SIM8500_Sensor_Porting_Guide_文档_V1.01文档

芯讯通无线科技(上海)有限公司
上海市长宁区临虹路289号3号楼芯讯通总部大楼
电话：86-21-31575100
技术支持邮箱：support@simcom.com
官网：www.simcom.com

名称：	SIM8500_Sensor_Porting_Guide_文档_V1.01
版本：	1.01
日期：	2022.03.09
状态：	已发布

版权声明

本手册包含芯讯通无线科技（上海）有限公司（简称：芯讯通）的技术信息。除非经芯讯通书面许可，任何单位和个人不得擅自摘抄、复制本手册内容的部分或全部，并不得以任何形式传播，违反者将被追究法律责任。对技术信息涉及的专利、实用新型或者外观设计等知识产权，芯讯通保留一切权利。芯讯通有权在不通知的情况下随时更新本手册的具体内容。

本手册版权属于芯讯通，任何人未经我公司书面同意进行复制、引用或者修改本手册都将承担法律责任。

芯讯通无线科技(上海)有限公司
上海市长宁区临虹路289号3号楼芯讯通总部大楼
电话：86-21-31575100
邮箱：simcom@simcom.com
官网：www.simcom.com

了解更多资料，请点击以下链接：
http://cn.simcom.com/download/list-230-cn.html

技术支持，请点击以下链接：
http://cn.simcom.com/ask/index-cn.html 或发送邮件至 support@simcom.com

关于文档

版本历史

版本	日期	作者	备注
1.00	2020.8.17	李玉龙	第一版
1.01	2022.03.09	伍文祥	更新版本

适用范围

本文档适用于 SIMCom SIM8500 系列。

目录

1 介绍

1.1 本文目的

基于 SIM8500 平台(展锐 sl8541e_1h10_32b)， 对 sensor 驱动移植简单介绍。
参考此应用文档，开发者可以很快理解并快速开发相关业务。

1.2 参考文档

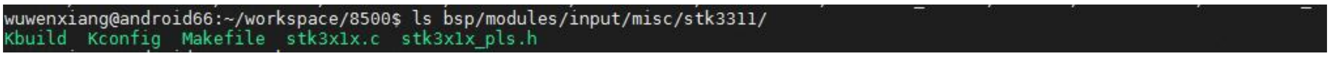
30727_SC2721GDeviceSpecification_V0.3.pdf
30969_SL8541E_GPIO_Spec_V1.1.xlsx

1.3 术语和缩写

2 驱动移植

移植 sensor 驱动代码，例如 光距感 sensor stk3311：

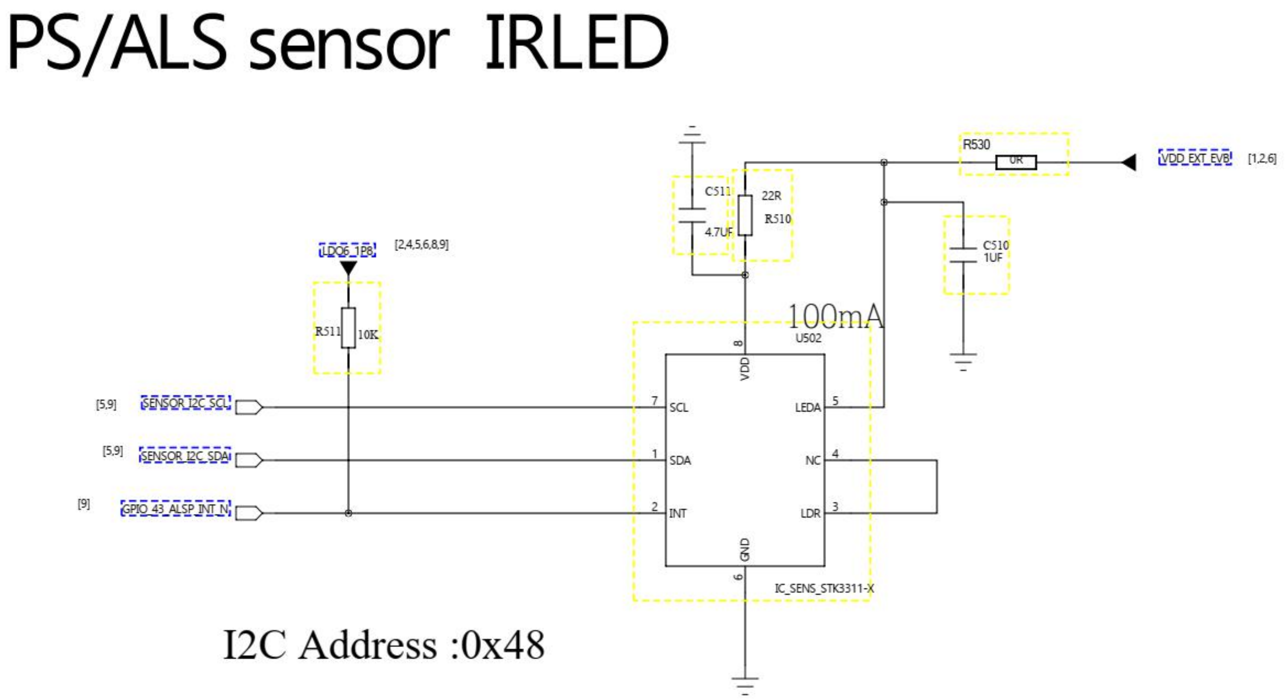
A、将供应商提供的代码拷贝到图示目录

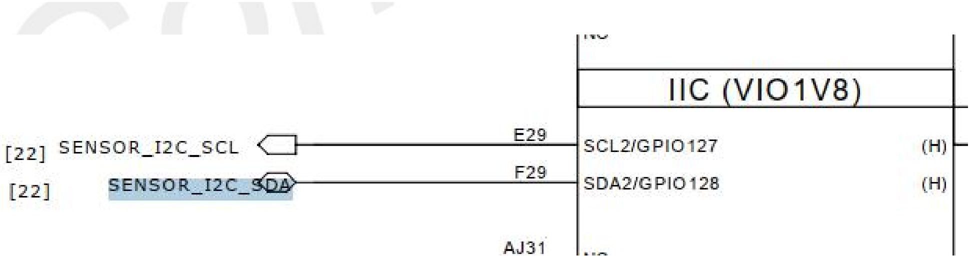


```
wuwenxiang@android66:~/workspace/8500$ ls bsp/modules/input/misc/stk3311/
Kbuild  Kconfig  Makefile  stk3x1x.c  stk3x1x_pls.h
```

