



How to Convert 3-Axis Directions and Swap X-Y Axis of Accelerometer Data within Android™ Driver

by: Gang Chen
Field Applications Engineer

1 Abstract

Android is one of the most popular Operating Systems (OS) for smart phones and other portable devices. It supports various sensors with standard APIs including accelerometers. The accelerometer standard API defines the coordinate system for accelerometer raw data. Users have to then convert the raw data read from sensors to the standard unit and make them comply with the defined coordinate directions. This application note introduces how the coordinate system on Android is defined, and how to convert the 3-axis directions on the driver code within the Android system. The sample code discussed is based on Freescale's driver code for Android 2.2 and 2.3, using the MMA8452Q as an example.

1.1 Keywords

Accelerometer, Sensor Driver, Android

Contents

1	Abstract	1
1.1	Keywords	1
1.2	Convert X, Y and Z-Axis Directions on Android HAL File	6
1.3	Swap X and Y-axis on Android 2.2 HAL File	6
1.4	Swap X and Y-axis on Android 2.3 HAL File	7
1.5	Swap X and Y-axis on Kernel Driver File	8
1.6	Convert X, Y and Z Axis Directions on Kernel Driver File	8
1.7	Conclusion	9
1.8	References	9

Abstract

A smart phone or portable device should have Wi-Fi and Internet capabilities, the ability to run applications, and must have built-in sensors. A high-end smart phone could also integrate a proximity sensor, an ambient light sensor, a 3-axis accelerometer, and a magnetometer. Android 2.3 adds API support for several new sensor types, including gyroscopes, sensors for rotational vector, linear acceleration, gravity, and barometer pressure. Applications can use these new sensors in combination with any other sensors available on the device, in order to implement advanced motion based functions with high precision and accuracy.

The 3-axis accelerometer or low-g sensor is one of the sensors supported by Android API, which has a specified coordinate system to supply a standard interface to applications. The coordinate space is defined relative to the screen of the phone in its default orientation as shown in [Figure 1](#).



Figure 1. Android Coordinate System for 3-axis Accelerometer

The origin is in the lower-left corner with respect to the screen, with the X-axis horizontal and pointing right, the Y-axis vertical and pointing up and the Z-axis pointing outside the front face of the screen. In this system, coordinates behind the screen have negative Z values.

The Android accelerometer data specification is: `Sensor.TYPE_ACCELEROMETER`

All values are in SI units (m/s^2) and measure the acceleration applied to the phone minus the force of gravity.







- `values[0]`: Acceleration minus G_x on the x-axis
- `values[1]`: Acceleration minus G_y on the y-axis
- `values[2]`: Acceleration minus G_z on the z-axis

For example, when the device lies flat on a table and is pushed on its left side toward the right, the x acceleration value is positive. When the device lies flat on a table, the acceleration value is $+9.81$, which correspond to the acceleration of the device ($0 m/s^2$) minus the force of gravity ($-9.81 m/s^2$).

When the device lies flat on a table and is pushed toward the sky with an acceleration of $A \text{ m/s}^2$, the acceleration value is equal to $A+9.81$ which correspond to the acceleration of the device ($+A \text{ m/s}^2$) minus the force of gravity (-9.81 m/s^2).

Table 1 lists the acceleration values read from the sensor corresponding to each position of a device. Users can check whether the accelerometer directions comply with the system coordinate with this table.

Table 1. Acceleration Values on each Axis for Different Positions

Position	X	Y	Z
UP: 	0	9.81m/s^2	0
LEFT: 	9.81m/s^2	0	0
DOWN: 	0	-9.81m/s^2	0
RIGHT: 	-9.81m/s^2	0	0
FRONT UP: 	0	0	9.81m/s^2
BACK UP: 	0	0	-9.81m/s^2

The 3-axis acceleration values are read from the accelerometer sensor, assuming that the sensor’s 3-axis direction complies with the system coordinate. However in an actual product, different sensor chips could be used, or the mounting directions could be different, so that the data direction differs. Figure 2 shows the orientation definition of Freescale’s MMA8452Q 3-axis accelerometer.

