PENMOUNT 触控控制器
LINUX / ANDROID
驱动来源码调适指南

Revision J

15/Aug/’23

PenMount®
TOUCH SCREEN SOLUTIONS

Preface

Disclaimer

The information in this document is subject to change without notice. The manufacturer makes no representations or warranties regarding the contents of this manual and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. Furthermore, the manufacturer reserves the right to revise this publication or make changes in the specifications of the product described within it at any time without notice and without obligation to notify any person of such revision.

Trademarks

AMT is the registered trademark of Apex Material Technology Corp. **PenMount** is a registered trademark of **SALT International Corp.** Microsoft and Windows are registered trademarks of Microsoft Corp. Other product names used in this manual are the properties of their respective owners and are acknowledged.

Copyright

Revision Table

Date	Revision	Changes
01/May/2019	A	第一版
02/Sep/2019	B	Add:为 PenMount 虚拟按键功能设置 hid-multitouch 内核驱动程序
20/Oct/2019	C	Add: Qt 支持
22/Jun/2020	D	Change: USB 界面设定 usbhid 驱动 hid_blacklist。
15/Jan/2021	E	加入 PenMount P3 相关内核支持说明。 更新了关于 Qt 设定的相关描述。
26/Mar/2021	F	调整 PenMount P3 为正式名称 PenMount K1。 调整驱动 Makefile 内容。
12/Oct/2021	G	更新了关于 Qt 设定的相关描述。
15/Jan/2022	H	更新了关于 Qt 使用 xcb 与 wayland 的设置描述。
17/Mar/2022	I	更新了關于 Qt 的手势支持说明。
26/Jul/2022	J	更新了关于使用 Qt5.6 搭配 libinput 的说明。
23/Aug/2022	K	更新了关于 ANDROID 定制驱动时，于内核 4.16 之后版本的 hid_ignore_list 设置说明。 增加了内核 6.X 的字段。 增加了 P2 HID over I2C 的字段。
02/May/2023	L	修正了**關於** KConfig 设定的误植
15/Aug/2023	J	增加了 5.2.1Android 驱动程序源码中对于边缘补偿的设定说明

Table of Content

1. 概述

在大部分情况下，触控功能都需仰赖内核驱动程序的支持。一般可由以下几点判断。

1. 内核版本
   下表中列出内核对于不同 PenMount 装置与接口的支持程度。灰色的部分代表需要使用 PenMount 提供的驱动程序。

Controller	Interface	Product ID	Kernel Version				
			2.6	3.X	4.X	5.X	6.X
PM9000	RS-232	---		V3.2 penmount			
PM6000	USB	0x6000	generic-usb	V3.5 hid-generic	V3.18 hid-penmount		
		0x6005	generic-usb	V3.5 hid-generic			
	RS-232	---		V3.2 penmount			
P2 / RMT	USB	0x3500		V3.0 hid-multitouch			
		0x3502	generic-usb	V3.5 hid-generic			
		0x3508		3.4 hid-multitouch			
	RS-232	---		V3.2 penmount			
	I2C	---					
	HID over I2C	0x3508		3.8 i2c-hid			
K1	USB	0x14E1	generic-usb (mouse mode)	3.4 hid-multitouch			
	HID over I2C	0x14E1		3.8 i2c-hid			

2. Linux Distribution
   - Red Hat Enterprise Linux / CENTOS 4：使用内核 V2.6.9，是唯一无法驱动 PenMount 6000 USB 的版本。因其 generic-usb 驱动判断问题，导致需额外修改后才能正常动作。
   - Red Hat Enterprise Linux / CENTOS 6：使用内核 V2.6.32。虽然标准的 Linux kernel V2.6.32 不支持 PenMount P2 USB，但因 RHEL6.6 版本后内建使用 hid-multitouch 驱动，因此可支持 PenMount P2 USB。[1]
   - Raspbian / SuSE Enterprise Server 12 SP3：该系统使用内核 4.X 的版本，但未包含 hid-penmount 驱动，导致 PenMount 6000 USB 无法动作的问题。

[1] 因 Xorg X Server 版本之故，PenMount P2 USB 会被 xinput evdev 驱动设定为错误的装置型态而发生动作异常的情况。

- CENTOS8：该系统使用内核 4.18 ，但预设不带 PenMount RS-232 内核驱动。以致在桌面系统下无法使用 RS-232 接口触摸屏。

3. 特殊韧体功能
   - Virtual Button : PenMount P2 USB V6.0 版本提供的 hot key 功能，需要搭配内核 V4.9 之后的版本。若使用之前的内核，需修改 hid-multitouch 驱动方可使用。

4. 搭配 APP 需求
   - Android APP：在 Android 系统上的动作基本上也视搭配的内核版本而定。但若需使用 APP 进行校准或其它设定，则需要事先安装额外的驱动程序。

2. USB HID 驱动程序设置

PenMount USB 接口在 V3.0 内核之后的版本已多数内建支持。可参阅下表：

<table>
<tr><th rowspan="2">Controller</th><th rowspan="2">Interface</th><th rowspan="2">USB Product ID</th><th colspan="5">Kernel Version</th></tr>
<tr><th>2.6</th><th>3.X</th><th>4.X</th><th>5.X</th><th>6.X</th></tr>
<tr><td rowspan="2">PM6000</td><td rowspan="2">USB</td><td>0x6000</td><td>generic-usb</td><td>V3.5
hid-generic</td><td colspan="3">V3.18
hid-penmount</td></tr>
<tr><td>0x6005</td><td>generic-usb</td><td colspan="4">V3.5
hid-generic</td></tr>
<tr><td rowspan="3">P2 / RMT</td><td rowspan="3">USB</td><td>0x3500</td><td></td><td colspan="4">V3.0
hid-multitouch</td></tr>
<tr><td>0x3502</td><td>generic-usb</td><td colspan="4">V3.5
hid-generic</td></tr>
<tr><td>0x3508</td><td></td><td colspan="4">3.4
hid-multitouch</td></tr>
<tr><td>K1</td><td>USB</td><td>0x14E1</td><td>generic-usb
(mouse mode)</td><td colspan="4">3.4
hid-multitouch</td></tr>
</table>

2.1. 启用内核驱动

下表中列出了各驱动程序与其对应的内核设置。启用时，可直接修改档案，或是透过 menuconfig 进行设定。

驱动程序	内核设置	说明
generic-usb	CONFIG_USB_HID=y	USB Human Interface Device (full HID) support
hid-generic	CONFIG_HID_GENERIC=m	Generic HID driver
hid-penmount	CONFIG_HID_PENMOUNT=m	Penmount touch device
hid-multitouch	CONFIG_HID_MULTITOUCH=m	HID Multitouch panels

注意事项：

- 如果在内建驱动支持的内核上触控无法动作，请先确认内核中相关的设置是否未被启用，以致驱动无法加载。
- 于内核 3.4 至 3.8，使用 USB PID 0x3508 的装置时，需设置系统于启动后执行加载 hid-multitouch 模块。

```
modprobe hid-multitouch
echo 3 14e1 3508 1 > /sys/bus/hid/drivers/hid-multitouch/new_id
```

2.2. 使用定制驱动

对于较旧的内核版本，需要额外的步骤修改设定以使用由 PenMount 提供的定制驱动。

项目	内容	
驱动程序	hid-penmount	
内核设置	CONFIG_HID_PENMOUNT=m	
程序代码	来源路径	<penmount_src_dir>/linux/driver/hid/hid-penmount.c
	复制至路径	<kernel_src_dir>/drivers/hid/hid-penmount.c
Kconfig 档案	路径	<kernel_src_dir>/drivers/hid/Kconfig
	新增内容	**config** HID_PENMOUNT tristate "Penmount touch device" **depends on** USB_HID **---help---** This selects a driver for the PenMount USB touch controller.
Makefile 档案	路径	<kernel_src_dir>/drivers/hid/Makefile
	新增内容	obj-$(CONFIG_HID_PENMOUNT) += hid-penmount.o