B、观察原理图，配置 device tree

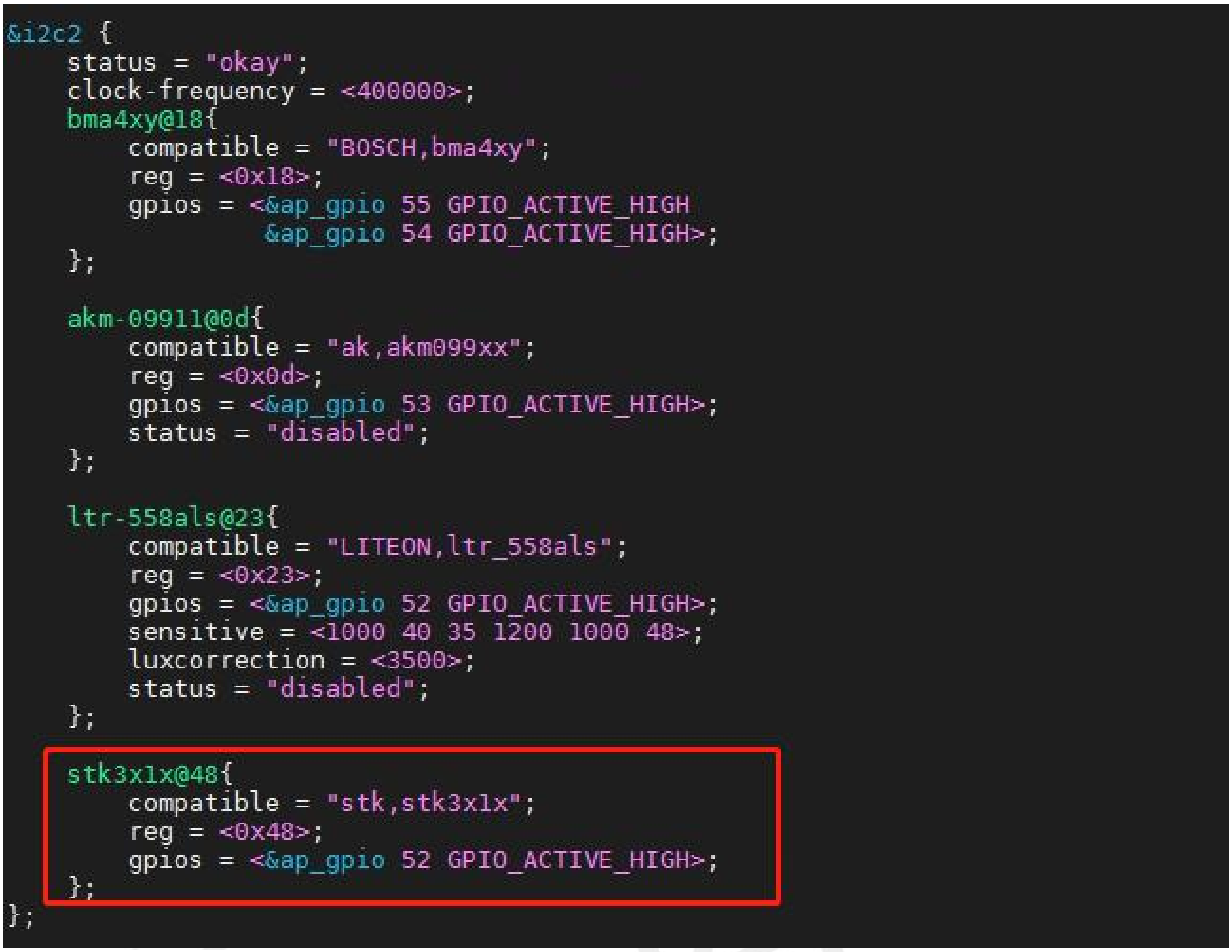
首先从硬件上看，这个器件挂载到哪组 I2C 总线上，然后配置该设备的 dts





从原理图可知该 sensor 是挂载到 I2C2 总线上，在 SIM8500 平台需要修改的文件：

bsp/kernel/kernel4.14/arch/arm/boot/dts/sl8541e-1h10_32b-overlay.dts



```
&i2c2 {
	status = "okay";
	clock-frequency = <400000>;
	bma4xy@18{
		compatible = "BOSCH,bma4xy";
		reg = <0x18>;
		gpios = <&ap_gpio 55 GPIO_ACTIVE_HIGH
			 &ap_gpio 54 GPIO_ACTIVE_HIGH>;
	};

	akm-09911@0d{
		compatible = "ak,akm099xx";
		reg = <0x0d>;
		gpios = <&ap_gpio 53 GPIO_ACTIVE_HIGH>;
		status = "disabled";
	};

	ltr-558als@23{
		compatible = "LITEON,ltr_558als";
		reg = <0x23>;
		gpios = <&ap_gpio 52 GPIO_ACTIVE_HIGH>;
		sensitive = <1000 40 35 1200 1000 48>;
		luxcorrection = <3500>;
		status = "disabled";
	};

	stk3x1x@48{
		compatible = "stk,stk3x1x";
		reg = <0x48>;
		gpios = <&ap_gpio 52 GPIO_ACTIVE_HIGH>;
	};
};
```

```
stk3x1x@48{//sensor_name@addr
    compatible = "stk,stk3x1x";//compatible 与驱动中设置的compatible对应
    reg = <0x48>;//芯片地址
    gpios = <&ap_gpio 52 GPIO_ACTIVE_HIGH>;//中断pin，该驱动中没有用到，使用的轮询机制，可
以不配，如果使用中断模式，就需要配置一下
};
```

