SensorHub各类型Sensor校准介绍

文档版本	V3.0
发布日期	2020-09-28

紫光展锐（上海）科技有限公司

紫光展锐（上海）科技有限公司

前　言

概述

本文档对平台支持的传感器基本校准流程、各类型传感器的校准原理、校准操作及校准数据存储进行说明。

读者对象

供传感器模块相关技术人员及展锐客户参考使用。

缩略语

缩略语	英文全名	中文解释
MCU	Micro Control Unit	微控制器单元，专用于管理嵌入式系统中多个传感器、收集传感器数据、向上上传数据和事件、向下部署配置。

变更信息

文档版本	发布日期	修改说明
V1.0	2016/08/22	初稿。
V2.0	2019/10/17	修改磁力计和光传感器部分。
V3.0	2020/09/28	技术变更： 扩充校准流程说明，增加校准数据格式。 文字变更： 调整文档格式，优化文档语言描述。

关键字

传感器/Sensor、校准。

目 录

图目录

表目录

1 概述

1.1 传感器类别

平台目前支持的传感器种类如下：

- Accelerator Sensor：加速度计
- Magnetic Sensor：磁力计
- Gyroscope Sensor：陀螺仪
- Ambient Light Sensor：光线传感器
- Proximity Sensor：距离传感器

1.2 基本校准流程

Sensorhub 是集中管理手机平台上各传感器驱动和算法的模块，它在独立的 MCU（例如 ARM Cortex m4）上运行。HAL 层实现 Android 定义的标准 Sensors HAL 特性参数配置和接口定义，Kernel 层将 HAL 层下发的命令及 Sensor 配置参数传递给 MCU，并将 MCU 反馈的状态信息、上报的 Sensor 事件传递给 HAL 层。

校准的目的是将待校准手机设备放置于标准状态位置或环境下，标定该 Sensor 工作基准，屏蔽由于硬件装配等因素造成的微小不一致，对齐 Sensor 的工作状态，同时将装配结构差异超出规定范围的机器个体筛除出去。

用户在 Android 侧或工程模式下发起校准流程，下发命令到 MCU，完成校准后将校准结果返回并上报给用户层，并保存获得的校准参数，以便传感器后续工作时使用这些校准参数。

HAL 层使用到的主要接口和相关参数定义如下。

```
int sensor_calibration(int enable, int cali_type, int sensor_type)
int SenCaliCmd(const char * cmd)
```

其中输入参数 cmd 由 calib_cmd 和 sensor_type 组成：

```
enum calib_cmd {
    CALIB_EN,                   //使MCU开始校准该sensor
    CALIB_CHECK_STATUS,         //让MCU检查校准操作是否已完成，通过calibrator_data读得
    CALIB_DATA_WRITE,
    CALIB_DATA_READ,            //让MCU传回校准完成后的数据，通过calibrator_data读得
    CALIB_FLASH_WRITE,
    CALIB_FLASH_READ,
};
```

```
enum calib_status {
        CALIB_STATUS_OUT_OF_RANGE = -2,          //已不使用
        CALIB_STATUS_FAIL = -1,                  //校准失败
        CALIB_STATUS_NON = 0,
        CALIB_STATUS_INPROCESS = 1,              //仍在校准过程中
        CALIB_STATUS_PASS = 2,                   //校准成功完成
};
```

- CALIB_CHECK_STATUS 命令返回的 calib_status 只反映校准流程是否成功走完，是否采集到足够的样本数据，不判断数值是否达标。
- 通过 CALIB_DATA_READ 命令读出的 calibrated_data 具体判断校准数值是否达标。
- 代码位于在 shub_core.c 中的 check_xxx_cali_data( )。

