海栎创芯片驱动指导文档

日期	2021/4/14
项目 ID	Xxx
文档版本	V3.1
驱动版本	V2.2
App 版本	V1.25
发布日期	2021/4/14
作者	Steven

公司地址以及联系方式：

上海总部：上海市浦东新区丹桂路 899 号国创中心主楼（1 栋）4 层 411 室

深圳分公司：深圳市宝安区洪浪北二路鼎新科技园 A 栋 5 楼

公司网址：

www.hynitron.com

版本历史：

Version	Date	Author
V2.0	2020/8/13	Steven
V3.0	2021/4/13	Steven
V3.1	2021/4/14	Steven

更改履历表：

版本 v2.0	初版
	1、整理芯片系列兼容，支持 CST1XX/CST2XX/CST3XX/CST7XX/CST8XX /CST1XXSE/CST2XXSE/CST3XXSE 2、支持 apk 升级固件、读取版本号、工厂测试、读取 raw 和 diff 功能。 3、支持 esd 检测功能。 4、支持手势唤醒功能。 5、支持接近感应功能。
版本 V3.0	1、支持 CST9XX 2、支持 factory 工厂测试开短路 3、强制 DTS 匹配获取 rst/int/分辨率
版本 V3.1	1、修改固件配置内容说明

目录

1、目标

本文档主要用于介绍海栎创触摸芯片驱动的框架架构，驱动的配置与调试，方便 FAE 同事和方案公司调试驱动。包括驱动的主要功能、驱动配置、文档结构、移植步骤。

2、适用芯片类型

基本信息	
支持的芯片类型	CST1XX: CST126 CST128 CST130 CST140 CST14055 CST148 CST1XXSE:CST128SE

	CST2XX: CST226 CST237 CST240 CST2XXSE: CST226SE CST3XX: CST326 CST328 CST340 CST348 CST7XX: CST716 CST726 CST736 CST8XX: CST816 CST826 CST836U CST9XX: CST912 CST918 CST6XX: CST6928S
支持的平台	支持安卓平台（MTK/高通/全志/瑞芯微/展讯）
ADB、apk 功能	支持
其他功能	GESTURE ESD PROXIMITY

3、文件结构

驱动文件存放在 hynitron 文件夹，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 调试所用到的接口等功能。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
Kconfig	必选	Kconfig 文件
xxx_core.c	必选	驱动主功能文件，实现驱动的挂载、读取触摸数据的上报、休眠唤醒功能。
xxx_core.h	必选	驱动主功能头文件，包括项目信息（每个项目都单独需要配置）、主要结构体类型。
Hynitron_config.h	必选	可配置功能模块的 Enable 和 Disable 头文件
Hynitron_common.h	必选	芯片类型及寄存器的定义，其他文件函数外部声明，打印信息的定义
Hynitron_esd_check.c	必选	ESD 检测功能文件
Hynitron_gesture.c	可选	手势唤醒功能文件
Hynitron_i2c.c	必选	IIC 通讯功能文件
Hynitron_proximity.c	可选	接近感应功能文件
Hynitron_tool_debug.c	可选	安卓 sys、pro 节点，用于 adb 和 apk 调试，强烈推荐增加该功能。
Hynitron_update_firmware.c	必选	固件更新功能文件
Hynitron_update_firmware.h	必选	固件更新头文件
/firmware	必选	固件升级所用到的固件文件
/docs	可选	Dts 配置

4、编译配置

4.1 修改编译文件

将驱动文件打包到 hynitron 文件夹，并将 hynitron 文件夹复制到： kernel/dirvers/input/touchscreen 目录下。

（1）修改 touchscreen 目录下的 Kconfig 文件，在这个文件的末尾增加如下一行：

Source “drivers/input/touchscreen/hynitron/Kconfig”

（2）修改 touchscreen 目录下的 Makefile 文件，在这个文件的末尾增加如下一行：

Obj-$(CONFIG_TOUCHSCREEN_HYNITRON_TS) +=hynitron/

或者 obj-y += hynitron/

4.2 编译命令

（1）调出 menuconfig，选择 TOUCHSCREEN_HYNITRON_TS

（2）编译 bootimage

$ make bootimage -j32

默认编译驱动文件，不需要修改上述选项。

5、驱动功能配置

5.1 DTS 配置

示例：

Example:

```
i2c@f9927000 {
        hynitron@1a{
                compatible = "hynitron,hyn_ts";
                reg = <0x1a>;
                hynitron,reset-gpio = <&gpio 12 0x01>;
                hynitron,irq-gpio = <&gpio 13 0x02>;
                hynitron,max-touch-number = <5>;
                hynitron,display-coords =    <1080 1920>;

                hynitron,have-key;
                hynitron,key-number = <3>;
                hynitron,key-code = <139 172 158>;
                hynitron,key-y-coord = <2000 2000 2000>;
                hynitron,key-x-coord = <200 600 800>;
        };
};
```

