Android Sensor Customization

Version: 1.0
Release date: 2011-10-10

Document Revision History

Revision	Date	Author	Description
1.0	2011-05-31	Weiqi Fu	Initial Draft

Table of Contents

Lists of Tables and Figures

Error! No table of figures entries found.

錯誤! 找不到圖表目錄。

1 Introduction

This document is used to descript how to configure the customization setting of android sensors and porting new sensors.
This document is for MT6573 load(after 1120MP). (Android2.3)

2 Customization setting

2.1 Configure file setting

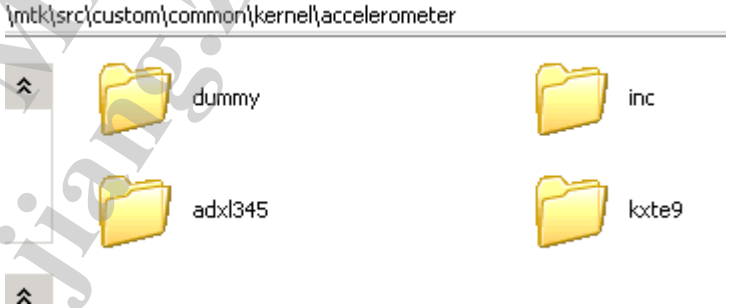
mtk\mak\$(project).mk file set the sensors' configure in each project. (for48MP load)
mediate\config\$(project)\ProjectConfig.mk (after 48MP use this path)

```
# Android sensor device
MTK_SENSOR_SUPPORT = yes

CUSTOM_KERNEL_MAGNETOMETER = ami304

CUSTOM_KERNEL_ACCELEROMETER = adxl345

CUSTOM_KERNEL_ALSPS = cm3623

CUSTOM_HAL_SENSORS = sensor

CUSTOM_HAL_MSENSORLIB = ami304
```

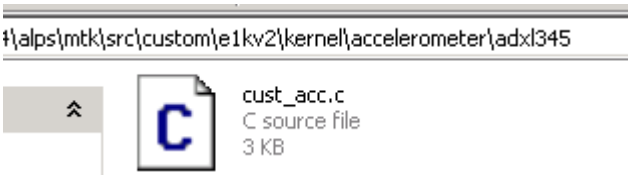
2.1.1 Kernel driver setting

If project have any sensor, the MTK_SENSOR_SUPPORT = yes, if have no sensor, set to no. Follow define the M-sensor, G-sensor, ALSPS sensors, if project have this sensor, define the sensor type, or set NULL. For example if project use adxl345 as the G-sensor, it define as follow:
CUSTOM_KERNEL_ACCELEROMETER = adxl345
If have no G-sensor, set as follow:
CUSTOM_KERNEL_ACCELEROMETER =

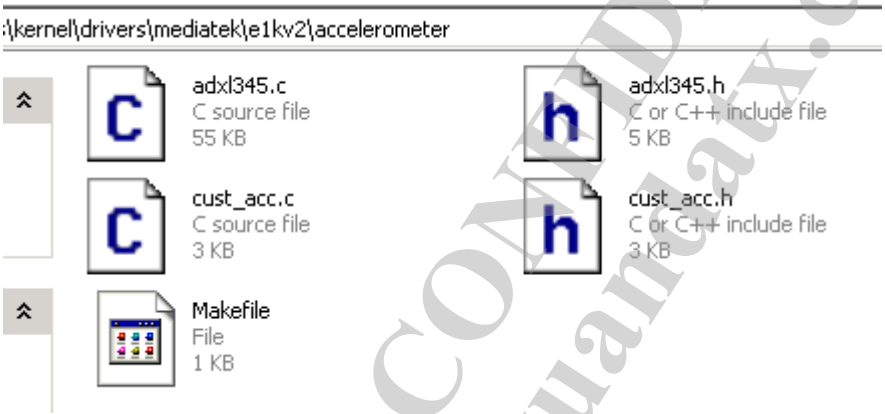
Notes: CUSTOM_KERNEL_XXXX will as the globe value in project, driver and android can use this value. CUST_KERNEL_XXX will soft link the name driver to kernel/drivers/xxx, and then project will use this sensor.
As adxl345 as example:
Mtk\src\custom\common\kernel\accelerometer store all the G-sensor drivers. (For 48MP load)
mediate\custom\kernel\accelerometer store all the G-sensor drivers. (After 48MP use this path)



Mtk\src\custom\&(project)\kernel\accelerometer folder have project customization configure file. (For 48MP load)
Mediate\custom\&(project)\kernel\accelerometer folder have project customization configure file. (After 48MP load)



In gen_cust phase, all adxl345 folder files and inc folder files can soft-link to path1 folder(for 48MP) or path2 folder(after 48MP). And auto creates the makefile. Then will build the adxl345 g_sensor driver with customization setting.



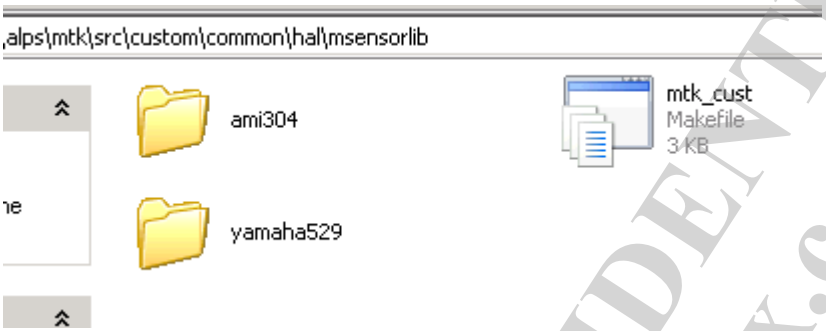
path1: kernel\drivers\mediatek\&(porject)\accelerometer
path2: mediatek\custom\out\\&(porject)\kenel\accelerometer

2.1.2 HAL build

There have only two values to set:
CUSTOM_HAL_SENSORS = sensor
Define the HAL customization folder, default sensor
CUSTOM_HAL_MSENSORLIB = yamaha529
Define which M-sensor daemon used, this should equal as CUSTOM_KERNEL_MAGNETOMETER.

Mtk\src\custom\&(project)\hal\sensors folder have project customization configure file(for48MP load)
Mediate\ custom\&(project)\hal\sensors folder have project customization configure file(afer 48MP load)

Mtk\src\custom\common\hal\msensorlib store all the msensor daemon source code.(for 48MP load)
mediatek\source\hardware\sensor\lib store all the msensor daemon source code.(after 48MP load)



Then they will soft-link the mach folder to follow path: (for 48MP load)

After 48MP msensor lib will not be soft-link to anywhere.
Then it will build the sensor HAL with configure setting and Msensor daemon application.