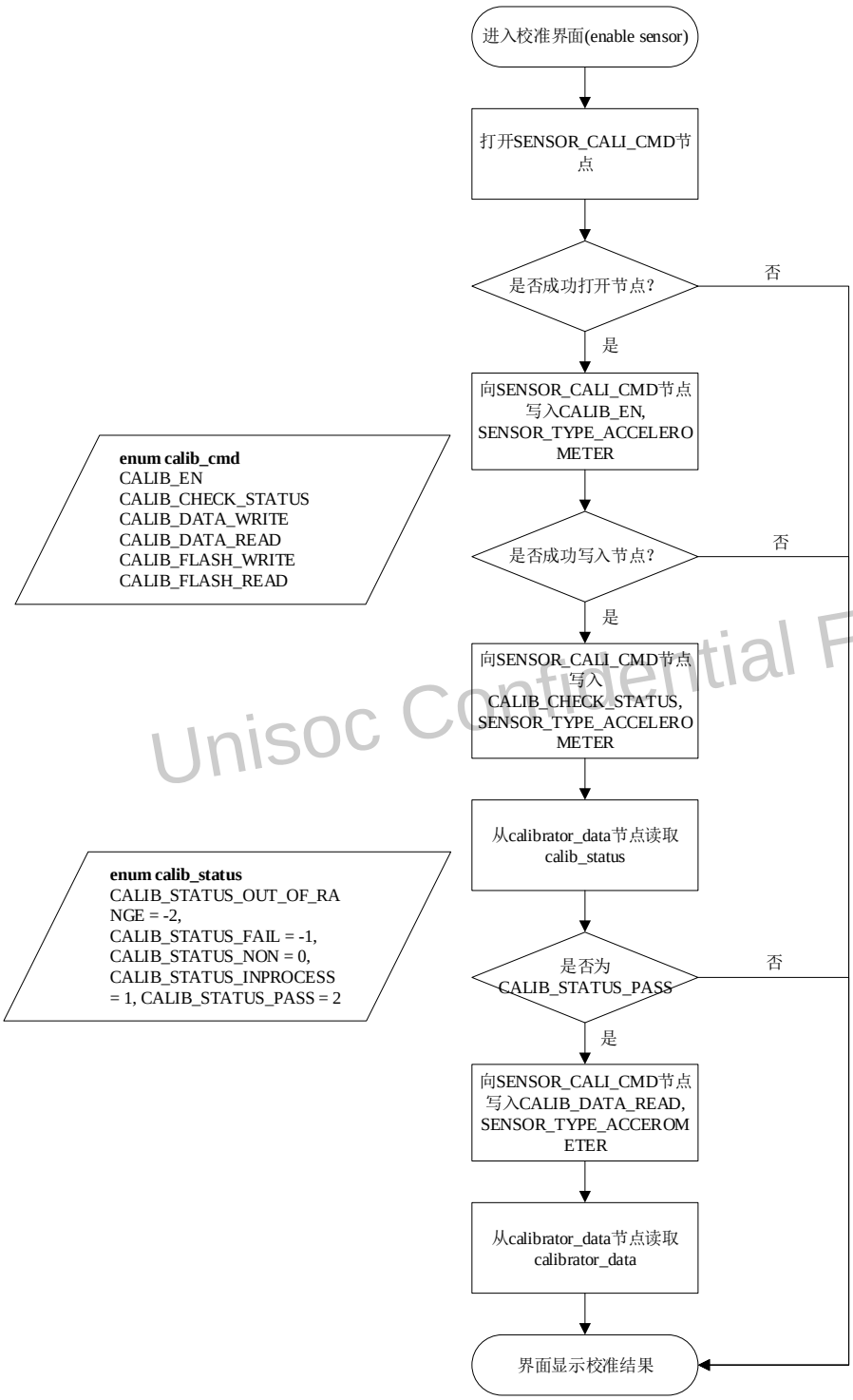
Kernel 层对文件节点 calibrator_cmd、calibrator_data 进行读写，与 MCU 进行通信，和与上层交互。

```
static ssize_t calibrator_cmd_store(struct device *dev,
                                    struct device_attribute *attr,
                                    const char *buf, size_t count)
static ssize_t calibrator_data_show(struct device *dev,
                                    struct device_attribute *attr, char *buf)
```