DTS 需包含如下信息：

1、IIC 地址（reg 默认都是 0x1A，特殊情况可更改）

2、属性名字（compatible 需与驱动内部定义一致，否则驱动无法加载）

3、中断引脚（hynitron,irq-gpio）

4、复位引脚（hynitron,reset-gpio）

5、最大触摸手指数（hynitron,max-touch-number）

6、分辨率（hynitron,display-coords）

7、按键信息（如有按键，必须配置）

备注：DTS 除了按键信息，其他信息必须配置否则 DTS 解析出错，无法加载。如果确认不能修改 DTS，可手动

修改函数 hyn_parse_dt，直接赋值。

5.2 功能模块以及项目信息配置（必选）

5.2.1 配置 Hynitron_core.h 文件（项目信息）

项目信息配置：

请选择相应的 IC 类型及项目 ID 信息（Hynitron_core.h）：

```
#define HYN_CHIP_TYPE_CONFIG                CST340
#define HYN_IRQ_TRIGGER_RISING_CONFIG       0x01
#define HYN_MAIN_IIC_ADDR_CONFIG            0x1A
```

项目配置如上，开始新项目时，请驱动工程师或者 FAE 工程师配置如下信息：

CHIP_TYPE 芯片类型： CST340 （必选，如果芯片类型选择错误，可能导致检测芯片失败、固件无法升级，驱动加载失败）

TRIGGER_RISING 上升沿： 0x00 （可选，默认下降沿触发中断）

```
#define HYN_X_DISPLAY_DEFAULT               720
#define HYN_Y_DISPLAY_DEFAULT               1280
#define HYN_X_REVERT                        0
#define HYN_Y_REVERT                        0
#define HYN_XY_EXCHANGE                     0
#define HYN_MAX_KEYS                        3
#define HYN_MAX_POINTS                      5
#define HYN_MAX_SELFCAP_ID                  2
```

X_DISPLAY：720 （默认，如 dts 获取失败，采用此值）

Y_DISPLAY：1280 （默认，如 dts 获取失败，采用此值）

X_REVERT：0 （默认，修改 X 方向的坐标方向）

Y_REVERT：0 （默认，修改 Y 方向的坐标方向）

XY_EXCHANGE：0（默认，交换 X 与 Y）

MAX_KEYS: 3 （默认，3 个）

MAX_POINTS: 5 （默认，自容需修改为 2）

其他宏配置：

HYN_RESET_SOFTWARE	使能软件看门狗复位功能	Disable
HYN_UPDATE_FIRMWARE_POEWRON_ENABLE	使能断电复位升级功能	Disable
HYN_UPDATE_FIRMWARE_ENABLE	使能固件升级功能	Disable
HYN_UPDATE_FIRMWARE_FORCE	使能固件强制升级功能	Disable
HYN_IIC_TRANSFER_LIMIT	使能 IIC 通讯字节长度限制功能	Disable

5.2.2 配置 hynitron_config.h 文件（驱动功能模块配置）

配置功能模块：

功能模块宏（hynitron_config.h）	功能	默认
HYN_DEBUG_EN	打印 debug log 信息，用于调试，user 版本建议关闭	Enable
HYN_MT_PROTOCOL_B_EN	Linux 多点触摸协议开关，enable（B 协议），disabe（A 协议）	Enable
HYN_REPORT_PRESSURE_EN	Multi-Touch A/B 上报 pressure 值，默认打开	Enable
HYN_GESTURE_EN	手势功能开关，enable（开启），disable（关闭）	Disable
HYN_PSENSOR_EN	接近感应开关，enable（开启），disable（关闭）	Disable
HYN_ESDCHECK_EN	ESD 静电保护机制，每隔 1s 检测一次，异常则复位 IC。	Enable
HYN_AUTO_FACTORY_TEST_EN	开机工厂测试验证功能，检测 tp 一致性。	Disable
HYN_EN_AUTO_UPDATE	固件自动升级功能。enable（开启），disable（关闭）	Disable
HYN_SYS_AUTO_SEARCH_FIRMWARE	固件自动查询升级功能。enable（开启），disable（关闭）	Disable
ANDROID_TOOL_SURPORT	安卓平台 proc 节点生成。用于 apk 调试。	Enable
HYN_SYSFS_NODE_EN	安卓平台 sys 节点生成。用于 adb 调试。	Enable

