



Unisoc Confidential For hiar

# Touch Panel 客制化指导手册

文档版本  
发布日期

V1.3  
2020-08-07

**版权所有 © 紫光展锐科技有限公司。保留一切权利。**

本文件所含数据和信息都属于紫光展锐所有的机密信息，紫光展锐保留所有相关权利。本文件仅为信息参考之目的提供，不包含任何明示或默示的知识产权许可，也不表示有任何明示或默示的保证，包括但不限于满足任何特殊目的、不侵权或性能。当您接受这份文件时，即表示您同意本文件中内容和信息属于紫光展锐机密信息，且同意在未获得紫光展锐书面同意前，不使用或复制本文件的整体或部分，也不向任何其他方披露本文件内容。紫光展锐有权在未经事先通知的情况下，在任何时候对本文件做任何修改。紫光展锐对本文件所含数据和信息不做任何保证，在任何情况下，紫光展锐均不负任何与本文件相关的直接或间接的、任何伤害或损失。

请参照交付物中说明文档对紫光展锐交付物进行使用，任何人对紫光展锐交付物的修改、定制化或违反说明文档的指引对紫光展锐交付物进行使用造成的任何损失由其自行承担。紫光展锐交付物中的性能指标、测试结果和参数等，均为在紫光展锐内部研发和测试系统中获得的，仅供参考，若任何人需要对交付物进行商用或量产，需要结合自身的软硬件测试环境进行全面的测试和调试。非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

Unisoc Confidential For hiar

# 紫光展锐科技有限公司



# 前言

## 概述

本文档主要介绍基于 Android 9.0+Kernel4.4、Android 10.0+Kernel4.14 和 Android R+Kernel4.14，客制化配置 Touch Panel，以及客制化项提示和一些常见问题的定位方法。

## 读者对象


本文档主要适用于需要客制化配置 Touch Panel 的开发人员。

## 缩略语

缩略语	英文全名	中文解释
TP	Touch Panel	触摸屏
DTS	Device Tree Source	设备树源码
KO	Kernel Object	内核模块

## 符号约定

在本文中可能出现下列标志，它所代表的含义如下。

符号	说明
 说明	用于突出重要/关键信息、补充信息和小窍门等。 “说明”不是安全警示信息，不涉及人身、设备及环境伤害。

## 变更信息

文档版本	发布日期	修改说明
V1.0	2019-05-20	初稿

文档版本	发布日期	修改说明
V1.1	2020-03-02	增加 Android 10.0 客制化配置 Touch Panel。
V1.2	2020-05-09	更新模板格式和内容优化。
V1.3	2020-08-07	主要是增加 2.3 节

## 关键字

Touch Panel、TP、客制化。

Unisoc Confidential For hiar

# 目 录

1 TP 工作流程 .....	1
2 TP 客制化配置.....	3
2.1 基于 Android 9.0+Kernel4.4.....	3
2.1.1 集成驱动文件.....	3
2.1.2 配置 Makefile/Kconfig/Defconfig .....	3
2.1.3 添加 TP 设备信息到 DTS .....	7
2.2 基于 Android 10.0+Kernel4.14.....	8
2.2.1 集成驱动文件.....	8
2.2.2 添加 KO 模块到工程 .....	10
2.2.3 动态加载 KO 模块.....	12
2.2.4 添加 TP 的休眠唤醒节点 .....	12
2.2.5 编译及调试技巧 .....	14
2.3 基于 Android R+Kernel4.14 .....	14
2.3.1 集成驱动文件.....	14
2.3.2 添加 KO 模块到工程 .....	14
2.3.3 动态加载 KO 模块.....	15
2.3.4 添加 TP 的休眠唤醒节点 .....	16
2.3.5 编译及调试技巧 .....	16
3 TP 客制化项提示 .....	17
3.1 TP 手势唤醒.....	17
3.2 TP 距感功能.....	17
3.3 TP 固件升级.....	17
4 常见问题.....	18

