全 志 科 技
Allwinner Technology

A20 平台 sensor hal 层使用说明文档

V1.0

2013-06-17

全 志 科 技
Allwinner Technology

Revision History

Version	Date	Changes compared to previous issue
V1.0	2013-06-17	初始版本

全 志 科 技
Allwinner Technology

目录

1. 概述

1.1 编写目的

文档针对 sensor hal 层的使用方法进行讲解，为达到添加新的模组以及 sensor 的使用。

1.2 使用范围

适用于 A20 对应平台。

1.3 相关人员

项目中 sensor 开发，维护以及使用人员应认真阅读该文档。

1.4 文档介绍

（由于文档不断补充，代码也不断更新，有些地方可能和实际代码中有细微差别，请注意）

2. SENSOR HAL 层介绍

2.1 功能介绍

Sensor hal 层的主要功能为接收驱动上传的数据，对数据进行相关的处理后，上报给 android 系统，供各种应用使用。

2.2 源码介绍

Sensor hal 层的源码目录如下：

android4.x\device\softwinner\common\hardware\libsensors

包括以下文件：

```
android4.x\device\softwinner\common\hardware\libsensors
├——— AccelSensor.cpp
├——— AccelSensor.h
├——— Android.mk
├——— chown.cpp
├——— GyroSensor.cpp
├——— GyroSensor.hs
├——— InputEventReader.cpp
├——— InputEventReader.h
├——— insmodDevice.cpp
├——— LibFusion_ARM_cpp.a
├——— LightSensor.cpp
├——— LightSensor.h
├——— MagnetoSensor.cpp
├——— MagnetoSensor.h
├——— MagSensor.cpp
├——— MagSensor.h
├——— MEMSAlgLib_eCompass.h
├——— MEMSAlgLib_Fusion.h
├——— PressSensor.cpp
├——— PressSensor.h
├——— ProximitySensor.cpp
```

```
├── ProximitySensor.h
├── SensorBase.cpp
├── SensorBase.h
├── sensorDetect.cpp
├── sensors.cpp
├── sensors.h
├── TempSensor.cpp
└── TempSensor.h
```

名称	作用
AccelSensor.cpp AccelSensor.h	Gsensor 数据处理文件
Android.mk	编译文件说明
chown.cpp	修改设备节点权限文件，目前尚不使用
GyroSensor.cpp GyroSensor.h	陀螺仪（l3gd20）数据处理文件
InputEventReader.cpp InputEventReader.h	Sensor 数据处理关键文件
insmodDevice.cpp	自动检测时，加载设备驱动文件
LibFusion_ARM_cpp.a MEMSAlgLib_eCompass.h MEMSAlgLib_Fusion.h	陀螺仪（l3gd20） 地磁传感器（lsm303d）静态库文件
LightSensor.cpp LightSensor.h	光感数据处理文件
MagnetoSensor.cpp MagnetoSensor.h	地磁传感器（lsm303d）数据处理文件
MagSensor.cpp MagSensor.h	地磁传感器（fxos8700）数据处理文件
PressSensor.cpp PressSensor.h	压力传感器数据处理文件
ProximitySensor.cpp ProximitySensor.h	接近式传感器数据处理文件
SensorBase.cpp SensorBase.h	Sensor hal 架构基础类
TempSensor.cpp TempSensor.h	温度传感器处理文件

3. 使用介绍

Sensor hal 架构中根据 input 系统中已经注册的 sensor 设备名称来进行对应的类注册以及将相关的传感器相关信息注册到 android 系统中。

3.1 关键数据结构介绍

3.1.1 struct sensor_info

```
struct sensor_info{
        char sensorName[64];
        char classPath[128];
        float priData;
};
```

该结构体主要用于存储支持的 sensor 的相关信息，在相关的 sensor 数据处理文件中使用。

sensorName：sensor 驱动中注册到 input 系统中的设备名称。

classPath：sensor 设备在 sys 中的路径，用于节点的操作。

priData：主要用于 sensor 处理时的转换常量。主要用于 gsensor 中，其他 sensor 有需要时也可以使用。

3.1.2 struct sensors_data

```
struct sensors_data{
        char name[64];
        float lsg;
};
```

