

Openmag User's Guide

Sean D'Epagnier

1

Copyright (C) 2007, 2008 Sean D'Epagnier

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with the Front-Cover texts being "A GNU Manual," and with the Back-Cover Texts as in (a) below. A copy of the license is included in the section entitled "GNU Free Documentation License."

Published by Sean D'Epagnier

Table of Contents

1	Software	1
1.1	Installation	1
1.2	DataClient	1
1.2.1	Example Session	1
1.2.2	Host Side Commands	2
1.3	DataViewer	3
2	Calibration	4
2.1	StillPoints	4
2.2	Verifying Calibration	4
2.3	Alignment	6
3	Emulation	7

1 Software

1.1 Installation

1. Windows OS – There is a binary distribution for windows. This is a zip (openmag_win32.zip) which contains all programs and drivers. The device can only function as one of: mouse, joystick, or cdc device. This limitation is because windows is broken. The best I can offer is to choose one device at a time. The cdc device is used for configuration, to force this mode, hold the second (right) button down when plugging in the device. This is only for the pointer model, the survey model is always in cdc mode. Windows will prompt for a driver the first time. Manually specify the downloaded folder containing the file OpenMag.inf. Once the driver is installed, you will be able to communicate via serial to the device.

Hyperterminal Setup – Normally you will use the dataclient, but if you want to use hyperterminal for some reason: go to File->Properties, click on Settings, then ASCII Setup. Check “Send line ends with line feeds” and “Echo typed characters locally”

2. Other OS – Most common OS’s support usb mouse, joystick, and cdc devices. This means you do not need to do anything special and no special driver is needed. For the serial device you should get a virtual comm port device, on linux it is /dev/ttyACM0 (the kernel module is cdc-acm), on freebsd it is /dev/ttyU0 (the kernel module is umodem). You will need Qt 4.3.0 or better to run DataViewer.

1.2 DataClient

Once configured the device will show up as a comm port. This means you can communicate to the device with any terminal program (eg hyperterminal)

The dataclient is the console version of the interface to device. The gui version is dataviewer in the next section. The dataclient program provides a more interactive user interface than a basic serial terminal. The main features are lineediting and completion via readline, data and reply separation (stdout and stderr), and the ability to change directories using cd, as well as see the directory via pwd.

The datainterface that the dataclient provides direct access to has two important concepts:

- operators – commands that operate on accessors, eg: get, set, clear, ls, ops ...
- accessors – data on the device which can be accessed eg: softwareversion

1.2.1 Example Session

To connect to a device, fifo or file:

```
./dataclient <file>
```

To connect to a server

```
./dataclient <host:port>
```

Once successfully connected you should see the prompt

```
$->
```

To autodetect the device, run the dataclient with no arguments.

It is now possible to execute commands and query data. Many of the commands are similar to unix commands. To list all possible commands issue “allops”

```
$-> allops
type ops ls allops mem get set values clear
```

Note: operators may be added or removed depending on software version

To list the accessors, issue “ls”

```
$-> ls
mouse/ joystick/ softwareversion settings/ stats/ calc/ calibration/ sensors/█
```

Items with a trailing / are directories. To list their contents, you may either cd into them, or ls them

```
$-> ls stats
freeram runtime mainloopfreq watchdog_resets
```

It should be apparent that “stats” is an accessor as is “stats/runtime”. For example you may “get stats/runtime”.

```
$-> get stats/runtime
51.81s
$-> cd stats/
stats/ $-> get runtime
54.54s
```

Not all accessors support all operators. To see which operators are supported, use the ops operator which is always supported

```
$-> ops softwareversion
type mem ops get
```

This means it is valid to replace “ops” with any of the valid operators in the above command. Notice that you cannot “set” softwareversion.

Whenever the sensors are set up to automatically output data, this streaming data appears on stderr from dataclient.

For example, you may log data

```
sean@sun ~ $ ./dataclient 2> log
$-> set sensors/accel/outputrate 10
$-> [ctrl-D to exit]
sean@sun ~ $ cat log
accel: 201 58 1273
accel: 204 65 1273
accel: 202 65 1271
...
```

This interface includes additional host side operators which work with the dataclient program, they will not work with a console program.