## 图目录

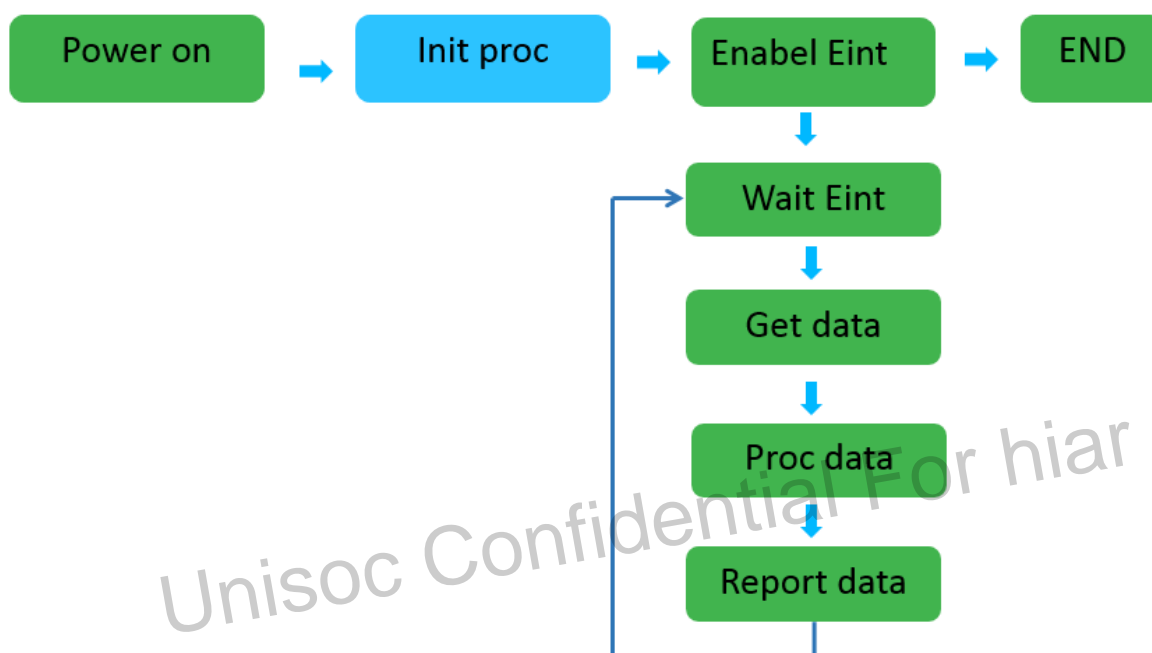
图 1-1 TP 的工作流程 .....	1
图 1-2 固件升级流程 .....	2
图 2-1 驱动文件路径 .....	3
图 2-2 Kconfig 文件 .....	5
图 2-3 Defconfig 配置界面 .....	6
图 2-4 Defconfig 文件修改前后比对 .....	7
图 2-5 DTS 文件中 TP 设备信息 .....	7

Unisoc Confidential For hiar

# 1 TP 工作流程

TP 硬件的初始化在 Kernel 启动阶段，基于 Linux 内核的 TP 工作流程如图 1-1。

图1-1 TP 的工作流程



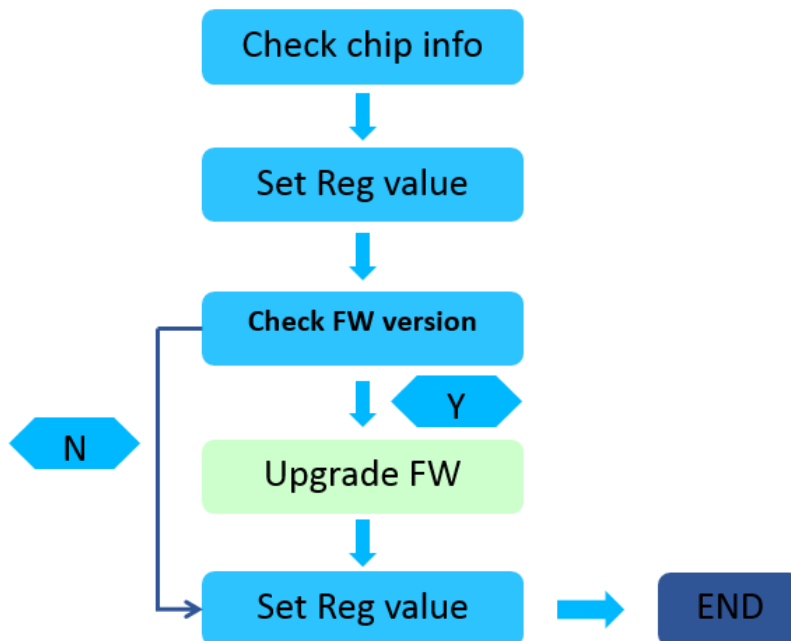
## Power on

上电：在 Power on 之前需要申请一些平台硬件资源，如 I2C 资源、GPIO 资源和中断资源等，然后进行 TP 的上电。一般在上电以后需要对 TP 做一次硬件的复位，即将 TP 的 reset 引脚进行一次“高低高”电平的处理。

## Init proc

初始化：这一部分主要进行一些寄存器的初始化，检测是否进行固件升级，注册 input 设备等，固件升级流程如图 1-2。

图1-2 固件升级流程



## Enable\_Eint

使能中断：TP 被触摸后，会触发一个外部中断，驱动的中断处理函数会向 TP 的工作队列添加一个新的任务。这个任务是通过 I2C 读取坐标以及其他相关原始数据，然后将其处理成坐标信息，并通过 input 子系统将数据上报给 framework 层。