该结构用于存储 sensor 的相关数据。主要用于结构体 struct sensor_extend_t 中，即增加 sensor 的描述信息。

name：sensor 驱动中注册到 input 系统中的设备名称。拷贝到结构体 struct sensor_info 中的 sensorName 中。

lsg：主要用于 sensor 处理时的转换常量。主要用于 gsensor 中，其他 sensor 有需要时也可以使用。拷贝到结构体 struct sensor_info 中的 priData 中。

3.1.3 struct sensor_extend_t

```
struct sensor_extend_t{
        struct sensors_data sensors;
        struct sensor_t sList;
};
```

该结构体用于描述一个传感器的相关信息。

Sensors：传感器的额外信息，填充注册到 input 系统中的设备名称以及 sensor 数据转换时的单位常量。

sList：系统中注册一个传感器时需要的传感器的描述信息。

3.1.4 struct status

```
struct status{
        bool isUsed;
        bool isFound;
};
```

该结构体用于记录 sensor 是否是否使用，是否找到的状态。

isUsed：系统中是否启用某一种 sensor。启用时设置为 true。

isFound：系统中是否已经找到了某一种 sensor。找到时，被标记为 true。

3.1.5 struct o_device

```
struct o_device{
        int isFind;
        char name[32];
};
```

该结构体用于储存那些 input 中不属于 sensor 的 input 设备，减少 sensor 设备查找的时间。

isFind：是否已经找到系统中固定注册的 input 设备。

name：注册到 input 系统中的不属于 sensor 设置的设备名称。

3.1.6 关键变量 sNumber

变量 sNumber 用于记录已经找到的 sensor 设备的个数。

3.2 sensor 使用说明

Sensor hal 中，根据变量 seStatus 来确定是否使用某一种 sensor 以及是否已经找到了相关的 sensor 设备。根据 seStatus 的变量来确定是否创建某一种 sensor 类。使用时，请确定 sensorDetect.cpp 中的 statusInit 函数，是否已经设置了该类的 seStatus 的 isUsed 变量为 true，该函数中 seStatus 变量中的 isFound 必须初始化为 false。

3.2.1 enable 与 delay 函数的说明

Sensor 驱动中需要实现 delay 和 enable 这两个函数，用于应用控制读取数据，以及上报数据的时间间隔。在各自的 sensor 处理函数中需要实现这两个函数的接口。驱动与 hal 层使用节点的方式进行读写，因此涉及到节点的读取权限问题。

节点权限修改的源文件为： android4.2.1\device\softwinner\wing-common\sensor.h

该文件将被复制到 system/bin 目录下。在 sensors.cpp 文件中的 open_sensors 函数通过 property_set 设置为 1 时调用节点权限的修改。

修改的名称为 delay 和 enable。路径为/sys/devices/virtual/input/下的 input 设备，因此使用时请注意驱动中实现的 delay 与 enable 的函数节点名称以及输入设备的路径是否与设置的路径相符。

3.2.2 Gsensor 设备

Gsensor 设备已经支持了 bma250，mma8452，mma7660，mma865x，afa750，lis3de_acc，lis3dh_acc，lsm303d_acc，FreescaleAccelerometer（fxos8700），kxtik，dmard10，dmard06，mxc622x，lis35de 等。使用时，请确认驱动注册 input 设备名称是否跟 struct sensor_extend_t gsensorList 中的设备名称匹配。结构体 struct sensor_extend_t 中的 sensors 变量中的 lsg 变量为存取 gsensor 设备将数据转换为单位（1G：9.8）时的转换常量。Lsg 的数值一般情况下由模组厂提供。该数值一般情况下为当将机器平放时，z 轴上报的数据。

3.2.3 gyr（陀螺仪）设备

陀螺仪设备目前支持的为 l3gd20。设备调试时的方向调整主要通过驱动端进行相关的调整。驱动位于\lichee\linux-3.x\drivers\input\gyr 下。

陀螺仪三个轴的方向，可以修改 drivers/input/gyr/l3gd20_gyr.c 文件中的三个变量。

default_l3gd20_gyr_pdata.negate_x
default_l3gd20_gyr_pdata.negate_y
default_l3gd20_gyr_pdata.negate_z