2.2 Gsensor customization

We choose adxl345 as an example
mediatek\&(project)\kernel\accelerometer folder have sensor project customization configure file cust_acc.c.

```
/*----------------------------------------------------------------------------*/
static struct acc_hw cust_acc_hw = {
    .i2c_num = 0,
    .direction = 1,
    .power_id = MT65XX_POWER_NONE,  /*!< LDO is not used */
    .power_vol= VOL_DEFAULT,        /*!< LDO is not used */
    .firlen = 16,                   /*!< don't enable low pass fileter */
};
/*----------------------------------------------------------------------------*/
struct acc_hw* get_cust_acc_hw(void)
{
    return &cust_acc_hw;
}
```

mediatek\ custom\common\kernel\accelerometer\inc\cust_acc.h file as follow:

```
#ifndef __CUST_ACC_H__
#define __CUST_ACC_H__

struct acc_hw {
    int i2c_num;    /*!< the i2c bus used by the chip
    int direction;  /*!< the direction of the chip */
    int power_id;   /*!< the LDO ID of the chip, MT65
    int power_vol;  /*!< the Power Voltage used by the
    int firlen;     /*!< the length of low pass filter
};

extern struct acc_hw* get_cust_acc_hw(void);
#endif
```

2.3 ALS_PS customization

Notes: for ALS_PS driver, there have some different.

Static struct alsps_hw cust_alsps_hw = {
.i2c_num =2, // Use I2C 2
.power_id = MT6516_POWER_NONE, // Sensor used LDO, now it is null.
.power_vol = VOL_DEFAULT, // LOD vol, now is null
.i2c_addr = {0xc0, 0x48, 0x78, 0x00}, // for multiple slave address sensor
.ps_threshold = 3;
.als_window_loss =0; //compensating for the loss when light is passing through the glasses which covered on the sensor
}

Notice:

a) Because cm3623 have multiple addresses for different mode, so it defines multiple slave address. CM3623 have two type addresses for different type. You can choose your type.
b) .als_level depend on specific sensor, it means how many light level the hardware support.
c) .als_value has no relationship with hardware, these values will report to sensor HAL and applications will use these values

als_level will mapping to als_value, you can see the details in abc123_get_als_value(). In different projects the mapping may be different.

2.4 Gyroscope customization

We choose mpu3000 as an example

mediatek\&(project)\kernel\gyroscope\mpu3000 folder have sensor project customization configure file cust_gyroscope.c.

```
static struct gyro_hw cust_gyro_hw = {
    .i2c_num = 0,
    .direction = 1,
    .power_id = MT65XX_POWER_NONE,   /*!< LDO is not used */
    .power_vol= VOL_DEFAULT,         /*!< LDO is not used */
    .firlen = 16,                    /*!< don't enable low pass fileter */
   // .power = cust_gyro_power,
};
```

We need to config i2c_num and direction

2.5 HAL customization

Mtk\src\custom\&(project)\hal\sensors folder(for 48MP);
mediatek\custom\&(project)\hal\sensors folder have project customization configure file, as follow

he

Hwmsen_custom.c file define this project support which sensor and its attribute. Detail information of sensor attribute please read mtk\src\custom\common\hal\sensors\readme.txt file.(for 48MP) ; mediatek\custom\source\hardware\sensor\hwmsen\sensor_list_readme.txt file.(after 48MP)

```
#include <hardware/sensors.h>
#include <linux/hwmsensor.h>
#include "hwmsen_custom.h"

struct sensor_t sSensorList[MAX_NUM_SENSORS] =
]{
]   {
        .name       = "YAMAHA Orientation sensor",
        .vendor     = "Yamaha",
        .version    = 1,
        .handle     = ID_ORIENTATION,
        .type       = SENSOR_TYPE_ORIENTATION,
        .maxRange   = 360.0f,
        .resolution = 1.0f,
        .power      = 0.25f,
        .reserved   = {}
    },
```

Hwmsen_custom.h file define project support how many sensor, and other information.

```
#ifndef __HWMSEN_CUSTOM_H__
#define __HWMSEN_CUSTOM_H__

#define MAX_NUM_SENSORS                 3
//#define MSENSOR_LIB

/* Only needed when Msensor need to run daemon in hal layer,
* If don't, set follow marco to NULL
*/
#define  MSENSOR_DAEMON_NAME
#define  MSENSOR_DAEMON_PROP
/*-----------------------------------------------------------------

#endif
```

MAX_NUM_SENSORS define support sensor number. If you support G-sensor, M-sensor, orientation sensor, you should define it as 3 and define detail information in hwmsen_custom.c file.

Notes: if project use AMI304 M-sensor, please define as follow:

```
#define MSENSOR_LIB

/*---If only needed when Msensor have library----------
#define  MSENSOR_DAEMON_NAME     "ami304d"
#define  MSENSOR_DAEMON_PROP     "init.svc.ami304d"
/*----------------------------------------------------
```

Or don't define MSENSOR_LIB, and set name and prop to null.

2.6 Msensor customization

For different msensor, because they use different daemon application and different device type, so we need configure for each Msensor.

Before configure each Msensor, we need to check the follow files, and make sure the init.rc in normal mode and factory mode use right configration.

1. check mediatek/config/$(project)/init.rc and make sure we have follow code, if have not we need to add these code at in init.rc

```
    # Sensor
    chmod 0666 /dev/hwmsensor
    chmod 0666 /dev/msensor
    chmod 0666 /dev/gsensor
    chmod 0666 /dev/alsps
    mkdir /data/amit/
    mkdir /data/misc/sensors 0777 system system
    mkdir /data/misc/akmd 0777 compass compass

.....
.....
    service ami304d /system/bin/logwrapper /system/bin/ami304d
    disabled
    oneshot
```

```
service memsicd /system/bin/memsicd
        disabled
        oneshot

service akmd8975 /system/bin/akmd8975
      disabled
      user compass
      group compass

service orientationd /system/bin/orientationd
      disabled
      user compass
      group input

service geomagneticd /system/bin/geomagneticd
      disabled
      user compass
      group system input
```

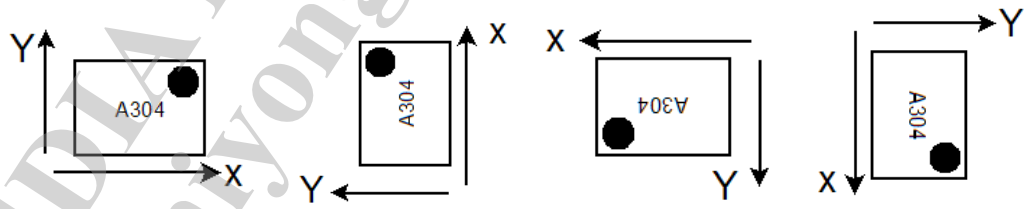
2. check mediatek/custom/$(project)/factory/init.rc
   Make sure we have the same configuration with the step 1.