如果装置是 PenMount P2，需额外设定以下档案：

路径	档案	设定
<kernel_src_dir>/drivers/hid	hid.h (~ 2.6.32)	#define IS_INPUT_APPLICATION(a) (((a >= 0x00010000) && (a <= 0x00010008)) || (a == 0x00010080) || (a == 0x000c0001) || **((a >= 0x000d0002) && (a <= 0x000d0006))**)
	hid-ids.h	#define USB_VENDOR_ID_PENMOUNT 0x14E1 #define USB_DEVICE_ID_PENMOUNT_P2 0x3500 #define USB_DEVICE_ID_PENMOUNT_P2_WIN8 0x3508

	hid-core.c
	(~ 2.6.37)
		static const struct hid_device_id hid_blacklist[] = {
		// …
		{ HID_USB_DEVICE(USB_VENDOR_ID_PENMOUNT, USB_DEVICE_ID_PENMOUNT_P2) },
		{ HID_USB_DEVICE(USB_VENDOR_ID_PENMOUNT, USB_DEVICE_ID_PENMOUNT_P2_WIN8) },
		{ }
		}

	hid-core.c
	(2.6.38 ~)
		static const struct hid_device_id hid_have_special_driver[] = {
		// …
		{ HID_USB_DEVICE(USB_VENDOR_ID_PENMOUNT, USB_DEVICE_ID_PENMOUNT_P2) },
		{ HID_USB_DEVICE(USB_VENDOR_ID_PENMOUNT, USB_DEVICE_ID_PENMOUNT_P2_WIN8) },
		{ }
		}

2.3. 启用 Virtual Key 功能

由于 hid-multitouch 内核驱动的设计架构，搭配 PenMount P2 V5.2 ~ V6.0 韧体版本时需要先行调整程序代码后，使用重新编译的版本才能正常启用 Virtual Key 功能。

2.3.1. 内核 3.14 及之前的版本设置

请开启 hid-multitouch.c 档案，并找寻以下函数：

```
static int mt_input_mapping(struct hid_device *hdev, struct hid_input *hi,
                struct hid_field *field, struct hid_usage *usage,
                unsigned long **bit, int *max)
{
        /* Only map fields from TouchScreen or TouchPad collections.
        * We need to ignore fields that belong to other collections
        * such as Mouse that might have the same GenericDesktop usages. */
        if (field->application != HID_DG_TOUCHSCREEN &&
           field->application != HID_DG_PEN &&
           field->application != HID_DG_TOUCHPAD) {
                return -1;
        }
}
```

由于 Virtual Key 的 Top Level Collection Usage Page 属于 HID_UP_KEYBOARD，因此在原始搭配原始程序会在 mapping 时因回传 -1 而被系统的 hid-core 驱动程序忽略不去处理。建议修改为以下回传数值 0，使 hid-core 改以系统默认的方式来处理 Virtual Key events。

```
static int mt_input_mapping(struct hid_device *hdev, struct hid_input *hi,
                struct hid_field *field, struct hid_usage *usage,
                unsigned long **bit, int *max)
{
        /* Only map fields from TouchScreen or TouchPad collections.
        * We need to ignore fields that belong to other collections
        * such as Mouse that might have the same GenericDesktop usages. */
        if (field->application != HID_DG_TOUCHSCREEN &&
           field->application != HID_DG_PEN &&
           field->application != HID_DG_TOUCHPAD) {
                if (hi->input->id.vendor == USB_VENDOR_ID_PENMOUNT) {
                        return 0;
                } else {
                        return -1;
                }
        }
}
```

2.3.2. 内核 3.15 至 4.9 的版本设置

请开启 hid-multitouch.c 档案，并找寻以下数组：

```
static const struct hid_device_id mt_devices[] = {

        /* Generic MT device */
        { HID_DEVICE(HID_BUS_ANY, HID_GROUP_MULTITOUCH, HID_ANY_ID, HID_ANY_ID) },
```

```
	/* Generic Win 8 certified MT device */
	{  .driver_data = MT_CLS_WIN_8,
		HID_DEVICE(HID_BUS_ANY, HID_GROUP_MULTITOUCH_WIN_8,
			HID_ANY_ID, HID_ANY_ID) },
	{ }
};
```

请在 Generic MT device 项目前加上以下项目：

```
static const struct hid_device_id mt_devices[] = {

	/* PenMount devices */
	{ .driver_data = MT_CLS_EXPORT_ALL_INPUTS,
		MT_USB_DEVICE(USB_VENDOR_ID_PENMOUNT,
			0x3508) },

	/* Generic MT device */
	{ HID_DEVICE(HID_BUS_ANY, HID_GROUP_MULTITOUCH, HID_ANY_ID, HID_ANY_ID) },

	/* Generic Win 8 certified MT device */
	{  .driver_data = MT_CLS_WIN_8,
		HID_DEVICE(HID_BUS_ANY, HID_GROUP_MULTITOUCH_WIN_8,
			HID_ANY_ID, HID_ANY_ID) },
	{ }
};
```

2.3.3. 内核 4.10 及之后的版本设置

此版本之后的 hid-multitouch 内核驱动已能正常支持 PenMount Virtual Key 功能，因此不需要额外进行修改。

3. RS-232 驱动程序设置

PenMount USB 接口在 V3.2 内核之后的版本已多数内建支持。可参阅下表：

Controller	Interface	USB Product ID	Kernel Version				
			2.6	3.X	4.X	5.X	6.X
PM9000	RS-232	---		V3.2 penmount			
PM6000	RS-232	---		V3.2 penmount			
P2 / RMT	RS-232	---		V3.2 penmount			

3.1. 启用内核驱动

下表中列出了各驱动程序与其对应的内核设置。启用时，可直接修改档案，或是透过 menuconfig 进行设定。

驱动程序	内核设置	说明
serport	CONFIG_SERIO=y	Serial I/O support
	CONFIG_SERIO_SERPORT=m	Serial port line discipline
serio-penmount	CONFIG_TOUCHSCREEN_PENMOUNT=m	Penmount serial touchscreen

3.2. 使用定制驱动

对于较旧的内核版本，则需要额外的步骤修改设定以使用由 PenMount 提供的定制驱动。该驱动与内核驱动名称相同，可直接进行置换

项目	内容	
驱动程序	serio-penmount	
内核设置	CONFIG_TOUCHSCREEN_PENMOUNT=m	
程序代码	来源路径	<penmount_src_dir>/linux/driver/serial/penmount.c
	复制至路径	<kernel_src_dir>/drivers/input/touchscreen/penmount.c
Kconfig 档案	路径	<kernel_src_dir>/drivers/input/touchscreen/Kconfig
	新增内容	使用暨有内容设定
Makefile 档案	路径	<kernel_src_dir>/drivers/input/touchscreen/Makefile
	新增内容	使用暨有内容设定

3.3. 加载内核驱动

对于 RS-232 驱动而言，同一个档案支持 PM9000 / PM6000 / P2，因此需要靠额外的程序加载并设定装置型态。

目前有两种程序可支持加载，请依需求选择其一进行编译，并设定系统于开机时执行之。

程序	Source Code 路径	范例
pmsAttach	<penmount_src_dir>/linux/driver/serial/pmsAttach	pmsAttach 9000 /dev/ttyS1 19200
		pmsAttach 6000 /dev/ttyS1 19200
		pmsAttach PCI /dev/ttyS1 38400
inputattach	https://github.com/flosse/linuxconsole/	inputattach –baud 19200 –pm9k /dev/ttyS1
		inputattach –baud 19200 –pm6k /dev/ttyS1
		inputattach –baud 38400 –pm3k /dev/ttyS1