这三个变量为 0 代表此轴方向取正，为 1 代表此轴方向取负。

3.2.4 compass(地磁传感器)设备

地磁传感器目前主要支持 fxos8700 以及 lsm303d，由于地磁传感器涉及较多数据的处理以及静态库的使用，因此两个设备都有自己的数据处理文件。

MagSensor.cpp 与 MagSensor.h 为 fxos8700 数据处理。8700 通过第三方的 magd 来接收地磁与加速度数据，转换后传给 hal 层。详细的使用说明参见 3.3.1 节。

MagnetoSensor.cpp 与 MagnetoSensor.h 为 lsm303d 的数据处理文件。默认情况下编译到 hal 层的处理文件。详细的使用说明，参见 3.3.2 节。

使用时请确认 delay 与 enable 相关接口的可操作性。

3.2.5 其他 sensor 设备

其他 sensor 设备需要关注的依旧是 delay 以及 enable 相关接口的可操作性以及对于数据的处理方式。

3.3 compass 使用说明

3.3.1 fxos8700 使用说明

Fxos8700 使用步骤如下所示：

1）sysconfig.fex 的配置

目录：lichee\tools\pack\chips\sun7i\configs\android\wing-xxx ）

```
[gsensor_para]
gsensor_used            = 1
gsensor_twi_id          = 1 //使用第一组 twi，根据硬件情况进行配置
```

因为 fxos8700 为地磁与加速度二合一传感器，模组驱动将读取 gsensor 配置文件即可。

2）驱动

目录：lichee\linux-3.x\drivers\input\e_compass\fxox8700.c ）

驱动应该编译为模块的形式。

3）hal 层以及 magd

magd 文件为 fxos8700 接受驱动的加速度数据以及地磁数据，进行转换后传到 hal 层供系统使用。magd 目录：

android4.2.1\device\softwinner\common\hardware\magd

编译后将在 android4.2.1\out\target\product\wing-evb-v20\system\bin 目录下 找到 magd

Hal 层编译后在\android4.2.1\out\target\product\wing-evb-v20\system\lib\hw 下找到 sensors.exDroid.so。

注意：当 hal 层已经编译好后，需要修改为支持 8700 的 hal 层，则需要在

编译的 android4.2.1\device\softwinner\common\hardware\目录下执行命令：find .| xargs touch 之后在进行项目的编译与打包。否则可能出现系统飞掉的情况。

4）配置文件的修改（目录：android4.2.1\device\softwinner\wing-xxx ）

（1）BoardConfig.mk 文件（表示编译 8700 相关文件）

在该文件中增加语句：

```
SW_BOARD_USES_MAGSENSOR_TYPE = fxos8700
```

（2）init.sun7i.rc 文件的修改

A、加载驱动：

```
#sensor
insmod /system/vendor/modules/fxos8700.ko
```

若使用自动检测时，需要加载驱动 sw_device.ko,即：

```
insmod /system/vendor/modules/sw_device.ko
```

B、启动 magd

```
service magdservice /system/bin/magd
        class main
        user root
        group root
        onshot
```

5)关于方向的调整

（1） gsensor 方向，可以调整 gsensor.cfg 文件，8700 对应的方向变量为：

```
;---------------------------
;name:fxos8700
;---------------------------
gsensor_name = FreescaleAccelerometer
gsensor_direct_x = false
gsensor_direct_y = false
gsensor_direct_z = false
gsensor_xy_revert = true
```

由于 layout 的原因，需要调整这组方向值，只有 true 与 false 两个值，当转动机器时，发现 X 轴反向，可以将 gsensor_direct_x 中原来的 false 修改为 true。

当发现 x 轴与 y 轴对调，可以将 gsensor_xy_revert 的 true 修改为 false。

当发现机器之后垂直 90°的时候，才发生方向的转动，说明 Z 轴方向不对，将 gsensor_direct_z 原来的 false 修改为 true。

方向调整的具体方法可以参照“A20 平台 G-sensor 模块开发说明文档 V2.0.doc”中的 6.1

节。

（2）compass 的方向调整

需要找一个支持 compass 的手机或者是机器作为参考，使用 Z-Devicetest 1.12 (v34).apk 中的指南针进行测试，如方向指向一致，且转动时，变化幅度一致，说明方向调整正确。