Abstract

From [Figure 2](#), we can see that the chip must be mounted with pin 1 at right-down position (PD), and mounted on the front of the PCB, in order to align with the default position of the Android Coordinate System. By doing this, the user ensures that the data direction complies with the system's coordinate definition. In any other cases, the data will not comply with the system definition and the direction of the data needs to be changed. In some cases, the X and Y-axis must be swapped or both direction changing and X-Y swapping are needed.

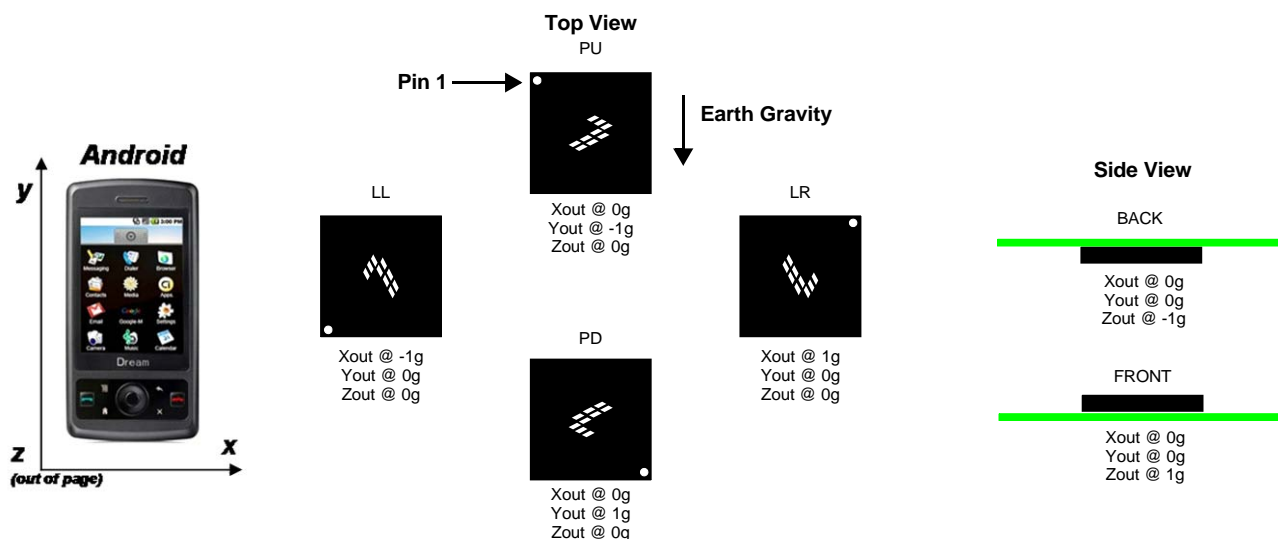


Figure 2. Orientation Definition of MMA8452Q

The way to judge whether the direction conversions or X-Y swapping are needed is described as below:

1. Put the device on its UP position as shown on [Table 1](#).
2. Read the 3-axis data from the sensor. If the data on Y-axis is about $\pm 1 \text{ g}$ ($\pm 9.81 \text{ m/s}^2$), and data on other two axes is about 0, then X-Y swapping is not needed. Otherwise X and Y-axis needs to be swapped, go to [Step 3](#).
 - a) On this position, if Y-axis data is read as $+1 \text{ g}$ ($+9.81 \text{ m/s}^2$), the Y-axis direction doesn't need to convert; whereas, if the data is negative, the Y-axis direction needs to be converted.
 - b) Put the device on its LEFT position as shown on [Table 1](#). The X-axis data should be read as $\pm 1 \text{ g}$ ($\pm 9.81 \text{ m/s}^2$), and the data on the other two axes should be about 0. If the X-axis data is positive, its direction doesn't need to be converted; otherwise the X-axis direction needs to be converted. Go to [Step 4](#) to judge the Z-axis direction.