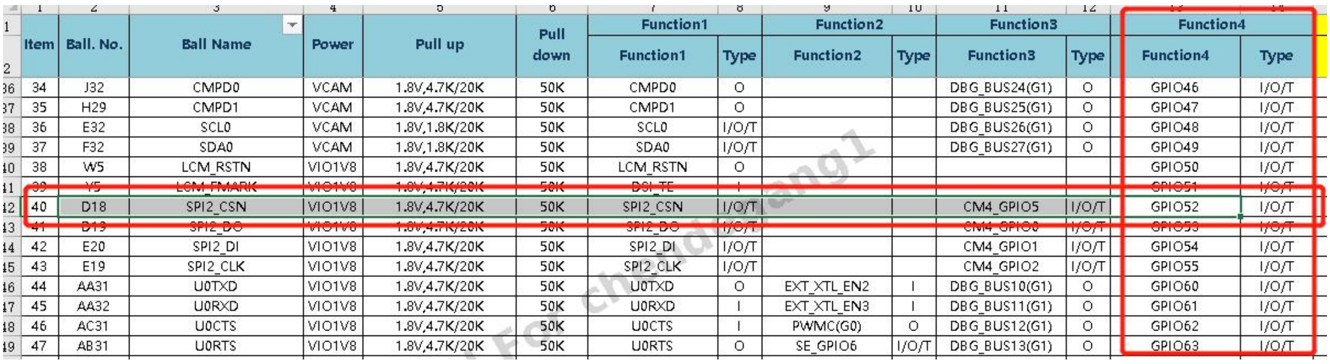
如果需要配置 gpio，需要在 Uboot 里面配置一下，配置路径：

bsp/bootloader/u-boot15/board/spreadtrum/sl8541e_1h10_32b/pinmap-sl8541e.c

通过查询 30969_SL8541E_GPIO_Spec_V1.1.xlsx

30969_SL8541E_
GPIO_Spec_V1.1.›

例如该驱动中配置 gpio52 为输入中断脚，查询该 excel



Item	Ball. No.	Ball Name	Power	Pull up	Pull down	Function1		Function2		Function3		Function4	
						Function1	Type	Function2	Type	Function3	Type	Function4	Type
34	J32	CMPD0	VCAM	1.8V,4.7K/20K	50K	CMPD0	O			DBG_BUS24(G1)	O	GPIO46	I/O/T
35	H29	CMPD1	VCAM	1.8V,4.7K/20K	50K	CMPD1	O			DBG_BUS25(G1)	O	GPIO47	I/O/T
36	E32	SCL0	VCAM	1.8V,1.8K/20K	50K	SCL0	I/O/T			DBG_BUS26(G1)	O	GPIO48	I/O/T
37	F32	SDA0	VCAM	1.8V,1.8K/20K	50K	SDA0	I/O/T			DBG_BUS27(G1)	O	GPIO49	I/O/T
38	W5	LCM_RSTN	VIO1V8	1.8V,4.7K/20K	50K	LCM_RSTN	O					GPIO50	I/O/T
39	Y5	LCM_FMARK	VIO1V8	1.8V,4.7K/20K	50K	DSI_TE	I					GPIO51	I/O/T
40	D18	SPI2_CSN	VIO1V8	1.8V,4.7K/20K	50K	SPI2_CSN	I/O/T			CM4_GPIO5	I/O/T	GPIO52	I/O/T
41	D19	SPI2_DO	VIO1V8	1.8V,4.7K/20K	50K	SPI2_DO	I/O/T			CM4_GPIO0	I/O/T	GPIO53	I/O/T
42	E20	SPI2_DI	VIO1V8	1.8V,4.7K/20K	50K	SPI2_DI	I/O/T			CM4_GPIO1	I/O/T	GPIO54	I/O/T
43	E19	SPI2_CLK	VIO1V8	1.8V,4.7K/20K	50K	SPI2_CLK	I/O/T			CM4_GPIO2	I/O/T	GPIO55	I/O/T
44	AA31	U0TXD	VIO1V8	1.8V,4.7K/20K	50K	U0TXD	O	EXT_XTL_EN2	I	DBG_BUS10(G1)	O	GPIO60	I/O/T
45	AA32	U0RXD	VIO1V8	1.8V,4.7K/20K	50K	U0RXD	I	EXT_XTL_EN3	I	DBG_BUS11(G1)	O	GPIO61	I/O/T
46	AC31	U0CTS	VIO1V8	1.8V,4.7K/20K	50K	U0CTS	I	PWMC(G0)	O	DBG_BUS12(G1)	O	GPIO62	I/O/T
47	AB31	U0RTS	VIO1V8	1.8V,4.7K/20K	50K	U0RTS	O	SE_GPIO6	I/O/T	DBG_BUS13(G1)	O	GPIO63	I/O/T

在 bsp/bootloader/u-boot15/board/spreadtrum/sl8541e_1h10_32b/pinmap-sl8541e.c

中搜索 SPI2_CSN

```
{REG_PIN_SPI2_CSN,                     BITS_PIN_AF(3)},// 3 对应的是gpio功能, (从0开始,
excel中描述的func4,即对应 BITS_PIN_AF(3) )
//pin name                            驱动能力 (0~16)  模式
{REG_MISC_PIN_SPI2_CSN,
BITS_PIN_DS(1)|BIT_PIN_NULL|BIT_PIN_WPU|BIT_PIN_SLP_AP|BIT_PIN_SLP_WPU|BIT_PIN_SLP_IE},//PROX_INT???????????
//其中 BIT_PIN_NULL|BIT_PIN_WPU|BIT_PIN_SLP_AP|BIT_PIN_SLP_WPU|BIT_PIN_SLP_IE 表示配
置为悬空, sleep 上拉输入模式, BIT_PIN_SLP_AP 为 AP 侧使用

/* 如果配置该gpio为输出, 则可配置如下 */
{REG_PIN_SPI2_CSN,                     BITS_PIN_AF(3)},
{REG_MISC_PIN_SPI2_CSN,
BITS_PIN_DS(1)|BIT_PIN_NULL|BIT_PIN_NUL|BIT_PIN_SLP_AP|BIT_PIN_SLP_NUL|BIT_PIN_SLP_OE},//PROX_INT???????????
```