应先调整好 gsnesor 的方向后在调整 compass 方向。

3.3.2 lsm303d 使用说明

Sensor hal 层默认情况下为编译 lsm303d 的相关处理文件。使用步骤如下：

1）sysconfig.fex 的配置

目录：lichee\tools\pack\chips\sun7i\configs\android\wing-xxx ）

```
[gsensor_para]
gsensor_used            = 1
gsensor_twi_id          = 1 //使用第一组 twi，根据硬件情况进行配置
```

因为 ls303d 为地磁与加速度二合一传感器，模组驱动将读取 gsensor 配置文件即可。

2）驱动

目录：lichee\linux-3.x\drivers\input\e_compass\lsm303d.c ）

驱动应该编译为模块的形式。

3）驱动的加载（目录：android4.2.1\device\softwinner\wing-xxx\init.sun7i.rc ）

```
#sensor
insmod /system/vendor/modules/lsm303d.ko
```

若使用自动检测时，需要加载驱动 sw_device.ko,即：

```
insmod /system/vendor/modules/sw_device.ko
```

4) 关于方向的调整

（1） gsensor 方向，可以调整 gsensor.cfg 文件，8700 对应的方向变量为：

```
;-------------------------
;name:lsm303d_acc
;-------------------------
gsensor_name = lsm303d_acc
gsensor_direct_x = false
gsensor_direct_y = false
gsensor_direct_z = false
gsensor_xy_revert = true
```

方向调整的具体方法可以参照“A20 平台 G-sensor 模块开发说明文档 V2.0.doc”中的 6.1 节。

（2）compass 的方向调整

需要找一个支持 compass 的手机或者是机器作为参考，使用 Z-Devicetest 1.12 (v34).apk 中的指南针进行测试，如方向指向一致，且转动时，变化幅度一致，说明方向调整正确。

应先调整好 gsnesor 的方向后在调整 compass 方向。

3.4 添加一种新 sensor 模组的方法

添加一种新的 sensor 模组，只需要添加该 sensor 的相关的描述信息即可。

需要在文件：sensorDetect.cpp 中找到该类 sensor 的结构体 struct sensor_extend_t 描述数组，填充该数据的内容即可。

如需要增加一种 lightsensor 设备 jsa1212，在类 sensor 的结构体 struct sensor_extend_t 描述数组名称为 ligSensorList[]，没有增加该设备的数组内容为：

```
struct sensor_extend_t ligSensorList[] = {
        {
                {
                        "al3010", 0,
                }, {
                         "AL3010 Light sensor",
                        "Intersil ",
                        1,
                        SENSORS_LIGHT_HANDLE,
                        SENSOR_TYPE_LIGHT,
                        300.0f, 1.0f,
                        0.35f, 0, { }
                }
        },
        {
                {
                        "lightsensor", 0,
                }, {
                        "LTR-C216R-14 sensor LIGHT",
                        "LTR ",
                        1,
                        SENSORS_LIGHT_HANDLE,
                        SENSOR_TYPE_LIGHT,
                        20000.0f, 20.0f,
```

```
                        0.2f, 0, { }
                }
        },

};
```

增加该设备的描述信息后的 ligSensorList[]数组如下：

```
struct sensor_extend_t ligSensorList[] = {
        {
                {
                        "al3010", 0,
                }, {
                         "AL3010 Light sensor",
                        "Intersil ",
                        1,
                        SENSORS_LIGHT_HANDLE,
                        SENSOR_TYPE_LIGHT,
                        300.0f, 1.0f,
                        0.35f, 0, { }
                }
        },
        {
                {//struct sensors_data 描述信息
                         "jsa1212als", 0,
                },
            {//struct sensor_t 描述信息
                    "JSA1212 Light sensor",
                        "SOLTEAM",
                        1,
                        SENSORS_LIGHT_HANDLE,
                        SENSOR_TYPE_LIGHT,
                        100000.0f, 1.0f,
                        0.005f, 0, { }
```

```
                }
        },

        {
                {
                        "lightsensor", 0,
                }, {
                        "LTR-C216R-14 sensor LIGHT",
                        "LTR ",
                        1,
                        SENSORS_LIGHT_HANDLE,
                        SENSOR_TYPE_LIGHT,
                        20000.0f, 20.0f,
                        0.2f, 0, { }
                }
        },

};
```