2.6.1 AMI304 Msensor

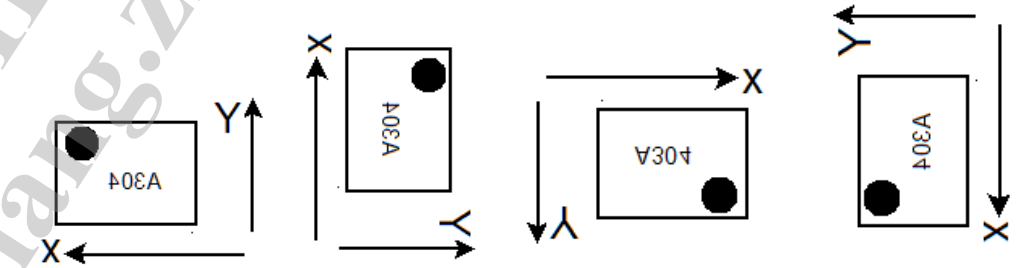
- In mediatek/config/$(project)/Projectconfig.mk
  Make sure MTK_AUTO_DETECT_MAGNETOMEER = yes
  CUSTOM_KERNEL_MAGNETOMETER = ami304
- Layout setting

If device install in board in different direction, the layout is different, follow will describe each setting (define in Mtk\src\custom\$(project)\kernel\magnetometer\mmc314x\cust_mag.c file [for 48MP]; mediatek\custom\$(project)\kernel\magnetometer\mmc314x\cust_mag.c file[after 48MP]).

Layout: 0~3 (Top-view)



Layout: 4~7 (bottom-view)



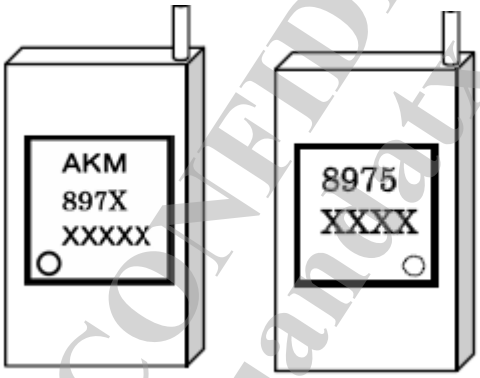
2.6.2 AKM8975 Msensor

- In mediatek/config/$(project)/Projectconfig.mk
  Make sure MTK_AUTO_DETECT_MAGNETOMEER = yes
  CUSTOM_KERNEL_MAGNETOMETER = akm8975

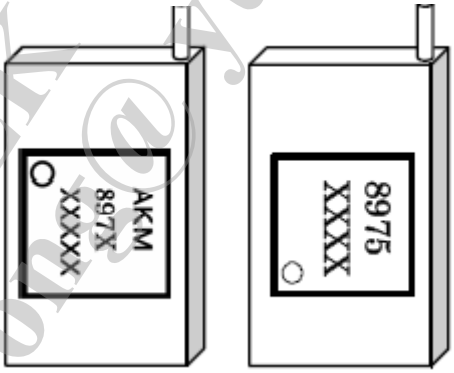
- Layout setting

AKM897X have two type, each type the layout setting is different, follow will describe each setting (define in mediatek\custom\$(project)\kernel\magnetometer\akm8975\akm8975_cust_mag.c file).