1.2.1 校准流程示意图

图 1-1 以加速度计为例，从上层操作视角展示了校准流程。

图1-1 校准流程示意图



1.2.2 校准流程说明

下面以加速度计为例介绍基本的校准流程。光传感器校准相对特殊，其校准流程参见“5 光线传感器校准”。

1. 向 MCU 下发使能校准的命令。
   a 进入工程模式界面，进入该类 Sensor 的校准页面，触发校准流程，自动在 sensor_native_mmi_test.cpp 中打开加速度计。
   b sensor_calibration（）中往节点/sys/class/sprd_sensorhub/sensor_hub/calibrator_cmd 写入 CALIB_EN，SENSOR_TYPE_ACCELEROMETER，FLAG
      - CALIB_EN：使能校准
      - FLAG = 1：加速度计校准、力计校准、陀螺仪校准、距离传感器自动校准
      - FLAG = 5：距离传感器手动校准 3cm
      - FLAG = 6：距离传感器手动校准 5cm
2. 向 MCU 查询校准是否成功完成。
   a sensor_calibration（）中往节点/sys/class/sprd_sensorhub/sensor_hub/calibrator_cmd 写入 CALIB_CHECK_STATUS，SENSOR_TYPE_ACCELEROMETER
   b Kernel 向 MCU 请求读取校准操作的状态 status，存入 calibrator_data 节点。
   c 上层从节点/sys/class/sprd_sensorhub/sensor_hub/calibrator_data 读出 status 值。若为 CALIB_STATUS_PASS，则到下一步。否则报错，校准流程未成功完成。
3. 从 MCU 读取校准数据并存储。
   a sensor_calibration（）中往节点/sys/class/sprd_sensorhub/sensor_hub/calibrator_cmd 写入 CALIB_DATA_READ，SENSOR_TYPE_ACCELEROMETER
   b Kernel 向 MCU 请求读取校准数据，对数据进行验证检查。
   c Kernel 将检查通过的数据存入/mnt/vendor/productinfo/sensor_calibration_data 节点。
   d Kernel 在/sys/class/sprd_sensorhub/sensor_hub/calibrator_data 记录操作结果状态。
   e 上层从节点/sys/class/sprd_sensorhub/sensor_hub/calibrator_data 获知校准的最终结果。

2 加速度计校准

2.1 校准原理

加速度计校准方法为平面放置校准。将手机放置于水平桌面，屏幕朝上，保持静止。

说明

水平桌面的要求是倾斜角在±1° 以内。

校准模式对收到的 Sensor 原始数据的范围要求由客户在 Kernel 层代码中进行配置。通过客制化定义以下两个宏值，来确定 X、Y、Z 的约束范围。方法见 shub_core.c 中的 check_acc_cali_data（）。

- ACC_MAX_X_Y_BIAS_VALUE
- ACC_MAX_Z_BIAS_VALUE

为使校准获得成功，上述两个宏值应满足以下要求，否则校准将失败。下面举例中所提供的数据仅供参考。

- X 轴和 Y 轴的数据在理想情况下应为 0.0m/s^2，其进入校准模式时采集到的数据绝对值应小于 2m/s^2。
- Z 轴数据在理想情况下应为 9.8m/s^2，其进入校准模式时采集到的数据绝对应要在 9.8±2.5 m/s^2 之间。

校准后的三轴数据应达到以下范围：

- X 轴和 Y 轴采集到数据的绝对值减去 offset 应小于 0.4m/s^2。
- Z 轴采集到数据的绝对值减去 offset 应在 9.4m/s^2 和 10.2m/s^2 之间。

2.2 校准操作

加速度计校准方式如下：

- 工厂模式下校准
- APK mmi 下校准

工厂模式下校准

步骤 1 将手机放置于水平桌面，静止不动。

步骤 2 在关机情况下，按住音量+键，再同时按下 power 键约 3 秒，此时手机会进入工厂测试模式。

步骤 3 进入“建议抽测项”，选择“加速度传感器校准”。待页面上显示“校准成功”之后，校准完成。

---结束

Apk mmi 下校准

步骤 1 正常开机之后，将手机放置于水平桌面，静止不动。

步骤 2 在拨号页面输入“*#*#83789#*#*”。

步骤 3 选择“Item Test”，然后点击“Acceleration Sensor Calibration”。待页面上显示“pass”时，校准完成。

---结束

2.3 校准数据存储

加速度计校准数据存储空间共 30Byte。如表 2-1 所示，每个 Index 代表一个 Byte（8 个 bit），其中 Index0 到 Index11（即 bit0～bit95）已经使用，Index12～Index29（即 bit96～bit239）保留。

表2-1 加速度计校准数据存储说明

Index（Byte）	格式	名称	说明
0	INT	Bias（X）	转换单位 Android 格式：/10000 示例：（int）data = 98000 Android 格式=98000/10000 = 9.8(m/s^2)
1			
2			
3			
4	INT	Bias（Y）	转换单位 Android 格式：/10000 示例：（int）data = 98000 Android 格式=98000/10000 = 9.8(m/s^2)
5			
6			
7			
8	INT	Bias（Z）	转换单位 Android 格式：/10000 示例：（int）data = 98000 Android 格式=98000/10000 = 9.8(m/s^2)
9			
10			
11			
12~29	无	无	保留