1.2.2 Host Side Commands

1. cd – change to a given directory if it exists, relative paths are supported, "cd ../../info" Lastdir is supported as well eg: "cd -"
2. pwd – display the current directory

It is recommended to run “dataclient --help” to show all the capabilities of dataclient.

1.3 DataViewer

The DataViewer is a graphical application used to query and interact with the device while it is running. When it is first run, you should see a tree view, and below it a console and an output window.

Hit populate to automatically completely query the device. This may take a few seconds. Now it is possible to view all of the data stored on the device. If you press “Get Values”, it will re-request just the values. This is useful because many of the values update continuously. You may also check certain values and only request those values. It is possible to modify certain values. The ones marked “write only” or “read/write” can be modified. For values with only certain possible settings, a dropdown is provided.

The Console window displays the actual data being sent and received to the device. The dataclient is actually running and doing all of the communication with the device, the dataviewer communicates with the dataclient. There may be operations that can only be performed from the console, for this you will have to run the dataclient.

The Output window displays streaming data coming from the device.

2 Calibration

One of the key features is autocalibration. The device may not come pre-calibrated, so for precise measurements, the user should understand how to perform calibration. If you are interested in how the calibration works, see the Calibration document. The device computes the unknown calibration coefficients needed to deliver useful data.

2.1 StillPoints

The device heavily uses stillpoints for calibration. What this means, is it can detect when it is not moving by looking at sensor noise levels. Once the unit is “still” the raw sensor data can be used in a meaningful way to calibrate the sensors. It is essential to place the device in various orientations and hold it perfectly still for 1-3 seconds. It is important to cover all possible orientations. To make it easier to make sure you do it correctly, I recommend each of the 6 sides in 2-3 rotations on each side. 10 positions is minimum, 12-15 positions will give better results.

Note: With the DUSI model it is possible to perform and validate all calibration using the menu system.

2.2 Verifying Calibration

The accelerometer only estimates 4 unknowns (or 7 if calibration over temperature is enabled) This means that it automatically calibrates as soon as you hold the device still for a second or so in a few orientations. Here is an example session:

```
$-> set calibration/accel/debugging true
$-> clear calibration/accel/calibration
$-> accel bias: (-1.428 18.66 3.059)
accel magnitude 1.33e+03
accel bias: (132.2 65.65 -202.8)
accel magnitude 1.33e+03
accel bias: (260.5 64.23 -109.1)
accel magnitude 1.33e+03
accel bias: (295.7 68.98 -86.69)
accel magnitude 1.33e+03
accel bias: (302.5 70.23 -87.32)
accel magnitude 1.33e+03
accel bias: (287.9 77.5 -111.9)
accel magnitude 1.33e+03
accel bias: (289.2 79.17 -112.6)
accel magnitude 1.33e+03
accel bias: (290.3 82.19 -111.2)
accel magnitude 1.33e+03
accel bias: (290.4 82.08 -115.1)
accel magnitude 1.33e+03
accel bias: (291 81.71 -116.6)
accel magnitude 1.33e+03
accel bias: (290 81.95 -118.3)
```

```
accel magnitude 1.33e+03
```

As you can see, the bias changes less as it approaches the true value. The calibration updates when you move the device to a new orientation and hold it still.

The magnetometer is more complicated to calibrate. It estimates 9 unknowns, as well as the alignment between the magnetometer and the accelerometer (4 more unknowns). The magnetometer should be calibrated away from magnetic distortions. For best results, change the bandwidth of the magnetometer to either slow or medium. When you turn mag debugging on:

```
$-> set calibration/mag/debugging true
$-> clear calibration/mag/calibration
```

There are two updates, the first gives you Residules

```
mag magnitude: 5.16e+04
mag magnitude ratios: [0.973 1.04]
mag cross coupling: {-0.118 0.172 -0.0517}
mag bias: (1.35e+04 4.59e+03 2.15e+03)
mag deviation: 0.00264
```

The closer the residue is to zero, the more correct the calibration is, a deviation of .01 or better is acceptable.