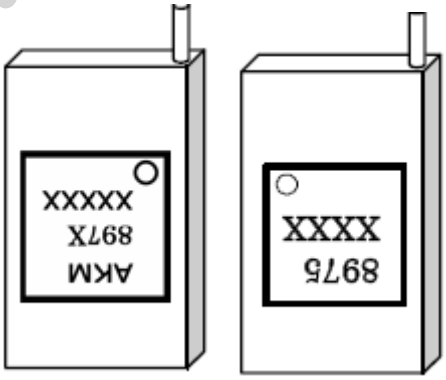
Layout = 1



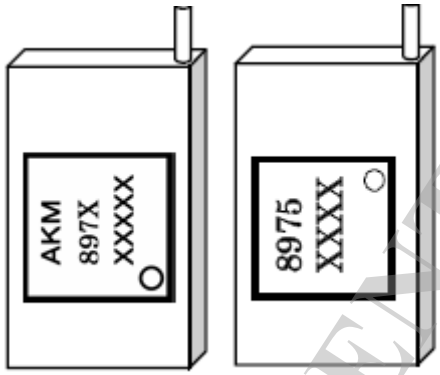
Layout = 2



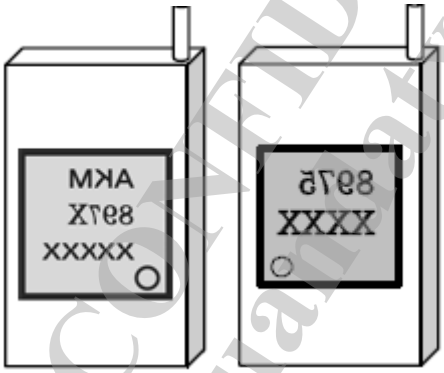
Layout = 3



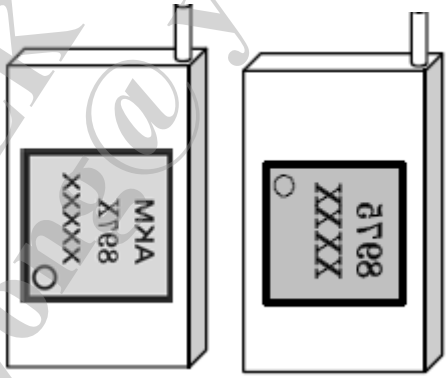
Layout = 4



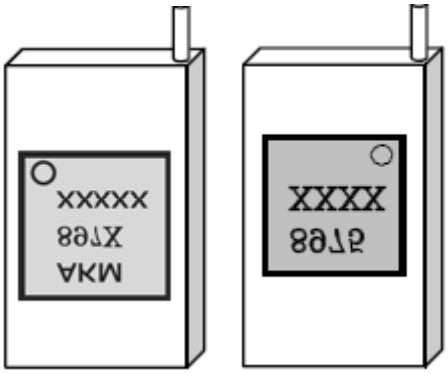
Layout = 5



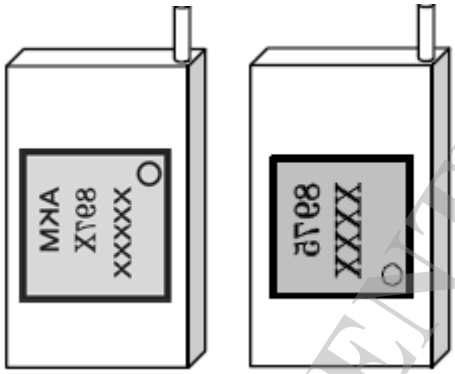
Layout = 6



Layout = 7



Layout = 8

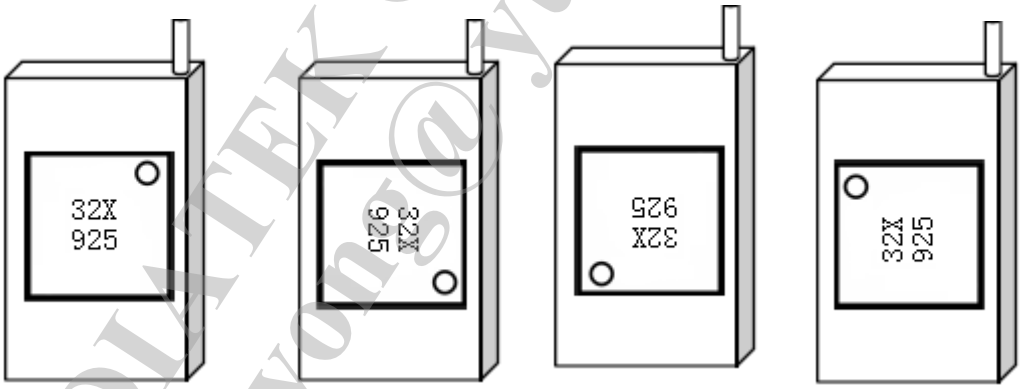


2.6.3 MMC328x Msensor

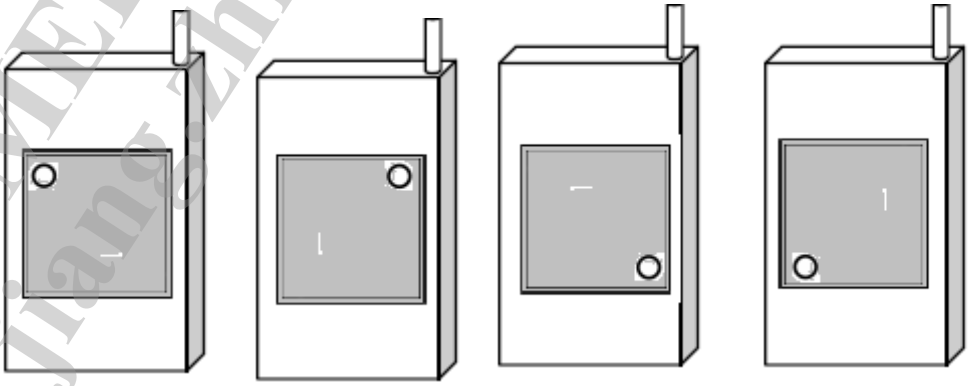
- In mediatek/config/$(project)/Projectconfig.mk
  Make sure MTK_AUTO_DETECT_MAGNETOMEER = yes
  CUSTOM_KERNEL_MAGNETOMETER = mmc328x
- Layout setting

If device install in board in different direction, the layout is different, follow will describe each setting (define in Mtk\src\custom\$(project)\kernel\magnetometer\mmc314x\cust_mag.c file[for 48MP]; define in mediatek\custom\$(project)\kernel\magnetometer\mmc314x\cust_mag.c file[after 48MP];).

Layout: 0~3 (Top-view)



Layout: 4~7 (bottom-view)



2.6.4 Yamaha529 Msensor

- In build/target/$(project)/init.rc file.(for 48MP); In mediatek/config/$(project)/init.rc file.(after 48MP).

Because Yamaha529 Msensor use input device, in different project the input device number is different. So should get the input device number after system is boot-up use adb shell.
cat /sys/class/input/inputX/name (X form 0 – maximum number of input device), get the "orientation", "geomagnetic" and "geomagnetic_raw" device name a, b and c. Then add follow code in #sensor part:
chown system /sys/class/input/inputa/enable
chown system /sys/class/input/inputa/delay
chown system /sys/class/input/inputa/wake
chown system /sys/class/input/inputb/enable
chown system /sys/class/input/inputb/wake
chown compass /sys/class/input/inputc/offsets
Add follow code in servicer part.
service orientationd /system/bin/orientationd
user compass
group input
service geomagneticd /system/bin/geomagneticd
user compass

- group system input In mtk\src\custom\&(project)\hal\sensors\sensor\hwmsen_custom.h file(for 48MP). mediatek\custom\&(project)\hal\sensors\sensor\hwmsen_custom.h file(after 48MP) Don't define MSENSOR_LIB, and set name and prop to null。

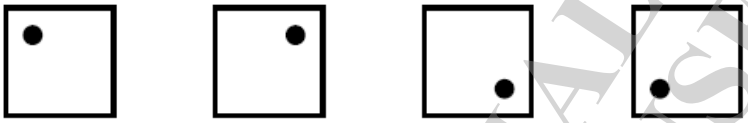
Notes: Yamaha529 M-sensor need translate information larger than 8 bytes once, mt6516 only I2C0 support, but I2C0 is used by other device on MT6516, so Yamaha529 M-sensor can only be used on MT6573.

- Layout setting

If device install in board in different direction, the layout is different, follow will describe each setting (define in Mtk\src\custom\$(project)\kernel\magnetometer\yamaha529\cust_mag.c file[for 48MP];
define in mediatek\custom\$(project)\kernel\magnetometer\yamaha529\cust_mag.c file[after 48MP];).
Layout: 0~3 (Top-view)
Layout: 4~7 (bottom-view)

Top View

Position of the sensors



Setting Value	0	1	2	3

Bottom View

Position of the sensors

Setting Value	4	5	6	7

2.6.5 Yamaha530 Msensor

- In mediatek/config/$(project)/Projectconfig.mk
  Make sure MTK_AUTO_DETECT_MAGNETOMEER = no
  CUSTOM_KERNEL_MAGNETOMETER = Yamaha530