# 2 TP 客制化配置

本章主要介绍如何进行 TP 客制化配置。

## 2.1 基于 Android 9.0+Kernel4.4

### 2.1.1 集成驱动文件

作为 input 设备，相应驱动文件均需放在 kernel4.4/drivers/input/touchscreen 目录下，如图 2-1。

图2-1 驱动文件路径

```
~/workspace/projects/sprdroid9.0_trunk/kernel4.4/drivers/
input/touchscreen$ ls
88pm860x-ts.c      cyttsp_spi.c      lpc32xx_ts.c      touchright.c
ad7877.c           da9034-ts.c       mainstone-wm97xx.c touchwin.c
ad7879.c           da9052_tsi.c      Makefile           tps6507x-ts.c
ad7879.h           dynapro.c         max11801_ts.c      tsc2004.c
ad7879-i2c.c       edt-ft5x06.c      mc13783_ts.c       tsc2005.c
ad7879-spi.c       eeti_ts.c         mcs5000_ts.c       tsc2007.c
adaptive-ts        egalax_ts.c       migor_ts.c         tsc200x-core.c
ads7846.c          elants_i2c.c      mk712.c            tsc200x-core.h
arl021_i2c.c       elo.c             mms114.c           tsc40.c
atmel_mxt_ts.c     ft6236.c          msg2xxx_ts         ucb1400_ts.c
atmel-wm97xx.c     fujitsu_ts.c     mtouch.c           usbtouchscreen.c
auo-pixcir-ts.c    goodix5           of_touchscreen.c   w90p910_ts.c
bcm_iproc_tsc.c    goodix.c          pcap_ts.c          wacom_i2c.c
bu21013_ts.c       gt1x_ts           penmount.c         wacom_w8001.c
chipone_icn8318.c  gunze.c           pixcir_i2c_ts.c    wdt87xx_i2c.c
colibri-vf50-ts.c  hampshire.c       rohm_bu21023.c     wm831x-ts.c
cy8ctmg110_ts.c    hp680_ts_input.c s3c2410_ts.c       wm9705.c
cyttsp4_core.c     htcpen.c          st1232.c           wm9712.c
cyttsp4_core.h     ili210x.c         stmpe-ts.c         wm9713.c
cyttsp4_i2c.c       imx6ul_tsc.c      sun4i-ts.c         wm97xx-core.c
cyttsp4_spi.c       inextio.c         sur40.c            zforce_ts.c
cyttsp_core.c       intel-mid-touch.c sx8654.c           zylonite-wm97xx.c
cyttsp_core.h       ipaq-micro-ts.c   synaptics_dsx      touchit213.c
cyttsp_i2c.c        jornada720_ts.c   ti_am335x_tsc.c
cyttsp_i2c_common.c Kconfig
```

#### 说明

Kernel4.4 上展锐建议用静态加载方式。

### 2.1.2 配置 Makefile/Kconfig/Defconfig

下面以 synaptics 为例，描述如何进行 Makefile/Kconfig/Defconfig 的配置。

## Makefile

修改 kernel4.4/drivers/input/touchscreen/Makefile 文件，添加 TP 驱动对应的 Makefile，增加下图中红色下划线一行。

```
obj-$(CONFIG_TOUCHSCREEN_TPS6507X) += tps6507x-ts.o
obj-$(CONFIG_TOUCHSCREEN_ZFORCE) += zforce_ts.o
obj-$(CONFIG_TOUCHSCREEN_COLIBRI_VF50) += colibri-vf50-ts.o
obj-$(CONFIG_TOUCHSCREEN_ROHM_BU21023) += rohm_bu21023.o
obj-$(CONFIG_TOUCHSCREEN_ADAPTIVE) += adaptive-ts/
obj-$(CONFIG_TOUCHSCREEN_SYNAPTICS_DSX) += synaptics_dsx/
obj-$(CONFIG_TOUCHSCREEN_GOODIX5) += goodix5/
obj-$(CONFIG_TOUCHSCREEN_GT1X_TS) += gt1x_ts/
obj-$(CONFIG_TOUCHSCREEN_MSG2XXX_TS) += msg2xxx_ts/
```

## Kconfig

修改 kernel4.4/drivers/input/touchscreen/Kconfig 文件，将 synaptics 驱动的 Kconfig 添加进来，增加下图中红色下划线一行。

```
source drivers/input/touchscreen/adaptive-ts/Kconfig
source drivers/input/touchscreen/synaptics_dsx/Kconfig
```

kernel4.4/drivers/input/touchscreen/synaptics\_dsx/Kconfig 文件中的内容如图 2-2，里面有多个 config，需要在 Defconfig 中进行配置。

图2-2 Kconfig 文件

```
#
# Synaptics DSX touchscreen driver configuration
#
menuconfig TOUCHSCREEN_SYNAPTICS_DSX
    bool "Synaptics DSX touchscreen"
    default n
    help
        Say Y here if you have a Synaptics DSX I2C touchscreen
        connected to your system

        if unsure, say N.

if TOUCHSCREEN_SYNAPTICS_DSX
config TOUCHSCREEN_SYNAPTICS_DSX_I2C
    tristate "Synaptics DSX core driver module"
    depends on I2C
    help
        Say Y here if you have a Synaptics DSX I2C touchscreen
        connected to your system

        If unsure, say N.

        To compile this driver as a module, choose M here: the
        module will be called synaptics_dsx_i2c.

config TOUCHSCREEN_SYNAPTICS_DSX_RMI_DEV
    tristate "Synaptics DSX RMI device module"
    depends on TOUCHSCREEN_SYNAPTICS_DSX_I2C
    help
        Say Y here to enable support for direct RMI register access.

        If unsure, say N.

        To compile this driver as a module, choose M here: the
        module will be called synaptics_dsx_rmi_dev.

config TOUCHSCREEN_SYNAPTICS_DSX_FW_UPDATE
    tristate "Synaptics DSX firmware update module"
    depends on TOUCHSCREEN_SYNAPTICS_DSX_I2C
    help
        Say Y here to enable support for doing firmware update.

        If unsure, say N.

        To compile this driver as a module, choose M here: the
        module will be called synaptics_dsx_fw_update.
```

