

## 海栎创芯片驱动指导文档

日期	2022/7/1
项目 ID	Xxx
文档版本	V3.5
驱动版本	V2.4
App 版本	V1.26
发布日期	2022/7/1
作者	Steven



## 公司地址以及联系方式:

上海总部: 上海市浦东新区学林路 36 弄 10 号

深圳分公司: 深圳市宝安区新安街道宝安 67 区留仙一路高新奇产业园 2 期 1 号楼 B 座 8 楼

公司网址:

<http://hailichuang.cent.uoeee.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
V3.2	2021/7/20	Steven
V3.3	2021/8/4	Steven
V3.4	2021/9/3	Steven
V3.5	2022/7/1	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
版本 V3.2	1、修改 debug 调试功能,支持 USB 连接调试。 2、增加工厂测试功能,支持 CST644K
版本 V3.3	1、修复 CST918/CST130 软件复位失效。
版本 V3.4	1、修改开机匹配固件机制。
版本 V3.5	1、增加支持芯片 CST3240。 2、增加 sdcard log 文件保存功能。

## 目录

1、目标 .....	4
2、适用芯片类型 .....	4
3、文件结构 .....	4
4、编译配置 .....	5
4.1 修改编译文件 .....	5
4.2 编译命令 .....	5
5、驱动功能配置 .....	5
5.1 配置 DTS .....	5
5.2 配置功能模块以及项目信息（必选） .....	6
5.2.1 配置 Hynitron_core.h 文件（项目信息） .....	6
5.2.2 配置 hynitron_config.h 文件（配置功能模块） .....	7
5.2.3 配置固件信息（新项目必须修改） .....	8
5.3 固件版本信息 .....	9
5.4 触摸功能调试 .....	9
5.5 固件升级功能调试 .....	9
5.6 配置量产测试（可选） .....	10
5.6.1、安装测试 apk .....	11
5.6.2、修改 selinux 权限 .....	11
5.6.3、配置 apk 工厂测试参数 .....	11
6、adb 调试节点 .....	11
7、apk 调试 .....	12
7.1 数据保存 .....	13
7.2 Rawdata/Diff 互容界面介绍 .....	13
7.3 APK 更新固件 .....	13
8、手势唤醒功能 .....	14
8.1 手势初始化 .....	14
8.2 手势上报 .....	14
9、接近感应功能 .....	15
9.1 接近感应初始化 .....	15
9.2 接近感应上报 .....	15
10、驱动加载流程 .....	17
10.1 驱动入口函数 .....	17
10.2 加载 I2C 驱动 .....	17
10.3 执行 probe 函数 .....	17
10.4 触摸信息上报 .....	17
11、寄存器说明 .....	18
11.1 互容产品寄存器： .....	18
11.1.1 触摸信息寄存器（ENUM_MODE_NORMAL 模式） .....	18
11.1.2 版本信息寄存器（ENUM_MODE_DEBUG_INFO 模式） .....	19
11.1.3 模式命令寄存器 .....	20
11.2 自容产品寄存器 .....	20
11.2.1 工作模式切换命令如下 .....	20
11.2.2 NOMAL 寄存器说明 .....	20

## 1、目标

本文档主要用于介绍海标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 CST644K: CST644K
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（ <b>每个项目都单独需要配置</b> ）、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/drivers/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 配置 DTS

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：

（1）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。

（2）DTS 需要根据具体平台调整，具体查看/doc 路径下 hynitron-ts.txt。

（3）互容芯片 7 位的 IIC 地址默认 0X1A 或者 0X5A，寄存器地址统一为 16 位。

## 5.2 配置功能模块以及项目信息（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）

```
2: #define HYN_X_DISPLAY_DEFAULT 720
3: #define HYN_Y_DISPLAY_DEFAULT 1280
4: #define HYN_X_REVERT 0
5: #define HYN_Y_REVERT 0
6: #define HYN_XY_EXCHANGE 0
7: #define HYN_MAX_KEYS 3
8: #define HYN_MAX_POINTS 5
9: #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（如芯片无复位引脚，必须打开）	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（不校验项目 ID、版本，谨慎使用）	Disable
HYN_IIC_TRANSFER_LIMIT	使能 IIC 通讯字节长度限制功能（MTK 有些平台限制通讯长度）	Disable

## 5.2.2 配置 hynitron\_config.h 文件（配置功能模块）

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	
HYN_EN_AUTO_UPDATE_CST644K	使能 CST644 芯片升级功能。	

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 配置固件信息（新项目必须修改）

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）替换或者增加对应芯片的.h 文件
- （2）修改 hynitron\_fw\_grp[] 对应固件的数组名字、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 固件版本信息

互容芯片系列-固件版本信息读取函数：cst3xx\_firmware\_info

第一步：进入 debug info 模式：wirte 0XD1 0X01

第二步：读取寄存器 0XD1FC 的 checkcode，判断是否为空芯片。

第三步：如芯片有固件，则继续读取芯片类型/项目 ID/版本信息/CHECKSUM。

第四步：退出 debug info 模式，回复 normal 模式：write 0xD1 0X09

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

## 5.4 触摸功能调试

互容芯片系列-触摸功能调试

第一步：报点输入设备/工作队列注册，支持报点 A/B 协议：hyn\_input\_dev\_int（）；

第二步：中断服务函数注册，支持上升沿/下降沿中断：hyn\_irq\_init（）/hyn\_eint\_interrupt\_handler（）；

第三步：当触摸中断来临时，中断服务函数响应，进入报点函数：cst3xx\_touch\_report（）；

第四步：报点函数解析触摸协议：

触摸信息解析示例：