- In mediatek/config/$(project)/init.rc file

Because Yamaha530 Msensor use input device, in different project the input device number is different. So should get the input device number after system is boot-up use adb shell.
cat /sys/class/input/inputX/name (X form 0 – maximum number of input device), get the "orientation", "geomagnetic" and "geomagnetic_raw" device name a, b and c. Then add follow code in #sensor part:

```
chown system /sys/class/input/inputa/enable
chown system /sys/class/input/inputa/delay
chown system /sys/class/input/inputa/wake
chown system /sys/class/input/inputb/enable
chown system /sys/class/input/inputb/wake
chown compass /sys/class/input/inputc/offsets
```

init.rc config in #sensor part Example:
1. use follow command:


```
# cd sys/class/input
cd sys/class/input
# ls
ls
mice
input0
event0
input1
event1
input2
mouse0
event2
input3
mouse1
event3
input4
event4
input5
event5
input6
event6
input7
event7
#
```

2. use follow command (not maybe we need try many times before we got the right input name which we needed)

```
# cat input7/name
cat input7/name
orientation
```

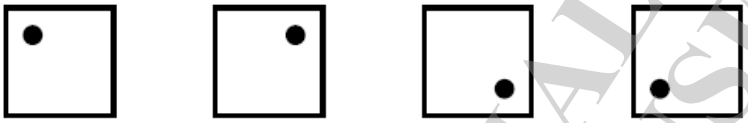
We can know that input7 is orientation,so we can config init.rc like this
Use input7 to replace inputa
chown system /sys/class/input/input7/enable
chown system /sys/class/input/input7/delay
chown system /sys/class/input/input7/wake
the others can be config like that.

- Layout setting

If device install in board in different direction, the layout is different, follow will describe each setting (define in Mtk\src\custom\$(project)\kernel\magnetometer\yamaha530\cust_mag.c file[for 48MP];
define in mediatek\custom\$(project)\kernel\magnetometer\yamaha530\cust_mag.c file[after 48MP];).
Layout: 0~3 (Top-view)
Layout: 4~7 (bottom-view)

Top View

Position of the sensors



Setting Value	0	1	2	3

Bottom View

Position of the sensors

Setting Value	4	5	6	7

2.7 How to use Msensor auto deteted

If we does not sure which MSensor will be use before SMT we can open Msensor auto detected feature （ this feature is default open，now only yamaha529 and yamaha530 cannot be auto detected）

If we want open open Msensor auto detected feature, make sure the fallow code is write in mediatek/config/$(project)/Projectconfig.mk(after1120MP)

```
# Android sensor device
MTK_SENSOR_SUPPORT = yes
MTK_AUTO_DETECT_ACCELEROMETER = yes
MTK_AUTO_DETECT_MAGNETOMETER = yes
CUSTOM_KERNEL_MAGNETOMETER = ami304_auto akm8975_auto
CUSTOM_KERNEL_ACCELEROMETER = adx1345_auto mma8453q_auto lis33de_auto
CUSTOM_KERNEL_ALSPS = cm3623
CUSTOM_KERNEL_BAROMETER = ms5607
CUSTOM_KERNEL_GYROSCOPE = mpu3000
CUSTOM_HAL_SENSORS = sensor
CUSTOM_HAL_MSENSORLIB = mmc328x akm8975 ami304 yamaha530
```

make sure this value is yes

add all sensor driver you wanted to build

2.8 How to use Gsensor auto detect

If we does not sure which GSensor will be use before SMT, than we can open Gsensor auto detected feature （ this feature is default close）

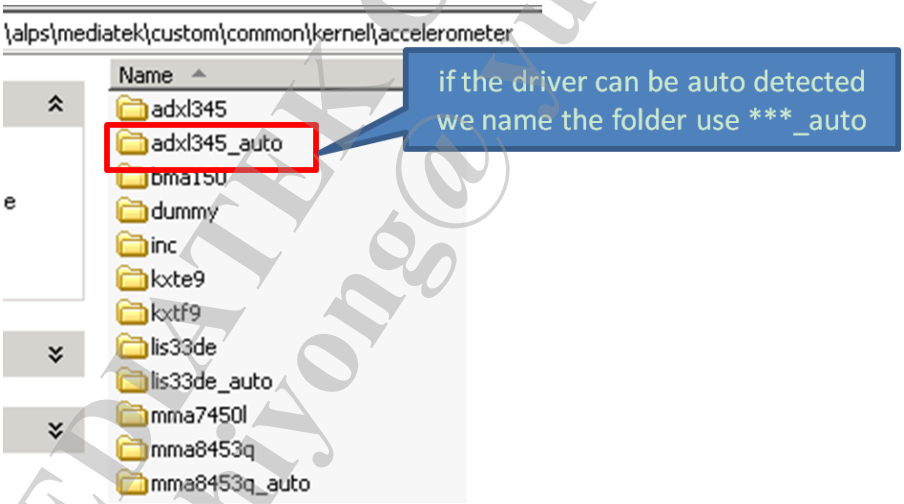
If we want open open Gsensor auto detected feature, make sure the follow code is write in mediatek/config/$(project)/Projectconfig.mk(after1126MP)

```
# Android sensor device
MTK_SENSOR_SUPPORT = yes
MTK_AUTO_DETECT_ACCELEROMETER = yes
MTK_AUTO_DETECT_MAGNETOMETER = no
CUSTOM_KERNEL_MAGNETOMETER = ami304
CUSTOM_KERNEL_ACCELEROMETER = adxl345_auto mma8453q_auto lis33de_auto
CUSTOM_KERNEL_ALSPS = cm3623
CUSTOM_KERNEL_BAROMETER = ms5607
CUSTOM_KERNEL_GYROSCOPE = mpu3000
CUSTOM_HAL_SENSORS = sensor
CUSTOM_HAL_MSENSORLIB = mmc328x akm8975 ami304 yamaha530
```

make sure this is yes

add all sensor driver you wanted to build
note: this driver must can be auto detected which is ported by developer

2.make sure we have gsensor driver that can be auto detected, and the gsensor driver that can be auto detected will be placed in alps\mediatek\costom\common\kernel\accelerometer\



