EETI eGTouch Linux Programming Guide v2.5a

TABLE OF CONTENTS

Sec 1: Introduction

EETI provides all kinds of touch solution. EETI eGTouchD is a touch daemon driver for EETI touch controller. Only is available for kernel **2.6.24** upward.

Support interfaces:

1. **USB**
2. **RS232**
3. **PS/2**

Having below features:

1. **Precise points.**
2. **Great calibration precision for Resistive controller.**
3. **Capable for 10+ points report.**
4. **Following Linux Standard Multitouch-protocol point report.**
5. **Rightclick, beep sound, constant touch filter, etc.**
6. **Support multi devices.**
7. **Available for detecting X-window rotation to do rotating coordinate.**
8. **Provide manually modify driver's behavior.**

This document would assist you to install eGTouchD.

If you encounter any problem as running eGTouchD driver, please help us follow the steps described in SEC 7 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

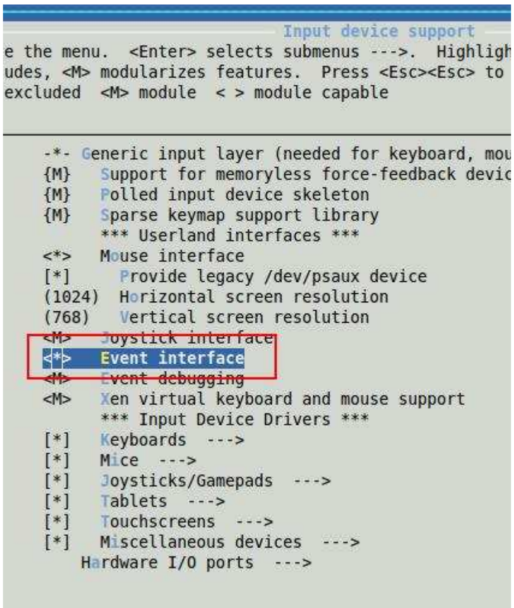
To install driver, we need below kernel modules support. Please make sure to enable these modules.

1. **EVDEV**
2. **UINPUT**
3. **HIDRAW (USB Interface)**

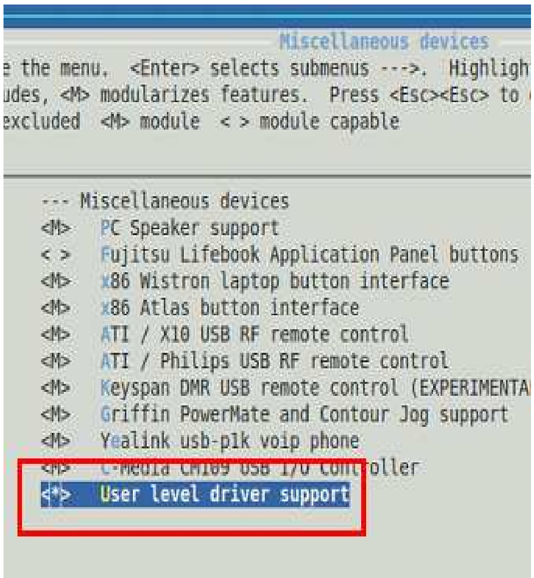
You could check this by "make menuconfig" command or modify Kconfig file. There is an example of "make menuconfig" below:

Make menuconifg:

[Device Drivers] / [Input device support] / [Event interface]

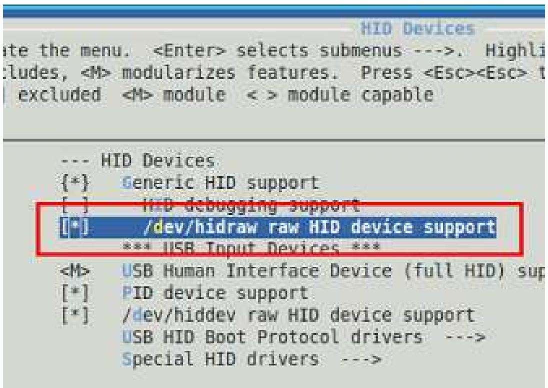


[Device Drivers] / [Input device support] / [Miscellaneous devices] / [User level driver support]



[Device Drivers] / [HID Devices] / [/dev/hidraw raw HID device support]

(for USB Interface)



2.2 Patch kernel source code

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

1.	**Interface is using USB**
2.	**X.org version < 1.8.7 or without X**

2.3 check device

1.) If you did above modification, please rebuild your kernel to make it effect.

2.) After that, you could check those kernel functions enable or not through below steps.

All interface.

a. 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#
```

USB interface only.

b. 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#
```

c. USB touch device handlers

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

If you need and have done the 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 Driver Package

3-1 Install Process

You could execute script file **setup.sh** to automatically install driver.