## Defconfig

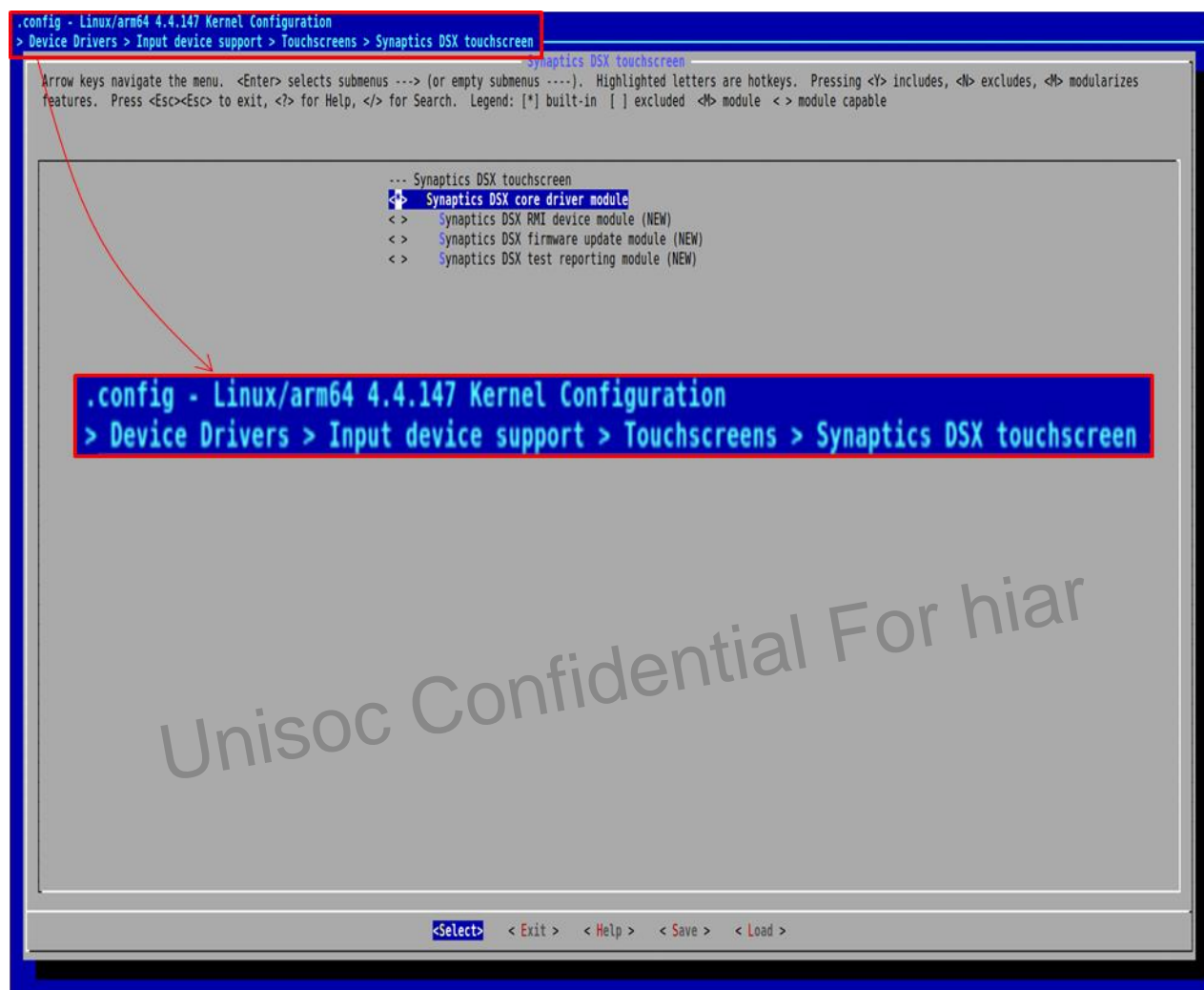
Defconfig 配置步骤如下：

步骤 1 lunch 对应的 board，以 s9863a3h10\_Natv-userdebug-native 工程为例：lunch

s9863a3h10\_Natv-userdebug-native（这里必须先选择项目）。

步骤 2 输入 kuconfig 命令进行 config 的配置，使用键盘逐级选择 Device driver > Input device support > Touchscreen > Synaptics DSX touchscreen，配置界面如下图 2-3。

图2-3 Defconfig 配置界面



步骤 3 使用键盘操作“Save”保存后退出。

步骤 4 检查 Defconfig 是否修改成功，使用“git diff”对比修改前后的 Defconfig 文件，如图 2-4 所示，图 2-4 中红色线框部分表示新的驱动已经添加到内核。

