

芯片驱动指导文档(展讯平台)

芯片驱动指导文档(展讯平台)	
Project name	Touch panel
Document ref	[Document ref]
Version	1.3.1
Release date	2015.10.26
Owner	mshl
Classification	
Distribution List	
Approval	

This document contains information proprietary to FocalTech Systems, Ltd., and may not be reproduced, disclosed or used in whole or part without the express written permission of FocalTech Systems, Ltd.

**Copyright © 2015, FocalTech Systems, Ltd
All rights reserved**

**3/F,Kingdom Sci-Tech Building,
5th Gaoxinnan Avenue, Hi-Tech Park,
Nanshan District ,Shenzhen, Gungdong, P.R. China**

**ZIP :518057
T +86 755 26588222
F +86 755 26712499
E support@focaltech-systems.com**

www.focaltech-systems.com

Revision History

Date	Version	List of changes	Author	Approved by
2014.09.25	1.0	Initial draft.	mshl	
2015.04.30	1.1	Add gesture function.	mshl	
2015.06.05	1.2	Add 8606 IC .	mshl	
2015.06.11	1.3	Update Makefile &add 64bitlib .	mshl	
2015.10.26	1.3.1	Add 8716 and 3x07 IC .	mshl	

目录

1	概述.....	4
2	基本信息.....	4
3	驱动文件介绍.....	4
4	整合到展讯平台.....	5
4.1	移植文件.....	5
4.1.1	主要功能移植.....	5
4.1.2	完整功能移植.....	5
4.2	如何编译.....	6
4.3	调试.....	7
5	与上层 APK 应用及 ADB 命令有关的宏和 ADB 命令的介绍.....	7
5.1	与上层 APK 应用及 ADB 命令有关的宏的介绍.....	7
5.2	ADB 命令介绍.....	7
6	详细功能介绍.....	9
6.1	手势唤醒功能.....	9
6.1.1	手势基本功能的描述.....	9
6.1.2	手势功能的基本规格.....	9
6.1.3	目前手势实现的字符,数字,符号(标准正楷字体).....	9
6.1.4	手势书写规范.....	10
6.1.5	手势功能驱动代码介绍.....	11

1 概述

本文档介绍敦泰科技（FocalTech）Android 驱动程序主要功能、文件结构以及如何移植到展讯平台。

2 基本信息

支持芯片系列：	FT5x06, FT5606, FT5x16, FT6x06, Ft6x36, FT5x06i, FT5336, FT3316, FT5436i, FT5336i, FT5x46, FT5x26, FT5822, FT8606, FT8716, FT3x07
支持平台：	展讯所有平台
APK/ADB 工具：	支持

3 驱动文件介绍

驱动文件存放在 src 文件夹里，实现了驱动挂载、触摸点上报、休眠唤醒、手势唤醒、FW 升级等功能及 APK 和 ADB 调试调用的接口。下面列表是每一个文件的功能简介：

文件名	属性	功能
Makefile	必选	Makefile 文件。
focaltech_core.h	必选	驱动主功能头文件。
focaltech_core.c	必选	驱动主功能文件，用来实现驱动的挂载、读取触摸数据的上报坐标、休眠唤醒处理等触摸屏驱动的基本功能。
focaltech_flash.c	可选	固件升级功能文件。
focaltech_ctl.c	可选	调试功能实现文件，用于支持 Android 应用程序（APK 工具）调试 TP。加入调试功能文件，可以在装成整机后在 Android 上层对触控 IC 进行测试、调试、检测等功能。
focaltech_ex_fun.c	可选	扩展功能实现文件，主要用于支持固件升级、ADB 调试。该文件不是必需的，但是推荐在驱动中增加该功能，以便于您使用的触控 IC 在必要时升级为最新版本的固件。
ft_gesture_lib.h	可选	手势唤醒功能头文件。
focaltech_gesture.c	可选	手势唤醒功能文件。
ft_gesture_lib_v1.0_20140820.a	可选	手势唤醒功能库文件。