The other type of update is the Still update. You need a minimum of 10 good readings before you will get this update. Once you get a still update, the deviation should be much better, and calibration is more complete.

The magalign calibration calculates the rotation angle from mag to accel coordinates as well as the dip angle (inclination).

```
magalign rotation: <-0.00269 0.0171 0.0345> 2.21 degrees
magalign dip angle: 68.9
magalign deviation: 0.00493
```

A deviation of .02 or better is acceptable.

You can typically get this with 11-14 orientations. This deviation is the same units as the residue for the normal update. As you can see the deviation is very low once a still update occurs. Notice that the magnitude ratios are well estimated, this typically doesn't occur until a still update. It is important to get a still update before using the device for accurate measurements.

If you have calibrated in a distortion-free area, and want to lock the calibration so that moving through areas with distortions will not alter the calibration, disable autocalibration:

```
$-> set calibration/mag/autocalenabled false
$-> set calibration/magalign/autocalenabled false
```

If you do not disable autocalibration, the calibration may become worse if it updates from not moving near a distortion.

You can monitor the magalign and magfast calibration the same way as the above. The accel must have decent calibration for pitch and roll to be accurate. The accel, mag, and magalign calibrations all must have decent calibration before the yaw calculation is accurate.

2.3 Alignment

Once calibration for the sensors is performed, accelerometer and magnetometer vectors are available in the sensor coordinate system. This coordinate system typically has some arbitrary rotation away from the desired coordinate system.

There are two alignments performed. One rotates from sensor coordinates into box coordinates. Box coordinates are used to calculate pitch, roll, yaw and dip (the current measured inclination). The other rotation rotates into laser coordinates. Unlike box coordinates, laser coordinates do not have a concept of roll since the alignment is to an axis (laser or sight). Laser coordinates provide incline and azimuth which are like pitch and yaw except they use laser coordinates not box coordinates.

You may perform alignments external to the device with your own software and not use this feature, in this case you might as well read from the sensors directly.

Before performing alignment calibration, the accelerometer should be well calibrated.

To set the box alignment there are two options.

- if you know magnetic north, place the device level and facing magnetic north:

```
set boxalignment 1
```

Now the boxalignment is set

- If you do not know magnetic north, then place the device level:

```
set boxalignment 2
```

Next place the device pointed straight up (against a wall should work):

```
set boxalignment 3
```

If you want to reset boxalignment to no rotation.

```
clear boxalignment
```

To set the laser alignment, you have to take shots around the laser axis. To take a shot, align through the laser axis to a known point and:

```
set laseralignment 1
```

repeat this command for at least 5 shots, or until the error is low enough.

If you make a bad shot, you will need to reset and start over. To reset the laser alignment:

```
clear laseralignment
```

3 Emulation

Emulation allows you to run the calibration algorithms, and the menu system on a pc computer rather than the device. If you had an embedded system running a regular OS talking to the device, it would be possible to run the algorithms on that system therefore offloading the floating point computations.

Note: Currently Emulation does not compile on win32.

The program “calibrationhost” can be run which runs a tcp server. Connect to this server using the dataclient.

```
./calibrationhost -q /dev/ttyACM0 &
[1] 12351
Listening on port 7029 for connections from telnet or dataclient
./dataclient localhost:7029
$-> ls
calc/ sensors/
```

The emulation provides the exact same interface. The device (/dev/ttyACM0) should previously be configured to output accel, mag, and temperature data. There is a script “runcalibrationhost.sh” which runs the calibrationhost program and sets up the device to output raw data correctly.

It is also possible to relay the data as a server:

```
./dataclient -p 3000 -q
Listening on port 3000 for connections from telnet or dataclient
$->
```

You may then connect to this dataclient with other dataclients from remote hosts on port 3000. These dataclients may in turn run as servers as well.

It is also possible to run the menu interface if you have opengl. This program “menuhost” automatically runs the calibration algorithms, so it is equivalent to calibrationhost, with the addition of the menu system.