HP TouchPad Sensor Setup for Android

Coordinate System

The Android device framework uses a 3-axis coordinate system to express data values. For the following HP TouchPad sensors, the coordinate system is defined relative to the device's screen when the device is held in its default orientation:

- Acceleration sensor (hardware)
- Gravity sensor (software)
- Gyroscope (hardware)
- Linear acceleration sensor (software)
- Magnetic field sensor (hardware)

Android phone devices use a natural (default) portrait orientation with the sensor coordinate system defined as shown in **Figure 1**. When the device is in the default orientation, the X axis is horizontal and points to the right, the Y axis is vertical and points up, and the Z axis points toward the outside of the screen face. The axes are not swapped when the device's changes orientation; that is, the sensor's coordinate system never changes as the device moves.

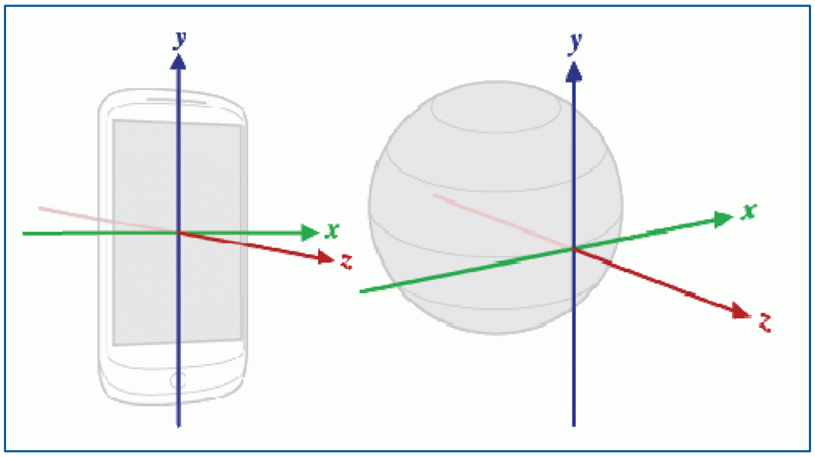

Figure 1 – Sensor Coordinate System Relative to the Device and Earth

HP originally designed the TouchPad to work under WebOS as a natural portrait orientation device. Current CyanogenMod Android builds for the HP TouchPad are for a tablet device with a natural landscape orientation. The relationship between tablet and portrait orientations is as follows:

- Tablet X-Axis = Portrait Y-Axis
- Tablet Y-Axis = Neg. Portrait X-Axis
- Tablet Z-Axis = Portrait Z-Axis

The polarity of rotation around the coordinate system axis is defined as positive when the device is rotated in the clockwise direction around the positive direction of the respective axis as shown in **Figure 2**.

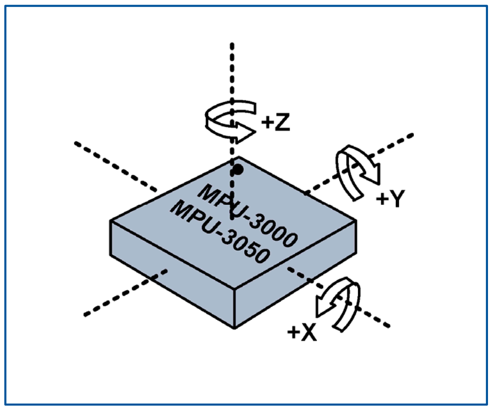

Figure 2 – Sensor Rotation System

Some sensors and methods use a coordinate system that is relative to the world's frame of reference (specifically, magnetic north) as opposed to the device's frame of reference. These sensors and methods return data that represent device motion or device position relative to the earth's magnetic field:

- Orientation sensor (software)
- Rotation sensor (software)

Figure 3 shows the earth's magnetic field coordinate system relative to true North. The angle between True North and H is the declination angle D and the angle between the geomagnetic field B and horizontal plane is the inclination angle I. Bx, By, and Bz are the three orthogonal magnetic field components. **Note** that in the Android geomagnetic coordinate system, the Z-Axis points up (toward the sky), the X-Axis points magnetic North (horizontal projection H of the geomagnetic field B), and the Y-Axis points magnetic East.