FT_Upgrade_App.i	可选	固件升级 App 文件。
FT8xx6_Pramboot_Vx.x_xxxx.i	可选	固件升级 Pramboot 文件。

4 整合到展讯平台

4.1 移植文件

本节讲述如何移植我司的驱动，包括主要功能移植与完整功能移植两部分，若是主要功能移植则驱动仅实现了如驱动挂载、触摸点上报、休眠唤醒功能，若是完整功能移植则驱动实现的功能有驱动挂载、触摸点上报、休眠唤醒、手势唤醒、FW 升级等功能及 APK 和 ADB 调试调用的接口。

4.1.1 主要功能移植

1. 将 Makefile、focaltech_core.c、focaltech_core.h 这三个文件复制到 /drivers/input/touchscreen/focaltech/ 目录下，如下图所示：



名称	大小	类型
focaltech_core.c	55 KB	C Source
focaltech_core.h	9 KB	C/C++ Header
Makefile	1 KB	文件

图 4.1.1

2. 修改 kernel/include/linux/input/touchscreen/focaltech/ 目录下的 Makefile 文件，在这个文件的末尾增加如下两行：

```
obj-$(CONFIG_TOUCHSCREEN_FTS) += fts_ts.o
fts_ts-y += focaltech_core.o
```

4.1.2 完整功能移植

1. 将 Makefile、focaltech_core.c、focaltech_core.h、focaltech_ctl.c、focaltech_ex_fun.c、focaltech_flash.c、focaltech_gesture.c、ft_gesture_lib.h、FT_Upgrade_App.i、FT8xx6_Pramboot_Vx.x_xxxx.i 和 ft_gesture_32bit_lib_v1.0_20140820.a (64 位系统则为 ft_gesture_64bit_lib_v1.0_20140820.a) 十一个文件复制到 kernel/include/linux/input/touchscreen/ 目录，如下图所示：

名称	大小	类型
focaltech_core.c	57 KB	C Source
focaltech_core.h	9 KB	C/C++ Header
focaltech_ctl.c	11 KB	C Source
focaltech_ex_fun.c	22 KB	C Source
focaltech_flash.c	101 KB	C Source
focaltech_gesture.c	12 KB	C Source
FT6xx6_Framboot_Vx.x.xx...	0 KB	Preprocessed C/C++ Source
ft_gesture_32bit_lib_v1...	27 KB	A 文件
ft_gesture_64bit_lib_v1...	54 KB	A 文件
ft_gesture_lib.h	1 KB	C/C++ Header
FT_Upgrade_App.i	243 KB	Preprocessed C/C++ Source
Makefile	1 KB	文件

图 4.1.2

2. 修改 kernel/include/linux/input/touchscreen/focaltech/目录下的 Makefile 文件, 在这个文件的末尾增加如下内容:

```
obj-$(CONFIG_TOUCHSCREEN_FTS) += fts_ts.o

fts_ts-y += focaltech_core.o focaltech_ctl.o focaltech_ex_fun.o
focaltech_gesture.o focaltech_flash.o
```

在kernel目录下Makefile文件的末尾增加如下内容:

```
vmlinux-main := $(core-y) $(libs-y) $(drivers-y) $(net-y)
$(PWD)/drivers/input/touchscreen/focaltech/ft_gesture_32bit_lib_v1.0_20140820.a
```

4.2 如何编译

在 spread/config/your_project_name 目录下(your_project_name 是当前编译的项目名), 打开 ProjectConfig.mk 文件, 找到 CUSTOM_KERNEL_TOUCHPANEL 条目, 参照以下方式修改, 其中 focaltech 必须与放置 TP 驱动源文件的文件夹名字相同。

```
CUSTOM_KERNEL_TOUCHPANEL = focaltech
```