3 Porting guideline

For all Android sensors, you should provide the basic information for this device. Custom will use this information to complete their custom file. This information should put in mtk\src\custom\common\hal\sensors\readme.txt file.(for 48MP)
; mediatek\custom\source\hardware\sensor\hwmsen\sensor_list_readme.txt file.(after 48MP).

```
struct sensor_t {
    /* name of this sensors */
    const char*     name;
    /* vendor of the hardware part */
    const char*     vendor;
    /* version of the hardware part + driver. The value of this field is
     * left to the implementation and doesn't have to be monotonicaly
     * increasing.
     */
    int             version;
    /* handle that identifies this sensors. This handle is used to activate
     * and deactivate this sensor. The value of the handle must be 8 bits
     * in this version of the API. Have follow items: (defined in
kernel/include/Linux/hwmsensor.h)
ID_ORITNTATION,
    ID_MAGNETIC,
    ID_ACCELEROMETER,
    ID_GYROSCOPE,
    ID_PROXIMITY,
    ID_LIGHT,
    ID_PRESSURE,
    ID_TEMPRESRATURE
     */
    int             handle;
    /* this sensor's type. Have follow items: (defined in
hardware\libhardware\include\sensors.h)
   SENSOR_TYPE_ACCELEROMETER
   SENSOR_TYPE_MAGNETIC_FIELD
   SENSOR_TYPE_ORIENTATION
   SENSOR_TYPE_GYROSCOPE
   SENSOR_TYPE_LIGHT
   SENSOR_TYPE_PRESSURE
   SENSOR_TYPE_TEMPERATURE
   SENSOR_TYPE_PROXIMITY
  */
    int             type;
```

```
        /* maximaum range of this sensor's value in SI units */
        float               maxRange;
        /* smallest difference between two values reported by this sensor */
        float               resolution;
        /* rough estimate of this sensor's power consumption in mA */
        float               power;
        /* reserved fields, must be zero */
        void*               reserved[9];
};
```

For example: ADXL345 3-axis accelerometer

```
{
        .name           = "ADXL345 3-axis Accelerometer",
        .vendor         = "ADI",
        .version        = 1,
        .handle         = ID_ACCELEROMETER,
        .type           = SENSOR_TYPE_ACCELEROMETER,
        .maxRange       = 32.0f,
        .resolution = 4.0f/1024.0f,
        .power          =130.0f/1000.0f,
        .reserved       = {}
},
```

3.1 Gsensor porting guideline

1. Create new folder in mediatek\custom\common\kernel\accelerometer, for example abc123. Then copy adxl345.h and adxl345.c files in adxl345 folder to your new folder abc123.
2. Replace adxl345 to abc123, ADXL345 to ABC123 in these two file, include the file name.
3. Now we complete the 50% of driver for new sensor, although we can't do anything for new device (only write the device name abc123). Now we start the core part for new device driver, it includes five parts: Hardware initial, Read sensor data, Sensor calibration, attribute files and trifling things.
4. Hardware initial: In abc123_i2c_probe function will call abc123_init_client() function, the function mainly is used to initial the device, your should complete this function, make the device into right mode (usually device will into standby mode in start-up phase). Also you should modify the hardware relation setting to your device in adc123.h file.
   In initial phase, you should define two variables: gsensor_gain. The gsensor_gain is the sensitivity of device, if you device is +-2g, 10bit, the gsensor_gain is 2^{10} /2 (+-) /2 (2g) = 256 count/g. If you device sensitivity will change in other part, please need modify it.
5. Read sensor data: there have two read sensor data function: ABC123_ReadSensorData() and ABC123_ReadRawData().
   a) ABC123_ReadSensorData() is used to get sample data, the unit is mg. You can only re-write ABC123_ReadData() function to get the raw data, then other part of this

function will count the data. If get raw data is 345, the sensitivity is 512, then the data is 345 * 9807 / 512 = 6608 mg.

b) In ABC123_ReadData() function, it will read the sample data through the I2C bus. Please pay attention to the data format. The CONFIG_ABC123_LOWPASS is used to enable or disable the low pass method, if your device sample has noise, you can use it.

c) ABC123_ReadRawData() is used to get the raw sample data, it will call ABC123_ReadData() function.

6. Sensor calibration. For Gsensor there have two calibration methods: hardware and software.

a) Hardware calibration, device have registry to store the offset value. If the sample sensitivity is different with offset registry sensitivity, the calibration value is different. If sample sensitivity is 256, offset registry is 128. The count value by different sensitivity use same offset value is different, so we need calculate the offset value. If sample sensitivity is larger than offset registry sensitivity, you can use the original code.

If sample sensitivity is less than offset registry sensitivity, you should modify calibration function. Search all the abc123_offset_resolution.sensitivity, for example in ABC123_ReadCalibration() function,

mul = obj->reso->sensitivity/abc123_offset_resolution.sensitivity; modify it to

mul = abc123_offset_resolution.sensitivity / obj->reso->sensitivity;

also modify follow *mul → /mul .

in ABC123_WriteCalibration() function, modify all the obj->cali_sw[] values to 0.

b) Software calibration, if device don't have offset registry, you can use software calibration.

7. Attribute files, the attribute file are mainly used for debug the device, you can add any attribute file for you device. In default attribute files, the self, sleftest files is for ADXL345, if your device don't have this feature, please delete these files. Other files can use for your driver.

8. trifling thing

a) Power manager control. modify ABC123_SetPowerMode() function, make it can support your device power control, when "enable" is false, device will into standby mode, when "enable" is true, device will into normal mode.

b) Sample rate. In gsensor_operate() function "SENSOR_DELAY", it will set device into different sample rate. If you device support different sample rate, please complete this part. The "value" is the sample gap, android define four values: 0, 20, 60 and 200 (ms). Our HAL define three types: 20, 60 and 200(ms) (put 0 to 20). So you can set your device sample rate.

c) Device sensitivity. In some part, driver will use "obj->reso->sensitivity", it is the sensitivity of device, if it doesn't change, and you can use the constant to define. Or you can change it will you sensitivity is change.

9. After modify these part, you driver can work. In mediate\config\$(project)\ProjectConfig.mk make sure "MTK_SENSOR_SUPPORT = yes".

Then modify "CUSTOM_KERNRL_ACCELEROMETER = ABC123". It will build your driver in the platform.

10. In mediatek\custom\$(project)\kernel\accelerometer folder, make sure there have abc123 folder (you can copy adxl345 folder to you). In this folder, the abc_cust_acc.c will configure the device.

    Static struct acc_hw abc123_cust_acc_hw = {
     .i2c_num =2, // Use I2C -2
     .direction = 4, // it is the sensor layout direction, you can refer hwmsen_helper.c
     .power_id = MT6516_POWER_NONE, // Sensor used LDO, now it is null.
     .power_vol = VOL_DEFAULT,   // LOD vol, now is null
     .firlen = 10,   // filter used in low pass, should less 15
    }
    Now you can build your driver, and debug it.

    Notes: In Android, if you want to get good performance, the G-sensor device sample rate must larger than 50 HZ, and resolution must larger than 256LSB (+/- 2G, 10 bit).

3.2 ALS_PS porting guideline

Because most of the ALS (Ambient Light Sensor) and PS (Proximity Sensor) in one device, we put ALS and PS driver in one file.