4. I2C 驱动程序设置

对于 I2C 接口而言，PenMount P2 需使用提供的定制内核模块。而 PenMount K1 则使用内核标准 i2c-hid 模块。

Controller	Interface	Product ID	Kernel Version				
			2.6	3.X	4.X	5.X	6.X
P2 / RMT	I2C	---					
	HID over I2C	0x3508		V3.8 i2c-hid			
K1	HID over I2C	---		V3.8 i2c-hid			
	I2C	ili251x				V5.1 Ili210x	

4.1. 启用内核驱动

下表中列出了各驱动程序与其对应的内核设置。启用时，可直接修改档案，或是透过 menuconfig 进行设定。

驱动程序	内核设置	说明
i2c-hid	CONFIG_I2C_HID=m	HID over I2C transport layer

4.2. 使用定制驱动

对于 PenMount I2C 接口装置而言，需要修改设定以使用由 PenMount 提供的定制驱动。

项目	内容	
驱动程序	penmount_i2c	
内核设置	CONFIG_TOUCHSCREEN_PENMOUNT_I2C=m	
程序代码	来源路径	<penmount_src_dir>/linux/driver/i2c/penmount-i2c.c
	复制至路径	<kernel_src_dir>/drivers/input/touchscreen/penmount-i2c.c
Kconfig 档案	路径	<kernel_src_dir>/drivers/input/touchscreen/Kconfig
	新增内容	**config** PENMOUNT_TOUCHSCREEN_I2C tristate "Penmount I2C touchscreen" **depends on** I2C **---help---** This selects a driver for the PenMount I2C touch controller.
Makefile 档案	路径	<kernel_src_dir>/drivers/input/touchscreen/Makefile
	新增内容	obj-$(CONFIG_TOUCHSCREEN_PENMOUNT_I2C) += penmount-i2c.o

4.3. 中断设置

对于 PenMount I2C 接口装置而言，可以将其 INT PIN 连接至系统的 GPIO，使驱动能在触控时实时读取坐标值。

由于各系统对于 GPIO 的设置方式不大相同，因此这部分需由系统整合厂商依实际设置作修改。可以选择以下其中一种方式进行：

方式	项目	说明	范例
直接修改驱动程序	PENMOUNT_I2C_GPIO_IRQ 定义	设定 INT PIN 对应之 GPIO	#define GPIO_TO_PIN(bank, gpio) (32 * (bank) + (gpio)) GPIO_TO_PIN(3,19)
	penmount_i2c_init_gpio () 函式	设置 GPIO 为 pull high，同时由 low level 触发	
使用 DeviceTree 配置文件	interrupt-parent interrupts	设置 GPIO 为 low level 触发	interrupt-parent = <&gpio3>; interrupts = <19 8>; /* IRQ_TYPE_LEVEL_LOW */

4.4. 加载内核驱动

对于 I2C 接口而言，需要额外设定系统，才能加载驱动程序。以下以几种 硬件平台为例进行说明。

方式	项目	档案路径（范例）	设置（范例）

直接修改系统设定	i2c_board_info[]	(S3C2440) <kernel_src_dir>/kernel/arch/arm/mach-s3c2440/mach-s3c2440.c	找到 friendly_arm_i2c_devices[] 并新增项目如下：
			#include <linux/i2c.h>
			
			static struct i2c_board_info friendly_arm_i2c_devices[] __initdata = {
			{ **I2C_BOARD_INFO("penmount_i2c", 0x38)** },
			};
			
			static void __init mini2440_machine_init(void) {
			i2c_register_board_info(0, friendly_arm_i2c_devices, ARRAY_SIZE(friendly_arm_i2c_devices));
			s3c_i2c0_set_platdata(NULL);
			}

使用 DeviceTree 配置文件	compatible reg	PenMount P2	compatible = "penmount,penmount_i2c"; reg = <0x38>; interrupt-parent = <&gpio>; interrupts = <24 8>;
		PenMount K1	compatible = "hid-over-i2c"; reg = <0x41>; hid-descr-addr = <0x0001>; interrupt-parent = <&gpio>; interrupts = <24 8>;

5. ANDROID 驱动程序设置

5.1. 使用定制驱动

对于 ANDROID 系统而言，若需搭配电阻式校准 APP，需要搭配特有的 PenMount ANDROID 定制驱动方可使用。ANDROID 版本的驱动包含多个档案，提供了 USB / RS-232 / I2C 接口的支持。

<table>
<tr><th>项目</th><th colspan="3">内容</th></tr>
<tr><td rowspan="3">驱动程序</td><td>USB</td><td colspan="2">usb-penmount</td></tr>
<tr><td>RS-232</td><td colspan="2">serio-penmount</td></tr>
<tr><td>I2C</td><td colspan="2">penmount_i2c</td></tr>
<tr><td>内核设置</td><td colspan="3">CONFIG_PENMOUNT_TOUCHSCREEN=m</td></tr>
<tr><td rowspan="6">程序代码</td><td rowspan="5">来源路径</td><td rowspan="5"><penmount_src_dir>/android/driver/</td><td>penmount-module.c</td></tr>
<tr><td>penmount-serio.c</td></tr>
<tr><td>penmount-usb.c</td></tr>
<tr><td>penmount-i2c.c</td></tr>
<tr><td>penmount.h</td></tr>
<tr><td>复制至路径</td><td colspan="2"><kernel_src_dir>/drivers/input/touchscreen</td></tr>
<tr><td rowspan="2">Kconfig 档案</td><td>路径</td><td colspan="2"><kernel_src_dir>/drivers/input/touchscreen/Kconfig</td></tr>
<tr><td>调整内容</td><td colspan="2">请找到以下 config 并调整增加 depends on：
config TOUCHSCREEN_PENMOUNT
tristate "Penmount touchscreen"
depends on USB
depends on SERIO
depends on I2C
---help---
This selects a driver for the PenMount touch controller.</td></tr>
<tr><td rowspan="2">Makefile 档案</td><td>路径</td><td colspan="2"><kernel_src_dir>/drivers/input/touchscreen/Makefile</td></tr>
<tr><td>新增内容</td><td colspan="2">原本有一行：
obj-$(CONFIG_TOUCHSCREEN_PENMOUNT)+= penmount.o
请再新增一行：
penmount-objs += penmount-module.o penmount-usb.o penmount-serio penmount-i2c</td></tr>
</table>

如果装置是 PenMount USB 接口，需额外设定以下档案：

<table>
<tr><th>路径</th><th>档案</th><th>设定</th></tr>
<tr><td rowspan="2"><kernel_src_dir>/drivers/hid/</td><td>hid-ids.h</td><td>#define USB_VENDOR_ID_PENMOUNT 0x14E1
#define USB_DEVICE_ID_PENMOUNT_6000 0x6000</td></tr>
<tr><td>（内核 4.15 及之前）
usbhid/hid-quirks.c</td><td>在hid_blacklist数组中添加新条目，请在第一行加入如下方粗体项目：
static const struct hid_blacklist {
__u16 idVendor;</td></tr>
</table>

		__u16 idProduct;
		__u32 quirks;
		} hid_blacklist[] = {
		{ USB_VENDOR_ID_PENMOUNT,
		USB_DEVICE_ID_PENMOUNT_6000,
		HID_QUIRK_IGNORE },
		// …
		{ USB_VENDOR_ID_AASHIMA,
		USB_DEVICE_ID_AASHIMA_GAMEPAD,
		HID_QUIRK_BADPAD },
		}