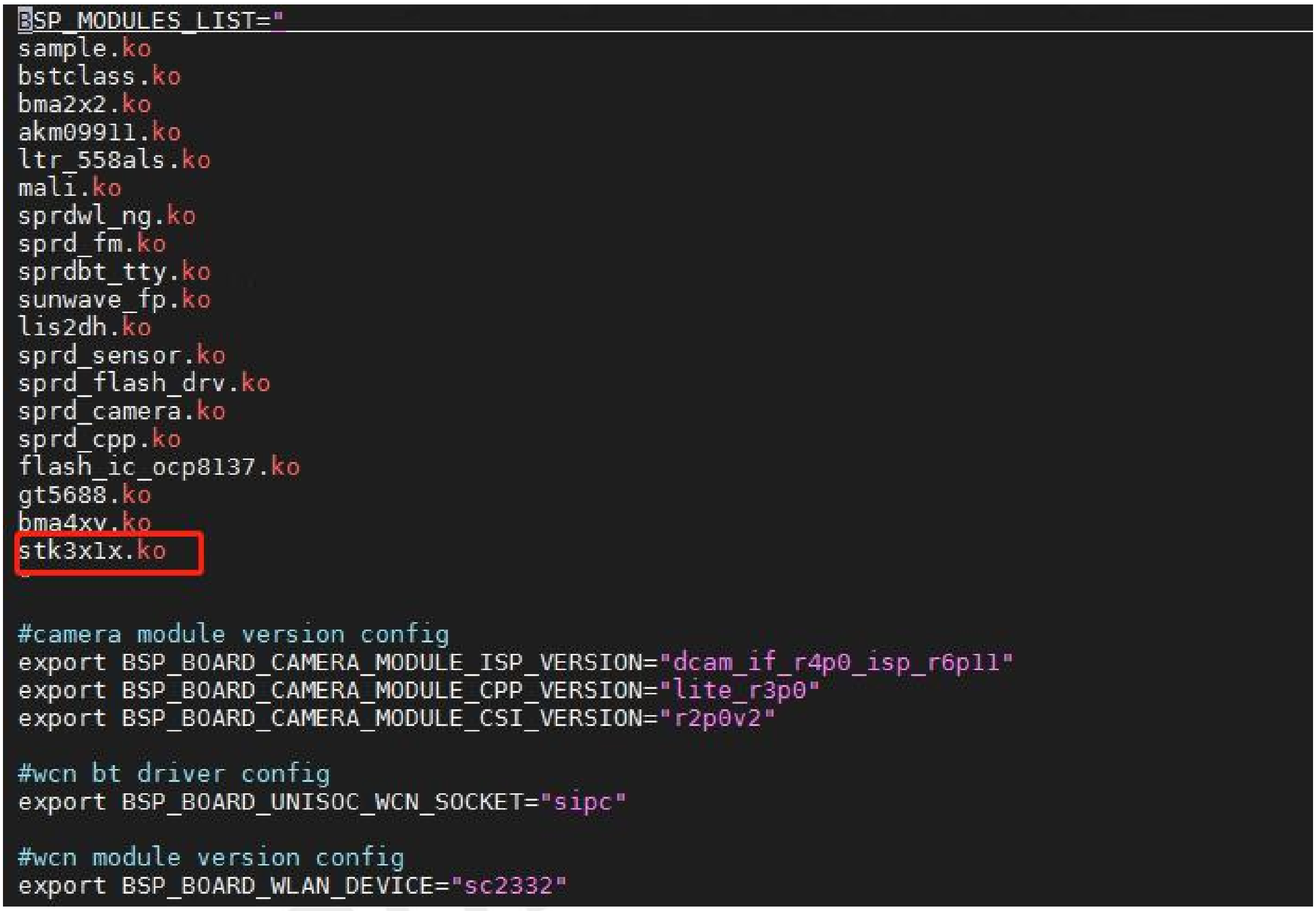
C、配置相关 cfg 文件，指定编译和打包进 sockoimage
修改文件：

bsp/device/sharkle/androidq/sl8541e_1h10_32b/sl8541e_1h10_32b_base/modules.cfg

device/sprd/sharkle/sl8541e_1h10_32b/BoardConfig.mk

device/sprd/sharkle/sl8541e_1h10_32b/sl8541e_1h10_32b_Natv.mk

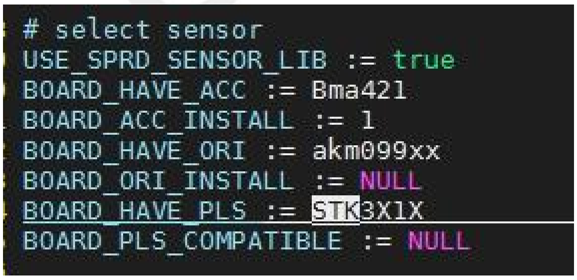
device/sprd/sharkle/sl8541e_1h10_32b/rootdir/root/init.sensors.rc bsp/device/sharkle/androidq/sl8541e_1h10_32b/sl8541e_1h10_32b_base/modules.cfg



```
BSP_MODULES_LIST="
sample.ko
bstclass.ko
bma2x2.ko
akm09911.ko
ltr_558als.ko
mali.ko
sprdwl_ng.ko
sprd_fm.ko
sprdbt_tty.ko
sunwave_fp.ko
lis2dh.ko
sprd_sensor.ko
sprd_flash_drv.ko
sprd_camera.ko
sprd_cpp.ko
flash_ic_ocp8137.ko
gt5688.ko
bma4xy.ko
stk3x1x.ko


#camera module version config
export BSP_BOARD_CAMERA_MODULE_ISP_VERSION="dcam_if_r4p0_isp_r6p11"
export BSP_BOARD_CAMERA_MODULE_CPP_VERSION="lite_r3p0"
export BSP_BOARD_CAMERA_MODULE_CSI_VERSION="r2p0v2"

#wcn bt driver config
export BSP_BOARD_UNISOC_WCN_SOCKET="sipc"

#wcn module version config
export BSP_BOARD_WLAN_DEVICE="sc2332"
```

在 BSP_MODULE_LIST 中添加需要编译的 ko 文件

device/sprd/sharkle/sl8541e_1h10_32b/BoardConfig.mk



```
# select sensor
USE_SPRD_SENSOR_LIB := true
BOARD_HAVE_ACC := Bma421
BOARD_ACC_INSTALL := 1
BOARD_HAVE_ORI := akm099xx
BOARD_ORI_INSTALL := NULL
BOARD_HAVE_PLS := STK3X1X
BOARD_PLS_COMPATIBLE := NULL
```

设置 BOARD_HAVE_PLS := STK3X1X，指定光距感 sensor 为目标 sensor， 该名字由 vendor\sprd\modules\sensors\libsensorclassic\pls\Pls_STK3X1X.cpp 决定，即 Pls_name.cpp 后面接的后缀名（name）有关系

device/sprd/sharkle/sl8541e_1h10_32b/rootdir/root/init.sensors.rc

```
on post-fs
    insmod ${ro.vendor.ko.mount.point}/socko/bstclass.ko
    insmod ${ro.vendor.ko.mount.point}/socko/bma4xy.ko
    insmod ${ro.vendor.ko.mount.point}/socko/stk3x1x_pls.ko
    insmod ${ro.vendor.ko.mount.point}/socko/akm09911.ko
+   insmod ${ro.vendor.ko.mount.point}/socko/stk3x1x.ko
    insmod ${ro.vendor.ko.mount.point}/socko/mir3da.ko

on factorytest
    insmod ${ro.vendor.ko.mount.point}/socko/bstclass.ko
    insmod ${ro.vendor.ko.mount.point}/socko/bma4xy.ko
    insmod ${ro.vendor.ko.mount.point}/socko/stk3x1x_pls.ko
    insmod ${ro.vendor.ko.mount.point}/socko/akm09911.ko
+   insmod ${ro.vendor.ko.mount.point}/socko/stk3x1x.ko
    insmod ${ro.vendor.ko.mount.point}/socko/mir3da.ko
```

自动挂载该 ko 文件

device/sprd/sharkle/sl8541e_1h10_32b/sl8541e_1h10_32b_Natv.mk

```
PRODUCT_SOCKO_KO_LIST := \
    $(BSP_KERNEL_MODULES_OUT)/bma4xy.ko \
+   $(BSP_KERNEL_MODULES_OUT)/stk3x1x.ko \
    $(BSP_KERNEL_MODULES_OUT)/mali.ko \
    $(BSP_KERNEL_MODULES_OUT)/sprdwl_ng.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_flash_drv.ko \
    $(BSP_KERNEL_MODULES_OUT)/sprd_camera.ko \
    $(BSP_KERNEL_MODULES_OUT)/sprd_cpp.ko \
    $(BSP_KERNEL_MODULES_OUT)/flash_ic_ocp8137.ko
```