配置固件升级功能支持芯片类型（对应芯片选择）：

升级功能宏（hynitron_config.h）	功能	默认
HYN_EN_AUTO_UPDATE_CST0xxSE	使能 CST0XXSE 系列芯片升级功能。 包括 CST016SE/CST026SE/CST036SE	Disable
HYN_EN_AUTO_UPDATE_CST0xx	使能 CST0XX 系列芯片升级功能。 包括 CST016/CST02E/CST036	Disable
HYN_EN_AUTO_UPDATE_CST1xx	使能 CST1XX 系列芯片升级功能。 包括 CST126/CS130/CST140/CST14055/CST148	Disable
HYN_EN_AUTO_UPDATE_CST1xxSE	使能 CST1XXSE 系列芯片升级功能。 包括 CST128SE/CST18858SE/CST18868SE	Disable
HYN_EN_AUTO_UPDATE_CST2xx	使能 CST2XX 系列芯片升级功能。 包括 CST226/CST237/CST240	Disable
HYN_EN_AUTO_UPDATE_CST2xxSE	使能 CST2XXSE 系列芯片升级功能。 包括 CST226SE	Disable
HYN_EN_AUTO_UPDATE_CST3xx	使能 CST3XX 系列芯片升级功能。 包括 CST326/CST328/CST340/CST348	Disable
HYN_EN_AUTO_UPDATE_CST3xxSE	使能 CST3XXSE 系列芯片升级功能。 包括 CST328SE	Disable
HYN_EN_AUTO_UPDATE_CST78xx	使能 CST78XX 系列芯片升级功能。 包括 CST716/CST726/CST736/CST816/CST826/CST836U	Disable
HYN_EN_AUTO_UPDATE_CST6xx	使能 CST6XX 系列芯片升级功能。 包括 CST6928S	

HYN_EN_AUTO_UPDATE_CST9xx	使能 CST9XX 系列芯片升级功能。 包括 CST912 CST918	

备注： 如升级时无法进入 bootloader 模式，请注意确认复位方式：

1、断电复位

2、复位引脚复位

3、看门狗复位

进入 bootloader 模式的窗口期：一般为复位芯片后，5ms~20ms 内发送命令有效。

5.2.3 固件信息配置（必须修改）

配置 IC 类型：

#define HYN_CHIP_TYPE_CONFIG CST340 （hynitron_core.h）

配置固件（hynitron_update_firmware.c）：

```
00026: #include "firmware/capacitive_hynitron_cst0xx_update.h"
00027: #include "firmware/capacitive_hynitron_cst2xx_update.h"
00028: #include "firmware/capacitive_hynitron_cst2xxse_update.h"
00029: #include "firmware/capacitive_hynitron_cst3xx_update.h"
00030: #include "firmware/capacitive_hynitron_cst3xxse_update.h"
00031: #include "firmware/capacitive_hynitron_cst6xx_update.h"
00032: #include "firmware/capacitive_hynitron_cst8xx_update.h"
00033: #include "firmware/capacitive_hynitron_cst9xx_update.h"
00034:
00035: //please config the chip series before using.
00036: struct hynitron_fw_array hynitron_fw_grp[] = {
00037:     //0-name; 1-fw; 2-project_id; 3-module_id; 4-chip_type; 5-fw_length;
00038:     { "capacitive_hynitron_cst0xx_update",    cst0xx_fw,    0x2843,0x01, CST016, (sizeof(cst0xx_fw))},
00039:     { "capacitive_hynitron_cst2xx_update",    cst2xx_fw,    0x0501,0x01, CST226, (sizeof(cst2xx_fw))},
00040:     { "capacitive_hynitron_cst2xxse_update", cst2xxse_fw, 0x0501,0x01, CST226SE, (sizeof(cst2xxse_fw))},
00041:     { "capacitive_hynitron_cst3xx_update",    cst3xx_fw,    0x2117,0x11, CST348, (sizeof(cst3xx_fw))},
00042:     { "capacitive_hynitron_cst6xx_update",    cst6xx_fw,    0x2117,0x11, CST6928S, (sizeof(cst6xx_fw))},
00043:     { "capacitive_hynitron_cst3xxse_update", cst3xxse_fw, 0x0501,0x01, CST328SE, (sizeof(cst3xxse_fw))},
00044:     { "capacitive_hynitron_cst8xx_update",    cst8xx_fw,    0x0501,0x01, CST836, (sizeof(cst8xx_fw))},
00045:     { "capacitive_hynitron_cst9xx_update",    cst9xx_fw,    0x2208,0x01, CST918, (sizeof(cst9xx_fw))},
00046:
00047: };
```