Syntex:

```
sh setup.sh                # To install the eGTouch driver.
sh setup.sh uninstall      # To remove the eGTouch driver.
```

You could also complete these steps manually.

1. Decompress eGTouch package which contains:
   - a) eGTouchD: a daemon service driver for EETI touch controller.
   - b) eGTouchL.ini: a parameter list loaded by driver
   - c) GetEvent.c: a sample code describes how to read EETI input event.

   If you have X-window, you may also be available for these:
   - d) eGTouchU: a X-window utility tool for eGTouchD (x86 system only)
   - e) eCalib: a command line X-window calibration tool.

2. Place “eGTouchL.ini” into Linux system directory “/etc/eGTouchL.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 eGTouchL.ini)

3. Place eGTouchD & eGTouchU & eCalib under /usr/bin. Place /usr/bin/eGTouchD execution in /etc/rc.local to make eGTouchD execute at system boot.

4. After launching eGTouchD with device plugged, check **/proc/bus/input/devices** file and you will find two virtual devices. Like below figures:

```
I: Bus=0006 Vendor=0eef Product=0020 Version=0001
N: Name="eGalaxTouch Virtual Device for Multi"
P: Phys=
S: Sysfs=/devices/virtual/input/input13
U: Uniq=
H: Handlers=event10
B: PROP=0
```

```
I: Bus=0006 Vendor=0eef Product=0010 Version=0001
N: Name="eGalaxTouch Virtual Device for Single"
P: Phys=
S: Sysfs=/devices/virtual/input/input14
U: Uniq=
H: Handlers=event11
```

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.

3-2 Tools

As you have **X-window**, these tools are available for use.

eGTouchU x86 system only	The tool eGTouchU is a utility tool which could help you modify driver's parameter through UI. The detail descriptions please refer to the document "EETI eGTouch Utility Guide" in driver package.
eCalib	The tool eCalib is a calibration tool with command line. Please type "eCalib -h" to see the usage content.

Sec 4: Touch Input Event Sequence

The eGTouchD daemon sends input event through kernel feature UINPUT so that the client program can get these events from /dev/input/eventX.

4-1 Two different event sequences

The eGTouchD daemon would report event based on different kernel version.

1. kernel version is 2.6.36 upwards:

Multi-touch Protocol Type B

```
ABS_MT_SLOT 0
ABS_MT_TRACKING_ID 0
ABS_MT_POSITION_X x[0]
ABS_MT_POSITION_Y y[0]
ABS_MT_SLOT 1
ABS_MT_TRACKING_ID 1
ABS_MT_POSITION_X x[1]
ABS_MT_POSITION_Y y[1]
```

you can see the detailed rule described in /Documentation/input/**multi-touch-protocol.txt** under Linux kernel source code.

2. kernel version is 2.6.35 downwards:

EETI protocol: Standard mouse event and custom extra event

Type = EV_KEY Code = BTN_LEFT Value = left mouse button state of **first point**, 1: pen down / 0: life off.	Type = EV_KEY Code = BTN_EXTRA Value = the touch state of **second point**, 1: pen down / 0: lift off.
Type = EV_ABS Code = ABS_X	Type = EV_ABS Code = ABS_RX

Value = the X axis position of **first point**. The range is from 0 to 2047.	Value = the X axis position of **second point**. The range is from 0 to 2047
Type = EV_ABS Code = ABS_Y Value = the Y axis position of **first point**. The range is from 0 to 2047.	Type = EV_ABS Code = ABS_RY Value = the Y axis position of **second point**. The range is from 0 to 2047.
Type = EV_SYNC Code = SYN_REPORT Value = 0 A Sync report event, all data will be valid after this event is received.	

4-2 How to read touch event

EETI provide a sample code **GetEvent.c** to show how the event sequence behaves. Please compile the sample code and execute it corresponding to the event node (/dev/input/eventX). You would see the event sequence as panel is touched and design your own application based on this input sequence as well.

Sec 5: eGTouchL.ini Parameter Explanations

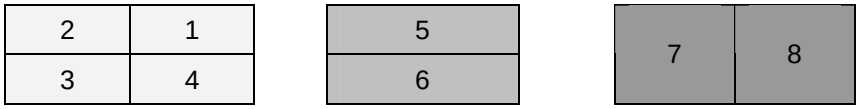
The file **eGTouchL.ini** has a parameter list which would be loaded by driver. Driver's behavior could be changed by these parameters. Please **DON'T** modify the front title as setting up eGTouchL.ini.