图2-4 Defconfig 文件修改前后比对

```
xingxing.luo@sh08839pcu:~/workspace/projects/sprdroid9.0_trunk/kernel4.4/arch/arm64/configs$ git diff sprd_sharkl3_defconfig
diff --git a/arch/arm64/configs/sprd_sharkl3_defconfig b/arch/arm64/configs/sprd
index 75abb25..cb5366a 100644
--- a/arch/arm64/configs/sprd_sharkl3_defconfig
+++ b/arch/arm64/configs/sprd_sharkl3_defconfig
@@ -1505,7 +1505,11 @@ CONFIG_INPUT_TOUCHSCREEN=y
 CONFIG_TOUCHSCREEN_ADAPTIVE=y
 CONFIG_TOUCHSCREEN_ADAPTIVE_VENDOR_FOCALTECH=y
 # CONFIG_TOUCHSCREEN_ADAPTIVE_VENDOR_MSTAR is not set
-# CONFIG_TOUCHSCREEN_SYNAPTICS_DSX is not set
+CONFIG_TOUCHSCREEN_SYNAPTICS_DSX=y
+CONFIG_TOUCHSCREEN_SYNAPTICS_DSX_I2C=y
+# CONFIG_TOUCHSCREEN_SYNAPTICS_DSX_RMI_DEV is not set
+# CONFIG_TOUCHSCREEN_SYNAPTICS_DSX_FW_UPDATE is not set
+# CONFIG_TOUCHSCREEN_SYNAPTICS_DSX_TEST_REPORTING is not set
 CONFIG_TOUCHSCREEN_PROPERTIES=y
 # CONFIG_TOUCHSCREEN_ADS7846 is not set
 # CONFIG_TOUCHSCREEN_AD7877 is not set
(END)
```

----结束

## 2.1.3 添加 TP 设备信息到 DTS

在 DTS 文件中添加 TP 设备信息，如图 2-5。

图2-5 DTS 文件中 TP 设备信息

```
&i2c3 {
    status = "okay";

    synaptics-touch@20 {
        compatible = "synaptics,dx";
        reg = <0x20>;
        gpios = <&ap_gpio 145 GPIO_ACTIVE_HIGH
                &ap_gpio 144 GPIO_ACTIVE_HIGH>;
    };
};
```

图 2-5 中，DTS 中描述了 TP 相关的硬件信息：

- TP 连接在 I2C-3 上
- I2C 从设备地址为 0x20
- 对应的中断和 reset pin

## 2.2 基于 Android 10.0+Kernel4.14

Android 10.0+Kernel4.14 上对 input 设备驱动集成不同于之前，不再以静态加载的方式编译进内核，要求以 KO 的形式动态加载到内核，因此，代码位置以及编译规则均与之前不同，但是，对于 TP 设备描述与 Android 9.0+Kernel4.4 完全相同，只是驱动部分有所变动。

### 2.2.1 集成驱动文件

驱动文件需存放在 bsp/modules/input/touchscreen/目录下，下面还是以 **synaptic** 为例，举例如何添加一款 TP 驱动。

步骤 1 添加 Android.mk 文件（红色下划线是新添加 KO 文件名字）

```
LOCAL_PATH:= $(call my-dir)

include $(CLEAR_VARS)
LOCAL_MODULE_TAGS := optional
LOCAL_MODULE := synaptics_dsx_td4310.ko
LOCAL_MODULE_CLASS := SHARED_LIBRARIES
LOCAL_MODULE_PATH := $(TARGET_OUT_VENDOR)/lib/modules
LOCAL_STRIP_MODULE := keep_symbols
LOCAL_SRC_FILES := $(LOCAL_MODULE)
include $(BUILD_PREBUILT)

ifeq ($(TARGET_BUILD_VARIANT),user)
DEBUGMODE := BUILD=no
else
DEBUGMODE := $(DEBUGMODE)
endif

#convert to absolute directory
PRODUCT_OUT_ABSOLUTE:=$(shell cd $(PRODUCT_OUT); pwd)

$(LOCAL_PATH)/synaptics_dsx_td4310.ko: $(TARGET_PREBUILT_KERNEL)
$(MAKE) -C $(shell dirname $@) ARCH=$(TARGET_KERNEL_ARCH) CROSS_COMPILE=$(KERNEL_CROSS_COMPILE) $(DEBUGMODE) KDIR=$(PRODUCT_OUT_ABSOLUTE)/obj/KERNEL clean
$(MAKE) -C $(shell dirname $@) ARCH=$(TARGET_KERNEL_ARCH) CROSS_COMPILE=$(KERNEL_CROSS_COMPILE) $(DEBUGMODE) KDIR=$(PRODUCT_OUT_ABSOLUTE)/obj/KERNEL
$(TARGET_STRIP) -d --strip-unneeded $@
```