需要根据固件修改以下内容：

（1）替换对应芯片的.h 文件

（2）修改 hynitron_fw_grp[ ]对应固件的，项目 ID，模组 ID，芯片类型。

（3）如一个项目存在多个 tp 厂，可增加对应头文件，根据项目 ID 和模组 ID 识别。

5.3 量产测试配置（可选）

量产测试参数配置，需要配合海栎创测试 apk 使用。

打开宏：ANDROID_TOOL_SURPORT（hynitron_config.h）

生成 proc 节点：/proc/cst1xx_ts/cst1x-update（互容）

/proc/cst8xx_ts/cst8xx-update（自容）

海栎创测试 APK: Hyntronic_TP_Tools(1.25).apk（版本不能低于 1.25）

5.3.1、安装测试 apk

adb install C:\Users\steven_wu\Desktop\Hyntronic_TP_Tools(1.25).apk

5.3.2、修改 selinux 权限

adb shell setenforce 0

如需要 user 版本使用，请修改上层 te 文件。设置为测试 apk 为系统 apk。

示例如下：

1.file_context 中添加：/proc/cst1xx_ts/cst1xx-update u:object_r:cst1xx_ts:s0

2.在 File.te 中添加：type cst1xx-update ,fs_type,proc_type;

3.在 untrusted_app_25.te 中添加

allow untrusted_app_25 cst1xx-update :file { read write getattr ioctl open};

具体请咨询安卓上层 apk 同事。

5.3.3、配置 apk 工厂测试参数

工厂测试数据保存路径：/sdcard/Android/data/com.hyn.tp_updatefw/cache/cstxxx/

配置文件命名：hyn_mutualcap_testconfig.ini

配置文件路径：/system/etc/hyn_mutualcap_testconfig.ini

Adb root

Adb remount

Adb push /..hyn_mutualcap_testconfig.ini /system/etc/

启动 APK 工厂自动测试：

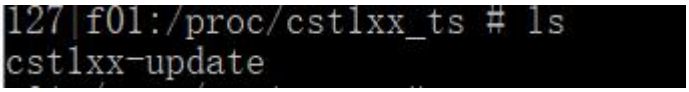
Adb shell am start -n com.hyn.tp_updatefw/.FactoryActivity

6、adb 调试节点

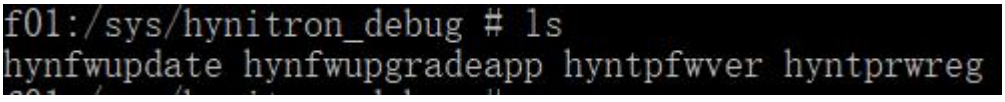
第一步：修改 selinux 权限： adb shell setenforce 0

生成的 adb 调试节点有两个：

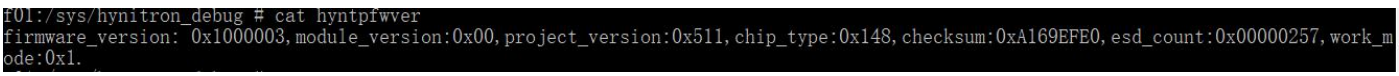
1、/proc/cst1xx_ts/cst1xx-update （打开宏 ANDROID_TOOL_SURPORT）



2、/sys/hynitron_debug（打开宏 HYN_SYSFS_NODE_EN）



查看版本号：cat hyntpfwver



固件升级：

Echo “1” >hynfwupdate //自动升级驱动合入的固件。

Echo “HYN_CST1_1.bin” >hynfwupgradeapp //自动升级/mnt/HYN_CST1_1.bin，确保 HYN_CST1_1.bin 的路径为/mnt.

中断 enable ： Echo “88” >hyntprwreg

中断 disable： Echo “99” >hyntprwreg

复位芯片： Echo “77” >hyntprwreg

7、apk 调试

第一步：修改 selinux 权限： adb shell setenforce 0

第二步：安装测试 apk： adb install C:\Users\steven_wu\Desktop\Hyntronic_TP_Tools(1.25).apk（版本不能低于 1.25）

1. DrawLine
2. 划线
3. MultiTouch
4. 多指触摸
5. Update Firmware
6. 固件更新
7. DataAnalysis
8. 数据分析，包括 rawdata diff，互容还包括自容信号
9. Manaul Test
10. 手动测试，互容会获取工厂数据 Delta High Short
11. Auto Test
12. 自动测试，顾名思义是自动获取工厂数据，然后对数据分析进行判断，该项依赖测试配置文件。
13. About
14. 软件介绍
15. Exit
16. 退出

7.1 数据保存

Diff 和 rawdata 默认不保存，当在 DataAnalysis 界面上，勾选 Savedata 后，会自动将数据保存至/sdcard/Android/data/com.hyn.tp_updatefw/cache/cstxxx/下，以.csv 格式保存。

工厂数据默认保存，目录同上。

7.2 Rawdata/Diff 互容界面介绍

Apk 在自动识别到互容芯片时，DataAnalysis 界面左上角会出现半透明的几个单选框和复选框。功能如下:

单选框：

Raw：读取 Rawdata 数据

Diff：读取 Diff 数据

复选框：

Reversal：将数据左右调换，因为平板和手机的 TX、RX 的相对位置会有所差别，另外方便调试，可选择勾选调整数据显示位置(是当前位置还是左右对称位置)