	（内核 4.16 及之后）
	hid-quirks.c
		在hid_ignore_list数组中添加新条目，请在第一行加入如下方粗体项目：
		
		static const struct hid_device_id hid_ignore_list[] = {
		{ HID_USB_DEVICE(USB_VENDOR_ID_PENMOUNT,
		USB_DEVICE_ID_PENMOUNT_6000) },
		// ….
		{ }
		};

若您建置的系统，dmesg 出现类似以下消息，代表上述 hid_blacklist 未设定正确。请确实设定使系统使用 PenMount USB 驱动。

```
<6>[    1.795837] usb 1-1.4: New USB device found, idVendor=14e1, idProduct=6000
<6>[    1.795855] usb 1-1.4: New USB device strings: Mfr=1, Product=2, SerialNumber=0
<6>[    1.795868] usb 1-1.4: Product: PenMount USB
<6>[    1.795878] usb 1-1.4: Manufacturer: DIALOGUE INC
<6>[    1.799029] input: DIALOGUE INC PenMount USB as /devices/ff500000.usb/usb1/1-1/1-1.4/1-1.4:1.0/input/input3
<6>[    1.799512] hid-generic 0003:14E1:6000.0002: input,hidraw1: USB HID v1.01 Mouse [DIALOGUE INC PenMount USB] on usb-ff500000.usb-1.4/input0
```

5.2. 调整驱动源码设置

PenMount Android 驱动源码提供参数值设定，可依情况动态设置，或是于系统建置阶段给予适当的默认值。

5.2.1. 边缘补偿设定

当发生画线无法达到屏幕边缘的情况，可依需求调整源码中相关定义与变数的设置。在 Android 系统运作时，亦可动态调整位于/sys/module/penmount/parameters/目录下的参数。

名称	参数	作用
PMDRIVER_CALIB_EDGE_MAX		边缘补偿启动的范围。默认值为 6，表示距边六分之一位置以内会作用。

g_EdgeAdjustLeft	`EdgeLeft`	左边缘的补偿设定。驱动程序会依数值调整触摸位置使其更接近边缘处。
g_EdgeAdjustRight	`EdgeRight`	右边缘的补偿设定。驱动程序会依数值调整触摸位置使其更接近边缘处。
g_EdgeAdjustTop	`EdgeTop`	上边缘的补偿设定。驱动程序会依数值调整触摸位置使其更接近边缘处。
g_EdgeAdjustBottom	`EdgeBottom`	下边缘的补偿设定。驱动程序会依数值调整触摸位置使其更接近边缘处。

6. 电阻式触控定位校准

使用传统电阻式触控时需要进行定位后，触控坐标才能达到一定的准确度。依据使用的是 Linux 或 Android 系统有两种校准工具可以使用。

6.1. LINUX 系统触控校准

在 Linux 系统下的定位程序有很多种，最常见的是使用 tslib 提供的校准功能。

6.1.1. 设定编译环境

tslib 在常用的 Linux distribution，例如 Debian / CENTOS / openSUSE / Ubuntu 中都提供编译好的二进制文件可供直接安装使用。

若您需要在订制的 Linux 系统中使用 tslib 触控校准功能，则需要利用源码进行编译。目前 tslib 来源码可自以下两处取得，两种版本是兼容的。

版本	取得方式
官方版本	由 tslib 网页下载，需使用 1.1 以后版本。
PenMount 订制版本	由 PenMount 提供。

- 编译所需套件：

套件名称	安装（Ubuntu）
automake	sudo apt-get install automake
autoconf	sudo apt-get install autoconf
libtool	sudo apt-get install libtool
gcc	请洽询您使用主板提供商取得相关 toolchain。

若编译使用的系统（platform）与程序预计执行的系统（xplatform）两者不同时，需要进行额外的交叉编译设定。

- 编译步骤

步骤	指令范例	说明
切换至 tslib source code 目录	cd linux/tslib	
清除旧编译设定	./autogen-clean.sh	
产生 configure 档案	./autogen.sh	
产生 Makefile 档案	./configure \	
	CC=**arm-linux-gnueabihf-gcc-4.7** \	使用 armhf 编译程序
	--host=**arm-linux-gnueabihf** \	交叉编译为 armhf 平台下使用之二进制

		文件
	--prefix=/usr/local/penmount/tslib \	预计安装于 /usr/local/penmount/tslib 目录下
	--enable-debug	开启 debug 选项
编译来源码	make	
将编译完成之二进制文件复制至目录	sudo make install	执行后可在 /usr/local/penmount/tslib 目录下取得编译完成之 armhf 系统使用的 tslib 二进制文件。 请将该目录档案复制至目标系统中。

6.1.2. 设定执行环境

执行 tslib 需要妥善设定环境变量，一般需要的设定如下：

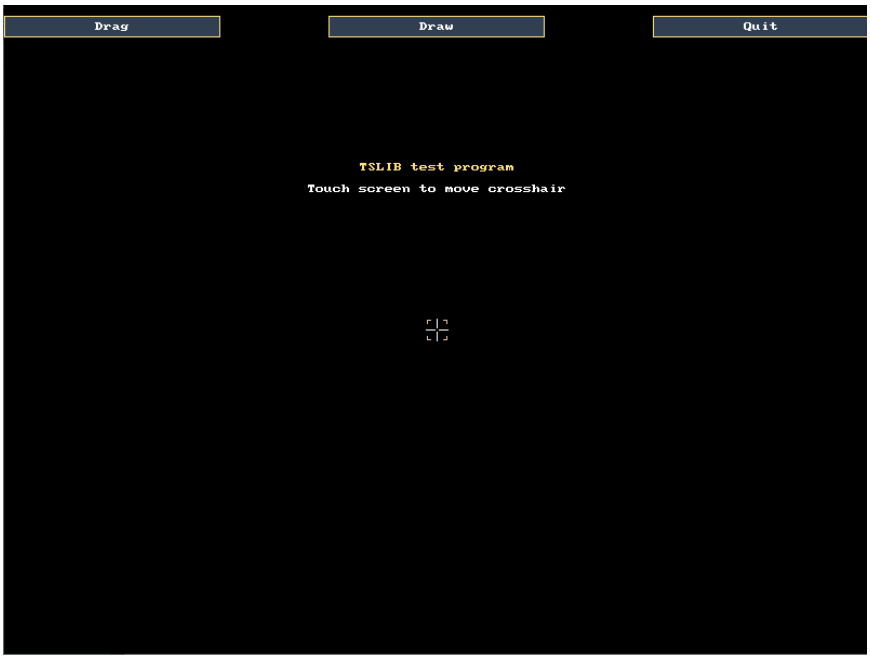
设定	范例	说明
TSLIB_DIR	export TSLIB_DIR=/usr/local/tslib	tslib 所在之目录。 在此假设上一步骤编译完成之档案复制至目标系统的 /uts/local/tslib 目录。
TSLIB_CONFFILE	export TSLIB_CONFFILE=$TSLIB_DIR/etc/ts.conf	tslib 配置文件之所在路径。
LD_LIBRARY_PATH	export LD_LIBRARY_PATH=$TSLIB_DIR/lib	tslib shared library 之所在路径。
TSLIB_TSDEVICE	export TSLIB_TSDEVICE=**/dev/input/event4**	触控装置名称。 **由于装置名称可能会依实际情况变动。因此建议使用 PenMount 版本 tslib，会自动找寻装置。可省略此设定步骤。**
TSLIB_PLUGINDIR	export TSLIB_PLUGINDIR=$TSLIB_DIR/lib/ts	tslib plugin 之所在路径。
TSLIB_CALIBFILE	export TSLIB_CALIBFILE=/etc/pointercal	执行 tslib 触控校准存取定位值之档案路径。

