T7510 Sensor Hub
客制化配置

修改历史 Revision History

版本号 Version	日期 Date	注释 Notes
V1.0	2019/10/30	初稿

文档信息 Document Information

适用产品信息 Chip Platform	适用版本信息 OS Version	关键字 Keyword
T7510	Android 9.0	sensor sensorhub

Contents

Sensor Hub 代码结构简介

Sensor Hub代码主要分三部分

Sensor Hub CM4代码

Sensor驱动架构及Sensor Hub算法放置在CM4侧，目前代码闭源，仅提供bin文件供客户直接打包使用，bin文件为roc1_cm4.bin；

Sensor Hub HAL层代码

HAL层实现Android定义的标准sensors HAL接口，sensor特性参数配置和接口定义配置也在HAL层；HAL层代码路径：

vendor/sprd/modules/sensors/libsensorhub$

Sensor Hub Kernel层代码

Kernel层代码主要负责将HAL层下发命令及sensor配置参数传递给CM4，并将CM4反馈的状态信息、上报的sensor事件传递给HAL层；kernel代码路径：

kernel4.14/drivers/iio/sprd_hub

对于大多数sensor，用户只需在board配置下，增加sensor类型配置、HAL层增加sensor特性参数配置文件和接口定义配置即可。
对于难于兼容的sensor，目前CM4系统留出sensor驱动调用接口，可以使用动态加载方式加载sensor驱动。这时需要用DS-5编译器，编译动态加载文件。
目前地磁sensor由于需要sensor厂商提供磁校准算法库，所以全部需要动态加载方式来添加。

Sensor Hub 客制化说明

Sensor Hub客制化配置

device/sprd/roc1/(工程名)/BoardConfig.mk配置需要支持的sensor具体型号

以ud710_3h10工程为例：
SENSOR_HUB_ACCELEROMETER := lsm6dsl
SENSOR_HUB_GYROSCOPE := lsm6dsl
SENSOR_HUB_LIGHT := ltr553als
SENSOR_HUB_MAGNETIC := akm09918_ud710
SENSOR_HUB_PROXIMITY := ltr553als
SENSOR_HUB_PRESSURE := null
SENSOR_HUB_CALIBRATION := ud710
其中最后一行CALIBRATION为校准文件名，可以自行命名，赋值null为不支持此类型sensor。同类型sensor，如需兼容多个型号，如加速度计需同时兼容lsm6dsl和icm20600，则按如下配置
SENSOR_HUB_ACCELEROMETER := lsm6dsl icm20600

device/sprd/roc1/(工程名)./工程名_base.mk配置需要支持的sensor type

frameworks/native/data/etc/android.hardware.sensor.proximity.xml:$(TARGET_COPY_OUT_VENDOR)/etc/permissions/android.hardware.sensor.proximity.xml \
frameworks/native/data/etc/android.hardware.sensor.light.xml:$(TARGET_COPY_OUT_VENDOR)/etc/permissions/android.hardware.sensor.light.xml \
frameworks/native/data/etc/android.hardware.sensor.accelerometer.xml:$(TARGET_COPY_OUT_VENDOR)/etc/permissions/android.hardware.sensor.accelerometer.xml \
。。。
以此类推

HAL层增加对应sensor特性参数配置文件

vendor/sprd/modules/sensors/libsensorhub/ConfigSensor/(sensor类型)目录增加(sensor类型)_(sensor型号).cpp文件
以ud710_3h10工程添加加速度计lsm6dsl为例，增加下述文件
vendor/sprd/modules/sensors/libsensorhub/ConfigSensor/accelerometer/accelerometer_lsm6dsl.cpp
Sensor型号名需要与BoardConfig配置中相同
其他sensor与此类同，另外需要在calibration文件夹增加此工程的calibration文件
calibration_ud710.cpp，ud710为BoardConifg SENSOR_HUB_CALIBRATION 的值。