修改 PRODUCT_SOCKO_KO_LIST， 添加目标 ko 文件打包到 socko

D、配置相关权限文件，使得第三方和系统 APP 可以正常访问

device/sprd/sharkle/common/rootdir/root/init.common.rc

device/sprd/sharkle/common/rootdir/root/ueventd.common.rc

device/sprd/sharkle/common/sepolicy/file_contexts

device/sprd/sharkle/common/rootdir/root/init.common.rc

```
on boot
    chown system system /sys/class/misc/gnss_common_ctl/gnss_power_enable
    chown system system /sys/class/misc/gnss_common_ctl/gnss_dump
    chown system system /sys/class/misc/gnss_common_ctl/gnss_subsys
    chown system system /dev/gnss_pmnotify_ctl
    chmod  220 /sys/class/misc/gnss_common_ctl/gnss_power_enable
    chmod 660 /sys/class/misc/gnss_common_ctl/gnss_dump
    chmod 660 /sys/class/misc/gnss_common_ctl/gnss_subsys
    chmod 660  /dev/gnss_pmnotify_ctl

    chmod 0660 /dev/AKM099XX
    chown system system /dev/AKM099XX

    chmod 0660 /dev/bma4xy_acc
    chown system system /dev/bma4xy_acc

+    chmod 0660 /dev/stk_ps
+    chown system system /dev/stk_ps
```

注： /dev/stk_ps 该节点是该驱动建立的， 查找对应的 i2c_driver 中设置的 name 即可，例如：

```
static struct i2c_driver stk_ps_driver =
{
    .driver = {
        .name = DEVICE_NAME,//会依据该name在dev目录下建立对应节点  DEVICE_NAME = stk_ps
        .owner = THIS_MODULE,
#ifdef CONFIG_OF
        .of_match_table = stk_match_table,
#endif
        .pm = &stk3x1x_pm_ops,
    },
    .probe = stk3x1x_probe,
    .remove = stk3x1x_remove,
    .id_table = stk_ps_id,
};
```

更改 /dev/stk_ps 节点权限和所在组权限

device/sprd/sharkle/common/rootdir/root/ueventd.common.rc

```
/dev/bma4xy_acc          0660     system      system
/dev/mir3da             0660     system      system
+/dev/stk_ps            0660     system      system
/dev/block/mmcblk0rpmb  0660     system      system
```

与前面的作用一样，设置该节点的权限和组权限

device/sprd/sharkle/common/sepolicy/file_contexts

```
#sprd sensors
/dev/lis3dh_acc         u:object_r:sensors_device:s0
/dev/bma4xy_acc         u:object_r:sensors_device:s0
/dev/AKM099XX           u:object_r:sensors_device:s0
/dev/stk_ps            u:object_r:sensors_device:s0
/dev/mir3da            u:object_r:sensors_device:s0
```

设置该节点的安全上下文

E、移植 HAL 层代码（这个也是供应商提供的）

vendor\sprd\modules\sensors\libsensorclassic\pls\

Pls_STK3X1X.cpp

1、关键函数： setDelay

设置 delay，即获取数据的时间间隔，展锐平台默认为 200ms

```
//光距感sensor:
int PlsSensor::readEvents(sensors_event_t * data, int count)
//Gsensor:
int AccSensor::setDelay(int32_t handle, int64_t delay_ns)
//MagSensor:
```

```
int OriSensor::setDelay(int32_t handle, int64_t ns)
//GyroSensor:
int GyroSensor::setDelay(int32_t handle, int64_t delay_ns)
```

在构造 HAL 层 sensor 对象的时候，也会去设置 delay 参数。例如光距感：

```
PlsSensor::PlsSensor() :
    SensorBase(STK_DEVICE_NAME, "proximity"),
        mEnabled(0),
        mPendingMask(0),
        mInputReader(32),
        mHasPendingEvent(false) {
    memset(mPendingEvents, 0, sizeof(mPendingEvents));

    mPendingEvents[Light].version = sizeof(sensors_event_t);
    mPendingEvents[Light].sensor = ID_L;
    mPendingEvents[Light].type = SENSOR_TYPE_LIGHT;

    mPendingEvents[Proximity].version = sizeof(sensors_event_t);
    mPendingEvents[Proximity].sensor = ID_P;
    mPendingEvents[Proximity].type = SENSOR_TYPE_PROXIMITY;

    for (int i=0 ; i<numSensors ; i++)
        mDelays[i] = 200000000; // 200 ms by default
}
```

Set_delay 函数实际是通过访问 /sys/class/input/inputX/delay 节点
从而通过 ioctl 操作底层的 set_delay 函数，有些厂家提供的驱动可能不包含该接口，可能会有些许差异
例如：
光距感 stk3311 PlsSensor::readEvents 直接返回 0
Gsensor AccSensor::readEvents 是正常访问 /sys/class/input/inputX/delay 节点，从而调用底层 mir3da_misc_ioctl

```
    case MIR3DA_ACC_IOCTL_SET_DELAY
```

2、获取底层 sensor 数据 readEvents

```
//光距感sensor:
int PlsSensor::readEvents(sensors_event_t * data, int count)
//Gsensor:
int AccSensor::readEvents(sensors_event_t *data, int count)
//MagSensor:
int OriSensor::readEvents(sensors_event_t *data, int count)
//GyroSensor:
int GyroSensor::readEvents(sensors_event_t *data, int count)


//光距感 sensor
int PlsSensor::readEvents(sensors_event_t * data, int count)
{
    if (count < 1)
        return -EINVAL;

    ssize_t n = mInputReader.fill(data_fd);
    if (n < 0)
        return n;

    int numEventReceived = 0;
    input_event const* event;

    while (count && mInputReader.readEvent(&event)) {//读取count个数据，通过InputReader读取event
        int type = event->type;
        if (type == EV_ABS) {
            processEvent(event->code, event->value);//将读到数据以一定的格式封装，交由InputDispatcher分发给相关应用
            mInputReader.next();
        } else if (type == EV_SYN) {
            int64_t time = timevalToNano(event->time);
            for (int j=0 ; count && mPendingMask && j<numSensors ; j++) {
                if (mPendingMask & (1<<j)) {
                    mPendingMask &= ~(1<<j);
                    mPendingEvents[j].timestamp = time;
                    if (mEnabled & (1<<j)) {
                        *data++ = mPendingEvents[j];
                        count--;
                        numEventReceived++;
                    }
                }
            }
            if (!mPendingMask) {
                mInputReader.next();
            }
        } else {
            ALOGE("stk: unknown event (type=%d, code=%d)",
                    type, event->code);
            mInputReader.next();
        }
    }

    return numEventReceived;
}
```