3. Put the device on its UP position and read the 3-axis data from the sensor. If the data on X-axis is about $\pm 1 \text{ g}$ ($\pm 9.81\text{m/s}^2$), and data on other two axes is about 0, then X-Y swapping is needed.
 - a) On this position, if X-axis data is read as $+1 \text{ g}$ ($+9.81\text{m/s}^2$), the X-axis direction doesn't need to be converted; otherwise it needs to be converted.
 - b) Put the device on its LEFT position as shown on Table 1. The Y-axis data should be read as $\pm 1 \text{ g}$ ($\pm 9.81\text{m/s}^2$), and the data on other two axes should be about 0. If the Y-axis data is positive, its direction doesn't need to be converted; otherwise it needs to be converted.
4. Put the device on its FRONT-UP position and read the 3-axis data from the sensor. If the data on Z-axis is about $+1 \text{ g}$ ($+9.81\text{m/s}^2$), and data on other two axes is about 0, then Z-axis direction doesn't need to be converted; whereas, if the Z-axis data is about -1 g (-9.81m/s^2), then Z-axis direction needs to be converted.

On an Android system, the sensor data is read by the Linux driver on the Kernel space, and sent to the API by the HAL driver. The layered structure is shown as Figure 3. Therefore, the sensor data could be converted on either the Linux driver level or HAL level.

