



DMARD18 Android Driver V1.38 Porting Guide

Files

Name	Version	Description
dmard18_AD_V1.38_PG.pdf	2012_10_03	This document
src/Driver/dmt030818.c	version 1.35	Linux I2C device driver
src/Driver/dmt030818.h	version 1.35	Linux I2C device driver header
src/libsensors/*	version 1.34	Android HAL
Calibration_applications_Operations_Guide.pdf	2012_09_11	APP Calibration document
DMT_Calibration/*	2012_09_11	APP Calibration source
DMT_Calibration_LAUNCHER.apk	2012_09_11	LAUNCHER APP
DMT_Calibration_DEFAULT.apk	2012_09_11	DEFAULT APP

Android Version

DMARD18 Android Driver V1.38 supports Android 2.3 (Gingerbread) system

And Android 4.0 (Ice-Cream-Sandwich) system.

Index

Revision History.....	2
Build Kernel Image.....	2
Sensor HAL Module.....	3
I2C Connection Circuit.....	3
Coordinate System.....	4
IOCTL Interface.....	5

Revision History

Ver.	Date	Updates	Descriptions
1.38	Oct/03 th , 2012	2.3 Gingerbread 4.0 Ice-Cream-Sandwich	1. remove IOCTL(_IOC_NR(cmd) > SENSOR_MAXNR) 2. Unified 1g = 1024 LSB 3. Default close AUTO_CALIBRATION FUNCTION 4. SET BANDWIDTH_REG to No filter 5. Fix sensorlayout's define
1.37	Jul/13 th , 2012	2.3 Gingerbread 4.0 Ice-Cream-Sandwich	First release

Build Kernel Image

1. Copy "src/Driver/dmt030818.c" to "\$KERNEL/drivers/misc/"
Copy "src/Driver/dmt030818.h" to "\$KERNEL/include/linux/"
For example in our system it's in "\$KERNEL/drivers/misc/"

```
$ cp src/Driver/dmt030818.c $KERNEL/drivers/misc/  
$ cp src/Driver/dmt030818.h $KERNEL/include/linux/
```

2. Specify the g-sensor layout by defining one of the following macro in "dmt030818.h". The example code uses layout pattern 1. See Figure 1 for other g-sensor layout pattern orientations.

g-senor layout configuration, choose one of the following configuration

```
#define CONFIG_GSEN_LAYOUT_PAT_1      1  
#define CONFIG_GSEN_LAYOUT_PAT_2      0  
#define CONFIG_GSEN_LAYOUT_PAT_3      0  
#define CONFIG_GSEN_LAYOUT_PAT_4      0  
#define CONFIG_GSEN_LAYOUT_PAT_5      0  
#define CONFIG_GSEN_LAYOUT_PAT_6      0  
#define CONFIG_GSEN_LAYOUT_PAT_7      0  
#define CONFIG_GSEN_LAYOUT_PAT_8      0
```

3. Modify "\$KERNEL/driver/misc/Makefile" with addition of the following line.
Refer to patch file in the "src/Makefile.patch"

```
obj-$(CONFIG_SENSORS_DMARD18) += dmt030818.o
```

4. Modify "\$KERNEL/driver/misc/Kconfig" with addition of the following line.
Refer to patch file in the "src/Kconfig.patch"

```
config SENSORS_DMARD18
    tristate "DMARD18 GSENSOR support"
    default y
    depends on I2C=y
    ---help---
    If you say yes here you get support for accelemereter
    sensor DMARD18.
```

5. Add I2C information to the board information. For example in our system it's in "\$KERNEL/arch/arm/mach-omap2/board-omap4panda.c". Locate your own board information file instead. DMARD18 has the option of selecting one of two slave addresses by setting SDO (pin#10) to high or low. Sample codes illustrate the case of setting SDO (pin#10) to low. In the case of SDO (pin#10) set to high, g-sensor's 7-bit slave address would be 0x1c. Refer to patch file in the "src/board-omap4panda.c.patch"

```
static struct i2c_board_info __initdata panda_i2c_boardinfo[] = {
    #if (defined(CONFIG_SENSORS_DMARD18) ||
    defined(CONFIG_SENSORS_DMARD18_MODULE))
        {I2C_BOARD_INFO("dmt", 0x1c), },
    #endif
};
```

