellaneous devices <M> PC Speaker support < > Fujitsu Lifebook Application Panel buttons <M> x86 Wistron laptop button interface <M> x86 Atlas button interface <M> ATI / X10 USB RF remote control <M> ATI / Philips USB RF remote control <M> Keyspan DMR USB remote control (EXPERIMENTA <M> Griffin PowerMate and Contour Jog support <M> Yealink usb-plk voip phone <M> C-Media CM109 USB I/O Controller <*> User level driver support </pre>
<p>[Device Drivers] / [HID Devices] / [/dev/hidraw raw HID device support] (for USB Interface)</p>	<p>[Device Drivers] / [HID Devices] / Special HID drivers / HID Multitouch panels (If Kernel Version 3.0 upwards & for USB Interface)</p>
 <pre> HID Devices ite the menu. <Enter> selects submenus --->. Highligh cludes, <M> modularizes features. Press <Esc><Esc> to excluded <M> module < > module capable --- HID Devices [*] Generic HID support [*] HID debugging support [*] /dev/hidraw raw HID device support *** USB Input Devices *** <M> USB Human Interface Device (full HID) sup [*] PID device support [*] /dev/hiddev raw HID device support USB HID Boot Protocol drivers ---> Special HID drivers ---> </pre>	 <pre> 6_64 3.4.4 Kernel Configuration Special HI igate the menu. <Enter> selects subm cludes, <N> excludes, <M> modularize Search. Legend: [*] built-in [] ex [*] Logitech wheels configuration <M> Apple MagicMouse multi-touch sup <M> Microsoft non-fully HID-complian <M> Monterey Genius KB20E keyboard <M> HID Multitouch panels <M> N-Trig touch screen <M> Ortek PKB-1700/WKB-2000/Skycable </pre>

2.1.b manually modify config file

In some specific Android bsp, `make menuconfig` is invalid for setting kernel config. In this case, we'll advise to set those kernel config by directly modifying the config file. You could see many kernel config files under this path:

/Your_Android_bsp/kernel/arch/arm/configs/

Please select the file relevant to your platform model. For example, if you're using "tegra harmony" platform, the file name would be "**tegra_harmony_android_defconfig**".

Find out the config file based on that rule and modify the file manually. Make sure necessary items are set correctly as below:

CONFIG_INPUT_EVDEV=y

CONFIG_INPUT_UINPUT=y

CONFIG_HIDRAW=y

CONFIG_HID_MULTITOUCH=m

2.2 Patch kernel source code

If you're **NOT** using **USB** interface, you could **SKIP** this part.

Please append following **RED** section into your source code.

If your kernel is 2.6.33 downwards , please follow section 6-1-1 .
--

If your kernel is 2.6.34 upwards , please follow section 6-1-2 .
--

2.2.a kernel 2.6.33 downwards

1. /SourceCode/drivers/input/ evdev.c
--

<pre>static struct input_device_id evdev_blacklist[] = { /* Added by EETI */ { .flags = INPUT_DEVICE_ID_MATCH_BUS INPUT_DEVICE_ID_MATCH_VENDOR, .bustype = BUS_USB, .vendor = 0x0EEF, }, {}, /* Terminating entry */ };</pre>

```
static struct input_handler evdev_handler = {  
    .event = evdev_event,  
    .connect = evdev_connect,  
    .disconnect = evdev_disconnect,  
    .fops = &evdev_fops,  
    .minor = EVDEV_MINOR_BASE,  
    .name = "evdev",  
    .id_table = evdev_ids,  
  
    .blacklist = evdev_blacklist, /* Added by EETI */  
};
```