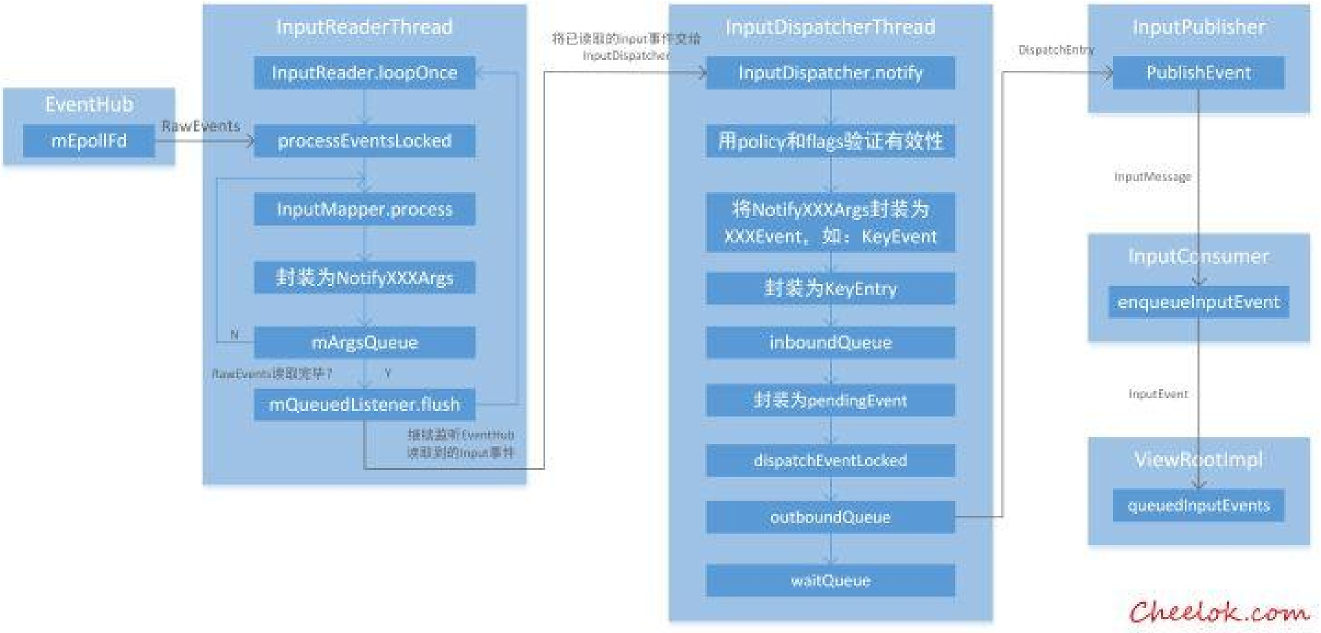
底层通过 input_report_abs/input_report_key 等函数向 EventHub 发送一些的封装的数据可以通过 getevent 获取到，一般在内核驱动中，会将调用发送数据的函数放到一个延时工作队列中来定时发送。

例如：Gsensor Mir3da:

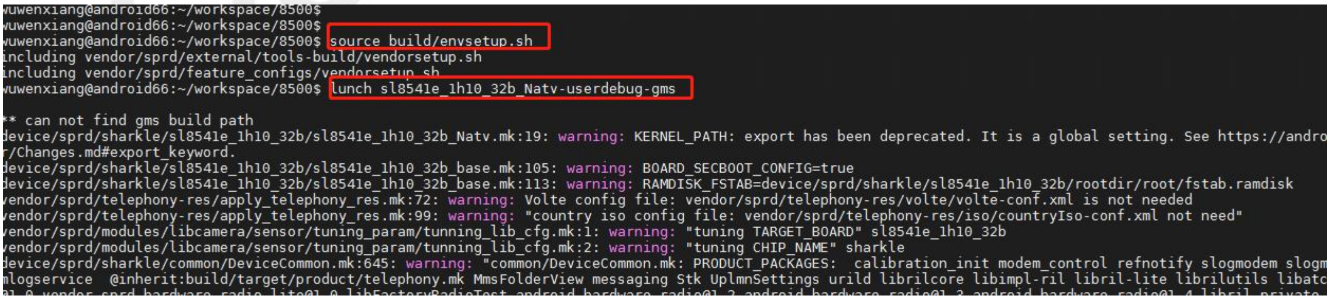
```
static void mir3da_work_func(struct work_struct *work)
{
    short x=0,y=0,z=0;
    struct mir3da_data *mir3da = container_of((struct delayed_work *)work,struct
mir3da_data, work);
    int map_para = 1;

  if (mir3da_read_data(mir3da->mir3da_i2c_client, &x,&y,&z) != 0) {
     MI_ERR("MIR3DA data read failed!\n");
  }
  else
  {
   map_para = mir3da_direction_remap(&x,&y,&z, direction_remap);

   if(bzstk)
     z = map_para*squareRoot(1024*1024 - (x)*(x) - (y)*(y));

   input_report_abs(mir3da->input, ABS_X, x);
   input_report_abs(mir3da->input, ABS_Y, y);
   input_report_abs(mir3da->input, ABS_Z, z);
   input_sync(mir3da->input);
  }

schedule_delayed_work(&mir3da->work,msecs_to_jiffies(atomic_read(&mir3da->delay)))
;
}
```

数据处理的大概流程如下:

InputReader 和 InputDispatcher 对象会分别创建两个线程（InputReaderThread 和 InputDispatcherThread），读取 EventHub 中的数据和分发从 InputReader 读取过来的数据给 InputPublisher，从而分发给监听的应用