5-1 Parameter Table

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

◆ DebugEnableBits	Debug message you want to show.
0	Close all Debug
1	Print initialization debug message **[Default]**
FFFF	Open all Debug
◆ ShowDebugPosition	Position you want to show/store Debug message
0	Print in file located at /tmp **[Default]**
1	Print in terminal
2	Print in above both
◆ Baudrate	Choose the BaudRate

0 X	Auto detect Baudrate **[Default]** Set Baudrate to X bps. (PCAP: 57600 , Resis: 9600)
◆ SerialPath	RS232 Serial Path
default /dev/serial/ttyS0	Default path /dev/ttySX (X could be equals to 0-10) **[Default]** Customized path. Please type in your specific serial path accordinating to the form.
◆ SupportPoints	The amount of points you want to report (This is also confined by Controller)
0 1 >=2	No point Single-touch Multi-touch **[Default = 5]**
◆ Direction	Change the X and Y direction
0 1 2 3 4	Don't make any invert **[Default]** Invert X Invert Y Invert both X and Y Swap X and Y
◆ Orientation	Change the orientation
0 1 2 3	0 degree **[Default]** 90 degree 180 degree 270 degree
◆ EdgeCompensate	Do edge compensate
0 1	Disable **[Default]** Enable
EdgeLeft, EdgeRight EdgeTop, EdgeBottom	Edge compensate value
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 1	Disable **[Default]** 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 
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
◆ DetectRotation (Only for x86 system)	**Enable:** Driver would map its coordinate corresponding to X window rotation. ***Please see Sec 6-2.** **Disable:** If there's no roation requirement, just disable it.
0 1	Disable **[Default]** Enable
◆ ReportMode	Set different report type
1 2 3	Normal Mode. Report point normally. **[Default]** Click on Touch. Only report point as touch down. Click on Release. Only report point as touch up.
◆ BtnType	Set EETI protocol BtnType
0 1 2	Report single event as BTN_LEFT. **[Default]** Report single event as ABS_PRESSURE. (Generally for Tslib) Report single event as BTN_TOUCH.
◆ RightClickEnable	Report mouse Right Click after constant touch for a while
0 1	Disable Right Click Enable Right Click **[Default]**
RightClickDuration	Constant touch duration to trigger Right Click
X	X milliseconds **[Default = 1500]**
RightClickRange	Valid area of trigger-RightClick constant touch
X	±X range of the point would lead to constant touch for RightClick **[Min 0 - 50 Max] [Default = 10]**
◆ BeepState	Make a beep sound as touch ***Please see Sec 6-3.**
0 1 2 3	Disable Beep Make a beep sound as "Touch Down" Make a beep sound as "Touch Up" Make a beep sound as both two above conditions.
BeepDevice	Choose the beep sound device

0 1 2 3	No device Send beep sound by from system buzzer Send beep sound by from sound card (Only for x86 system) Send beep sound from both devices.
BeepFreq	You can modify buzzer beep frequency here.
X	(Only for buzzer) The buzzer beep frequency. **[Default = 1000]**
BeepLen	You can modify buzzer beep time length here.
X	(Only for buzzer) The buzzer beep time length (ms). **[Default = 200]**

Sec 6: Annotation

6-1 Kernel source patch

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.**

6-1-1 kernel 2.6.33 downwards

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

```
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 */
};

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,
};
```

6-1-2 kernel2.6.34 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,
};
```

6-2 DetectRotation Note

As DetectRotation is enabled, eGTouch driver have to be executed after X-server is ready.(We use Xlib to do detection). You have to remove the eGTouch execution in rc.local because it would not work out. Please manually put eGTouch execution in the sequence after OS's X server is ready.

We provides **gdm** solution since it's a general startup.

1. Modify the file "**Default**" under **/etc/gdm/Init**
2. Add eGTouch execution **/usr/bin/eGTouchD** at the end of file but before "**exit 0**"
3. Reboot system.

Since the ready time sequence of Xlib is different among diverse startup. We're sorry that we couldn't provide solution correspond to all startup. If there's any further problem as setting up please contact us for technical support.

6-3 Rotation and Beep for Embedded System

If you are using an embedded system (ex: ARM CPU), and you need support for rotation detection. There's a necessary condition: **Xrandr** lib support since eGTouch detect rotation event by Xrandr lib.

And so on. If you are using an embedded system (ex: ARM CPU), you need support for sound card beep. There's a necessary condition: **ALSA** lib support since eGTouch send beep sound by ALSA lib.

If you need this support and your system got target library, please contact us for a customized driver. Thanks.

Sec 7: 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.

7-1 Environment Information

Fill in this chart

1. CPU type	
2. Kernel version	
3. Linux Distribution version.	
4. If having X-window, X version? (X --version)	
5. If have no-X, what's your GUI system? (QT?)	
6. Touch Controller Interface	
7. Touch Controller Type	

7-2 Register input devices

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

7-3 Driver debug log

1. Execute eGTouchD with parameter like below:

   eGTouchD -D -f
2. Touch four corner of the touch panel
3. The log file would be saved as /tmp/eGTouchXXX
4. Send us the log file