在完成 4.1 节所述的工作后就可以编译内核了, 在命令行下输入编译命令如下图所示:

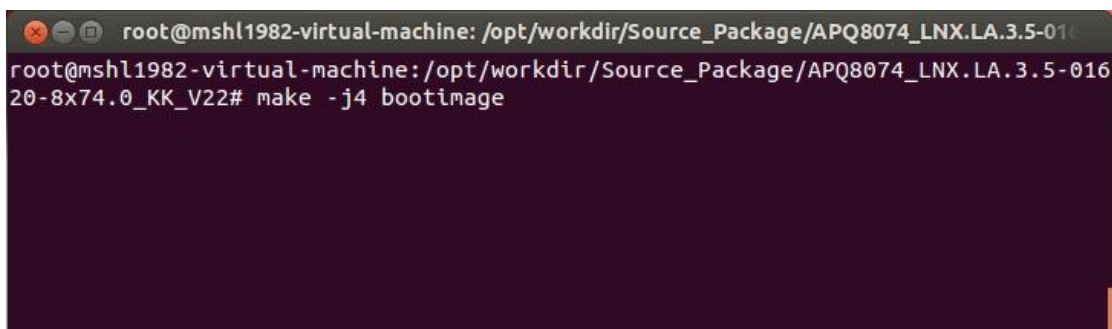


图 4.2.1

等待几分钟之后, 编译成功会生成 boot.img。

4.3 调试

把在 4.2 节所生成的 boot.img 烧录到开发平台, 验证驱动编译成功与否最简单的方法就是用手来触摸触摸屏来看报点功能是否正常, 若是完整移植则可参考 5.2 节的 ADB 命令介绍通过在命令行键入相关命令来验证。

5 与上层 APK 应用及 ADB 命令有关的宏和 ADB 命令的介绍

5.1 与上层 APK 应用及 ADB 命令有关的宏的介绍

focaltech_core.h 文件中的 FTS_CTL_IIC 与 FTS_APK_DEBUG 这两个宏均与上层 APK 应用有关其中由 FTS_CTL_IIC 这个宏控制的相关代码为我司的读数据及升级用的 APK 提供了驱动层的接口而由 FTS_APK_DEBUG 这个宏控制的相关代码则为用 APK 来实现整机测试提供驱动层的接口(我们提供了两套整机测试方案一个是在驱动端实现另一个则也可以通过我司提供的 APK 来实现), 而 FTS_SYSFS_DEBUG 这个宏控制的相关代码则为 ADB 命令调用提供驱动层的接口, 故建议客户在移植代码时把这三个宏都打开。在 focaltech_core.c 文件中的 `static int fts_probe(struct i2c_client *client, const struct i2c_device_id *id)`, 在这个函数中要加入以下语句:

```
#ifdef FTS_SYSFS_DEBUG
    fts_create_sysfs(client);
#endif

#ifdef FTS_CTL_IIC
    if (ft_rw_iic_drv_init(client) < 0)
    {
        dev_err(&client->dev, "%s:[FTS] create fts control iic driver failed\n",
            __func__);
    }
#endif

#ifdef FTS_APK_DEBUG
    fts_create_apk_debug_channel(client);
#endif
```

FTS_CTL_IIC 这个宏的实现是由 focaltech_ctl.c 这个文件来实现的, FTS_APK_DEBUG 这个宏的功能是在 focaltech_ex_fun.c 文件中实现的。

5.2 ADB 命令介绍

为了方便项目的调试我们在驱动端提供了供 ADB 命令调用的接口, 这样在实现的项目调试阶段可以大幅度降低调试难度并还可以加快项目的进度。

与生成相关调试节点有关的代码在 focaltech_ex_fun.c 文件中如下所示:

THIS DOCUMENT CONTAINS INFORMATION PROPRIETARY TO FOCALTECH SYSTEMS, LTD., AND MAY NOT BE REPRODUCED, DISCLOSED OR USED IN WHOLE OR PART WITHOUT WRITTEN PERMISSION OF FOCALTECH SYSTEMS, LTD.

```
static struct attribute *fts_attributes[] = {
    &dev_attr_ftstpfwver.attr,
    &dev_attr_ftstpdrierver.attr,
    &dev_attr_ftsfwupdate.attr,
    &dev_attr_ftstprwreg.attr,
    &dev_attr_ftsfwupgradeapp.attr,
    &dev_attr_ftsgetprojectcode.attr,
    NULL
};

static struct attribute_group fts_attribute_group = {
    .attrs = fts_attributes
};

/*create sysfs for debug*/
int fts_create_sysfs(struct i2c_client *client)
{
    int err;
    err = sysfs_create_group(&client->dev.kobj, &fts_attribute_group);
    if (0 != err) {
        dev_err(&client->dev, "%s() - ERROR: sysfs_create_group() failed.\n",
            __func__);
        sysfs_remove_group(&client->dev.kobj, &fts_attribute_group);
        return -EIO;
    } else {
        mutex_init(&g_device_mutex);
        pr_info("fts:%s() - sysfs_create_group() succeeded.\n",
            __func__);
    }
    return err;
}

void fts_release_sysfs(struct i2c_client *client)
{
    sysfs_remove_group(&client->dev.kobj, &fts_attribute_group);
    mutex_destroy(&g_device_mutex);
}
```

生成的节点在系统的/sys/bus/i2c/devices/1-0038/这个目录下，在PC上在运行里键入cmd回车，键入adb shell回车，键入cd /sys/bus/i2c/devices/1-0038/回车，下面介绍具体如何通过ADB命令来调用驱动所提供的接口：

- A. 键入cat ftstpfwver回车即可得到当前FW的版本号
- B. 键入cat ftstpdrierver回车即可得到当前驱动的版本号
- C. 键入echo 1 > ftsfwupdate回车即可通过.i文件来升级FW
- D. 键入echo 88 > ftstprwreg回车即可读到0X88寄存器里的值

THIS DOCUMENT CONTAINS INFORMATION PROPRIETARY TO FOCALTECH SYSTEMS, LTD., AND MAY NOT BE REPRODUCED, DISCLOSED OR USED IN WHOLE OR PART WITHOUT WRITTEN PERMISSION OF FOCALTECH SYSTEMS, LTD.

- E. 键入echo *_app.bin > ftsfwupgradeapp回车即可通过.bin文件来升级FW, 在键入此命令之前必需先把*_app.bin文件放到/sdcard/目录下才行

6 详细功能介绍

6.1 手势唤醒功能

6.1.1 手势基本功能的描述

在手机/Pad LCD 息屏,进入休眠状态后, FW 实现捕捉用户在 TP 上的触摸轨迹并采用相应的算法对其进行分析, 在分析之后若触摸轨迹合法则会把触摸轨迹数据准备好并通过发中断通知 Driver 去获取, Driver 通过调用 ft_gesture_lib_v1.0_20140820.a 库文件中的 int fetch_object_sample(unsigned char *buf, short pointnum) 这个函数来对获取到的触摸轨迹信息进行分析并输出相应的结果如: 是特定的字符(比如说双击,直线,字母等),就通知主机端,主机端可以根据手势快速进入对应的应用程序,实现特殊应用的快捷启动; 若触摸轨迹不合法 FW 会忽略此次的触摸轨迹。

6.1.2 手势功能的基本规格

- A. 目前只实现了单指单笔的手势.
- B. 手势书写时间不能超过 1.5s.
- C. 书写不能太快,速度不能大于 200mm/s.
- D. 符号不能写的太小,比如现在规定不能小于 1.3cm*1.3cm.
- E. 待机功耗小于 1mA(TBD)(无触摸状态. 有触摸时恢复到正常工作状态的功耗大小)
- F. 字符符号书写尽量规范,正楷书写,比例协调,不能写的太随意,不接受花体写法.