3 磁力计校准

硬磁失真由恒磁体或者被磁化铁材料接近传感器造成。硬磁干扰主要来源于载体上及其周围的永磁体、磁化的铁或钢。这种类型的失真是恒定的，硬磁失真相当于在传感器输出的各个方向都增加了一个恒定幅值的分量，可以用简单的静态运算来进行补偿。

软磁失真是地球磁场和其他靠近传感器的“软”磁性材料共同作用的结果。软磁失真导致依赖方向的变化的场强。随着载体方位的变化，这种相互作用的强度也发生改变。因此，软磁干扰对磁罗盘原始测量值输出的影响随着磁罗盘方位的改变而改变。经过广泛的实测数据分析，这种影响在坐标系上表现为 x、y 轴的输出轨迹类似于一个斜椭圆。

在实际的嵌入式项目中，每一家磁力计厂商的产品设计原理和数据处理算法都有所不同。为满足选料差异性，展锐使用动态加载方式调用厂商提供的库接口，来对磁力计进行使能，关闭，读取数据等操作。

```
//磁力计初始化，分配资源。在enable时调用。
int (*mag_init)(int32 sampleTime_ms, float *mag_offset);
//传入MAG的raw data到厂商算法中处理，得到经过校准之后的mag data。
int (*mag_update)(double *mag_raw_data, double *acc, double *gyro, float *cali_mag, int *mag_accuracy, unsigned long long currTime, uint32 odr);
//磁力计关闭，释放资源。在disable时调用。
int (*mag_close)(float *mag_offset);
//传递软磁校准参数，该接口也可用于下发其他参数数据到库中。在enable时调用。
int (*mag_update_cfg)(int *arg);
```


关于详细的接口说明，请参考适用自己开发平台和软件版本的《SensorHub 动态加载驱动指导》。

各家厂商磁力计的校准数据内容及格式各不相同，需要与厂商确认。

4 陀螺仪校准

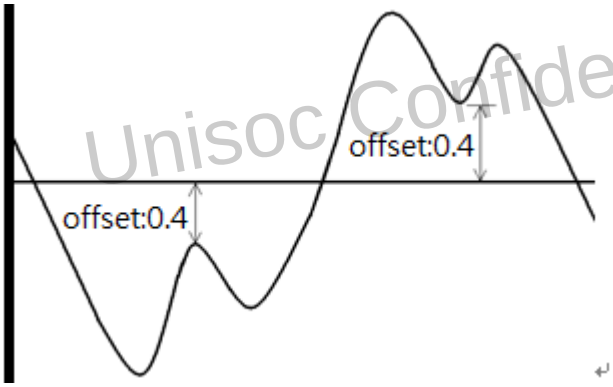
4.1 校准原理

陀螺仪采集的是角速度，无论其处于什么倾斜角度，只要其静止不动，其三轴采集到的数据理想情况下就应该无限趋近于零，目前 Sensorhub 陀螺仪校准的具体条件如下所示：

- 手机放置在稳定的桌面上，确保手机保持静止。
- 采集若干笔陀螺仪的 raw data，取平均值，然后与理论值（0.0rad/s）取差值，得到三轴的 offset。
  - 在整个测试过程中，若所有 offset 的绝对值小于 0.4rad/s，说明校准成功。以下面 BMI160 某次校准结果为例，三轴 offset 绝对值均小于 0.4rad/s，说明校准成功。

    X: 0.001 rad/s Y:-0.027 rad/s Z: 0.028 rad/s
  - 在整个测试过程中若有任何一个 offset 绝对值超过 0.4rad/s，则说明此 Sensor 有问题或是手机晃动，说明校准失败。图 4-1 所示就是陀螺仪数据异常的情况。