Savedata: 保存数据

SelfCapData: 勾选后，会在右侧和下侧显示 Rx 和 Tx 的自容信号。

7.3 更新固件

- 进入 Update Firmare 前，请先将固件通过 adb push 至/sdcard/目录下，参考命令如下:

adb push **.bin /sdcard/

- 然后通过”OPEN FILE”按钮选择固件
- 单击”UPDATE”来更新，等待进度条结束后，界面会自动刷新 tx rx 的个数以及固件版本。

启动命令：

am start -n com.hyn.tp_updatefw/.MainActivity

am start -n com.hyn.tp_updatefw/.DrawLineActivity

am start -n com.hyn.tp_updatefw/.UpdateFwActivity

am start -n com.hyn.tp_updatefw/.DataAnalyzeActivity

am start -n com.hyn.tp_updatefw/.ManualTestActivity

am start -n com.hyn.tp_updatefw/.FactoryActivity

附表：

常用 adb 命令：

adb shell cat /proc/kmsg | grep "HYN"

adb shell cat /proc/kmsg > /mnt/sdcard/log

adb pull /mnt/sdcard/log C:\Users\Administrator\Desktop

adb logcat > C:\Users\twl\Desktop\log\logcat.log

Adb shell settings put system show_touches 1 打开划线界面

adb shell settings put system pointer_location 1 打开指针位置

adb shell setprop debug.layout true 打开布局

adb shell ll /sys/bus/i2c/drivers 查看设备挂载

adb shell getevent

adb shell getevent -l 报点事件

adb shell getevent -r 报点率

adb shell getevent -r /dev/input/event5

adb shell input keyevent 3 返回键

adb shell input keyevent 4 主界面键

adb shell input keyevent 26 power 键

adb shell setenforce 0 关闭 selinux

adb push C:\Users\steven_wu\Desktop\apk\hyn_mutualcap_testconfig.ini /system/etc/ push 配置文件

Adb shell input tap x y 点击

Adb shell Input swipe x1 y1 x2 y2 滑动

adb shell settings put system screen_off_timeout 600000 设置液晶灭屏时间

adb shell am start com.android.settings/com.android.settings.Settings //打开手机设置

8、手势唤醒功能

8.1 手势初始化

支持的手势事件：

#define KEY_GESTURE_U KEY_U

#define KEY_GESTURE_UP KEY_UP

#define KEY_GESTURE_DOWN KEY_DOWN

```
#define   KEY_GESTURE_LEFT                  KEY_LEFT
#define   KEY_GESTURE_RIGHT                 KEY_RIGHT
#define   KEY_GESTURE_O                     KEY_O
#define   KEY_GESTURE_E                     KEY_E
#define   KEY_GESTURE_M                     KEY_M
#define   KEY_GESTURE_W                     KEY_W
#define   KEY_GESTURE_S                     KEY_S
#define   KEY_GESTURE_V                     KEY_V
#define   KEY_GESTURE_C                     KEY_C
#define   KEY_GESTURE_Z                     KEY_Z
#define   KEY_GESTURE_DOUBLECLICK           KEY_POWER
```

手势上报的手势码：

生成手势信息的节点：

/sys/hynitron_debug/hyn_gesture_mode　　//手势模式状态节点

/sys/hynitron_debug/hyn_gesture_buf　　//手势上报数据节点

8.2 手势上报

手势数据结构体：

```
struct hyn_gesture_st
{
    u8 header[HYN_GESTRUE_POINTS_HEADER];
    u16 coordinate_x[HYN_GESTRUE_POINTS];
    u16 coordinate_y[HYN_GESTRUE_POINTS];
     u16 report_key;            //驱动上报的事件码 ID
     u8   gestrue_id;           //芯片上报的手势码 ID
     u8   mode;                 //手势唤醒功能开关
     u8   active;               //手势开启检测状态 ，灭屏时置位
};
```

手势功能流程：

第一步：hyn_probe 中注册手势唤醒支持的事件，注册手势节点。

第二步：灭屏状态下，下发手势检测命令。设置中断：低电平触发，不睡眠。（低电平要保持 200ms）

第三步：中断触发，读取手势码，上报手势码对应的 input 层事件码，唤醒屏幕。

9、接近感应功能

9.1 接近感应初始化

接近感应的功能实现主要依附于驱动与安卓上层的匹配，来实现接近感应功能的打开与关闭，以及接近与远离数据的传递。

在 hyn_proximity_init 函数中：

（1）初始化实现接近感应的 input 输入设备，设置支持事件 EV_ABS

```
hyn_proximity_data->ps_input_dev = input_allocate_device();
__set_bit(EV_ABS, hyn_proximity_data->ps_input_dev->evbit);
```

```
input_set_abs_params(hyn_proximity_data->ps_input_dev, ABS_DISTANCE, 0, 1, 0, 0);
ret= input_register_device(hyn_proximity_data->ps_input_dev);
```