6. Rebuild the kernel image

Sensor HAL Module

1. Modify "src/libensors/Android.mk" with addition of the following line.

```
LOCAL_CFLAGS := -DLOG_TAG=\"Sensors\"
               -DSENSORHAL_ACC_D18 \
               -Wall \
```

2. Copy "src/libensors" directory to "\$ANDROID/hardware/libhardware/module/libensors"

```
$ cp -r src/libensors hardware/libhardware/module/
```

3. Add the following to the file "\$ANDROID/system/core/rootdir/init.rc"

```
chown system system /sys/class/acceleometer/dmt/enable_acc
chown system system /sys/class/acceleometer/dmt/delay_acc
mkdir /data/misc/dmt 0777 system system
chmod 0755 /dev/dmt
```

4. Rebuild the Android image

I2C Connection Circuit

Please refer to “AN007: Application Circuit Examples and Basic Operations of DMARD18” for connection circuit example.

Coordinate System

The coordinate system used by the Android SensorEvent API is defined relative to the screen of the phone in its default orientation, as shown in the Figure 1. Specify the g-sensor layout to one of the layout patterns illustrated below.

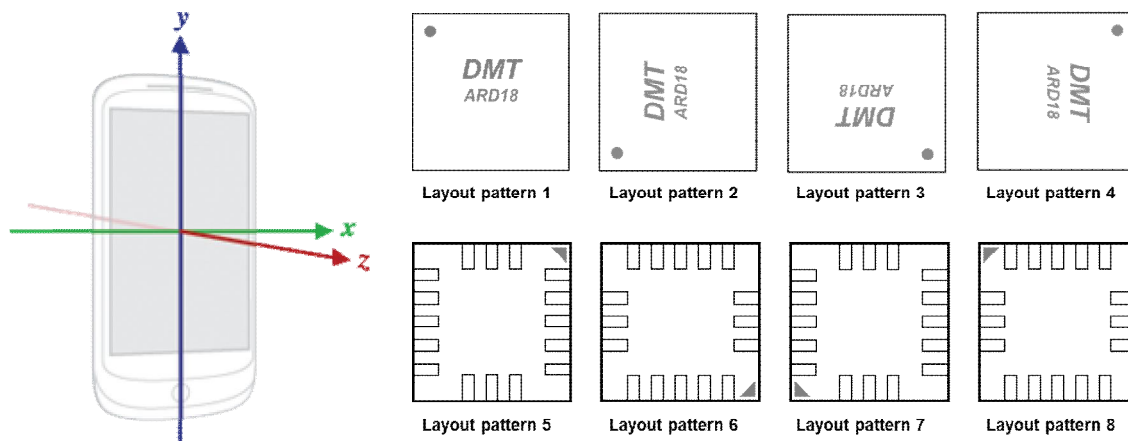


Figure 1: Android SensorEvent Coordinate System and DMARD18 Layout Patterns

IOCTL Interface

DMARD18 Android Driver implements the following five specific operations to the device.

- I. **SENSOR_RESET:** All g-sensor internal registers will be restored to default values after the reset. A typical program segment to reset g-sensor is

```
int fd = open("/dev/dmt", O_RDONLY);
ioctl(fd, SENSOR_RESET);
```

- II. **SENSOR_CALIBRATION:** Conduct static g-sensor offset calibration. Offset values will be estimated and returned to caller. Furthermore, the offset values will be saved as "/data/misc/dmt/offset.txt" and subsequent g-sensor readings will be automatically compensated by these offset values.