3 编译 & 下载



```
wuwenxiang@android66:~/workspace/8500$
wuwenxiang@android66:~/workspace/8500$
wuwenxiang@android66:~/workspace/8500$ source build/envsetup.sh
including vendor/sprd/external/tools-build/vendorsetup.sh
including vendor/sprd/feature_configs/vendorsetup.sh
wuwenxiang@android66:~/workspace/8500$ lunch sl8541e_1h10_32b_Natv-userdebug-gms

** can not find gms build path
device/sprd/sharkle/sl8541e_1h10_32b/sl8541e_1h10_32b_Natv.mk:19: warning: KERNEL_PATH: export has been deprecated. It is a global setting. See https://andro
r/Changes.md#export_keyword.
device/sprd/sharkle/sl8541e_1h10_32b/sl8541e_1h10_32b_base.mk:105: warning: BOARD_SECBOOT_CONFIG=true
device/sprd/sharkle/sl8541e_1h10_32b/sl8541e_1h10_32b_base.mk:113: warning: RAMDISK_FSTAB=device/sprd/sharkle/sl8541e_1h10_32b/rootdir/root/fstab.ramdisk
vendor/sprd/telephony-res/apply_telephony_res.mk:72: warning: Volte config file: vendor/sprd/telephony-res/volte/volte-conf.xml is not needed
vendor/sprd/telephony-res/apply_telephony_res.mk:99: warning: "country iso config file: vendor/sprd/telephony-res/iso/countryIso-conf.xml not need"
vendor/sprd/modules/libcamera/sensor/tuning_param/tunning_lib_cfg.mk:1: warning: "tuning TARGET_BOARD" sl8541e_1h10_32b
vendor/sprd/modules/libcamera/sensor/tuning_param/tunning_lib_cfg.mk:2: warning: "tuning CHIP_NAME" sharkle
device/sprd/sharkle/common/DeviceCommon.mk:645: warning: "common/DeviceCommon.mk: PRODUCT_PACKAGES:  calibration_init modem_control refnotify slogmodem slogm
mlogservice  @inherit:build/target/product/telephony.mk MmsFolderView messaging Stk UplmnSettings urild librilcore libimpl-ril libril-lite librilutils libatc
@1.0 vendor.sprd.hardware.radio.lite@1.0 libFactoryRadioTest android.hardware.radio@1.2 android.hardware.radio@1.3 android.hardware.radio@1.4 libril_private
```

```
wuwenxiang@android66:~/workspace/8500$ source build/envsetup.sh
wuwenxiang@android66:~/workspace/8500$ lunch sl8541e_1h10_32b_Natv-userdebug-gms
```