（2）注册接近感应的节点：

/sys/hynitron_debug/hyn_proximity_mode　　//接近感应模式状态节点

/sys/hynitron_debug/hyn_proximity_buf　　//接近感应上报数据节点

```
hyn_create_proximity_sysfs(hyn_ts_data->client);
```

节点操作：

```
cat  hyn_proximity_mode          //查看节点信息，查看接近感应状态
echo 01 > hyn_proximity_mode     //写入节点信息，打开接近感应，用于模拟调试
```

（3）初始化接近感应上报的接口函数

根据平台移植对应的接近感应上报实现方法，具体见参考驱动。

9.2 接近感应上报

目前接近感应的上报方式根据平台的不同，实现方法也不一样。主流的上报方法如下：

（1）展讯平台

第一种方法： 注册字符杂项设备：

```
err = misc_register(&tp_ps_device); //结构体 tp_ps_device 定义了操作接口函数。
static int tp_ps_release(struct inode *inode, struct file *file);
static long tp_ps_ioctl(struct file *file, unsigned int cmd, unsigned long arg)
static struct file_operations tp_ps_fops = {
    .owner                  = THIS_MODULE,
    .open                   = tp_ps_open,
    .release          = tp_ps_release,
    .unlocked_ioctl   = tp_ps_ioctl,
};
static struct miscdevice tp_ps_device = {
    .minor = MISC_DYNAMIC_MINOR,
    .name    = TP_PS_DEVICE,              //通常设备名字为 ltr_558als
    .fops    = &tp_ps_fops,
};
```

在上述实现方法中，上层会访问设备：ltr_558als，通过 ioctl 的操作接口函数来进行读写操作，进而实现检测接近和远离。

第二种方法：注册 class 节点：

```
firmware_class = class_create(THIS_MODULE,"sprd-tpd");//client->name
firmware_cmd_dev = device_create(firmware_class, NULL, 0, NULL, "device");//device
if(device_create_file(firmware_cmd_dev, &dev_attr_proximity) < 0)                    // /sys/class/sprd-tpd/device/proximity
input_dev = input_allocate_device();
static DEVICE_ATTR(proximity, S_IRUGO | S_IWUSR, show_proximity_sensor, store_proximity_sensor);
```

注册 class 设备，生成节点： /sys/class/sprd-tpd/device/proximity

注册输入设备，上报接近和远离事件：

```
input_report_abs(tp_ps->input, ABS_DISTANCE, dps_data);
input_sync(tp_ps->input);
```

上层访问 sys 节点 proximity 来下发接近感应开启和关闭命令。

此方法实现写数据与读数据的分离，即写数据通过节点 proximity 下发，读数据通过 input 设备读取。

（2）Mtk 平台

第一种方法：

定义结构体：struct hwmsen_object obj_ps;

obj_ps.polling = 0;//interrupt mode

obj_ps.sensor_operate = tpd_ps_operate;

绑定 ID_PROXIMITY： if((err = hwmsen_attach(ID_PROXIMITY, &obj_ps)))

static int tpd_ps_operate(void* self, uint32_t command, void* buff_in, int size_in,void* buff_out, int size_out, int* actualout)

此方法依赖 MTK 平台的 hwmsen 设计，通过操作接口函数：tpd_ps_operate，来实现读写数据。

需要包含的头文件：

#include <hwmsensor.h>

#include <hwmsen_dev.h>

#include <sensors_io.h>

第二种方法：

注册光感驱动： alsps_driver_add(&ps_init_info);

```
struct alsps_init_info ps_init_info = {
    .name     = "hyn_ts",
    .init     = ps_local_init,
    .uninit = ps_local_uninit,
};
```

定义接口函数：

struct ps_control_path ps_ctl = { 0 };

struct ps_data_path ps_data = { 0 };

```
    ps_ctl.open_report_data = ps_open_report_data;    //上报数据
    ps_ctl.enable_nodata = ps_enable_nodata;   //下发命令数据
    ps_data.get_data = ps_get_data;      //读取数据
```

此方法是参考光感驱动加载过程，来定义对应的操作函数，进而实现上层与驱动的交互。但是此方法无法实现兼容，而且必须把注册函数放在驱动入口处，不能放在 probe 里面。

需要包含的头文件：

```
    #include <alsps.h>
```

10、驱动加载流程

10.1 驱动入口函数

static int __init hynitron_driver_init(void)

10.2 加载 I2C 驱动

ret = i2c_add_driver(&hynitron_i2c_driver); //注意检查 dts 配置 compatible，必须与 hyn_dt_match 数组相同。