2. /SourceCode/drivers/input/mousedev.c

```
static struct input_device_id mousedev_blacklist[] =  
{  
    /* Added by EETI */  
    {  
        .flags = INPUT_DEVICE_ID_MATCH_BUS | INPUT_DEVICE_ID_MATCH_VENDOR,  
        .bustype = BUS_USB,  
        .vendor = 0x0EEF,  
    },  
    {  
        .flags = INPUT_DEVICE_ID_MATCH_BUS | INPUT_DEVICE_ID_MATCH_VENDOR,  
        .bustype = BUS_VIRTUAL,  
        .vendor = 0x0EEF,  
    },  
    {}, /* Terminating entry */  
};  
  
static struct input_handler mousedev_handler = {  
    .event = mousedev_event,  
    .connect = mousedev_connect,  
    .disconnect = mousedev_disconnect,  
    .fops = &mousedev_fops,  
    .minor = MOUSEDEV_MINOR_BASE,  
    .name = "mousedev",  
    .id_table = mousedev_ids,  
  
    .blacklist = mousedev_blacklist, /* Added by EETI */  
};
```

3. /SourceCode/drivers/input/joydev.c

```
static const struct input_device_id joydev_blacklist[] =
{
    {
        .flags = INPUT_DEVICE_ID_MATCH_EVBIT | INPUT_DEVICE_ID_MATCH_KEYBIT,
        .evbit = { BIT_MASK(EV_KEY) },
        .keybit = { [BIT_WORD(BTN_TOUCH)] = BIT_MASK(BTN_TOUCH) },
    },    /* Avoid itouchpads and touchscreens */
    {
        .flags = INPUT_DEVICE_ID_MATCH_EVBIT | INPUT_DEVICE_ID_MATCH_KEYBIT,
        .evbit = { BIT_MASK(EV_KEY) },
        .keybit = { [BIT_WORD(BTN_DIGI)] = BIT_MASK(BTN_DIGI) },
    },    /* Avoid tablets, digitisers and similar devices */

    {
        .flags = INPUT_DEVICE_ID_MATCH_BUS | INPUT_DEVICE_ID_MATCH_VENDOR,
        .bustype = BUS_VIRTUAL,
        .vendor = 0x0EEF,
    },    /* Added by EETI */
    { }    /* Terminating entry */
};

static struct input_handler joydev_handler = {
    .event = joydev_event,
    .connect = joydev_connect,
    .disconnect = joydev_disconnect,
    .fops = &joydev_fops,
    .minor = JOYDEV_MINOR_BASE,
    .name = "joydev",
    .id_table = joydev_ids,
    .blacklist = joydev_blacklist,
};
```

2.2.b kernel 2.6.34 upwards

1. /SourceCode/drivers/input/**evdev.c**

```
static bool evdev_match(struct input_handler *handler, struct input_dev *dev)
{
    /* Avoid EETI USB touchscreens */
    #define VID_EETI 0x0EEF
    if ((BUS_USB == dev->id.bustype) && (VID_EETI == dev->id.vendor))
        return false;
    return true;
}

static struct input_handler evdev_handler = {
    .event = evdev_event,
    .match = evdev_match, /* Added by EETI */
    .connect = evdev_connect,
    .disconnect = evdev_disconnect,
    .fops = &evdev_fops,
    .minor = EVDEV_MINOR_BASE,
    .name = "evdev",
    .id_table = evdev_ids,
};
```

2. /SourceCode/drivers/input/**mousedev.c**

```
static bool mousedev_match(struct input_handler *handler, struct input_dev *dev)
{
    /* Avoid EETI USB touchscreens */
    #define VID_EETI 0x0EEF
    if ((BUS_USB == dev->id.bustype) && (VID_EETI == dev->id.vendor))
        return false;
    /* Avoid EETI virtual devices */
    if ((BUS_VIRTUAL == dev->id.bustype) && (VID_EETI == dev->id.vendor))
        return false;
    return true;
}

static struct input_handler mousedev_handler = {
    .event = mousedev_event,
    .match = mousedev_match, /* Added by EETI */
};
```

```
.connect = mousedev_connect,  
.disconnect = ousedev_disconnect,  
.fops = &mousedev_fops,  
.minor = MOUSEDEV_MINOR_BASE,  
.name = "mousedev",  
.id_table = mousedev_ids,  
};
```