6.1.3. 执行触控校准

tslib 的校准程序名称为 ts_calibrate。请执行后依屏幕画面提示点击五个校准点，结束后程序会生成 pointercal 档案。

6.1.4. 测试校准效果

tslib 的测试程序名称为 ts_test。可用于测试 执行完成 ts_calibrate 校准程序后的效果。



若您未连接装置执行时，需要手动输入指令才能关闭该程序：

sudo killall ts_test

6.2. ANDROID 系统触控校准

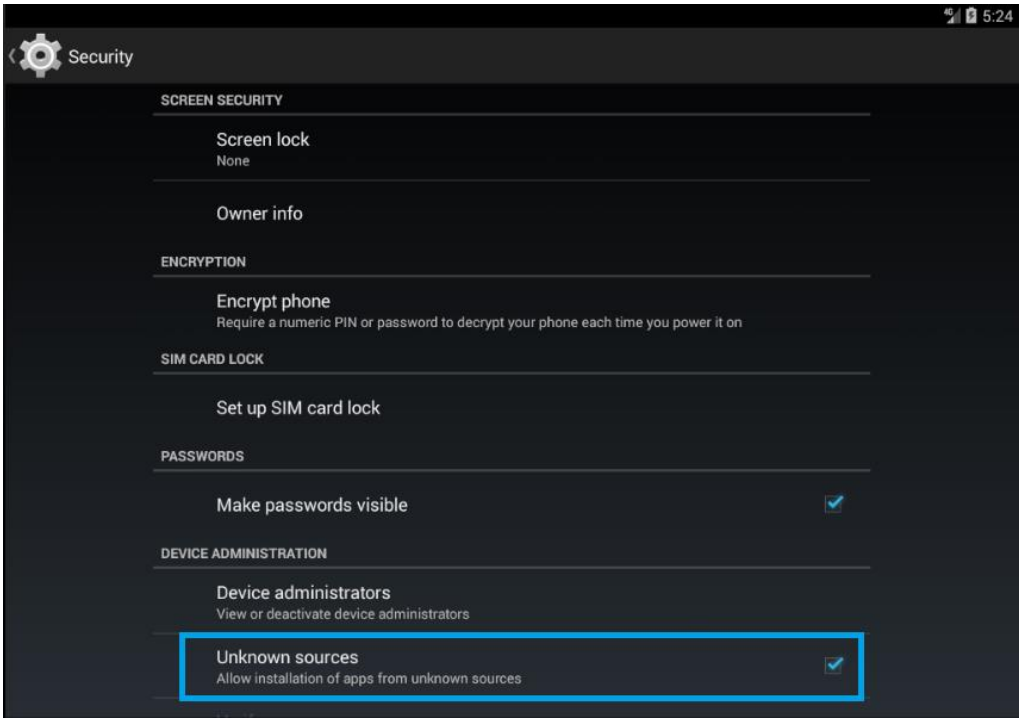
在 ANDROID 系统中，PenMount 提供订制的校准 APP 位于以下路径。

android/calibration/pmCalib2.apk

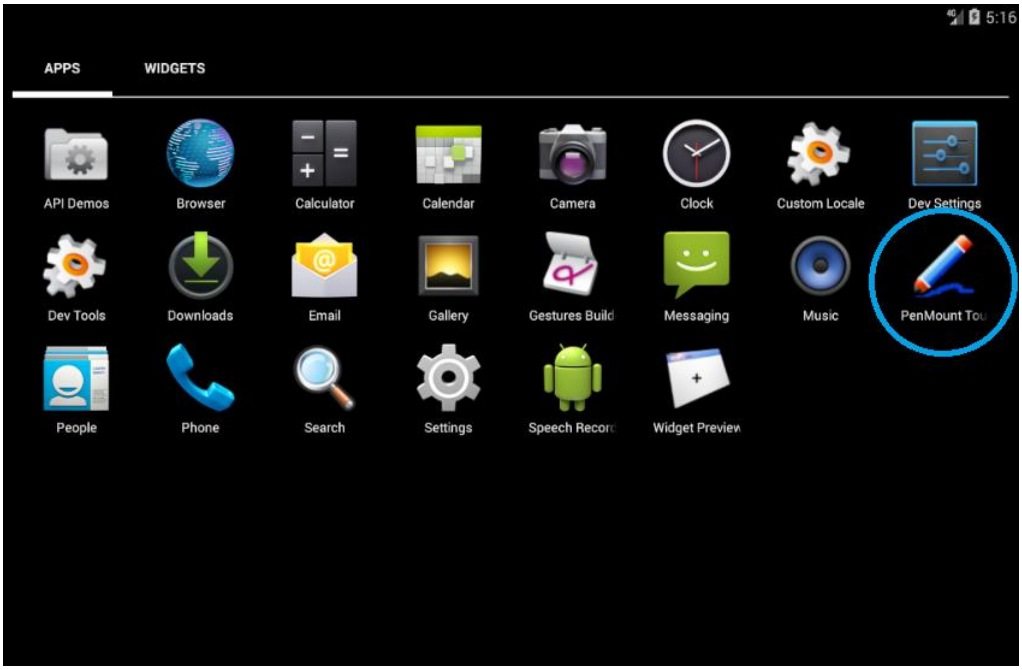
6.2.1. 安装

安装校准程序前需要先启用系统允许安装“Unknown sources”。

请至“Settings”=> “【Personal】 Security”=> “【DEVICE ADMINISTRATION】”，并勾选“Unknown sources”。



点击并安装 pmCalib2.apk 后，会出现以下“PenMount TouchScreen Calibration”图示。



6.2.2. 设定权限

Android 5 及之后的版本中 SELinux 若设置为 enforcing，会造成校准程序失去作用。

- 检查方式

若不确定系统是否设置了 SELinux enforcing，可以在校准程序结束后，自后台取得 logcat，其中会有类似下的“avc: denied”讯息：

```
06-16 20:00:15.693 W/com.penmount( 1401): type=1400 audit(0.0:25): avc: denied { read } for name="version" dev="proc" ino=4026531943 scontext=u:r:untrusted_app:s0:c512,c768 tcontext=u:object_r:proc:s0 tclass=file permissive=0
```

- 对策

请在执行校准程序前，于后台暂时将 SELinux 设置为 permissive 模式。

```
setenforce 0
```

6.2.3. 设定选项

程序启动后的接口如下图。

各设定说明如下。

选项	说明	数值	批注
Calibration Mode	定位点个数	5：5 点定位	
		9：9 点定位	**默认模式**
		16：16 点定位	
Calibration Offset	定位点内缩比例	2：内缩 2%	部分可能会被 Navigation Bar 遮蔽
		5：内缩 5%	

		N：内缩 N%	**默认值。一般为 7%。 该数值依 Navigation Bar 与屏幕比例调整**
		10：内缩 10%	
		15：内缩 15%	
Output Debug Message	输出更多 Debug 讯息		可由 logcat 指令取得 Debug 讯息

6.2.4. 执行校准

按下“Calibrate”按钮后变会启动校准程序，请依序点击程序显示的定位点，并依照提示的说明文字完成校准。

由于校准程序是将校准值储存于装置中，因此本程序仅支持以下韧体版本。

装置	韧体版本
PM9000	C2.3
	C2.4
	E2.2
	E2.3
PM6000	6000.3
	6000.6
	6010.1

7. Qt 设置

Qt 是 C++应用程序开发框架，主要常用于开发嵌入式系统的图形接口。目前常用的 Qt 有 4 与 5 两个版本，两者对于触摸屏的支持度不大相同，可以表列如下：

Plugin	configure	Qt4	Qt 5.0 ~ 5.5	QT5.6 ~ Qt6	支援电阻屏校准	支援电容屏多指触摸
tslib	-tslib	V	V	V	V	
evdevtouch	-evdev		**V**	V		V
libinput	-libinput			**V**	V	V