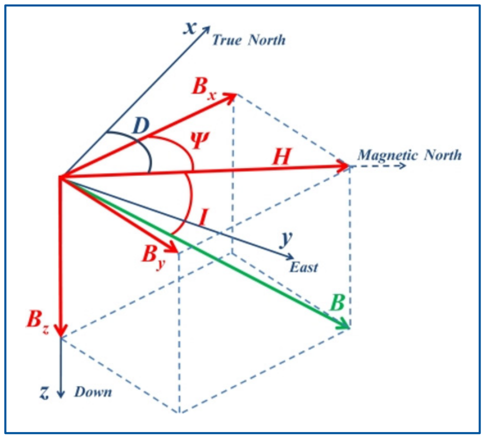

Figure 3 – Geomagnetic field Coordinate System

Hardware Sensor Outputs

Acceleration Sensor

The acceleration sensor measures the force of gravity in meters/sec^2 in the direction of the each axis. Accelerations can be a combination of linear or centripetal acceleration of the tablet and gravity. When each axis is pointed up with respect to the earth and the tablet is motionless, the acceleration sensor output for that axis will nominally be 9.81 m/s^2 (1g) but will vary somewhat with altitude and location on the earth. The HP TouchPad sensor module (ST lsm303dlh) outputs acceleration in g which is converted at the Android device driver level into m/s^2.

Magnetic Field Sensor

The geomagnetic field sensor measures the magnetic field strength in the direction of each axis in microtesla (μT), where 1 μT = 10 mG (milligauss). The value of the earth's geomagnetic field varies between 25 and 65 μT. The same HP TouchPad sensor module (ST lsm303dlh) outputs acceleration in gauss (1G = 100 μT) which is converted at the Android device driver level into μT.

Gyroscope

The gyroscope measures the angular rate of rotation around each axis in radians/sec, regardless of gravity. The sensor output is positive when the device is rotated in the clockwise direction around the positive direction of each axis. The HP TouchPad gyroscope sensor (InvenSense MPU-3050) outputs the angular rate in degrees/sec which is converted at the Android device driver level into radians/sec.

Software Sensor Outputs

Orientation Sensor

The orientation sensor measures the position of a device relative to magnetic north in degrees. This software sensor derives its data by using the magnetic field sensor in combination with the acceleration sensor. The orientation sensor provides data for the following three dimensions:

- Pitch (degrees of rotation around the x axis). This value is positive when the positive z axis rotates toward the positive y axis, and it is negative when the positive z axis rotates toward the negative y axis. The range of values is 180 degrees to -180 degrees.
- Roll (degrees of rotation around the y axis). This value is positive when the positive z axis rotates toward the positive x axis, and it is negative when the positive z axis rotates toward the negative x axis. The range of values is 90 degrees to -90 degrees.
- Azimuth (degrees of rotation around the z axis). This is the angle between magnetic north and the device's y axis. For example, if the device's y axis is aligned with magnetic north this value is 0, and if the device's y axis is pointing south this value is 180. Likewise, when the y axis is pointing east this value is 90 and when it is pointing west this value is 270.

Rotation Sensor

The rotation sensor measures the rotation vector which represents the orientation of the device as a combination of an angle and an axis, in which the device has rotated through an angle θ around an axis (x, y, or z). This software sensor derives its data using the acceleration sensor, gyroscope sensor, and magnetic field sensor. The values returned are unitless as follows:

- X-Axis value = x * sin(θ/2); tangential to the ground and points approximately East.
- Y-Axis value = y * sin(θ/2); tangential to the ground and points magnetic North.
- Z-Axis Value = z * sin(θ/2); perpendicular to the ground and points toward the sky.

Linux/Android Code Setup

General Approach

I modified Tomasz Rostanski's "Update MPU3050 kernel driver"[1] and "Add Gyroscope sensor userspace driver"[2] CyanogenMod code commits to develop CM 10 (jellybean) builds for the HP TouchPad 4G in both natural portrait and natural landscape orientations. The general process included the following steps:

- Establish parameters for the Linux kernel level drivers based on the natural portrait orientation used for the original HP TouchPad hardware development.
- Modifying the Android device level drivers as necessary to result in sensor outputs with the proper magnitude and polarity for either natural portrait or natural landscape orientation.