3. /SourceCode/drivers/input/joydev.c

```
static bool joydev_match(struct input_handler *handler, struct input_dev *dev)  
{  
    /* Avoid touchpads and touchscreens */  
    if (test_bit(EV_KEY, dev->evbit) && test_bit(BTN_TOUCH, dev->keybit))  
        return false;  
    /* Avoid tablets, digitisers and similar devices */  
    if (test_bit(EV_KEY, dev->evbit) && test_bit(BTN_DIGI, dev->keybit))  
        return false;  
    /* Avoid EETI virtual devices */  
    #define VID_EETI 0x0EEF  
    if ((BUS_VIRTUAL == dev->id.bustype) && (VID_EETI == dev->id.vendor))  
        return false;  
    return true;  
}  
  
static struct input_handler joydev_handler = {  
    .event = joydev_event,  
    .match = joydev_match,  
    .connect = joydev_connect,  
    .disconnect = joydev_disconnect,  
    .fops = &joydev_fops,  
    .minor = JOYDEV_MINOR_BASE,  
    .name = "joydev",  
    .id_table = joydev_ids,  
};
```


2.2.c Kernel 3.8.0 upwards

Important! If your system fulfill below two conditions, please refer to **Sec 6-1-2** to do kernel HIDCORE patch first, or driver would **NOT** be functional.

1.	Interface is USB
2.	Kernel Version > 3.8.0
3	Resistive or SCAP controller

If your Linux kernel version is **3.8.0** upwards and using **resistive** or **SCAP** touch controller, please comment the following **RED** section in your source code.

/SourceCode/drivers/hid/hid-core.c

```
bool hid_ignore(struct hid_device *hdev)
{
    ...
    switch (hdev->vendor) {
        ...
        /*case USB_VENDOR_ID_DWAV:*/
            /* These are handled by usbtouchscreen. hdev->type is probably
             * HID_TYPE_USBNONE, but we say !HID_TYPE_USBMOUSE to match
             * usbtouchscreen. */
            /*if ((hdev->product == USB_DEVICE_ID_EGALAX_TOUCHCONTROLLER ||
                 hdev->product == USB_DEVICE_ID_DWAV_TOUCHCONTROLLER) &&
                 hdev->type != HID_TYPE_USBMOUSE)
                return true;
            break;*/
        ...
    }
    ...
}
```

2.3 check device

Please make sure these check items are passed, if not so, the driver would not be executed successfully.

- 1.) After patching kernel, build new kernel and reboot for taking effect on changes.
- 2.) Check uinput functions enable or not

UINPUT device node

You should see uinput under **/dev/input/uinput** or **/dev/uinput**.

For example:

```
File Edit View Terminal Help
root@william-desktop:/dev/input# pwd
/dev/input
root@william-desktop:/dev/input# ls uinput -al
crw-r----- 1 root root 10, 223 2010-01-05 15:43 uinput
root@william-desktop:/dev/input#
```

- 3.) If interface is **USB**. After plug in an USB device, check below functions.

HIDRAW device node

As the usb device is plug-in, there would be a **hidraw** node generated under **/dev**

```
File Edit View Terminal Help
root@william-desktop:/dev# pwd
/dev
root@william-desktop:/dev# ls hidraw* -al
crw-rw---- 1 root root 251, 0 2010-01-05 17:02 hidraw0
root@william-desktop:/dev#
```

USB touch device handlers

Type command "**cat /proc/bus/input/devices**" and see the result.

If you need and have done the kernel source code patch at section 2.2, you would see a **blank content** behind the **Handlers** item.

```
I: Bus=0003 Vendor=0eef Product=720c Version=0100
N: Name="eGalax Inc. USB TouchController"
P: Phys=usb-0000:00:1d.0-2/input0
S: Sysfs=/devices/pci0000:00/0000:00:1d.0/usb2/2-2/2-2:1.0/input/input7
U: Uniq=
H: Handlers=
B: EV=1b
B: KEY=421 0 30001 0 0 0 0 0 0 0
B: ABS=100 3f
B: MSC=10
```

Sec 3: Install Process

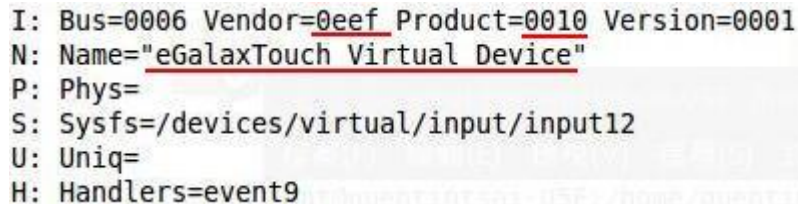
This section describes how to install eGTouch into Android. Before following this, please make sure referring to **"Section 2 Patch Kernel"** to rebuild kernel for supporting necessary features.