对于电阻触摸屏需要额外的校准，因此在 Qt 框架上需要搭配额外的 tslib 或 libinput 设置才能动作正常。

7.1. 重新编译 Qt

一般嵌入式系统提供的 Linux SDK 会包含 Qt 库。然而有些为节省空间，未包含触摸屏支持，此时就需要自行以源码重新编译带有支持的版本。

确认触摸屏支持

以下列举常用的 Qt 版本源码下载连结作为参考。

Qt 版本	源码下载连结
4.6	https://download.qt.io/archive/qt/4.6/qt-everywhere-opensource-src-4.6.4.tar.gz
4.7	https://download.qt.io/archive/qt/4.7/qt-everywhere-opensource-src-4.7.4.tar.gz
4.8	https://download.qt.io/archive/qt/4.8/4.8.7/qt-everywhere-opensource-src-4.8.7.tar.gz
5.0	https://download.qt.io/new_archive/qt/5.0/5.0.2/single/qt-everywhere-opensource-src-5.0.2.tar.xz
5.1	https://download.qt.io/new_archive/qt/5.1/5.1.1/single/qt-everywhere-opensource-src-5.1.1.tar.xz
5.2	https://download.qt.io/new_archive/qt/5.2/5.2.1/single/qt-everywhere-opensource-src-5.2.1.tar.xz
5.3	https://download.qt.io/new_archive/qt/5.3/5.3.2/single/qt-everywhere-opensource-src-5.3.2.tar.xz
5.4	https://download.qt.io/new_archive/qt/5.4/5.4.2/single/qt-everywhere-opensource-src-5.4.2.tar.xz
5.5	https://download.qt.io/new_archive/qt/5.5/5.5.1/single/qt-everywhere-opensource-src-5.5.1.tar.xz
5.6	https://download.qt.io/new_archive/qt/5.6/5.6.3/single/qt-everywhere-opensource-src-5.6.3.tar.xz
5.7	https://download.qt.io/new_archive/qt/5.7/5.7.1/single/qt-everywhere-opensource-src-5.7.1.tar.xz
5.8	https://download.qt.io/new_archive/qt/5.8/5.8.0/single/qt-everywhere-opensource-src-5.8.0.tar.xz
5.9	https://download.qt.io/archive/qt/5.9/5.9.9/single/qt-everywhere-opensource-src-5.9.9.tar.xz
5.12	https://download.qt.io/archive/qt/5.12/5.12.11/single/qt-everywhere-src-5.12.11.tar.xz
5.15	https://download.qt.io/archive/qt/5.15/5.15.2/single/qt-everywhere-src-5.15.2.tar.xz
6.0	https://download.qt.io/archive/qt/6.0/6.0.4/single/qt-everywhere-src-6.0.4.tar.xz
6.1	https://download.qt.io/archive/qt/6.1/6.1.3/single/qt-everywhere-src-6.1.3.tar.xz
6.2	https://download.qt.io/archive/qt/6.2/6.2.4/single/qt-everywhere-src-6.2.4.tar.xz
6.3	https://download.qt.io/archive/qt/6.3/6.3.2/single/qt-everywhere-src-6.3.2.tar.xz
6.4	https://download.qt.io/archive/qt/6.4/6.4.3/single/qt-everywhere-src-6.4.3.tar.xz
6.5	https://download.qt.io/archive/qt/6.5/6.5.2/single/qt-everywhere-src-6.5.2.tar.xz

7.1.1. 编译 Qt4

Qt4 的 tslib mouse driver 能支援 触摸屏。若有以下两种情况是需要重新编译 Qt4。

1. 系统的 tslib 库版本过于老旧，不支持 PenMount 触摸屏。
2. 使用的 Qt4 不包含 tslib 支持，

编译 Qt4 时需指定 tslib library 与头文件的路径。以下为于 x86_64 系统中编译 ARM 版本的 Qt4 所进行的设置范例。在此以定制的环境变量表示相关路径，需要依实际情况定义这些档案路径： TSLIB_DIR 与 QTDIR 表示：

- QTDIR：编译完成 Qt Library ， make install 时的安装路径。
- $ TSLIB_DIR/include：tslib 头文件路径。
- $ TSLIB_DIR/lib ： tslib 库路径。

```
./configure \
-prefix $QTDIR \
-opensource \
-confirm-license \
-release -shared \
-embedded arm \
-platform linux-g++-64 \
-xplatform linux-arm-gnueabihf-g++ \
-depths all \
-fast \
-optimized-qmake \
-pch \
-qt-sql-sqlite \
-qt-libjpeg \
-qt-zlib \
-qt-libpng \
-qt-freetype \
-little-endian -host-little-endian \
-no-qt3support \
-no-libtiff -no-libmng \
-no-opengl \
-no-mmx -no-sse -no-sse2 -no-sse3 \
-no-ssse3 -no-sse4.1 -no-sse4.2 \
-no-3dnow \
-no-openssl \
-no-webkit \
-no-qvfb \
-no-phonon \
-no-nis \
-no-opengl \
-no-cups \
-no-glib \
-no-xcursor -no-xfixes -no-xrandr -no-xrender \
-no-separate-debug-info \
-nomake examples -nomake tools -nomake docs -nomake demos \
-qt-gfx-linuxfb \
-qt-gfx-transformed \
-qt-mouse-tslib \
-qt-mouse-linuxinput \
-qt-kbd-linuxinput \
-I$TSLIB_DIR/include \
```

-L$TSLIB_DIR/lib
make
sudo make install

其它的设置可参考以下连结：

https://doc.qt.io/archives/qt-4.8/configure-options.html

7.1.2. 编译 Qt5 / Qt6

Qt5 的 tslib 与 Qt5.6 开始的 libinput 提供了电阻触摸屏校准支持。请先检视您使用的 Qt5 是否包含以下档案：

Plugin	档案路径	校准工具。
libinput	$QT_PLUGINS_DIR/generic/libqlibinputplugin.so	pm-qcalib
tslib	$QT_PLUGINS_DIR/generic/libqtslibplugin.so	ts_calibrate

以 Ubuntu 20.04 系统为例，默认的 Qt5 plugins 路径为 /usr/lib/x86_64-linux-gnu/qt5/plugins。只需其中之一即可搭配电阻屏工具进行校准。

若使用的 Qt5 未包含上述支持，则需重新编译 Qt5，设置方式与 Qt4 有些许不同，也许要指定对应的 tslib 路径。

```
./configure \
-prefix $QTDIR \
-opensource \
-confirm-license \
-release -shared \
-platform linux-g++-64 \
-xplatform linux-arm-gnueabihf-g++ \
-optimized-qmake \
-pch \
-qt-sql-sqlite \
-qt-libjpeg \
-qt-zlib \
-qt-libpng \
-qt-freetype \
-no-opengl \
-no-openssl \
-no-cups \
-no-glib \
-skip webkit \
-skip webkit-examples \
-nomake examples \
-nomake tests \
-no-xcursor -no-xfixes -no-xrandr -no-xrender \
-no-separate-debug-info \
-tslib \
-I$TSLIB_DIR/include \
-L$TSLIB_DIR/lib
```

```
make
```

```
sudo make install
```

其它的设置可参考以下连结：

https://wiki.qt.io/Building_Qt_5_from_Git#Configuring_and_Building

7.2. 电阻触摸屏设置

如未特别设置，Qt 会使用默认的 Input Handler 来驱动触摸屏，其中只有 libinput 提供校准机制。电阻式触摸屏在校准之前皆可能会发生触摸位置不准确的现象，因此对大部分电阻屏而言都需要额外的设置来进行校准。