10.3 执行 probe 函数

1、hyn_platform_data_init(ts_data); //初始化平台相关的数据，解析 dts 配置

2、hyn_gpio_configure(); //初始化 gpio_request 申请 IRT,RST 引脚

3、hyn_ts_data_init(client); //初始化 hyn_ts_data 结构体数据，包括项目 ID、芯片类型等

4、hyn_detect_bootloader(client); //检测是否可进入芯片 boot 模式，确认芯片类型。

5、hyn_input_dev_int(ts_data); //初始化输入设备 input，设置报点 A/B 协议，创建报点工作队列。

6、hyn_irq_init(client); //初始化中断注册，包括上升沿、中断服务例程。

7、hyn_update_firmware_init(client); //升级固件，必须配置正确的芯片类型、项目 ID，才可升级。

8、hynitron_proc_fs_init(); // 生成 proc/节点信息，用于 apk 调试。

9、hyn_create_sysfs(client); //生成 sys/hynitron_debug 节点，用于 debug 调试。

10、hyn_gesture_init(hyn_ts_data->input_dev, client); //手势唤醒功能初始化，生成手势节点。

11、hyn_proximity_init(); //接近感应功能初始化，生成接近感应节点。

12、hyn_init_esd_protect(); //ESD 保护功能初始化，时间周期 1s。

10.4 触摸信息上报

1、触摸芯片拉中断脉冲。

2、触发驱动的中断服务例程。

3、添加报点 work 到对应的工作队列。

4、执行触摸数据上报 input 层。

5、安卓层处理并显示坐标。

11、寄存器说明

11.1 互容产品寄存器：

触摸信息寄存器（ENUM_MODE_NORMAL 模式）

（1）触摸信息必须是 normal 模式，否则数据异常（write 0xD109 进入）。

（2）数据读取必须按照 0xD000 读取第一个手指的 7 个字节，包括手指数量、按键数量。

（3）下发读取数据完成同步数据（write 0xD000AB）。

示例如下：

0x1A	W	0xD0	0x00					
0x1A	R	0x06	0x33	0x56	0x68	0x8F	0x01	0xAB
0x1A	W	0xD0	0x00	0xAB				

（4）需读取后面的多指数据，每个手指分配 5 个字节，按地址读取。

具体可参考报点函数 cst3xx_touch_report 处理。

寄存器地址	高四位				低四位			
	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0xD000	1st 手指的 ID				1st 手指的状态：按下(0x06)或者抬起			
0xD001	1st 手指的 X 坐标值高八位： X_Position>>4							
0xD002	1st 手指的 Y 坐标值高八位： Y_Position>>4							
0xD003	1st 手指的 X 坐标值 X_Position&0x0F				1st 手指的 Y 坐标值 Y_Position&0x0F			
0xD004	1st 手指的压力值							
0xD005	上报按键标志(0x80)				上报手指数量			
0xD006	固定 0xAB							
0xD007	2nd 手指的 ID				2nd 手指的状态：按下(0x06)或者抬起			
0xD008	2nd 手指的 X 坐标值高八位： X_Position>>4							

0xD009	2nd 手指的 Y 坐标值高八位： Y_Position>>4	
0xD00A	2nd 手指的 X 坐标值 X_Position&0x0F	2nd 手指的 Y 坐标值 Y_Position&0x0F
0xD00B	2nd 手指的压力值	
0xD00C	3rd 手指的 ID	3rd 手指的状态：按下(0x06)或者抬起
0xD00D	3rd 手指的 X 坐标值高八位： X_Position>>4	
0xD00E	3rd 手指的 Y 坐标值高八位： Y_Position>>4	
0xD00F	3rd 手指的 X 坐标值 X_Position&0x0F	3rd 手指的 Y 坐标值 Y_Position&0x0F
0xD010	3rd 手指的压力值	
0xD011	4th 手指的 ID	4th 手指的状态：按下(0x06)或者抬起
0xD012	4th 手指的 X 坐标值高八位： X_Position>>4	
0xD013	4th 手指的 Y 坐标值高八位： Y_Position>>4	
0xD014	4th 手指的 X 坐标值 X_Position&0x0F	4th 手指的 Y 坐标值 Y_Position&0x0F
0xD015	4th 手指的压力值	
0xD016	5th 手指的 ID	5th 手指的状态：按下(0x06)或者抬起
0xD017	5th 手指的 X 坐标值高八位： X_Position>>4	
0xD018	5th 手指的 Y 坐标值高八位： Y_Position>>4	
0xD019	5th 手指的 X 坐标值 X_Position&0x0F	5th 手指的 Y 坐标值 Y_Position&0x0F
0xD01A	5th 手指的压力值	