1. EETI eGTouch package contains:
 - a) eGTouchD: a daemon service driver for EETI touch controller.
 - b) eGTouchA.ini : a parameter list could be loaded by driver
 - c) eGalaxTouch_VirtualDevice.idc: a file necessary for Android 3.0 upwards
 - d) eGalaxCalibrator: a tool provides calibration and line drawing.
2. Place "eGTouchA.ini" into Android system directory "/data/eGTouchA.ini" where driver would load it. We can change driver behavior by modifying this file. **The detail descriptions of parameters are described in Section 5.** (You can see brief definitions in eGTouchA.ini)
3. To make eGTouchD driver run from the beginning of system operation. Please put eGTouchD driver into Android system directory "/system/bin". Modify the file **"init.rc"** in Android system and add service:

Below Android 4.0	service eGTouchD /system/bin/eGTouchD user root group root oneshot
Above Android 4.0	service eGTouchD /system/bin/eGTouchD class main user root group root oneshot

4. [Android 3.0 upwards] Please put file "eGalaxTouch_VirtualDevice.idc" into Android system directory "/system/usr/idc/".

5. After launching eGTouchD, check **/proc/bus/input/devices** file and we will find a virtual devices called "eGalaxTouch Virtual Device". Information like below figure:



```
I: Bus=0006 Vendor=0eef Product=0010 Version=0001
N: Name="eGalaxTouch Virtual Device"
P: Phys=
S: Sysfs=/devices/virtual/input/input12
U: Uniq=
H: Handlers=event9
```

We could check event node which was assigned to the virtual device and read/get input event through this device node, e.g. /dev/input/eventX.

6. If in item 5, the virtual device is not generated, please check whether there is hidraw generated under /dev/ directory as plug-in the device. If not, this may cause by invalid kernel patch, please do kernel patch based on Sec 2.1.b.

Sec 4: Calibration Tool

If you're using the "PCAP controller", please **SKIP** this section and **DO NOT** install this calibration tool. The PCAP controller doesn't need to do any calibration.

4-1 USB interface

Please install the tool **eGalaxCalibrator.apk** under the **/CalibrationTool/USB/** directory.

4-2 RS232 interface

Please install the tool **eGalaxAutoCalib.apk** under the **/CalibrationTool/UART/** directory.

As the calibration couldn't work properly, please contact EETI touch_fae@eeti.com for technical support.

Sec 5: eGTouchA.ini Parameter Explanations

The file **eGTouchA.ini** has a parameter list which would be loaded by driver. Driver's behavior could be changed by these parameters. Please **DON'T** modify the front title as setting up eGTouchA.ini.

This table describe the detailed usage of all parameters. There is also a simple description in eGTouchA.ini.

◆ DebugEnableBits		Debug message you want to show.
0	Close all Debug	
1	Print initialization debug message [Default]	
FFFF	Open all Debug	
F		
◆ ShowDebugPosition		Position you want to show/store Debug message
0	Print in file located at /data/	
1	Print in logcat [Default]	
2	Print in above both	
◆ Baudrate		Choose the BaudRate
0	Auto detect Baudrate [Default]	
X	Set Baudrate to X bps. (PCAP: 57600 , Resis: 9600)	
◆ ScanInterface		Choose scan interface
0	Scan all interface [Default] (USB / RS232 / PS/2)	
1	Scan USB interface only.	
2	Scan UART interface only.	
3	Scan PS/2 interface only.	