Qt 版本	预设 Handler
Qt 4	pc
Qt 5.0 ~ 5.5	evdevtouch
Qt 5.6 ~ 6	libinput

7.2.1. Qt4 设置 tslib 范例

以下以设置 tslib mouse plugin 为范例说明。

请注意使用 tslib mouse plugin 时，还需额外设置 tslib 的相关路径设定。

項目	设置	说明与范例
tslib	TSLIB_TSDEVICE	触摸屏的 event 装置路径。 export **TSLIB_TSDEVICE** =/dev/input/event4
	TSLIB_CALIBFILE	（可选）触摸屏的校准数据档案路径。 export **TSLIB_CALIBFILE** =/etc/pointercal
	TSLIB_CONFFILE	（可选）触摸屏的设置档案路径。 export **TSLIB_CONFFILE**=/etc/ts.conf ts.conf 档案建议为 input **加上** grab_events 设置： module_raw input **grab_events=1**
Qt4	QWS_MOUSE_PROTO	指定 Qt4 使用的 Mouse Plugin 名称。 export **QWS_MOUSE_PROTO**=tslib:/dev/input/event4

7.2.2. Qt5 / Qt6 设置 tslib 范例

若使用 Qt 5.6 之后的版本，预设的 libinput 即可直接搭配 pm-qcalib 程序校准电阻式触摸屏。

若为需使用 tslib 搭配 ts_calibrate 定位程序，则需参考本节内容设置包含 tslib 与 Qt。主要项目如下，需依实际情况调整数值。以下范例中假设 PenMount input 装置名称为 /dev/input/event4。

平台	设置	说明与范例	注
x11	QT_QPA_PLATFORM	export QT_QPA_PLATFORM=xcb	预设使用 X Window **驅動**，使用 PenMount X Window 驱动与校准程序。不需额外设置 tslib。[2]
wayland	QT_QPA_PLATFORM	export QT_QPA_PLATFORM=wayland	预设使用 libinput，校准程序可搭配 weston-calibrator。**不需额外设置 tslib。**
eglfs	QT_QPA_PLATFORM	export QT_QPA_PLATFORM=eglfs	
	QT_QPA_EGLFS_NO_LIBINPUT	**（Qt5.6 ~ 6 需设定）**设定不使用预设的 libinput。 export QT_QPA_EGLFS_NO_LIBINPUT=1	
	QT_QPA_EGLFS_TSLIB	**（Qt5.5~才支持）**指定 input handler 使用 tslib **來處理** input。 export QT_QPA_EGLFS_TSLIB=1	二择一设置即可。
	QT_QPA_GENERIC_PLUGINS	**（Qt5.0 即支持）**指定加载 tslib plugin。 export QT_QPA_GENERIC_PLUGINS=tslib:/dev/input/event4 需求档案：	

[2] 在 xcb platform 下若设置 QT_QPA_GENERIC_PLUGINS=tslib 会发生 Qt 程序收到重复输入事件的情况，造成动作异常。请检查并移除设置。

		$QT_PLUGIN_DIR/plugins/generic/libqtslibplugin.so	
linuxfb	QT_QPA_PLATFORM	export QT_QPA_PLATFORM=linuxfb	
	QT_QPA_FB_NO_LIBINPUT	**（Qt5.6 ~ 6 需设定）**设定不使用预设的 libinput。 export QT_QPA_FB_NO_LIBINPUT=1	
	QT_QPA_FB_TSLIB	**（Qt5.5~才支持）**指定 input handler 使用 tslib **來處理** input。 export QT_QPA_FB_TSLIB=1	二择一设置即可。
	QT_QPA_GENERIC_PLUGINS	**（Qt5.0 即支持）**指定加载 tslib plugin。 export QT_QPA_GENERIC_PLUGINS=tslib:/dev/input/event4 需求档案： $QT_PLUGIN_DIR/plugins/generic/libqtslibplugin.so	

对于需要使用 tslib 校准程序的平台，需额外设置如下：

項目	设置	说明与范例	注
tslib	TSLIB_TSDEVICE	触摸屏的 event 装置路径。 export TSLIB_TSDEVICE =/dev/input/event4	
	TSLIB_CALIBFILE	（可选）触摸屏的校准数据档案路径。 export **TSLIB_CALIBFILE** =/etc/pointercal	
	TSLIB_CONFFILE	（可选）触摸屏的设置档案路径。 export **TSLIB_CONFFILE**=/etc/ts.conf ts.conf 档案建议为 input **加上** grab_events 设置： module_raw input **grab_events=1**	

更多的设定方式可参考以下连结：

https://doc.qt.io/qt-5/embedded-linux.html

7.3. 手势支援

Qt 自版本 4.6 开始支持触摸屏多指手势。可参考官方网站与范例的说明：

https://doc.qt.io/qt-5/gestures-overview.html

7.3.1. 手势类别

类型	Qt::GestureType	说明	头文件	指数
Tap	Qt::TapGesture	单击	QTapGesture	1
Tap and Hold	Qt::TapAndHoldGesture	久压	QTapAndHoldGesture	1
Pan	Qt::PanGesture	平移	QPanGesture	2+
Pinch	Qt::PinchGesture	缩放 旋转	QPinchGesture	2
Swipe	Qt::SwipeGesture	滑动	QSwipeGesture	3

7.3.2. 启用手势支援

Qt 软件启动时，输入设备需以能产生 Touch 事件的模块驱动，例如 libinput。同时使用前软件需呼叫 grabGesture()，不然软件会无法收到手势事件。

```
grabGesture(Qt::GestureType::PanGesture);
grabGesture(Qt::GestureType::PinchGesture);
grabGesture(Qt::GestureType::SwipeGesture);
```

7.3.3. 处理手势事件

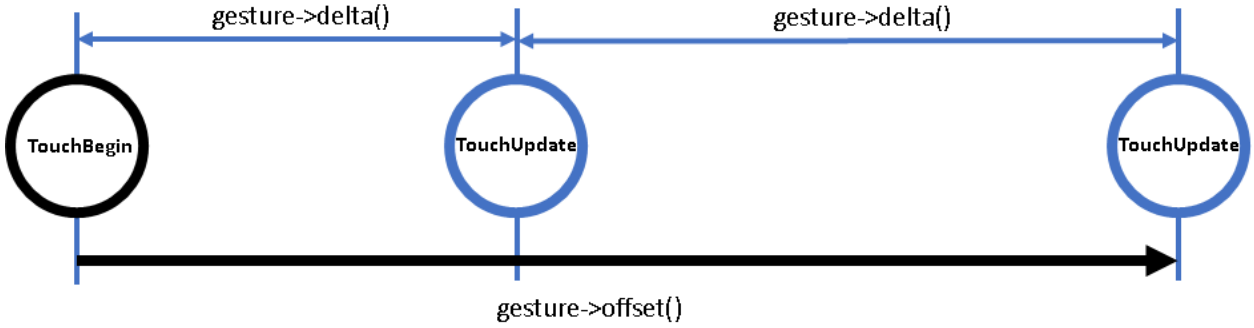
软件的事件处理函式可由以下方式判断是否为手势事件。

```
#include <QGestureEvent>

if (event->type() == QEvent::Gesture) {
        QGestureEvent* gesture = GestureEvent(static_cast<QGestureEvent*>(event));
        if (QGesture *swipe = gesture->gesture(Qt::SwipeGesture)) {
                // Swipe Event
        } else if (QGesture *pan = gesture->gesture(Qt::PanGesture)) {
                // Pan Event
        }
        if (QGesture *pinch = gesture->gesture(Qt::PinchGesture)) {
                // Pinch Event
        }
}
```