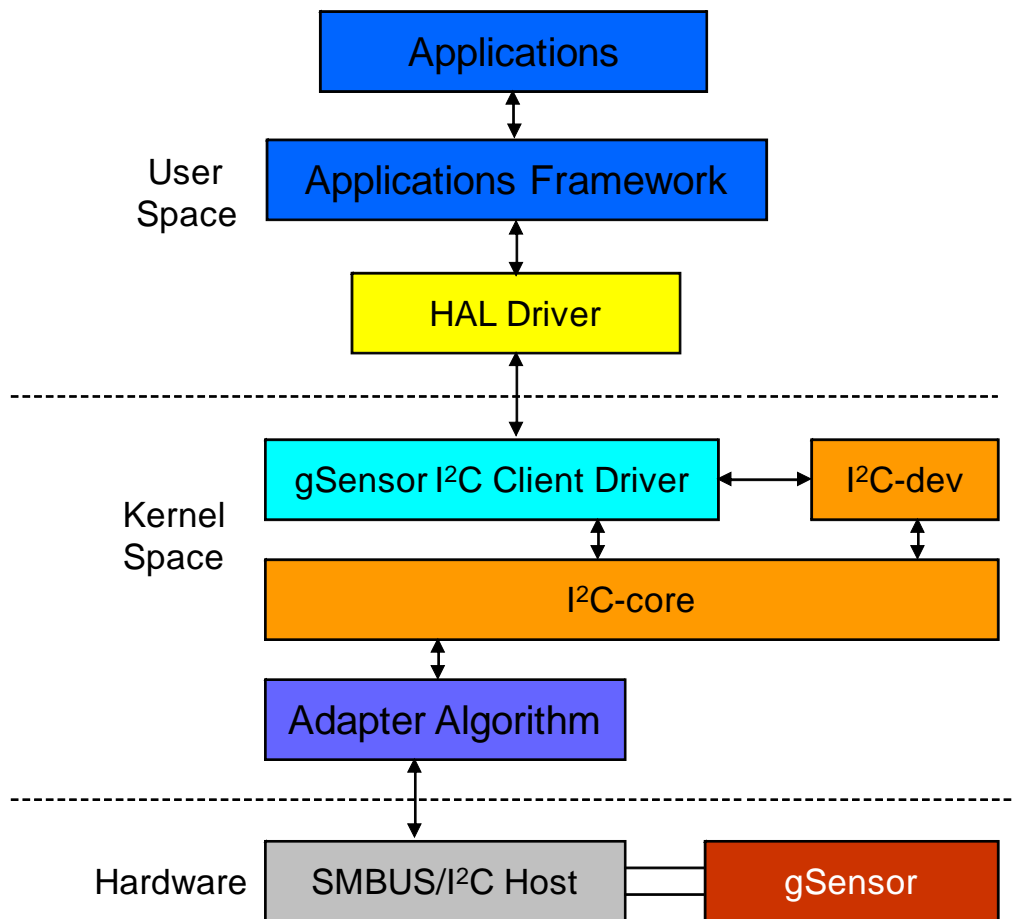


Figure 3. Android Driver Architecture

1.2 Convert X, Y and Z-Axis Directions on Android HAL File

On the HAL file, there would be a group of macro definitions that are used to convert the acceleration data read from the sensor to the standard unit (m/s^2). Like the below code:

```
// conversion of acceleration data to SI units (m/s^2)
#define CONVERT_A          (GRAVITY_EARTH / LSG)
#define CONVERT_A_X       (-CONVERT_A)
#define CONVERT_A_Y       (CONVERT_A)
#define CONVERT_A_Z       (CONVERT_A)
```

On this macro definition, the constant *GRAVITY_EARTH* is a standard gravity value, which is $9.81m/s^2$, and *LSG* is the least significant count for one gravity, for example, 1024 counts for MMA8452 on normal mode. So *CONVERT_A* is used to convert the data read from accelerometer sensors, from digital counts to standard gravity unit.

We can easily change directions of X, Y and Z-axis by modifying *CONVERT_A_X*, *CONVERT_A_Y* and *CONVERT_A_Z* respectively. If the direction of an axis is opposite to the system definition, use (*-CONVERT_A*) to convert it. If their directions are accordant, use (*CONVERT_A*) to keep the direction unchanged.

The macro definition is located on the HAL file *sensor.c* for FSL Android 9 (Android 2.2) driver. For FSL Android 10 (Android 2.3), you can find it on the HAL file *Sensor.h* in folder 'ibensors'.

1.3 Swap X and Y-axis on Android 2.2 HAL File

In some cases the X and Y-axis must be swapped to make the sensor data direction comply with the system coordinate.

For FSL Android 9 (Android 2.2) driver, Swapping X and Y axis can be easily implemented. First, find the code below on the function *sensor_poll()* on the HAL file *sensor.c*:

```
switch (event.code) {
    case ABS_X:
        sSensors.acceleration.x = event.value * CONVERT_A_X;
        break;
    case ABS_Y:
        sSensors.acceleration.y = event.value * CONVERT_A_Y;
        break;
    case ABS_Z:
        sSensors.acceleration.z = event.value * CONVERT_A_Z;
        break;
}
```

Second, modify the code as shown below:

```
switch (event.code) {
    case ABS_X:
        sSensors.acceleration.y = event.value * CONVERT_A_Y;
        break;
    case ABS_Y:
        sSensors.acceleration.x = event.value * CONVERT_A_X;
        break;
    case ABS_Z:
        sSensors.acceleration.z = event.value * CONVERT_A_Z;
        break;
}
```

1.4 Swap X and Y-axis on Android 2.3 HAL File

It's a little more complicated to swap X and Y-axis on Android 2.3 HAL files, because it has a more complicated HAL file structure. All the HAL files are in folder '*libsensors*'. Two functions in file *AccelSensor.cpp* need to be modified.

First, modify the code for the function *AccelSensor()* as shown below:

```

if (accel_is_sensor_enabled(SENSOR_TYPE_ACCELEROMETER)) {
    mEnabled |= 1<<Accelerometer;
    if (!ioctl(data_fd, EVIOCGABS(EVENT_TYPE_ACCEL_X), &absinfo)) {
        mPendingEvents[Accelerometer].acceleration.y = absinfo.value * CONVERT_A_Y;
    }
    if (!ioctl(data_fd, EVIOCGABS(EVENT_TYPE_ACCEL_Y), &absinfo)) {
        mPendingEvents[Accelerometer].acceleration.x = absinfo.value * CONVERT_A_X;
    }
    if (!ioctl(data_fd, EVIOCGABS(EVENT_TYPE_ACCEL_Z), &absinfo)) {
        mPendingEvents[Accelerometer].acceleration.z = absinfo.value * CONVERT_A_Z;
    }
}