◆ SerialPath		RS232 Serial Path
default	Default path /dev/ttySX (X could be equals to 0-10) [Default]	
/dev/serial/ttyS0	Customized path. Please type in your specific serial path according to the form.	
◆ DeviceNums		How many devices you want to plug-in to the system. If you want more than one device, please modify this value.
1	Only one device [Default]	
2-10	More than one device. [Max = 10]	
◆ SupportPoints		The amount of points you want to report (This is also confined by Controller)
0	No point	
1	Single-touch	
>=2	Multi-touch [Default = 5]	
◆ Direction		Change the X and Y direction
0	Don't make any invert [Default]	
1	Invert X	
2	Invert Y	
3	Invert both X and Y	
4	Swap X and Y	
◆ Orientation		Change the orientation
0	0 degree [Default]	
1	90 degree	
2	180 degree	
3	270 degree	
◆ EdgeCompensate		Do edge compensate
0	Disable [Default]	
1	Enable	
EdgeLeft, EdgeRight	Edge compensate value	
EdgeTop, EdgeBottom		
X	If equals to 100, it means no change. If you set Left=50, you'll see the left-edge points are shrinks inward. And vice versa. [Min 50 - 150 Max] [Default = 100]	
◆ HoldFilterEnable		Filter out constant touch or not
0	Disable [Default]	
1	Enable	
HoldRange	Constant touch valid area	
X	±X range of the point which would lead to constant touch [Min 0 - 50 Max] [Default = 10]	

◆ SplitRectMode		Split the display into Specific Rect. Touch would just show on the specific Rect.									
0	No change (Full Display) [Default]										
1-8	Driver in-built split Rect										
	<table><tr><td>2</td><td>1</td></tr><tr><td>3</td><td>4</td></tr></table>	2	1	3	4	<table><tr><td>5</td></tr><tr><td>6</td></tr></table>	5	6	<table><tr><td>7</td><td>8</td></tr></table>	7	8
2	1										
3	4										
5											
6											
7	8										
9	Customized Rect.										
CustomRectLeft CustomRectRight CustomRectTop CustomRectBottom		Theses parameters are valid as SplitRectMode=9. You can customize the Rect by these parameters.									
0-2047	Four sides of the customized Rect										
◆ VKEYEnable		Enable this option if there's virtual key on your touch sensor.									
0	Disable [Default]										
1	Enable										
VKEYReportMod		Virtual Key Sensitivity.									
X	Smaller value means more sensitive. On the other hand, larger is less sensitive.										
VKEY_X	Y	<div>1. The value of X refer to the vkey package reported by controller as touching the specified virtual key on your sensor.</div> <div>2. The value of Y refer to the keyevent code in input.h which you want to report to system. You can choose the keyevent you want and fill in the code.</div>									
Example A: VKEY_0	139	<div>1. As controller report vkey package [0], we'll send keyevent code [139] to system.</div> <div>2. The event code [139] in input.h refers to MENU_KEY.</div> <div>Note: 139 is a decimal number.</div>									
Example B: VKEY_1	0x160	<div>1. You can fill in Hex number in Y by adding a symbol [0x] before the number.</div> <div>2. As controller report vkey package [1], we'll send keyevent code [0x160] to system</div> <div>3. The event code [0x160] in input.h refer to KEY_OK.</div>									
Note: If you're not sure controller's vkey package value, please contact EETI vendor.											

Sec 6: Technical Support

If you encounter any problem as running eGTouchD driver, please help us follow below steps to collect debug information. Send those information to us and tell us your problem.

With these information we could help you solve the problem faster.

Technical support and any question: touch_fae@eeti.com

Please provide us below information and the description of the problem you encountered for us.

6-1 Environment Information

Fill in this chart

1. CPU type	
2. Kernel version	
3. Android version.	
4. Touch Controller Interface	
5. Touch Controller Type	

6-2 Register input devices

1. Execute command ``cat /proc/bus/input/devices``
2. Send us the output result.

6-3 Driver debug log

1. Modify file eGTouchA.ini. Change the value of the parameter DebugEnableBits from 1 to FFFFF. Change the value of ShowDebugPosition from 1 to 0.

As below

```
[eGTouchA.ini]
DebugEnableBits      FFFFF
ShowDebugPosition    0
```

2. Reboot your system. After rebooting, please touch four corner of the touch panel.
3. The log file would be printed in `/data/eGTouch_[year]_[date]_[time]`
4. You may see one more log named `eGTouch_[year]_[date]_[time]`. Please send us the Newest one for analyzing. Thanks.