步骤 2 添加 Kbuild 文件（红色下划线是新添加 KO 文件名字）



```
#
# synaptics_dsx_td4310.ko
#
# Kbuild: for kernel building external module
#
# Note:
# - Please refer to modules/sample/Kbuild to find out what should be
#   done is this Kbuild file
#

#
# Source List
#
KO_MODULE_NAME := synaptics_dsx_td4310
KO_MODULE_PATH := $(src)
KO_MODULE_SRC :=

KO_MODULE_SRC += $(wildcard $(KO_MODULE_PATH)/*.c)

#
# Build Options
#
ccflags-y += -DDEBUG

#
# Final Objects
#
obj-m := $(KO_MODULE_NAME).o
# Comment it if the only object file has the same name with module
$(KO_MODULE_NAME)-y := $(patsubst $(src)/%.c,%.o,$(KO_MODULE_SRC))
```

步骤3 修改 Makefile 文件（红色下划线是新添加 KO 文件名字）

```
#
# synaptics_dsx_td4310.ko
#
# Makefile: for external make invocation
#
# Note:
# - Please refer to modules/sample/Makefile to find out what should be
#   done in this Makefile
#

KO_MODULE_NAME := synaptics_dsx_td4310
KO_MODULE_OUT := $(BSP_MODULES_OUT)/$(KO_MODULE_NAME)
KO_MODULE_KBUILD := $(CURDIR)/Kbuild

.PHONY: modules modules_install clean

modules:
    @mkdir -p $(KO_MODULE_OUT) && ln -snf $(KO_MODULE_KBUILD) $(KO_MODULE_OUT)/Kbuild
    @ln -snf $(CURDIR) $(KO_MODULE_OUT)/source
    $(MAKE) -C $(BSP_KERNEL_PATH) M=$(KO_MODULE_OUT) src=$(CURDIR) $@

modules_install:
    $(MAKE) -C $(BSP_KERNEL_PATH) M=$(KO_MODULE_OUT) $@

# Remove the out directory wholly
clean:
    @$(MAKE) -C $(BSP_KERNEL_PATH) M=$(KO_MODULE_OUT) src=$(CURDIR) $@
    rm -rf $(KO_MODULE_OUT)
```

## 说明

一份驱动文件中只允许有一个 module\_init。

----结束

## 2.2.2 添加 KO 模块到工程

下面以 ums512\_1h10 为例描述如何添加 KO 模块到工程。

### 步骤 1 添加 KO 文件到 modules.cfg

在 bsp/device/sharkl5Pro/androidq/ums512\_1h10/ums512\_1h10\_base/modules.cfg 中添加对应的.ko 文件名，如下图中红色下划线所示。



```

BSP_MODULES_LIST="
flash_ic_sc2703.ko
gpio_test.ko
mali_gondul.ko
sprd_sensor.ko
sprd_camera.ko
sprd_cpp.ko
sprd_fd.ko
sprd_flash_drv.ko
flash_ic_ocp8137.ko
sprdwl_ng.ko
synaptics_dsx_td4310.ko
sprdbt_tty.ko
sprd_fm.ko
microarray_fp.ko
mmdvfs.ko
tcs3430.ko
stmvl5310.ko
sprd_vdsp.ko
vdsp_sipc.ko
vdsp_spipe.ko
akm4377.ko
ssd20xx_ts.ko
"
  
```

## 步骤 2 添加 KO 文件到系统编译中

在 `device/sprd/sharkl5Pro/ums512_1h10/ums512_1h10_Naty.mk` 中添加 KO 文件，如下图中红色下划线所示。

```

PRODUCT_SOCKO_KO_LIST := \
$(BSP_KERNEL_MODULES_OUT)/flash_ic_ocp8137.ko \
$(BSP_KERNEL_MODULES_OUT)/mali_gondul.ko \
$(BSP_KERNEL_MODULES_OUT)/sprd_flash_drv.ko \
$(BSP_KERNEL_MODULES_OUT)/sprdwl_ng.ko \
$(BSP_KERNEL_MODULES_OUT)/flash_ic_sc2703.ko \
$(BSP_KERNEL_MODULES_OUT)/sprdbt_tty.ko \
$(BSP_KERNEL_MODULES_OUT)/sprd_fm.ko \
$(BSP_KERNEL_MODULES_OUT)/sprd_sensor.ko \
$(BSP_KERNEL_MODULES_OUT)/sprd_camera.ko \
$(BSP_KERNEL_MODULES_OUT)/sprd_fd.ko \
$(BSP_KERNEL_MODULES_OUT)/sprd_cpp.ko \
$(BSP_KERNEL_MODULES_OUT)/mmdvfs.ko \
$(BSP_KERNEL_MODULES_OUT)/tcs3430.ko \
$(BSP_KERNEL_MODULES_OUT)/stmvl5310.ko \
$(BSP_KERNEL_MODULES_OUT)/sprd_vdsp.ko \
$(BSP_KERNEL_MODULES_OUT)/vdsp_sipc.ko \
$(BSP_KERNEL_MODULES_OUT)/vdsp_spipe.ko \
$(BSP_KERNEL_MODULES_OUT)/synaptics_dsx_td4310.ko \
$(BSP_KERNEL_MODULES_OUT)/ssd20xx_ts.ko
  