HAL 层对Sensor接口注册与fusion_mode配置

vendor/sprd/modules/sensors/libsensorhub/ConfigSensor/目录
以加速度计lsm6dsl为例
在SensorHubOpCodeExtrator.h增加
```
extern void SensorHubOpcodeRegisterAccelerometer_lsm6dsl();
```
在SensorHubOpCodeExtrator.cpp增加
```
#ifdef SENSORHUB_WITH_ACCELEROMETER_lsm6dsl
    strcat(accelerometer_list, "accelerometer_lsm6dsl,");
    SensorHubOpcodeRegisterAccelerometer_lsm6dsl();
#endif
```
在calibration/calibration_ud710.cpp文件中，对fusion_mode值进行修改配置
如加速度计、陀螺仪、地磁3种sensor都有，置为ACC_GYRO_MAG_FUSION；只有加速度计：ACC_FUSION；加速度计、陀螺仪： ACC_GYRO_FUSION，加速度计、地磁：ACC_MAG_FUSION

Sensor 特性参数opcode配置详细说明

- Sensor 功能寄存器配置
- Sensor 特性信息配置
- Sensor 校准参数配置

Sensor 功能寄存器配置

在特性参数配中，需将sensor初始化、使能、关闭、采样速率等寄存器配置到对应功能配置数组

- **iic_unit原型**

```
typedef struct iic_unit {
    uint8_t operate;
    uint8_t addr;
    uint8_t val;
    uint8_t mask;
} iic_unit_t;
```

iic_unit成员取值说明

operate ：

operate取值	定义
0x01	Read，读操作
0x02	Write，写操作
0x03	Check，检测操作，比如检测某一位是否置1
0x04	Delay，延时操作，单位为ms
0x05	操作控制CM4侧GPIO高低电平
0xff	在getrawdata配置中，如果sensor i2c不支持连续读寄存器，则在最后增加一行operate值为0xff的配置

addr：寄存器地址；GPIO端口号；

val：operate为写操作或检测操作时，需要写入的值，或要检测对比的值；或是GPIO电平配置；

mask：（1）operate为delay操作时，延时时间值：要延时多少ms；

（2）进行write、read、check操作时，需要处理的bit位置1，其余都置0.

功能类型说明

Opcode类型	数组名称	定义
CMD_HWINIT	eCmdInitArray	第一次上电，初始化sensor的配置，只在刚开机的时候执行一次。
CMD_ENABLE	eCmdEnableArray	填写每次enable的时候需要操作的寄存器，每次enable sensor的时候都会执行
CMD_DISABLE	eCmdDisableArray	填写每次disable的时候需要操作的寄存器，每次disable的时候都会执行
CMD_GET_BYPASS	eCmdGetRawDataArray	填写读取raw data的寄存器地址，顺序为从低位到高位，以加速度为例：X_L,X_H,Y_L,Y_H,Z_L,ZH.
CMD_SET_STATUS	eCmdSetStatusArray	有些sensor需要写入才能清掉状态位
CMD_SET_SELFTEST	eCmdSetSelftestArray	填写selftest的相关信息
CMD_SET_OFFSET	eCmdSetOffsetArray	设置sensor的offset寄存器，如果使用sensor自身的校准功能会用到，但目前有专门的校准算法，所以此项配置不用填写
CMD_SET_RATE	eCmdSetRateArray	设置sensor odr相应信息，填写顺序为由快到慢，目前加速度此项的填写顺序为：100Hz，50Hz，16Hz，4Hz
CMD_SET_MODE	eCmdSetModeArray	改变工作模式
CMD_GET_STATUS	eCmdGetStatusArray	当读取sensor raw data前会读sensor status寄存器，status ok代表当前raw data寄存器可读
CMD_CHECK_ID	eCmdCheckIdArray	填写sensor ID寄存器地址及ID值

Sensor 功能寄存器配置

以bmi160 acc enable的enable操作寄存器配置为例进行说明：


```
static struct iic_unit eCmdEnableArray[] = {
        /* 写操作，往0x7E寄存器写入0x11*/
        {
                .operate = 0x02, .addr = 0x7E, .val = 0x11, .mask = 0xFF,
        },
        /* delay操作，delay 20ms*/
        {
                .operate = 0x04, .addr = 0x00, .val = 0x00, .mask = 0x14,
        },
        /* check操作， 检测bit4 & bit5位是否分别为“1” 和 “0”*/
        {
                .operate = 0x03, .addr = 0x03, .val = 0x10, .mask = 0x30,
        },
        /* 写操作，往0x41寄存器的低四位写入“1000” ，高四位保持不变*/
        {
                .operate = 0x02, .addr = 0x41, .val = 0x08, .mask = 0x0F,
        },
};
```