Linux Kernel

The approach taken for setting up the Linux kernel code (*kernel/hp/tenderloin*) is to have settings which 1) are based on the original design natural portrait orientation device and 2) provide the same values for both the WiFi and 4G HP TouchPad platforms.

The code which has been modified is primarily in *arch/arm/mach-msm/board-tenderloin.c* and code snippets are provided in **Appendix A**. For the WiFi platform, these are generally the default values provided in the InvenSense MPU-3050 drivers. For the 4G platform, the X-Axis and Z-Axis polarity is reversed for each of the hardware sensors. Full-scale levels set at the kernel level are set as follows:

- lsm303dlh accelerometer sensor full-scale set to ±2.0 g.
- lsm303dlh magnetic field full-scale set to ±1.3 gauss.
- mpu-3050 gyroscope sensor full-scale set to ±2000 degrees/sec.

[1] http://review.cyanogenmod.org/29733
[2] http://review.cyanogenmod.org/29838

An additional code change was made in *include/linux/i2c/lsm303dlh.h* such that definitions in *lsm303dlh.h* match the latest ST MicroElectronics drivers (and as used in *board-tenderloin.c*).

Android Device

Setting up the HP TouchPad to operate as a native portrait orientation device or as a native landscape device requires code changes at the Android device level in *system.prop* and *touchscreen_drv/ts_srv.c*, as provided in **Appendix B.**

Code changes at the Android device level (*device/hp/tenderloin/libsensors*) were made to provide the proper sensor responses in either native portrait or native landscape orientation. Code snippets for *MLPSensor.cpp*, *lsm303dlh_acc.c*, and *lsm303dlh_mag.c* are provided in **Appendix C** for both native portrait and landscape orientations. Code snippets required for proper sensor unit conversions in *sensors.c* (based on the full-scale levels set at the kernel level) are also provided in **Appendix C**.

Device Response

When set up as described herein, the HP TouchPad 4G builds respond as follows:

- Acceleration and gyroscope sensor outputs have the correct orientation and values.
- Screen rotation responds correctly to device orientation when Z axis is near horizontal.
- Gyrospace 3D wall paper responds correctly to rotation around X and Y axis.
- Smart Compass works when the HP TouchPad is held with the camera/power switch end pointed up.

Limitation

A limitation when the HP TouchPad is set up for a native landscape orientation is that Android applications which do not recognize and adjust for this native orientation will not work correctly when in portrait mode. They may work properly if forced into landscape orientation.

For example, Android Sensor Box application Orientation Sensor (level bubbles) and Acceleration Sensor (rolling ball) will only work if the application is forced into landscape mode when the HP TouchPad is operating in its normal a native landscape orientation. They work correctly when the HP TouchPad is operating in a native portrait orientation.

Appendix A – Linux Kernel Code Snippets

```
==============================================================================
kernel/hp/tenderloin/include/linux/i2c/lsm303dlh.h
==============================================================================
#define LSM303DLH_ACC_MIN_POLL_PERIOD_MS       1
#define LSM303DLH_MAG_MIN_POLL_PERIOD_MS       5

#define LSM303DLH_ACC_DEFAULT_INT1_GPIO (-EINVAL)
#define LSM303DLH_ACC_DEFAULT_INT2_GPIO (-EINVAL)

==============================================================================
kernel/hp/tenderloin/arch/arm/mach-msm/board-tenderloin.c
==============================================================================
static struct lsm303dlh_acc_platform_data lsm303dlh_acc_pdata = {
       .poll_interval = 100,
       .min_interval = LSM303DLH_ACC_MIN_POLL_PERIOD_MS,
       .g_range = LSM303DLH_ACC_G_2G,
       .axis_map_x = 0,
       .axis_map_y = 1,
       .axis_map_z = 2,
       .negate_x = 0,
       .negate_y = 0,
       .negate_z = 0,
       .gpio_int1 = LSM303DLH_ACC_DEFAULT_INT1_GPIO,
       .gpio_int2 = LSM303DLH_ACC_DEFAULT_INT2_GPIO,
};

static struct lsm303dlh_mag_platform_data lsm303dlh_mag_pdata = {
       .poll_interval = 100,
       .min_interval = LSM303DLH_MAG_MIN_POLL_PERIOD_MS,
       .h_range = LSM303DLH_MAG_H_1_3G,
       .axis_map_x = 0,
       .axis_map_y = 1,
       .axis_map_z = 2,
       .negate_x = 0,
       .negate_y = 0,
       .negate_z = 0,
};
```