1. Create new folder in mtk\src\custom\common\kernel\alsps(for 48MP);
   mediatek\custom\common\kernel\alsps(after 48MP), for example abc123.
2. If your sensor has multiple slave address then copy cm3623.h and cm3623.c files in alsps folder. And if your sensor only has one slave address then copy al3006.h and al3006.c files in alsps folder.
3. Replace cm3623 or al3006 to abc123, CM3623 or AL3006 to ABC123 in these two file, include the file name.
4. Now we can start the core part for new device driver, it includes five parts: Hardware initial, Read sensor data, choose interrupt mode or polling mode, attribute files and trifling things.
5. Hardware initial: In abc123_i2c_probe function will call abc123_init_client( ) function, the function is mainly used to initial the device, you should complete this function, make the sensor into right mode (usually device will into standby mode in start-up phase). Also you should modify the hardware relation setting to your device in adc123.h file (registry address, device I2C address and other information).
6. Read sensor data: there have two types of function related to ALS_PS sensor data: the first type is read raw data, such as abc123_read_ps(), abc123_read_als() ; and the second type is convert raw data to another type of data which the HAL layer is needed, such as abc_get_ps_value() and abc_get_als_value(), so we should not change the logic of this two functions unless the HAL layer is changed.
   a. We should pay attention to abc_get_ps_value() and abc_get_als_value() , this two function is also used to filter the inaccuracy sensor value after the sensor is enable. At abc123_i2c_probe function we can set the delay time through ps_deb_on ps_debounce, als_deb_on als_debounce to get first sensor data after sensor was enabled.
7. choose interrupt mode or polling mode:

If your diver is used interrupt mode you should check your code with fallow steps:

a. In abc123_i2c_probe() function make sure obj_ps.polling=0; obj_als.polling =0;
b. In abc123_init_client() function you should enable sensor interrupt function
c. In abc123_enable_als() and abc123_enable_ps() function make sure use schedule_delayed_work() to get sensor data. And the delayed time depend on your sensor.
d. Finish abc_eint_work() function.

If your diver is used polling mode you should check your code with fallow steps:

a. In abc123_i2c_probe() function make sure obj_ps.polling=1; obj_als.polling =1;
b. In abc123_enable_als() and abc123_enable_ps() function you should do nothing

8. Attribute files, the attribute file are mainly used for debug the device, you can add any attribute file for you device.
9. trifling thing
   a. Power manager control. modify abc123_enable_ps() and abc123_enalbe_als() function, make it can support your device power control, when “enable” is false, device will into standby mode, when “enable” is true, device will into normal mode.
   b. Sensor reg related function. At this part you should finish some function to set sensor register depend on datasheet.
10. In mtk\src\custom\$project\kernel\alsps folder(for 48MP); mediatek\custom\$project\kernel\alsps folder(after 48MP), make sure there have abc123 folder (you can copy cm3623 folder to you and rename it). In this folder, the cust_alsps.c will configure the device.

```
Static struct alsps_hw cust_alsps_hw = {
 .i2c_num =2,  // Use I2C 2
 .power_id = MT6516_POWER_NONE, // Sensor used LDO, now it is null.
 .power_vol = VOL_DEFAULT,          // LOD vol, now is null
 .i2c_addr = {0xc0, 0x48, 0x78, 0x00}, // for multiple slave address sensor
 .ps_threshold = 3;
 .als_window_loss =0;              //compensating for the loss when light is passing through the glasses which covered on the sensor
}
```

Notice:

a) Because cm3623 have multiple addresses for different mode, so it defines multiple slave address. Refer your device and use this information in your driver.
b) .als_level depend on specific sensor, it means how many light level the hardware support.
c) .als_value has no relationship with hardware, these values will report to sensor HAL and applications will use these values
d) als_level will mapping to als_value, you can see the details in abc123_get_als_value(). In different projects the mapping may be different.

Now you can build your driver, and debug it.

4 Appendix

4.1 Msensor calibration process*[1]

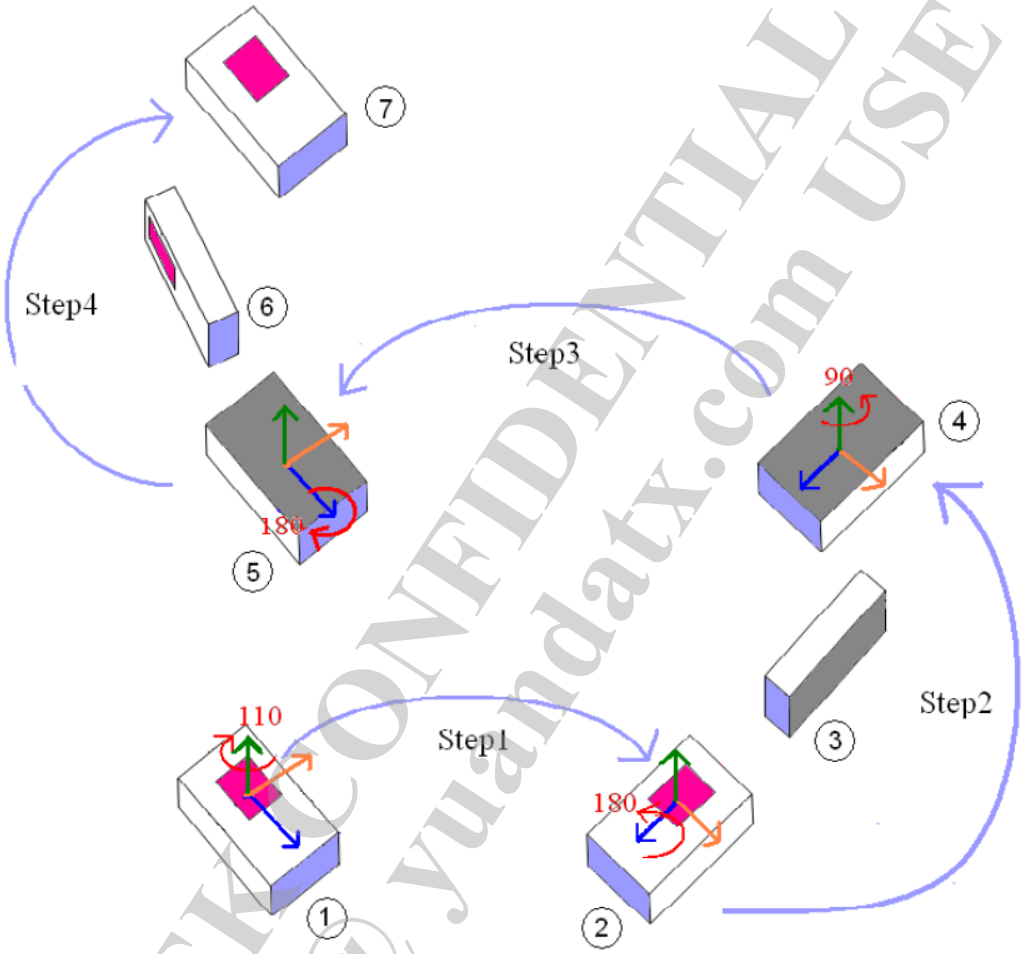
When you launch compass or other msensor application, you should do the calibration first. There are four steps to complete the calibration process.