7.3.3.1.Pan Gesture

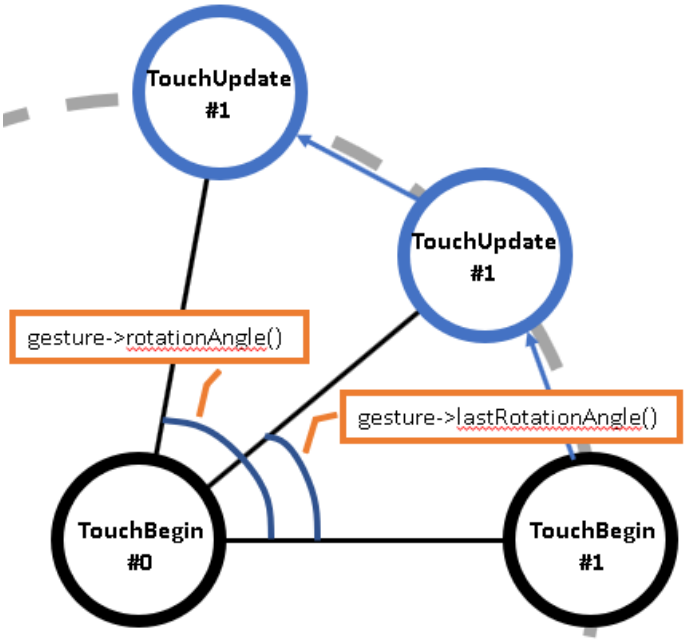
Pan 事件需使用者以二指触摸才会产生。收到 Pan 类别手势事件时，软件可由 delta() 值判断触摸的移动速度，并由 offset() 值判断总共移动的长度。



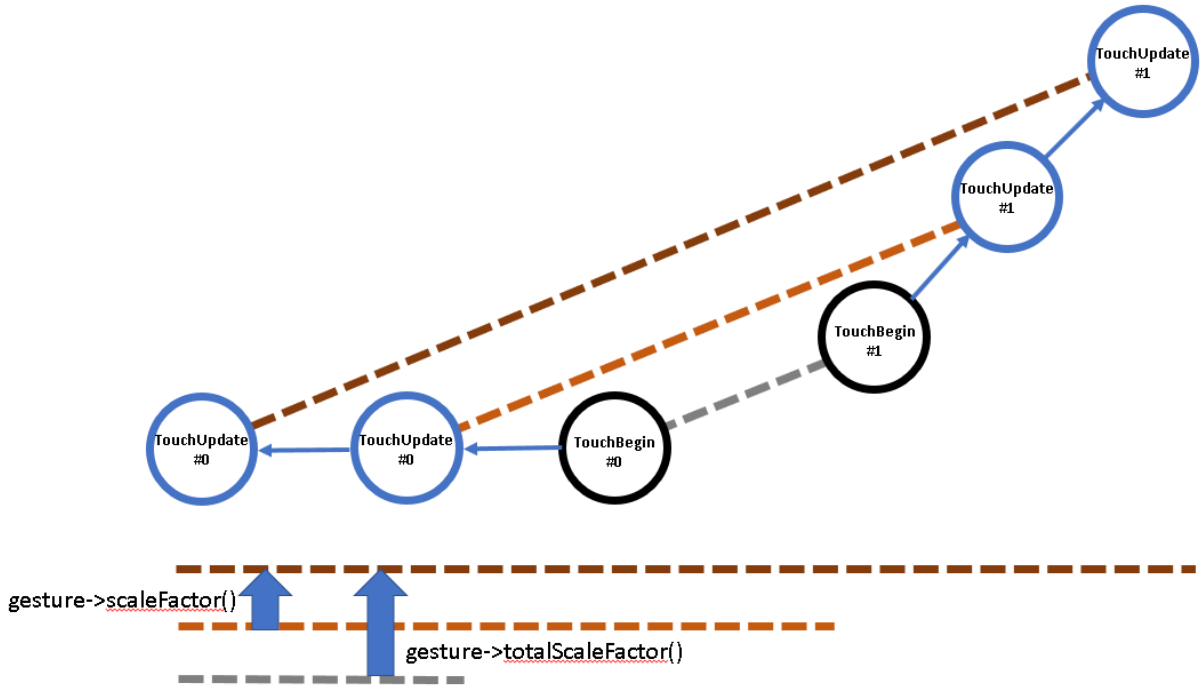
7.3.3.2.Pinch Gesture

Pinch 事件需使用者以二指触摸才会产生。收到 Pinch 类别手势事件时，依据 gesture->changeFlags() 值判断包含 Rotate 或 Zoom。

- QPinchGesture::RotationAngleChanged
  Rotate 产生了变化，可由 rotationAngle 判断自 TouchBegin 开始转的角度，顺时针转数值为正，逆时针转数值为负。

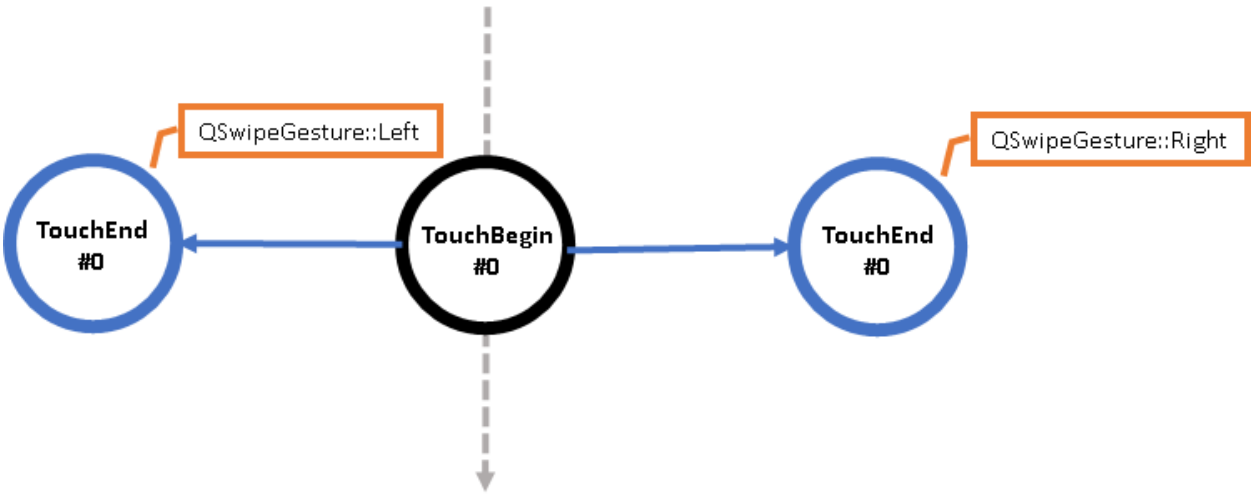


- QPinchGesture::ScaleFactorChanged
  双指的距离产生了变化，如果刚 TouchBegin 时的距离为 d，经 TouchUpdate 移动后，双指间距变为 d * scaleFactor()。 再经多次 TouchUpdate 后，双指间距变为 d* totalScaleFactor()。
  因此若 scaleFactor 大于 1，表示进行 Zoom In，若小于 1，表示进行 Zoom Out。



7.3.3.3.Swipe Gesture

手指在时限内往**左或右移動並** TouchEnd。



需注意不是所有的系统中软件都能收到 swipe 手势事件，因有时该事件可能会被系统拦截。

7.4. 简易测试与问题排除

1. 使用电阻式校准时请参考第**錯誤! 找不到參照來源。** 章，以 ts_calibrate 程序进行。若系统自带的 tslib 为 1.0 或更早的版本，会无法支持 PenMount 触摸屏。
2. 使用 Qt4 若不确定是否包含上述的 mouse plugin。可在执行前设定：

```
export QT_DEBUG_PLUGINS=1
```

3. 使用 Qt5 若不确定是否包含 tslib 支持。可在执行前设定：

```
export QT_DEBUG_PLUGINS=1

export QT_LOGGING_RULES=qt.qpa.input=true
```