图4-1 陀螺仪数据异常示意图



4.2 校准操作

陀螺仪校准方式如下：

- 工厂模式下校准
- APK mmi 下校准

工厂模式下校准

步骤 1 将手机放置于水平桌面上，静止不动。

步骤 2 在关机情况下，按住音量+键，再同时按下 power 键约 3 秒，此时手机会进入工厂测试模式。

步骤 3 进入“建议抽测项”，选择“陀螺仪传感器校准”。当页面上显示“校准成功”时，校准完成。

---结束

Apk mmi 下校准

步骤 1 正常开机之后，将手机放置于水平桌面上，静止不动。

步骤 2 在拨号页面输入“*#*#83789#*#*”。

步骤 3 选择“Item Test”，然后点击“Gyroscope Sensor Calibration”。当页面上显示“pass”时，校准完成。

---结束

4.3 校准数据存储

陀螺仪校准数据存储空间共 30Byte。如表 4-1 所示，每个 Index 代表一个 Byte（8 个 bit），其中 Index0 到 Index11（即 bit0～bit95）已经使用，Index12～Index29（即 bit96～bit239）保留。

表4-1 陀螺仪校准数据存储说明

Index（Byte）	格式	名称	说明
0	INT	Bias（X）	转换单位 Android 格式：/10000 示例：（int）data = 10000 Android 格式=10000/10000 = 1.Of(radians/second)
1			
2			
3			
4	INT	Bias（Y）	转换单位 Android 格式：/10000 示例：（int）data = 10000 Android 格式=10000/10000 = 1.Of(radians/second)
5			
6			
7			
8	INT	Bias（Z）	转换单位 Android 格式：/10000 示例：（int）data = 10000 Android 格式=10000/10000 = 1.Of(radians/second)
9			
10			
11			
12~29	无	无	保留

5 光线传感器校准

5.1 校准原理

目前每台手机都需要进行单独的光线传感器校准。如果 Sensor、手机结构、结构部件、透光部份部件改变，均需要重新校准参数。

说明

UMS312 不支持光线传感器校准。

校准目标

在均匀光照环境中，Light Sensor 上报数据尽量接近仪器读出的数值。

校准流程

光线传感器校准流程与其他传感器校准流程略有不同，具体如下。

1. light_sensor_calibration（）中，往节点/sys/class/sprd_sensorhub/sensor_hub/light_sensor_calibrator 写入 CALIB_DATA_WRITE, SENSOR_TYPE_LIGHT
2. Kernel 向 MCU 请求读取校准数据（若干轮取平均值），对数据进行验证检查。
3. Kernel 将比例系数 400/A 放大 10000 倍保存在/mnt/vendor/productinfo/sensor_calibration_data/light
4. 将比例系数同步到 MCU 侧。

校准方法

准备可调功率光源环境，和一只照度计。在同一位置，对照测取照度计读数和手机未校准时的读数。

目前光感元件厂商建议选取照度为 400 lux 作为基准，以此为例。

表5-1 光感校准数据表

照度计(仪器)数值	400	y(cali data)
手机数值(未校准)	A	x(raw data)

1. 调节光源强度，使照度计读出数值与基准 400 lux 接近。
2. 在照度计相同位置，获取手机此时未校准数值 A。
3. 获得 A 后，自动将 4000000/A 的比值保存在节点：/mnt/vendor/productinfo/sensor_calibration_data/light

说明

由于 Kernel 不支持浮点计算，因此将 400/A 放大 10000 倍保存，MCU 侧同步后会还原成 400/A。

校准计算

raw data 与 cali data 成比例：400/A = y/x

所以 y = (400/A)*x

实际使用时，通过 Sensor 获取 raw data x，MCU 将使用还原后的比例系数(400/A)，会自动计算出校准后的光照数据 y 给到用户上层。

5.2 校准操作

光线传感器需在工厂模式下进行校准，具体操作如下：

步骤 1 将手机放置于水平桌面，静止不动。

步骤 2 在关机情况下，按住音量+键，再同时按下 power 键约 3 秒，此时手机会进入工厂测试模式。

步骤 3 进入“建议抽测项”，选择“光传感器自动校准”。当页面上显示“校准成功”时，校准完成。

---结束

5.3 校准数据存储

光线传感器校准数据存储空间共 30Byte。如表 5-2 所示，每个 Index 代表一个 Byte（8 个 bit），其中 Index0 到 Index3（即 bit0～bit31）已经使用，Index4～Index29（即 bit32～bit239）保留。

