IMU Documentation

**INTRODUCTION**

This documentation is intended for utilizing the 3DM-GQ7-GNSS/INS. It has dual GNSS antenna capability and supports raw IMU data as well as computed metrics.

SensorConnect works very well for connecting and reading the data from the module, but we instead use the MSCL SDK in python, since we need to start automatically and are operating on a Linux system. Instead, SensorConnect is used to configure the IMU and select the channels of importance.

Here is a link to a nice quick start guide for MSCL: <https://lord-microstrain.github.io/MSCL/Documentation/Getting%20Started/index.html?python#collecting-data-213> , and documentation: <https://lord-microstrain.github.io/MSCL/Documentation/MSCL%20API%20Documentation/index.html#CClass:MipChannel> .

The current state of the codebase can be found on my personal GitHub, linked: <https://github.com/hhgarret/IMU_MSCL>. The code is annotated, but the main two fronts are MSCLrecord and MSCLdecode. MSCLrecord is used on the upboard in order to collect the data, while MSCLdecode is a parser meant to read through the files generated by MSCLrecord. MSCLdecode in its current form goes through and reads through every file and then collects each 1000th sample (a sample from each second) into a CSV which is human legible. It is my intention that MSCLdecode serves as a way for others to quickly understand and tinker with the way to parse through the code, to easily bend it to their needs.

**HOW TO USE**

MSCLrecord is a python script which is ran as a system service in order to continually record data. Instructions for how to set it up can be found on my personal GitHub linked above. Changes likely have to be made in order to customize the directory where data will be stored as the current solution is specific to our USB drive.

MSCLdecode is a python script which reads through a directory of data from MSCLrecord and outputs two csv files. One of those files is formed by extracting the header data from each block, and the other from decimating the sample data. One can run MSCLdecode as simply as being in the proper directory and executing, “python MSCldecode.py”.

MSCLdecode is currently configured to read from the data recorded between May 22nd and May 23rd.

**FILE FORMAT**

Files are stored in a directory corresponding to the current date, found in “UNIFIEDdata/MSCL\_samples/”, and are entitled based on the current hour and minute, in the format "%Y%m%d/%H%M00.bin". (00 are meant to represent seconds but are not recorded at the file level. Precise timestamps are included in the file.)

The general structure of each file is that it is a series of blocks of data. Each block of data contains a header and several samples. The number of blocks per file is specified in MSCLrecord as ‘blocksperfile’, currently 300. The number of samples per block is specified in ‘timetosample’ (in seconds, so it's multiplied by ‘samplingrate’, here 1000hz), and is currently 1 second, or 1000 samples per block. Thus, the 300 blocksperfile represents 5 minutes worth of data.

The header contains the number of channels in each sample (‘numfields’), timetosample, samplingrate, the sequence of the block in the file, then svcount\* (number of satellites visible). It then contains {numfields} channel ids, explaining what each channel is. Each of these previous numbers are 2 bytes. Finally, it contains a 10byte timestamp. Thus, the total header size is equivalent to 20 + {numfields}\*2. There are currently 19 channels, representing a wide range of data points from acceleration, attitude, and ECEF positioning.

The channel ids are used to determine the size and datatype of each channel when decoding. For example, a channel representing the IMU’s gyroscope data in quaternion form contains a 4-length vector of floats, while data from the barometer might be a solitary float. For more on how this is done, see ‘determine\_total\_size’ in MSCLdecode. By determining the size and datatype for each field code in the header an overall pattern to parse the samples is created.

When reading bytes of samples, a float is a 4byte floating point number and a double is 8bytes. The pattern of bytes in a single sample currently is ['3f', 'f', '3f', '4f', '3f', '3f', '3f', '3f', '2f', '3d', '3f', '4f', '3f', '3f', '3f', '3f', '3f', '3f', '3f'], where the first ‘3f’ is 3 floats (a total of 12 bytes) corresponding to the first channel (SCALED\_MAG\_VEC), then ‘f’ is a single float corresponding to SCALED\_AMBIENT\_PRESSURE.

The graphic below summarizes the structure of a single file:

A screenshot of a computer

AI-generated content may be incorrect.

In total the header is 58 bytes, and each sample is 236 bytes. Since there are 1000 samples per block, the size of each block is 236,058 bytes, and with 300 blocks per file each file should be of size 70,817,400 bytes, or about 67.5 MB. Extrapolating further, this means that for every 24 hour period there should be about 20.4 GB.0

NOTE: svcount is a new addition. The data recorded before June 2026 does not include svcount in the header. The parser provided with this documentation should reflect this fact, but beware if sourcing code from the GitHub repository directly.

**CHANNEL BREAKDOWN**

Below is a snippet of output used for debugging that shows the breakdown of every field currently being used in the raw and estimated data outputs of the IMU.

Active channels of type:  128 ,  CLASS\_AHRS\_IMU

# of active channels:  10

Channel Field:  32979  =  CH\_FIELD\_SENSOR\_SHARED\_GPS\_TIMESTAMP (in the header, not part of the sample)

Channel Field:  32774  =  CH\_FIELD\_SENSOR\_SCALED\_MAG\_VEC

Channel Field:  32791  =  CH\_FIELD\_SENSOR\_SCALED\_AMBIENT\_PRESSURE

Channel Field:  32772  =  CH\_FIELD\_SENSOR\_SCALED\_ACCEL\_VEC

Channel Field:  32778  =  CH\_FIELD\_SENSOR\_ORIENTATION\_QUATERNION

Channel Field:  32773  =  CH\_FIELD\_SENSOR\_SCALED\_GYRO\_VEC

Channel Field:  32788  =  CH\_FIELD\_SENSOR\_TEMPERATURE\_STATISTICS

Channel Field:  32775  =  CH\_FIELD\_SENSOR\_DELTA\_THETA\_VEC

Channel Field:  32776  =  CH\_FIELD\_SENSOR\_DELTA\_VELOCITY\_VEC

Channel Field:  32832  =  CH\_FIELD\_SENSOR\_ODOMETER\_DATA

Sample Rate:  1kHz

Active channels of type:  130 ,  CLASS\_ESTFILTER

# of active channels:  11

Channel Field:  33491  =  CH\_FIELD\_ESTFILTER\_SHARED\_GPS\_TIMESTAMP (in the header, not part of the sample)

Channel Field:  33344  =  CH\_FIELD\_ESTFILTER\_ECEF\_POS

Channel Field:  33345  =  CH\_FIELD\_ESTFILTER\_ECEF\_VEL

Channel Field:  33308  =  CH\_FIELD\_ESTFILTER\_COMPENSATED\_ACCEL

Channel Field:  33283  =  CH\_FIELD\_ESTFILTER\_ESTIMATED\_ORIENT\_QUATERNION

Channel Field:  33299  =  CH\_FIELD\_ESTFILTER\_ESTIMATED\_GRAVITY\_VECTOR

Channel Field:  33293  =  CH\_FIELD\_ESTFILTER\_ESTIMATED\_LINEAR\_ACCEL

Channel Field:  33287  =  CH\_FIELD\_ESTFILTER\_ESTIMATED\_ACCEL\_BIAS

Channel Field:  33294  =  CH\_FIELD\_ESTFILTER\_ESTIMATED\_ANGULAR\_RATE

Channel Field:  33286  =  CH\_FIELD\_ESTFILTER\_ESTIMATED\_GYRO\_BIAS

Channel Field:  33335  =  CH\_FIELD\_ESTFILTER\_ECEF\_VEL\_UNCERT

Sample Rate:  1kHz