A typical program segment to conduct static calibration is

```
int fd = open("/dev/dmt", O_RDONLY);
int v[3] = {0, 0, 0};
//the first element is set to the static calibration orientation
v[0] = CONFIG_GSEN_CALIBRATION_GRAVITY_ON_Z_NEGATIVE;
ioctl(fd, SENSOR_CALIBRATION, &v);
//the estimated offset values will be returned in v
printf("Offset@X/Y/Z: %04d , %04d , %04d\n", v[0], v[1], v[2]);
```

The static g-sensor offset calibration requires the device to be static. And the caller needs to inform which static orientation the device is positioned when conducting such calibration. The example code illustrates the case in which gravity is acting on the -Z axis. Please refer to Figure 2 for all other static orientations.

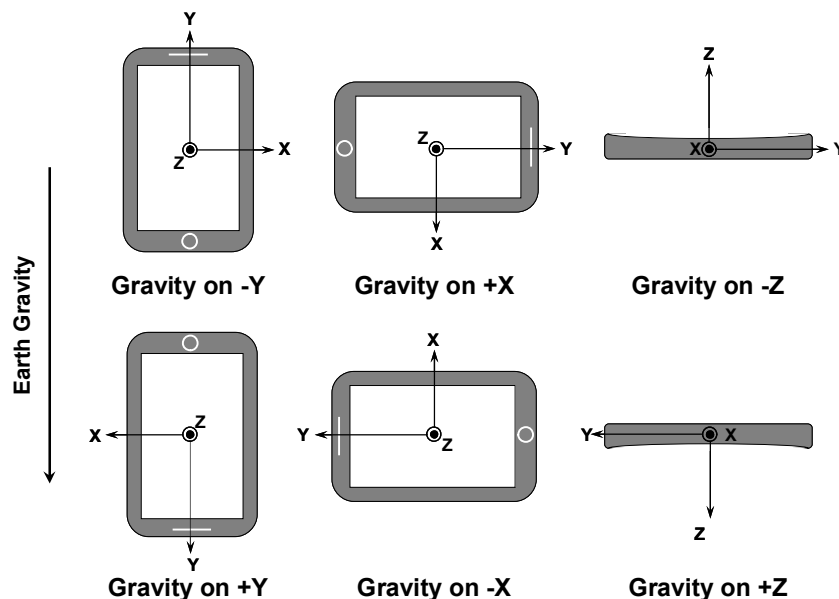


Figure 2: Static Calibration Orientations

- III. **SENSOR_GET_OFFSET:** Get the offset values currently used in the device driver. The g-sensor readings will be automatically compensated by these offset values. A typical

program segment to get offset is

```
int fd = open("/dev/dmt", O_RDONLY);
int v[3] = {0, 0, 0};
ioctl(fd, SENSOR_GET_OFFSET, &v);
//the offset values used in driver will be returned in v
printf("Offset@X/Y/Z: %04d , %04d , %04d\n", v[0], v[1], v[2]);
```

- IV. **SENSOR_SET_OFFSET**: Set the offset values into the device driver. Furthermore, the offset values will be saved as "/data/misc/dmt/offset.txt" and subsequent g-sensor readings will be automatically compensated by these offset values. A typical program segment to set offset is

```
int fd = open("/dev/dmt", O_RDONLY);
int v[3] = {20, -10, 30}; //the offset values to be set to device driver
ioctl(fd, SENSOR_SET_OFFSET, &v); //the device driver offset is set
```

- V. **SENSOR_READ_ACCEL_XYZ**: Read the acceleration values from the device driver. Note the g-sensor readings will be automatically compensated by the offset values set in the device driver. A typical program segment to read XYZ is

```
int fd = open("/dev/dmt", O_RDONLY);
int v[3];
ioctl(fd, SENSOR_READ_ACCEL_XYZ, &v);
//the acceleration values will be returned in v
printf("Acceleration@X/Y/Z: %04d , %04d , %04d\n", v[0], v[1], v[2]);
```