```

Second, modify the code on the function *processEvent()* as shown below:

```

void AccelSensor::processEvent(int code, int value)
{
    switch (code) {
        case EVENT_TYPE_ACCEL_X:
            mPendingMask |= 1<<Accelerometer;
            mPendingEvents[Accelerometer].acceleration.y = value * CONVERT_A_Y;
            break;
        case EVENT_TYPE_ACCEL_Y:
            mPendingMask |= 1<<Accelerometer;
            mPendingEvents[Accelerometer].acceleration.x = value * CONVERT_A_X;
            break;
        case EVENT_TYPE_ACCEL_Z:
            mPendingMask |= 1<<Accelerometer;
            mPendingEvents[Accelerometer].acceleration.z = value * CONVERT_A_Z;
            break;
    }
}

```

Upon completion, the X and Y-axis are swapped.

1.5 Swap X and Y-axis on Kernel Driver File

X and Y-axis data could be swapped at the very beginning when the sensor data is read in the low level Linux driver. With this method, the HAL file could keep its consistency no matter how the sensor chip mounted is on the PCB, or what kind of sensor is used.

For both Android 2.2 and 2.3, the most convenient way to do this is to modify the code on function *report_abs()*. On this function, sensor data is read by calling function *mma8452_read_data()* as below (in case the used sensor is MMA8452Q):

```
if (mma8452_read_data(&x,&y,&z) != 0) {
    //DBG("mma8452 data read failed\n");
    return;}

```

X and Y-axis can be swapped easily as below:

```
if (mma8452_read_data(&y,&x,&z) != 0) {
    //DBG("mma8452 data read failed\n");
    return;}

```

For Android 2.2, the Linux driver file of MMA8452 is *mma8452.c*; for Android 2.3, it's *mxc_mma8452.c* in folder 'hwmon'.

1.6 Convert X, Y and Z Axis Directions on Kernel Driver File

The sensor data directions can also be changed on the Linux driver file. Below sentences with comments could be added into function *report_abs()* to change the data directions:

```
if (mma8452_read_data(&y,&x,&z) != 0) {
    //DBG("mma8452 data read failed\n");
    return;}
}
x *= -1; //Reverse X direction
y *= -1; //Reverse Y direction
z *= -1; //Reverse Z direction
input_report_abs(mma8452_iddev->input, ABS_X, x);
input_report_abs(mma8452_iddev->input, ABS_Y, y);
input_report_abs(mma8452_iddev->input, ABS_Z, z);
input_sync(mma8452_iddev->input);

```


1.7 Conclusion

Android system has defined its coordinate system for the accelerometer, so users must convert the data read from actual sensors to comply with it. Both X, Y, Z-axis directions and X-Y-axis orientation should be checked whether conversions are needed or not. We can modify either HAL file or Kernel driver file to change axis directions or swap X and Y-axis, but do not modify HAL file and Kernel driver both at same time.

1.8 References

1. Android Coordinate System
<http://developer.android.com/reference/android/hardware/SensorEvent.html>
2. Reference Code and User's Guide on '*Android MMA8452.tar.gz*'.
3. Reference Code and User's Guide on '*mma8452_imx51_android10_v100.tar.gz*'.
4. Data sheet '*MMA8452Q.pdf*'.

How to Reach Us:

Home Page:

www.freescale.com

Web Support:

<http://www.freescale.com/support>

USA/Europe or Locations Not Listed:

Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:

Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:

Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:

Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:

Freescale Semiconductor Literature Distribution Center
1-800-441-2447 or +1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals", must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

Freescale and the Freescale logo are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. Xtrinsic is a trademark of Freescale Semiconductor, inc. All other product or service names are the property of their respective owners.

© Freescale Semiconductor, Inc. 2011. All rights reserved.

RoHS-compliant and/or Pb-free versions of Freescale products have the functionality and electrical characteristics of their non-RoHS-compliant and/or non-Pb-free counterparts. For further information, see <http://www.freescale.com> or contact your Freescale sales representative.

For information on Freescale's Environmental Products program, go to <http://www.freescale.com/epp>.