6.1.3 目前手势实现的字符,数字,符号(标准正楷字体)

Name	ID
←	0x20
→	0x21
↑	0x22
↓	0x23
⊙	0x24
@	0x50
<	0x34
>	0x52
^	0x53
v	0x54

△	0x55
o	0x30
w	0x31
m	0x32
e	0x33
C	0x34
g	0x35
a	0x36
d	0x37
n	0x40
z	0x65
b	0x42
q	0x43
L	0x44
p	0x45
s	0x46
u	0x47
h	0x70
k	0x71
y	0x72
r	0x73
3	0x60
6	0x61
9	0x62
7	0x63
8	0x64
2	0x65

6.1.4 手势书写规范

- 手机/平板要正方向拿,不能旋转方向
- 双击:两点距离不大于 1.0cm, 第一次点击触摸>100ms, 间隔>100ms, 第二次触摸抬起后 200ms 内不能有触摸
- 向上划直线:长度不小于 Y 分辨率的一半, X 方向的偏差不大于 X 分辨率的四分之一
- 向下滑动直线: 长度不小于 Y 分辨率的一半, X 方向的偏差不大于 X 分辨率的四分之一

THIS DOCUMENT CONTAINS INFORMATION PROPRIETARY TO FOCALTECH SYSTEMS, LTD., AND MAY NOT BE

REPRODUCED, DISCLOSED OR USED IN WHOLE OR PART WITHOUT WRITTEN PERMISSION OF FOCALTECH SYSTEMS, LTD.

- E. 向左滑动直线: 长度不小于 X 分辨率的一半, Y 方向的偏差不大于 Y 分辨率的八分之一
- F. 向左滑动直线: 长度不小于 X 分辨率的一半, Y 方向的偏差不大于 Y 分辨率的八分之一
- G. >上下基本对称,尽量画直
- H. ^左右基本对称,尽量画直
- I. v 左右基本对称,尽量画直
- J. △左右基本对称,尽量画直
- K. 圆圈符号(字母 O):尽量画圆,不能是扁形的,(X:Y <1.2); 封口尽量好(起始点接近终结点),要么封口在 X 和 Y 方向都不能大于字符区域的 1/4
- L. 字母符号 C: 允许写直线(<),开口够大,
- M. 字母符号 e:弧线尽量圆,不接受大头 e,和尾巴太长或太短的 e
- N. 字母符号 w:允许写折线的大写 W)
- O. 字母符号 m:允许写折线的大写 M)
- P. 字母符号 g:上下两部分大体相当,回勾不宜太大
- Q. 字母符号 a:尾巴不宜过长
- R. 字母符号 d:竖线不宜太短
- S. 字母符号 n:左右基本对称
- T. 字母符号 z:上横线可以为弧线(2)
- U. 字母符号 b:竖线不宜太短
- V. 字母符号 q:落笔必须为右勾
- W. 字母符号 p:竖线不宜太短
- X. 字母符号 s:尾巴不宜回勾
- Y. 字母符号 u:左右基本对称
- Z. 字母符号 h:竖线不宜太短
- AA. 字母符号 k:暂无
- BB. 字母符号 y: 暂无
- CC. 字母符号 r:暂无
- DD. 字母符号 3:暂无
- EE. 字母符号 6:暂无
- FF. 字母符号 9:暂无
- GG. 字母符号 7:横线要直
- HH. 字母符号 8:右上角起笔开始书写

6.1.5 手势功能驱动代码介绍

把 focaltech_core.h 文件中的 FTS_GESTRUE_EN 这个宏打开并在该文件中查看与该宏有关的代码如下为变量声明:

THIS DOCUMENT CONTAINS INFORMATION PROPRIETARY TO FOCALTECH SYSTEMS, LTD., AND MAY NOT BE REPRODUCED, DISCLOSED OR USED IN WHOLE OR PART WITHOUT WRITTEN PERMISSION OF FOCALTECH SYSTEMS, LTD.

```
#if FTS_GESTRUE_EN
#define GESTURE_LEFT      0x20
#define GESTURE_RIGHT     0x21
#define GESTURE_UP        0x22
#define GESTURE_DOWN      0x23
#define GESTURE_DOUBLECLICK 0x24
#define GESTURE_O          0x30
#define GESTURE_W          0x31
#define GESTURE_M          0x32
#define GESTURE_E          0x33
#define GESTURE_C          0x34
#define FTS_GESTRUE_POINTS 255
#define FTS_GESTRUE_POINTS_ONETIME 62
#define FTS_GESTRUE_POINTS_HEADER 8
#define FTS_GESTURE_OUTPUT_ADRESS 0xD3
#define FTS_GESTURE_OUTPUT_UNIT_LENGTH 4
short pointnum = 0;
unsigned short coordinate_x[150] = {0};
unsigned short coordinate_y[150] = {0};
extern int fetch_object_sample(unsigned char *buf, short pointnum);
extern void init_para(int x_pixel, int y_pixel, int time_slot, int cut_x_pixel, int cut_y_pixel);
suspend_state_t get_suspend_state(void);
#endif
```

与手势功能实现相关的函数主要有：

```
static int fts_probe(struct i2c_client *client, const struct i2c_device_id *id),
```

在这个函数中要加入以下语句：

```
#if FTS_GESTRUE_EN
    fts_Gesture_init(input_dev);
    if (fts_updateinfo_curr.CHIP_ID != 0x54 && fts_updateinfo_curr.CHIP_ID !=
0x58 && fts_updateinfo_curr.CHIP_ID != 0x86 && fts_updateinfo_curr.CHIP_ID != 0x87)
    {
        init_para(720, 1280, 0, 0, 0);
    }

#endif
```

init_para该函数是设置X与Y方向分辨率的，后面三个参数为预留的，其中第一个与第二个参数的意义分别表示当前触摸屏的X与Y方向的实际的分辨率；

```
static void fts_ts_suspend(struct early_suspend *handler),
```

在这个函数中要加入以下语句：

```
#if FTS_GESTRUE_EN
```

```
fts_write_reg(this_client, 0xd0, 0x01);
if (fts_updateinfo_curr.CHIP_ID==0x54 || fts_updateinfo_curr.CHIP_ID==0x58 ||
fts_updateinfo_curr.CHIP_ID==0x86 || fts_updateinfo_curr.CHIP_ID==0x87)
{
    fts_write_reg(this_client, 0xd1, 0xff);
    fts_write_reg(this_client, 0xd2, 0xff);
    fts_write_reg(this_client, 0xd5, 0xff);
    fts_write_reg(this_client, 0xd6, 0xff);
    fts_write_reg(this_client, 0xd7, 0xff);
    fts_write_reg(this_client, 0xd8, 0xff);
}

return 0;
#endif
```

```
static void fts_resume_work(struct work_struct *work),
```

在这个函数中加入以下语句:

```
#if FTS_GESTRUE_EN
    fts_write_reg(this_client, 0xD0, 0x00);
#endif

fts_reset();
```

```
static int touch_event_handler(void *unused),
```

在这个函数中加入以下语句:

```
#if FTS_GESTRUE_EN
    if(bsuspend == 1)
    {
        fts_read_reg(data->client, 0xB0, &state);
        printk("tpd fts_read_Gestrue data state=%d\n", state);
        if(state ==1)
        {
            fts_read_Gestrue data();
            return 1;
        }
    }
}

#endif
```

fts_read_Gestrue data(), 该函数为获取FW提供的用户的触摸轨迹的, static void fts_check_gesture(int gesture_id) 该函数是把通过库文件分析得出的手势的ID上报给上层系统, 这些函数供客户在移植代码时参考。