表5-2 光线传感器校准数据存储说明

Byte	格式	名称	说明
0	INT	系数	放大了 10000 倍的比例系数：10000*(400/a)
1			
2			
3			
4~29	无	无	保留

6 距离传感器校准

6.1 校准原理

Proximity Sensor 通过发射红外光，测量反射回来的红外线强度来感应距离。遮挡物体越近，反射回来的红外线的能量就越高，其值就越大，反之越远越小。若无遮挡，因为结构关系，也会有一定的反射，所以测得的数值一般不为 0，此值称为 noise。为了判断接近、远离，还需要分别设置接近和远离对应的门限值：

- 当测得数据大于接近门限，判为接近。
- 当测试数据小于远离门限值，判为远离。

所以数值上：接近门限>远离门限> noise。

距离传感器的校准分为静态校准和动态校准两种。对于结构一致性比较好的手机，手机之间的接近门限、远离门限与 noise 的差距不是很大，所以使用批量静态校准就可以满足此情况。对于动态校准，在 enable & get raw data 的时候会有相应的校准算法，详见《SensorHub 距感动态校准设计介绍》。

静态校准分为两种方法：

- 第一种方法是校准底噪：因为结构、Sensor 一致性的问题，每台手机的底噪都有所不同。无遮挡情况下采集底噪，通过静态校准参数中的 noise_low_add、noise_high_add 参数，计算出接近和远离的阀值。
- 第二种方法是校准接近和远离：通过用户自己设定的接近(一般为 3cm)和远离(一般为 5cm)的距离，然后使用 18%的灰卡去标定，直接把判断接近和远离的阀值计算出来。

6.2 校准操作

6.2.1 校准底噪

校准底噪有两种方式：

- 工厂模式下校准
- APK mmi 下校准

工厂模式下校准

步骤 1 将手机放置于水平桌面，静止不动。

步骤 2 在关机情况下，按住音量+键，再同时按下 power 键约 3 秒，此时手机会进入工厂测试模式。

步骤 3 进入“建议抽测项”，选择“距离传感器自动校准”。当页面上显示“校准成功”时，校准完成。

---结束

Apk mmi 下校准

步骤 1 正常开机之后，将手机放置于水平桌面，静止不动。

步骤 2 在拨号页面输入“*#*#83789#*#*”。

步骤 3 选择“Item Test”，然后点击“Proximity Sensor Calibration”，选择“自动校准”。当页面上显示“pass”时，校准完成。

---结束

6.2.2 校准接近远离

校准接近和远离有两种方式：

- 工厂模式下校准
- APK mmi 下校准

工厂模式下校准

步骤 1 将手机放置于水平桌面，静止不动。

步骤 2 在关机情况下，按住音量+键，再同时按下 power 键约 3 秒，此时手机会进入工厂测试模式。

步骤 3 进入“建议抽测项”，选择“距离传感器手动校准”，然后根据页面上的提示进行操作。当页面上显示“校准成功”之后，校准完成。

---结束

Apk mmi 下校准

步骤 1 正常开机之后，将手机放置于水平桌面，静止不动。

步骤 2 在拨号页面输入“*#*#83789#*#*”。

步骤 3 选择“Item Test”，然后点击“Proximity Sensor Calibration”。

步骤 4 选择“手动校准”，并根据页面上的提示进行操作。当页面上显示“pass”时，校准完成。

---结束

6.3 校准数据存储

距离传感器校准数据存储空间共 30Byte。如表 6-1 所示，每个 Index 代表一个 Byte（8 个 bit），其中 Index0 到 Index12（即 bit0～bit103）已经使用，Index13～Index29（即 bit104～bit239）保留。

表6-1 距离传感器校准数据存储说明

Byte	格式	名称	说明
0	INT		

Byte	格式	名称	说明
1		自动校准	自动校准的 calibrator
2			
3			
4	INT	手动校准（3cm）	手动校准 3cm 的 calibrator
5			
6			
7			
8	INT	手动校准（5cm）	手动校准 5cm 的 calibrator
9			
10			
11			
12	BYTE	位 Flag	• bit0：自动校准 • bit1：手动校准 3cm • bit2：手动校准 5cm 所在位为 1 表示执行过对应类型的校准操作
13~29	无	无	保留

7 参考文档

1. 《SensorHub 动态加载驱动指导》

说明

根据 Android 版本选择对应的 SensorHub 动态加载驱动指导文档。

2. 《SensorHub 距感动态校准设计介绍》