Appendix A – Linux Kernel Code Snippets

```
static struct mpu3050_platform_data mpu_pdata = {
       .int_config  = 0x10,
       .orientation = {   1,  0,  0,
                        0,  1,  0,
                        0,  0,  1 },
       .accel = {
              .get_slave_descr = get_accel_slave_descr,
              .adapt_num   = 0,
              .bus         = EXT_SLAVE_BUS_SECONDARY,
              .address     = 0x18,
              .orientation = {  1,  0,  0,
                               0,  1,  0,
                               0,  0,  1 },
        },
       .compass = {
              .get_slave_descr = get_compass_slave_descr,
              .adapt_num   = 0,
              .bus         = EXT_SLAVE_BUS_PRIMARY,
              .address     = 0x1E,
              .orientation = {  1,  0,  0,
                              0,  1,  0,
                              0,  0,  1 },
       },
};

--------------------------------------------------------------------------------
       if (machine_is_tenderloin() && boardtype_is_3g()) {
#ifdef CONFIG_INPUT_LSM303DLH
              lsm303dlh_acc_pdata.negate_x = 1;
              lsm303dlh_acc_pdata.negate_z = 1;
              lsm303dlh_mag_pdata.negate_x = 1;
              lsm303dlh_mag_pdata.negate_z = 1;
#endif
              mpu_pdata.orientation[0] = -mpu_pdata.orientation[0];
              mpu_pdata.orientation[8] = -mpu_pdata.orientation[8];
       }
#endif
```

Appendix B – Device Orientation Code Snippets

Portrait Orientation

```
================================================================================
Device/hp/tenderloin/system.prop
================================================================================

ro.sf.hwrotation=270

================================================================================
Device/hp/tenderloin/touchscreen_drv/ts_srv.c
================================================================================

#define USERSPACE_270_ROTATE 1
```

Landscape Orientation

```
================================================================================
Device/hp/tenderloin/system.prop
================================================================================

#ro.sf.hwrotation=270

================================================================================
Device/hp/tenderloin/touchscreen_drv/ts_srv.c
================================================================================

#define USERSPACE_270_ROTATE 0
```

Appendix C – Device Libsensor Code Snippets

Portrait Orientation

```
==================================================================================
device/hp/tenderloin/libsensors/MLPSensor.cpp
==================================================================================
void MPLSensor::gyroHandler(sensors_event_t* s, uint32_t* pending_mask, int index)
{
    VFUNC_ALOG;
    inv_error_t res;
    float temp[3];
    ALOGV_IF(EXTRA_VERBOSE, "gyroHandler");
    res = inv_get_float_array(INV_GYROS, temp);
    s->gyro.v[0] = temp[0] * M_PI / 180.0;
    s->gyro.v[1] = temp[1] * M_PI / 180.0;
    s->gyro.v[2] = temp[2] * M_PI / 180.0;
    s->gyro.status = mMpuAccuracy;
    if (res == INV_SUCCESS)
        *pending_mask |= (1 << index);
}

==================================================================================
device/hp/tenderloin/libsensors/lsm303dlh_acc.c
==================================================================================
void Lsm303dlhGSensor::processEvent(int code, int value)
{
    switch (code) {
        case EVENT_TYPE_ACCEL_X:
            mPendingEvent.acceleration.x = value * CONVERT_A_X;
            break;
        case EVENT_TYPE_ACCEL_Y:
            mPendingEvent.acceleration.y = value * CONVERT_A_Y;
            break;
        case EVENT_TYPE_ACCEL_Z:
            mPendingEvent.acceleration.z = value * CONVERT_A_Z;
            break;
    }
}
```