对于只改了 bsp/module 下面的文件，直接

make sockoimage ：编译生成 socko.img，存放 soc external modules 文件。

可以通过 fastboot flash socko socko.img 刷入镜像， 最好是整个目录全编译一下

全编：

```
    make update-api -j8 && make -j8
```

打包镜像：

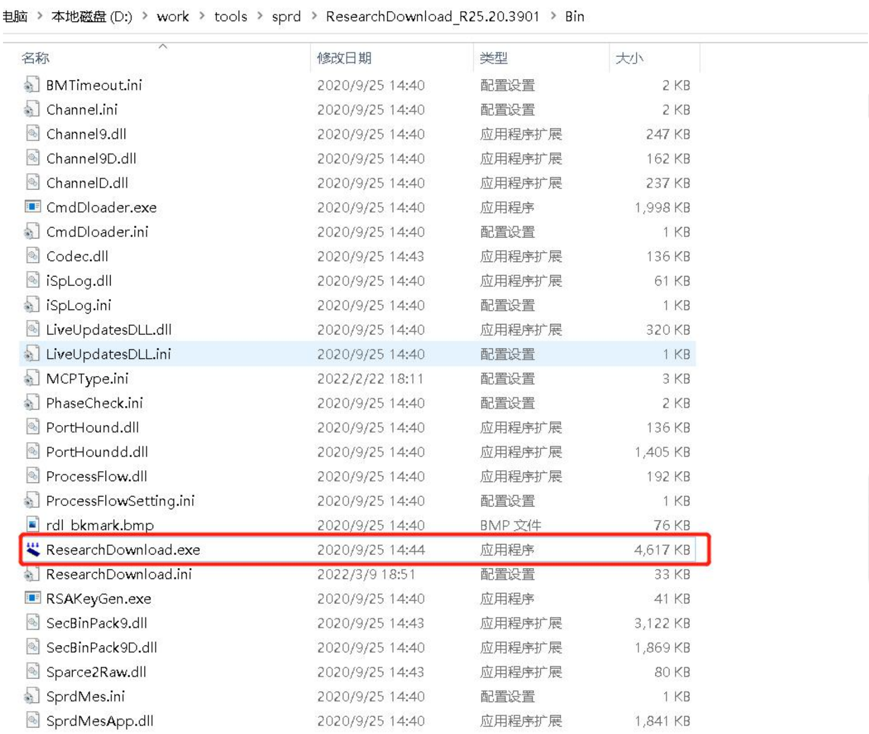
编译完成后，继续执行:cp_sign,

成功完成后继续执行：makepac

生成的 PAC 文件在 out/target/product/sl8541e_1h10_32b_Natv-userdebug-gms/cp_sign/../下

使用展锐下载工具 ResearchDownload_R25.20.3901，

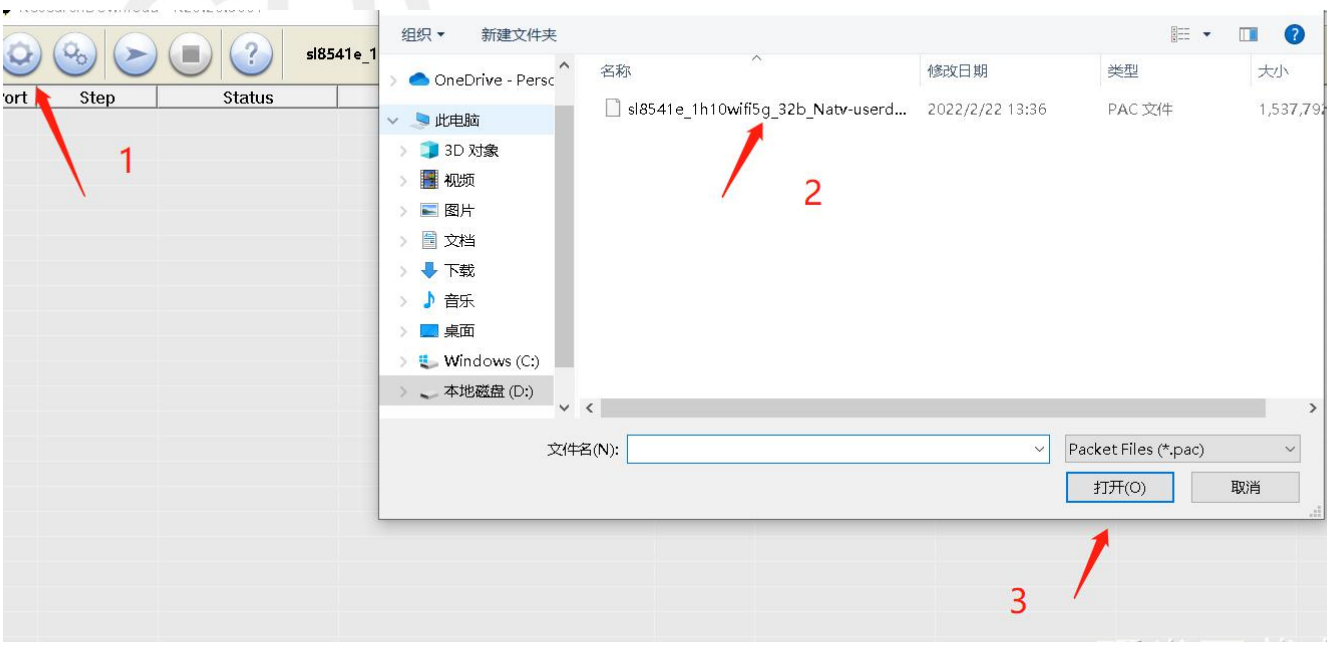
1、双击打开 ResearchDownload.exe



电脑 › 本地磁盘 (D:) › work › tools › sprd › ResearchDownload_R25.20.3901 › Bin

名称	修改日期	类型	大小
BMTimeout.ini	2020/9/25 14:40	配置设置	2 KB
Channel.ini	2020/9/25 14:40	配置设置	2 KB
Channel9.dll	2020/9/25 14:40	应用程序扩展	247 KB
Channel9D.dll	2020/9/25 14:40	应用程序扩展	162 KB
ChannelD.dll	2020/9/25 14:40	应用程序扩展	237 KB
CmdDloader.exe	2020/9/25 14:40	应用程序	1,998 KB
CmdDloader.ini	2020/9/25 14:40	配置设置	1 KB
Codec.dll	2020/9/25 14:43	应用程序扩展	136 KB
iSpLog.dll	2020/9/25 14:40	应用程序扩展	61 KB
iSpLog.ini	2020/9/25 14:40	配置设置	1 KB
LiveUpdatesDLL.dll	2020/9/25 14:40	应用程序扩展	320 KB
LiveUpdatesDLL.ini	2020/9/25 14:40	配置设置	1 KB
MCPType.ini	2022/2/22 18:11	配置设置	3 KB
PhaseCheck.ini	2020/9/25 14:40	配置设置	2 KB
PortHound.dll	2020/9/25 14:40	应用程序扩展	136 KB
PortHoundd.dll	2020/9/25 14:40	应用程序扩展	1,405 KB
ProcessFlow.dll	2020/9/25 14:40	应用程序扩展	192 KB
ProcessFlowSetting.ini	2020/9/25 14:40	配置设置	1 KB
rdl_bkmark.bmp	2020/9/25 14:40	BMP 文件	76 KB
ResearchDownload.exe	2020/9/25 14:44	应用程序	4,617 KB
ResearchDownload.ini	2022/3/9 18:51	配置设置	33 KB
RSAKeyGen.exe	2020/9/25 14:40	应用程序	41 KB
SecBinPack9.dll	2020/9/25 14:43	应用程序扩展	3,122 KB
SecBinPack9D.dll	2020/9/25 14:40	应用程序扩展	1,869 KB
Sparce2Raw.dll	2020/9/25 14:43	应用程序扩展	80 KB
SprdMes.ini	2020/9/25 14:40	配置设置	1 KB
SprdMesApp.dll	2020/9/25 14:40	应用程序扩展	1,841 KB

2、单击设置按钮，选择前面拷贝的 pac 包

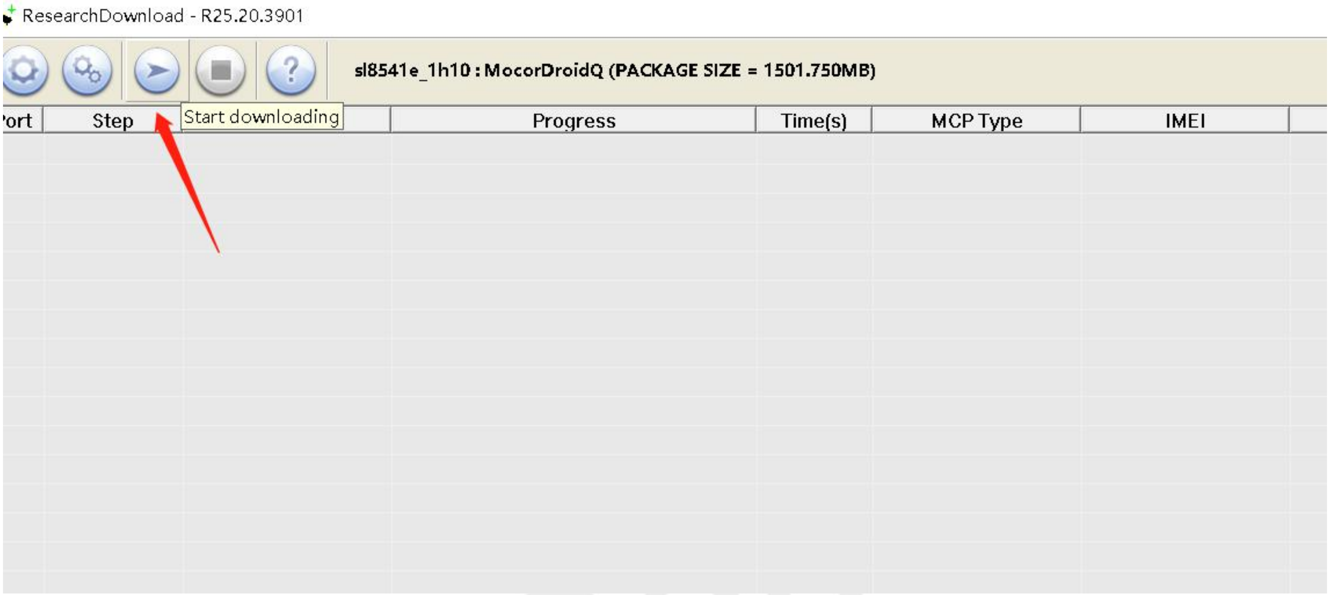


名称	修改日期	类型	大小
sl8541e_1h10wifi5g_32b_Natv-userd...	2022/2/22 13:36	PAC 文件	1,537,79

文件名(N): Packet Files (*.pac)

打开(O) 取消

3、选择开始下载，即可以开始下载，等下载完毕后重启设备即可


ResearchDownload - R25.20.3901
sl8541e_1h10 : MocorDroidQ (PACKAGE SIZE = 1501.750MB)
Start downloading
Port
Step
Progress
Time(s)
MCP Type
IMEI

4 Debug

1、 首先确认驱动 probe 是否成功
    可通过 adb 查看 是否有生成 /dev/xxx 节点
    如果没有生成，则加 log 调试，看是哪个阶段出了错误

常见出错项：

A、I2c 读写出错

首先排查硬件供电是否正常？ 然后用示波器量取波形看第一个读 ID 的信号是否正常？

如果设备返回 NACK，大概率是地址不对或者器件损坏，建议和硬件工程师一起检查、同时可联系供应商一同排查。

B、其他出错

例如 request gpio 时出错，这就需要排查是否有其他驱动占用该 gpio

可通过 cat /d/gpio 进行快速排查

2、Probe 成功，dev/xxx 节点也成功建立，但是不出数据
首先检查底层是否有数据，一般驱动都会有使能节点，可手动使能，然后通过 log 进行检查
通过 getevent 进行查看，如果 getevent 可以接收到数据，而第三方 app 没有获取到数据，这大概率是 hal 的代码有问题，可加 log 进行调试。