Step1：Put the sensor in the horizontal surface and rotate at least 110 degree around Green axis (Yaw).

Step2：Rotate the sensor 180 degree around Blue axis (Roll)

Step3：Rotate the sensor at least 90 degree around Green axis (Yaw)

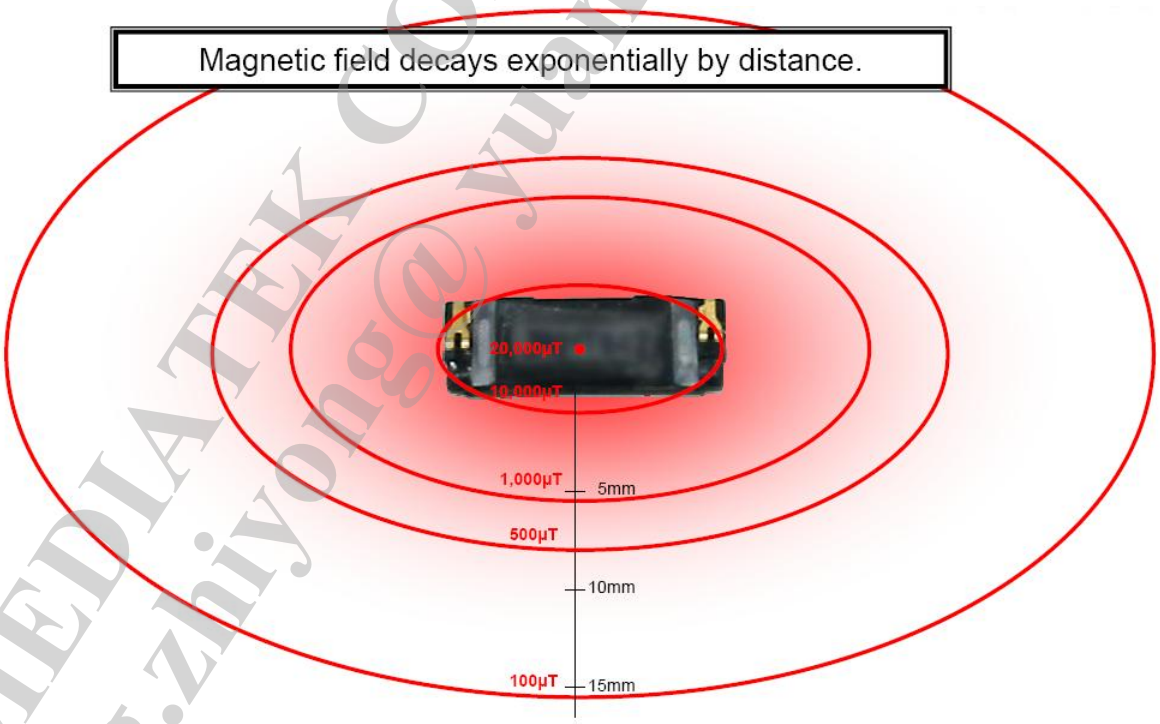
Step4：Rotate the sensor with 180 degree around Blue axis (Roll), so that the sensor has the original attitude.



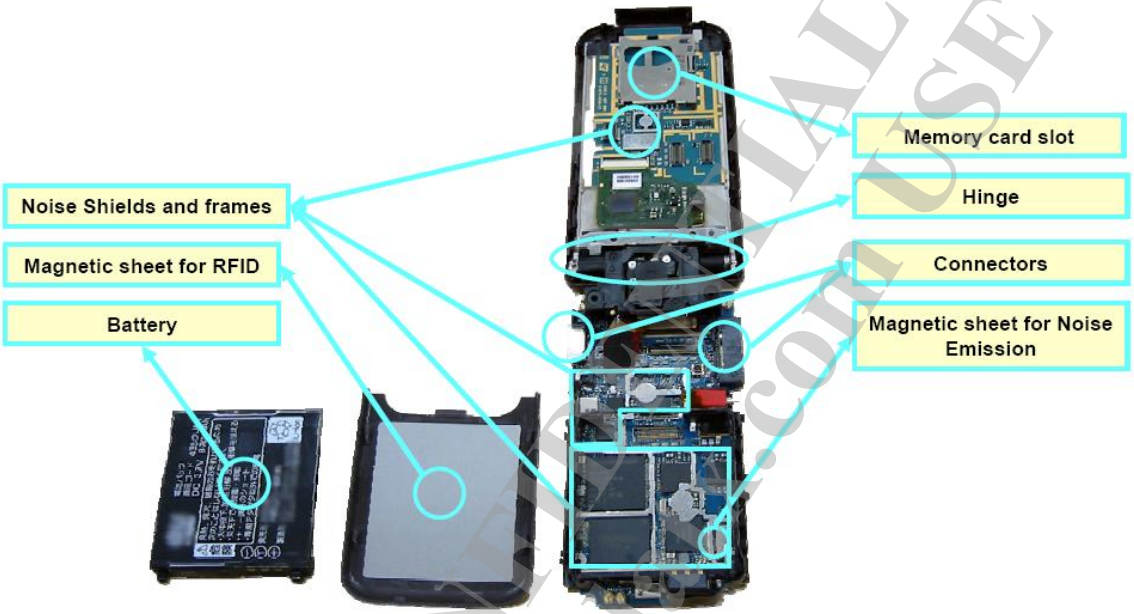
4.2 Msensor layout guideline*[2]

Typical components in a cell phone that cause Hard Iron Effect：

Example of magnetic field spreading around a speaker



Typical components in a cell phone that cause Soft Iron Effect.



Design guide

Safety distance guideline between electronic compass device and **magnetic components**

Magnetic parts	Standard recommended distance
Magnetic Open/Close SW	~ 30mm ~
Speaker	10 ~ 20mm
Vibrator motor	~ 10mm ~
Camera module	~ 10mm ~
Memory Card slot	~ 5mm ~
Magnetic sheet	~ 10mm ~

Above values are for reference.

Safety distance guideline between Electronic compass device and **Power line on PWB**

Current Fluctuation [mA]	Standard recommended distance from line [mm]
2	0
10	3
50	7
100	18
200	25 or more

*1: refer to AMI msensor document.
*2: refer to AKM msensor document.