```

----结束

## 2.2.3 动态加载 KO 模块

在 device/sprd/sharkl5Pro/common/rootdir/root/init.common.rc 中，添加加载 KO 的命令，如下图中红色下划线所示。

```
on post-fs
    insmod ${ro.vendor.ko.mount.point}/socko/mali_gondul.ko
    #insmode synaptics TP ko
    insmod ${ro.vendor.ko.mount.point}/socko/synaptics_dsx_td4310.ko
    insmod ${ro.vendor.ko.mount.point}/socko/akm4377.ko
```

### 说明

对于具体的 board，需要在各自 board 的 init.rc 文件中添加。

## 2.2.4 添加 TP 的休眠唤醒节点

Kernel4.14 以后，由于 display 架构的变化，display 不再通知 TP 进行 resume/suspend，为此，展锐平台单独做了一套 TP 休眠的唤醒的流程，驱动层需要做以下处理：

1. 在 TP 的驱动文件中创建文件节点/sys/touchscreen/ts\_suspend
2. 实现 ts\_suspend 对应的 xxx\_store

### TP 的休眠与唤醒

TP 的休眠与唤醒是通过写节点的方式进行控制：

- 往节点写“0”，表示退出睡眠。
- 往节点写“1”，表示进入睡眠。

### 说明

这一套休眠唤醒的方案适用于 TP PM runtime 机制无法满足所有使用场景的情况。

### 举例说明

步骤 1 创建/sys/touchscreen/目录。

```
err= sysfs_create_link(NULL, &client->dev.kobj, "touchscreen");
if (err < 0) {
    pr_err("%s sysfs_create_link failed!\n", __func__);

    goto class_create_failed;
}
```

步骤 2 创建 ts\_suspend 节点，增加下图中红色下划线一行。

```
static struct attribute *ssl_attrs[] = {
    &dev_attr_version.attr,
#ifdef SUPPORT_LPM
    &dev_attr_gesture.attr,
#endif /* SUPPORT_LPM */
    &dev_attr_testing.attr,
    &dev_attr_ssdtouch.attr,
    &dev_attr_touchmode.attr,
    &dev_attr_mptest.attr,
    &dev_attr_esdtime.attr,
    &dev_attr_fail_reason.attr,
    &dev_attr_ts_suspend.attr,
    NULL,
};
```

步骤3 实现节点的回调函数，在 store 回调函数中根据传进来的 input 参数，调用 TP 的 resume/suspend 接口，如下图红色线框所示。

```
static ssize_t solomon_suspend_store(struct device *dev,
    struct device_attribute *attr, const char *buf, size_t count)
{
    unsigned int input;

    if (kstrtouint(buf, 10, &input))
        return -EINVAL;

    if (input == 1)
        solomon_suspend_ex();
    else if (input == 0)
        solomon_resume_ex();
    else
        return -EINVAL;

    return count;
}

static ssize_t solomon_suspend_show(struct device *dev,
    struct device_attribute *attr, char *buf)
{
    struct solomon_device *ftdev = dev_get_drvdata(dev);

    return snprintf(buf, PAGE_SIZE, "%s\n", ftdev->is_suspend ? "suspend" : "resume");
}

static DEVICE_ATTR(ts_suspend, 0664, solomon_suspend_show, solomon_suspend_store);
```

----结束

## 2.2.5 编译及调试技巧

### 单独编译 KO 模块

由于 Android 10.0 及以上版本，BSP 可以单独编译，因此，在 bsp 目录下执行 source 和 lunch 操作之后，可以单独编译 KO 模块，命令如下：

```
make modules -m synaptics_dsx_td4310.ko
```

编译成功会在 bsp/out/ums512\_1h10/obj/modules 下生成对应的 KO 文件。

### 避免全编

由于修改了 device 部分，因此会涉及到全编，但是 device 部分的修改只是在开机时自动加载.ko 文件，因此可以用其他方式代替，避免全编，步骤如下：

步骤 1 添加一个 init.synaptics\_dsx\_td4310.rc 文件，文件里的内容如下：

```
on post-fs
    insmod /mnt/vendor/socko/synaptics_dsx_td4310.ko
```

步骤 2 依次执行以下命令：

```
adb root
adb remount
adb push synaptics_dsx_td4310.ko /mnt/vendor/socko
adb push init.synaptics_dsx_td4310.rc /vendor/etc/init
adb reboot
```

#### 说明

手机需要先用解锁工具解锁，否则/mnt/vendor/socko 区域无法 mount 成功。

----结束

## 2.3 基于 Android R+Kernel4.14

### 2.3.1 集成驱动文件

在 Android R 上，Kernel4.14 的 TP 驱动文件放在 bsp/modules/kernel4.14/input/touchscreen/目录下。集成驱动文件的步骤与 Android 10.0 的一致，请参考 2.2.1 集成驱动文件。