```

0x1A  W      0xD0  0x00
0x1A  R      0x06  0x0F  0x25  0x2A  0x71  0x01  0xAB
0x1A  W      0xD0  0x00  0xAB

```

触摸信息解析如下：

X 坐标：input\_x = (unsigned int)((buf[1] << 4) | ((buf[3] >> 4) & 0x0F));

Y 坐标：input\_y = (unsigned int)((buf[2] << 4) | (buf[3] & 0x0F));

压力值：input\_w = (unsigned int)(buf[4]);

状态值：sw = (buf[idx] & 0x0F) >> 1;

ID 值：finger\_id = (buf[idx] >> 4) & 0x0F;

具体可参考附录寄存器地址说明。

## 5.5 固件升级功能调试

互容芯片系列-固件升级功能调试-函数入口：hyn\_update\_firmware\_init（）；

第一步：读取芯片里面的固件版本信息：hyn\_firmware\_info（）；

第二步：根据配置的芯片类型，查找数组 hynitron\_chip\_type\_grp[] 的对应固件 h 文件。

第三步：如果芯片为空芯片，则烧录数组 hynitron\_chip\_type\_grp[] 中芯片类型匹配的第一个 h 文件。

第四步：如果芯片已烧录固件，则根据芯片固件中的项目 ID/芯片类型来寻找对应的.h 文件。

第五步：烧录前判断待烧录固件 h 文件中的固件版本，是否高于芯片中的固件版本，是否需要执行烧录：cst3xx\_update\_judge（）。

第六步：确认执行升级动作，操作复位函数使芯片进入 bootloader 阶段，此阶段只有 15ms 窗口期：cst3xx\_update\_firmware（）。

第七步：设置芯片进入 bootloader 模式：cst3xx\_into\_program\_mode（）。

第八步：擦除芯片中的久固件，此动作为全部擦除，不可恢复：cst3xx\_erase\_program\_area（）。

第九步：写入芯片新固件，通常为每次写入 1K 或者 512 字节，具体情况参考不同芯片烧录流程，如平台 IIC 通讯长度受限制，可打开 HYN\_IIC\_TRANSFER\_LIMIT 宏，此时将 1K 数据分为多次写入，单次写入 6 个字节：cst3xx\_write\_program\_data（）。

第十步：校验已写入芯片中的固件 checksum：cst3xx\_check\_checksum（）。

第十一步：如校验成功则推出烧录模式：cst3xx\_exit\_program\_mode（）。

第十二步：重新读取芯片中的固件版本信息：cst3xx\_firmware\_info（）。

备注：

调试阶段可通过 adb 操作/sys/hynitron\_debug 阶段执行固件升级动作，具体参考 adb 调试节点章节。

## 5.6 配置量产测试（可选）

第一种方法-apk 测试：

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\_SUPPORT（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）

第二种方法-sys 节点测试：

量产测试参数配置，需要打开宏：HYN\_AUTO\_FACTORY\_TEST\_EN 和 HYN\_SYSFS\_NODE\_EN。

测试参数说明：

#define HYN_FACTORY_TEST_LOG_TXT	"/sdcard/hyn_factory_test.txt"	//生产的测试数据 log 文件
#define HYN_TP_OPEN_VALUE_MAX	2500	//rawdata 测试数据上限
#define HYN_TP_OPEN_VALUE_MIN	1300	//rawdata 测试数据下限
#define HYN_TP_OPEN_LOW_VALUE_MIN	200	//低压测试数据下限
#define HYN_TP_OPEN_DRV_VALUE_MIN	1200	//高压测试数据下限
#define HYN_TP_OPEN_DROP_RATIO	15	//水平相邻节点落差 1.5
#define HYN_TP_OPEN_TX_DROP_RATIO	15	//垂直相邻节点落差 1.5
#define HYN_TP_SHORT_RX_TX_RESISTANCE	600	//短路阻值下限

以上测试参数，需要根据采样的 log 文件/sdcard/hyn\_factory\_test.txt，来确定具体的数值。

测试方法：

Echo "1" > /sys/hynitron\_debug/hyntpfactorytest //触发工厂测试

Cat /sys/hynitron\_debug/hyntpfactorytest //显示测试结果

### 5.6.1、安装测试 apk

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

### 5.6.2、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。

### 5.6.3、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）