Sensor 特性信息配置

- **对sensorInfoConfigArray 进行配置**

sensorInfoConfigArray 原型：

```
typedef struct {
  uint8_t Interface;
  uint16_t Interface_Freq;
  uint8_t ISR_Mode;
  uint8_t GPIO_ISR_Num;
  uint8_t Slave_Addr;
  uint8_t Chip_Id;
  float Resolution;
  uint16_t Range;
  uint8_t Postion;
  uint8_t Vendor;
} sensor_info_t;
```

Sensor 特性信息配置

sensor_info_t成员说明：

- **Interface**：通信接口，目前只支持I2C，填0即可;
- **Interface_Freq**：通信频率，其以KHz为单位，比如目前I2C以400KHz的频率去读写，目前填写400即可;
- **ISR_Mode**：是否使用中断模式，目前中断模式尚未支持;
- **GPIO_ISR_Num**：使用的中断模式时，用到的GPIO号;
- **Slave_Addr**：sensor 的I2C写地址;
- **Chip_Id**：sensor的chip id;
- **Resolution**：sensor的精度值。从sensor中读取的raw data拼接之后会直接乘以此参数，得到的就是标准单位的值：acc(m/s^2), gyro(rad/s), mag(uT), pressure(hpa)，light(als)。此值可以让sensor的FAE提供，也可以自己根据spec计算得出;
- **Range**：量程，根据google的标准，ACC一定要配置成 ±8G，ACC的range要填写8，其余的按照spec上的实际配置填写，目前gyro的为2000，mag的为4，其余的不需填写；

Sensor 特性信息配置

- **Postion :** 调整sensor数据的方向。

Position	方向调整
0	X : Y : Z
1	Y : -X : Z
2	-X : -Y : Z
3	-Y : X : Z
4	Y : X : -Z
5	X : -Y : -Z
6	-Y : -X : -Z
7	-X : Y : -Z

Sensor 特性信息配置

- **Vendor :**
  - **磁力计:** 此值需与磁力计校准库的mag_init接口返回值一致；
  - **Prox sensor：**
    - **0:** 不需将低噪值回写到sensor，使用sprd内置的快速校准算法；**1:** 需要回写工厂校准的低噪值到sensor，使用sprd内置的快速校准算法；**0xFE:** 使用动态加载机制。
  - **Acc、Gyro：**
    - **0：** 非动态加载机制； **0xFE:** 使用动态加载机制。
  - **Light：**
    - **0:** LTR553；**1:** sensor的raw data寄存器读出来的直接是als单位，不需要转换；**0xFE:** 使用动态加载机制。
  - **pressure：**
    - 默认支持bmp280，倘若opcode架构可以满足其他sensor的需求，直接填0即可；**0xFE：**动态加载机制。

对于动态加载机制，需与我司签订NDA协议之后，我司会release 相应的IDH包和使用文档，才能正常开发使用。

Sensor 校准参数配置

- **目前只需对距感sensor校准参数配置，其他sensor无需配置，下面对距感校准参数取值进行说明**

  在进行校准参数配置前，建议使用10台左右整机，获取以下数据：

  1. 用18%灰卡遮挡距感3cm处，sensor测量值；
  2. 用18%灰卡遮挡距感5cm处，sensor测量值；
  3. 无遮挡时sensor测量值(底噪)

  上述值可以将手机root后，在shell命令下

  cat dev/slog_pm | grep “DRIVER_COMM_PS_GetData”