### 2.3.2 添加 KO 模块到工程

下面以 ums512\_1h10 为例，描述如何添加 KO 模块到工程。

步骤 1 添加 KO 文件到 modules.cfg

在/bsp/device/sharkl5Pro/androidr/common/modules.cfg 中添加对应的.ko 文件名，如下图中红色下划线所示。

```

9  if [ $BSP_KERNEL_VERSION == "kernel4.14" ]; then
10     BSP_MODULES_LIST="
11         flash_ic_sc2703.ko
12         gpio_test.ko
13         mali_gondul.ko
14         sprd_sensor.ko
15         sprd_camera.ko
16         sprd_cpp.ko
17         sprd_fd.ko
18         sprd_flash_drv.ko
19         flash_ic_ocp8137.ko
20         sprdwl_ng.ko
21         synaptics_dsx_td4310.ko
22         sprdbt_tty.ko
23         sprd_fm.ko
24         microarray_fp.ko
25         mmdvfs.ko
26         tcs3430.ko
27         stmv15310.ko
28         sprd_vdsp.ko
29         vdsp_sipc.ko
30         vdsp_spipe.ko
31         akm4377.ko
32         ssd20xx_ts.ko
33     "
34 elif [ $BSP_KERNEL_VERSION == "kernel5.4" ]; then
35     BSP_MODULES_LIST="
36         sample.ko
37         synaptics_dsx_td4310.ko
38     "
39 fi

```

## 步骤 2 添加 KO 文件到系统编译中

在/device/sprd/sharkl5Pro/ums512\_1h10/module/bsp/mfeature/kernel/kernel4.14/kernel4.14.mk 中添加 KO 文件，如下图中红色下划线所示。

```

PRODUCT_SOCKO_KO_LIST += \
    $(BSP_KERNEL_MODULES_OUT)/flash_ic_ocp8137.ko \
    $(BSP_KERNEL_MODULES_OUT)/flash_ic_sc2703.ko \
    $(BSP_KERNEL_MODULES_OUT)/tcs3430.ko \
    $(BSP_KERNEL_MODULES_OUT)/stmv15310.ko \
    $(BSP_KERNEL_MODULES_OUT)/synaptics_dsx_td4310.ko \
    $(BSP_KERNEL_MODULES_OUT)/ssd20xx_ts.ko

```

----结束

## 2.3.3 动态加载 KO 模块

在/device/sprd/sharkl5Pro/ums512\_1h10/module/bsp/mfeature/kernel/kernel4.14/kernel4.14.rc 中，添加加载 KO 的命令，如下图中红色下划线所示。

```
on post-fs
    insmod ${ro.vendor.ko.mount.point}/socko/stmv15310.ko
    insmod ${ro.vendor.ko.mount.point}/socko/tcs3430.ko
    insmod ${ro.vendor.ko.mount.point}/socko/akm4377.ko
    #insmode synaptics TP ko
    insmod ${ro.vendor.ko.mount.point}/socko/synaptics_dsx_td4310.ko
    insmod ${ro.vendor.ko.mount.point}/socko/ssd20xx_ts.ko
```

#### 📖 说明

对于具体的 board，需要在各自 board 的 kernel4.14.rc 文件中添加。

## 2.3.4 添加 TP 的休眠唤醒节点

请参考 2.2.4 添加 TP 的休眠唤醒节点

## 2.3.5 编译及调试技巧

请参考 2.2.5 编译及调试技巧。

Unisoc Confidential For hiar

# 3

## TP 客制化项提示

---

本章主要给出一些常用 TP 客制化项提示。

### 3.1 TP 手势唤醒

- 需要 TP 修改固件，当有手势时，会上报中断，展锐平台对于中断时间的要求是 200ms 以上。
- 手势唤醒时其 irq type 必须包含 IRQF\_NO\_SUSPEND 标志位，即深睡后，在 lateresume 中把 irq type 切换成高电平+NOSUSPEND，有中断触发后，会将中断拉低 200ms，唤醒系统，再将 irq type 置成原来的触发方式。
- TP 手势唤醒中断脚 GPIO 配置参考以下配置：{REG\_PIN\_EXTINT0, PIN\_NULL, FUNC3, PIN\_NULL, PIN\_NULL, DS\_L1, SLP\_IE, PIN\_NULL, SLP\_AP}。

### 3.2 TP 距感功能

即 TP 模拟 Psensor，Hal 层无需改动，只需修改 TP 驱动文件，在 TP 驱动中添加 TP 距感功能的相关功能代码。

### 3.3 TP 固件升级

TP 固件升级的方法主要分为以下两种：

- 将固件集成到驱动文件中，以.h 形式或者以数组的形式存放。
- 使用 request\_firmware 接口申请 firmware，在加载驱动的时候进行 load。

# 4

## 常见问题

---

1. **问题描述：**开机 log 显示 TP 进行 I2C 通讯时出现 NACK。

**问题分析及解决方法：**

- 检查 I2C number 是否配置正确。
- 确认 TP 上电正常，reset pin 处于高电平状态。

2. **问题描述：**开机后触摸 TP 无响应。

**问题分析及解决方法：**

- 确认有中断信号，且驱动里的中断配置正确。
- 通过命令 `adb shell getevent`，触摸屏幕看是否有事件上报，如果有事件上报，查看上报的坐标是否正常，上报数据包是否正常。

Unisoc Confidential For hiar