```
127|f01:/proc/cst1xx_ts # ls
cst1xx-update
```

2、/sys/hynitron\_debug （打开宏 HYN\_SYSFS\_NODE\_EN）

```
f01:/sys/hynitron_debug # ls
hynfwupdate hynfwupgradeapp hyntpfwver hyntprwreg
```

查看版本号：cat hyntpfwver

```
f01:/sys/hynitron_debug # cat hyntpfwver
firmware_version: 0x1000003,module_version:0x00,project_version:0x511,chip_type:0x148,checksum:0xA169EFE0,esd_count:0x00000257,work_m
ode:0x1.
```

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 disable : Echo "1" >hyntprwreg
中断 enable: Echo "2" >hyntprwreg
复位芯片: Echo "3" >hyntprwreg
其他命令:
正常触摸模式: Echo "10" >hyntprwreg
FACTORY 模式: Echo "11" >hyntprwreg
RAWDATA 模式: Echo "12" >hyntprwreg
DIFF 模式: Echo "13" >hyntprwreg
查看调试寄存器: Cat hyntprwreg
```

附表-常用 adb 命令:

adb shell cat /proc/kmsg   grep "HYN"	显示 HYN 标志 kernel log
adb shell cat /proc/kmsg > /mnt/sdcard/log	保存 kernel log 到文件 log
adb pull /mnt/sdcard/log C:\Users\Administrator\Desktop	将 log 文件从手机 pull 出来
adb logcat > C:\Users\twl\Desktop\log\logcat.log	保存 logcat 到文件 logcat
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	显示 input 上报事件
adb shell getevent -l	显示报点事件
adb shell getevent -r	显示报点率
adb shell getevent -r /dev/input/event5	
adb shell input keyevent 3	模拟上报返回键
adb shell input keyevent 4	模拟上报主界面键
adb shell input keyevent 26	模拟上报 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	模拟点击事件
Adb shell Input swipe x1 y1 x2 y2	模拟滑动滑动
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

## 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: 划线
2. MultiTouch: 多指触摸

3. **Update Firmware:** 固件更新，可显示当前芯片内的固件信息。
4. **DataAnalysis:** 数据分析，包括 rawdata diff，互容还包括自容信号，音量上键 diff/rawdata 切换，音量下键暂停/继续。
5. **Manual Test:** 手动测试，互容会获取工厂数据 Delta High Short
6. **Auto Test:** 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。
7. **About:** 软件介绍
8. **Exit:** 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。

备注： 音量上键：切换 rawdata 模式和 diff 模式。

音量下键：切换暂停和继续。

## 7.3 APK 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/（或者/mnt 路径）
- 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。

启动安卓 apk 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
```

## 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\_firmware\_info(client); //读取芯片固件信息, 如失败, 检测 bootloader 确认芯片类型
- 5、hyn\_input\_dev\_init(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, 固件必须芯片类型、项目 ID 一致, 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。----问题分析: 逻辑分析仪抓取触摸时芯片上报 INT 信号是否正常。
- 2、触发驱动的中断服务例程。----问题分析: 查看 kernel log 中 irq 是否触发, cat/proc/interrupts 是否累加。
- 3、添加报点 work 到对应的工作队列。----问题分析: 查看 kernel log 中报点函数是否正常。
- 4、执行触摸数据上报 input 层。----问题分析: getevent 或者 kernel log 坐标是否上报正常。
- 5、安卓层处理并显示坐标。----问题分析: 查看安卓上层 log 坐标是否正常。

## 11、寄存器说明

### 11.1 互容产品寄存器：

#### 11.1.1 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 normal 模式）。
- （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）下发同步命令，完成此次坐标读取（write 0xD000AB）。

示例如下：

- （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。

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

触摸数据寄存器如下：

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	

### 11.1.2 版本信息寄存器（ENUM\_MODE\_DEBUG\_INFO 模式）

- （1）版本信息读取必须在 main 程序阶段，且进入 debug info 模式（先 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）。
- （4）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

### 11.1.3 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 自容产品寄存器

### 11.2.1 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

### 11.2.2 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