  再打开距感获取

Sensor 校准参数配置

```
static prox_cali_t eProxCaliArray[] = {
        {
                .collect_num = 7, //无需修改
                .ps_threshold_high = 0x70, //遮挡物距sensor约3cm时的值，建议使用多台样机中较大的值，否则底噪太大时可能导致误判接近
                .ps_threshold_low = 0x30, //遮挡物距sensor约5cm时的值，建议使用多台样机中较大的值，否则底噪太大时可能不能识别远离
                .dyna_cali = 0xa00, //取远大于sensor被完全遮挡时或sensor满量程值即可，软件自动校准时会自动调整实际值。
                .dyna_cali_add = 0x10, //环境不变时sensor测量值(raw_data)的浮动范围，如果基本不变取值0亦可
                .noise_threshold = 0xC0, //手机距感正常时sensor可能出现的最大噪声参考值，可以联系sensor厂商确认
                .noise_high_add = 0x58, //ps_threshold_high – noise，noise指无遮挡时的底噪值
                .noise_low_add = 0x18, // ps_threshold_low – noise，noise指无遮挡时的底噪值
                .ps_threshold_high_def = 0x70, //一般与ps_threshold_high取相同值；作为动态校准失效时的ps_threshold_high值
                .ps_threshold_low_def = 0x30, //一般与ps_threshold_low取相同值；作为动态校准失效时的ps_threshold_low值
                .dyna_cali_threahold = 0x18, //取正常样机底噪值最小参考值，底噪值小于此值就不再进行动态校准
                .dyna_cali_reduce = 0x8, //建议取值为介于noise_low_add和noise_high_add之间的值
                .value_threshold = 0xC0, //取值可以和noise_threshold相同
                .noise_reference = 0x20, //设置值应为正常sensor底噪值上限，取正常样机无遮挡时噪声最大值，建议与sensor厂商一起确认。此值用于工厂校准时判断
                sensor是否正常，手机底噪大于此值，说明sensor有问题或产线组装异常
        },
};
```

Sensor Hub CM4 log和debug方法

- **Sensor Hub CM4 log**
  - **Log存放位置**

    如果开机前未插sd卡

    1. 开机阶段前期（未检测到storage/emulated/0）sensorhub的log保存在 data/ylog/sensorhub/
    2. 开机阶段后期（检测到storage/emulated/0）sensorhub的log保存在 storage/emulated/0/ylog/sensorhub/
    3. user版本如果抓完log要导出，可插sd卡后使用adb shell log_ctr slogmodem COLLECT_LOG导到sd卡ylog目录

    如果开机前插sd卡，sensorhub log默认保存在sd卡ylog目录
  - **实时读取sensor hub CM4 log**
    - 用adb命令root和remount手机后，在shell下执行cat dev/slog_pm即可查看实时log输出，不对sensor进行操作时log输出量应该很少。
    - 如果上述操作没有log输出，应该是log处于关闭状态，可以执行echo “log on” > sctl_pm打开log输出

Sensor Hub CM4 log和debug方法

- **在线读写sensor寄存器**
  - 用adb命令将手机root和remount后，adb shell，进入d/sensor目录
  - 执行cat shub_debug，会有如下输出

```
ud710_3h10:/d/sensor # cat shub_debug
cat shub_debug

Description:
        echo "op sid reg val mask" > shub_debug
        cat shub_debug

Detail:
         op(operator): 0:open 1:close 2:read 3:write
         sid(sensorid): 0:acc 1:mag 2:gyro 3:proximity 4:light 5:pressure 6:new_dev
         reg: the operate register that you want
         value: The value to be written or show read value
         mask: The mask of the operation

         status: the state of execution. 1:success 0:fail

[shub_debug]operate:0x0 sensor_id:0x0 reg:0x0 value:0x0 status:0x0
```

Sensor Hub CM4 log和debug方法

– 根据提示执行需要的操作，举例如下：

- 读加速度计 0x75寄存器：执行echo “2 0 0x75 0x00 0xff” > shub_debug，然后cat shub_debug，最后一行输出value值0x11即所读寄存器值，status为0x1表示读取成功；

```
        value: The value to be written or show read value
        mask: The mask of the operation

        status: the state of execution. 1:success 0:fail

[shub_debug]operate:0x2 sensor_id:0x0 reg:0x75 value:0x11 status:0x1
```

- 将0x13写入加速度计0x19寄存器：执行echo “3 0 0x19 0x13 0xff” > shub_debug即可；

THANKS

WWW.UNISOC.COM
紫 光 展 锐 科 技