版本信息寄存器（ENUM_MODE_DEBUG_INFO 模式）

（1）版本信息读取必须是 debug info 模式（write 0xD101）

（2）读取对应寄存器地址的信息

（3）退回正常 normal 模式（write 0xD109）

具体可参考报点函数 cst3xx_firmware_info 处理。

寄存器地址	寄存器说明	寄存器（4 个字节）			
0xD1F4	按键、TX、RX 通道数量	KEY_NUM	TP_NRX	NC	TP_NTX
0xD1F8	X/Y 分辨率	TP_RESY		TP_RESX	
0xD1FC	固件校验码、Bootloader 时间	0xCACA		BOOT_TIMER	
0xD204	芯片类型、固件项目 ID	IC_TYPE		PROJECT_ID	
0xD208	芯片固件版本号	FW_MAJOR	FW_MINOR	FW_BUILD	
0xD20C	芯片固件 checksum	checksum_H	checksum_H	checksum_L	checksum_L

模式命令寄存器

模式命令用于进入不同工作模式，通常为内部调试用，客户端常用为 normal 模式 0xD109。

命令	命令说明	命令格式
0xD101	ENUM_MODE_DEBUG_INFO 模式，进入读取固件信息模式。	Write 0xD1 0x01
0xD102	System_Reset 标志，复位芯片。	Write 0xD1 0x02
0xD104	Redo_Calibration 标志，重新初始化算法。	Write 0xD1 0x04
0xD105	Deep sleep，进入睡眠模式。	Write 0xD1 0x05
0xD108	ENUM_MODE_DEBUG_POINTS，进入 debug 报点模式。	Write 0xD1 0x08
0xD109	ENUM_MODE_NORMAL，进入正常报点模式，为默认模式。	Write 0xD1 0x09
0xD10A	ENUM_MODE_DEBUG_RAWDATA,进入读取 rawdata 数据模式。	Write 0xD1 0x0A
0xD10B	ENUM_MODE_DEBUG_WRITE，进入 debug write 模式。	Write 0xD1 0x0B
0xD10C	ENUM_MODE_DEBUG_CALIBRATION,进入 redo 调试模式。	Write 0xD1 0x0C
0xD10D	ENUM_MODE_DEBUG_DIFF	Write 0xD1 0x0D
0xD119	ENUM_MODE_FACTORY	Write 0xD1 0x19

11.2 自容产品 CST8XX/CST7XX 寄存器

工作模式切换命令如下

工作模式	切换命令	描述
NOMAL	0000	正常报点和手势上报
DBG_IDAC	0004	工厂测试数据获取
DBG_POS	00E0	工厂测试按键和坐标获取
DBG_RAW	0006	原始值获取
DBG_SIG	0007	differ 值获取

NOMAL 寄存器说明

Address	Name	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	说明	Access
00h	Work_mode									Write: 00: NOMAL 04: DBG_IDAC E0: DBG_POS 06: DBG_RAW 07: DBG_SIG	R/W

01h	Proximity ID	[7:0]	default：00 far away：C0 near：E0	R
02h	touch num	touch points[3:0]		R
03h	touch1_XH	event_flg X_position[11:8]		R
04h	touch1_XL	X_position[7:0]		R
05h	touch1_YH	touch_ID[3:0] Y_position[11:8]		R
06h	touch1_YL	Y_position[7:0]		R
07h			default：00	R
08h			default：00	R
9h	touch2_XH	event_flg X_position[11:8]		R
10h	touch2_XL	X_position[7:0]		R
11h	touch2_YH	touch_ID[3:0] Y_position[11:8]		R
12h	touch2_YL	Y_position[7:0]		R
13h			default：00	R
14h			default：00	R
…	…			
A5h	sleep	deepsleep[7:0]	write 03 进入 deepsleep	W
A6h	fw_version	fw_version[7:0]	固件版本号	R
A7h	fw_version	fw_version[15:8]		R
A8h	module_ID	module_version[7:0]	模组 ID	R
A9h	project_name	project_name[7:0]	default：00	R
AAh	chip_type	chip_type[7:0]	芯片型号	R
ABh	chip_type	chip_type[15:8]		R
ACh	checksum	checksum[7:0]	固件 checksum	R
ADh	checksum	checksum[15:8]		R
…	…			
B0h	Prox_state	Prox_state[7:0]	write 01H 进入 Proximity 模式 00H 退出 Proximity 模式	W
…	…			
D0h	ges_state	ges_state[7:0]	write 01H 进入 gesture 识别模式 00H 退出 gesture 模式	W
…	…			

D3h	gesture ID	gesture[7:0]								手势模式使能才有效 double klick:0x24 up:0x22 down:0x23 left:0x20 rignt:0x21 C:0x34 e:0x33 m:0x32 O:0x30 S:0x46 V:0x54 W:0x31 Z:0x65	R
D4h	gesture data									做预留兼容别家驱动	R
D5h											R
D6h											R
D7h											R
D8h											R
D9h											R
DAh											R