Appendix C – Device Libsensor Code Snippets

```
=================================================================================
device/hp/tenderloin/libsensors/lsm303dlh_mag.c
=================================================================================
void Lsm303dlhMagSensor::processEvent(int code, int value)
{
    switch (code) {
        case EVENT_TYPE_MAGV_X:
            mPendingEvent.magnetic.x = value * CONVERT_M_X;
            break;
        case EVENT_TYPE_MAGV_Y:
            mPendingEvent.magnetic.y = value * CONVERT_M_Y;
            break;
        case EVENT_TYPE_MAGV_Z:
            mPendingEvent.magnetic.z = value * CONVERT_M_Z;
            break;
    }
}
```

Appendix C – Device Libsensor Code Snippets

Landscape Orientation

```
==================================================================================
device/hp/tenderloin/libsensors/MLPSensor.cpp
==================================================================================
void MPLSensor::gyroHandler(sensors_event_t* s, uint32_t* pending_mask, int index)
{
    VFUNC_ALOG;
    inv_error_t res;
    float temp[3];
    ALOGV_IF(EXTRA_VERBOSE, "gyroHandler");
    res = inv_get_float_array(INV_GYROS, temp);
    s->gyro.v[0] = temp[1] * -M_PI / 180.0;
    s->gyro.v[1] = temp[0] * M_PI / 180.0;
    s->gyro.v[2] = temp[2] * M_PI / 180.0;
    s->gyro.status = mMpuAccuracy;
    if (res == INV_SUCCESS)
        *pending_mask |= (1 << index);
}

==================================================================================
device/hp/tenderloin/libsensors/lsm303dlh_acc.c
==================================================================================
void Lsm303dlhGSensor::processEvent(int code, int value)
{
    switch (code) {
        case EVENT_TYPE_ACCEL_X:
            mPendingEvent.acceleration.y = value * CONVERT_A_X;
            break;
        case EVENT_TYPE_ACCEL_Y:
            mPendingEvent.acceleration.x = value * -CONVERT_A_Y;
            break;
        case EVENT_TYPE_ACCEL_Z:
            mPendingEvent.acceleration.z = value * CONVERT_A_Z;
            break;
    }
}
```

Appendix C – Device Libsensor Code Snippets

```
==================================================================================
device/hp/tenderloin/libsensors/lsm303dlh_mag.c
==================================================================================
void Lsm303dlhMagSensor::processEvent(int code, int value)
{
    switch (code) {
        case EVENT_TYPE_MAGV_X:
            mPendingEvent.magnetic.y = value * CONVERT_M_X;
            break;
        case EVENT_TYPE_MAGV_Y:
            mPendingEvent.magnetic.x = value * -CONVERT_M_Y;
            break;
        case EVENT_TYPE_MAGV_Z:
            mPendingEvent.magnetic.z = value * CONVERT_M_Z;
            break;
    }
}
```

Appendix C – Device Libsensor Code Snippets

Sensor Values

```
=================================================================================
device/hp/tenderloin/libsensors/sensors.c
=================================================================================
static const struct sensor_t sSensorList[] = {
        { "LSM303DLH 3-axis Accelerometer",
                "ST Microelectronics",
                1, SENSORS_HANDLE_BASE+ID_A,
                SENSOR_TYPE_ACCELEROMETER, 2.0f*GRAVITY_EARTH, (2.0f*GRAVITY_EARTH)/2048.0f, 0.5f, 10000/*10ms*/, { } },
        { "LSM303DLH Magnetometer",
                "ST Microelectronics",
                1, SENSORS_HANDLE_BASE+ID_M,
                SENSOR_TYPE_MAGNETIC_FIELD, 130.0f, 0.05f, 0.5f, 10000/*10ms*/, { } },
        { "ISL29023 Light sensor",
                "Intersil",
                1, SENSORS_HANDLE_BASE+ID_L,
                SENSOR_TYPE_LIGHT, 3000.0f, 1.0f, 0.75f, 0, { } },
        { "MPL Gyroscope",
                "Invensense",
                1, SENSORS_HANDLE_BASE+ID_GY,
                SENSOR_TYPE_GYROSCOPE, 2000.0f*RAD_P_DEG, 32.8f*RAD_P_DEG, 0.5f, 10000/*10ms*/, { } },
        { "MPL Temperature sensor",
                "Invensense",
                1, SENSORS_HANDLE_BASE+ID_T,
                SENSOR_TYPE_AMBIENT_TEMPERATURE, 80.0f, 0.5f, 0.0f, 10000/*10ms*/, {} },
};
```