*Versions of the prototyped case. This is an operational size with sufficient batteries for at least one month operation in comfortable temperatures and extended runtime if solar power is available. Smaller sizes may be possible but have not been explored.*

# Sapphire Mini Guide

**Version 5.3.0 Project Sapphire** [1]

2022.07.04

—


Chris Hayward

Dept Earth Sciences

Southern Methodist University

Dallas, Tx 75275-0395


[1] Project Sapphire's name is a takeoff on the GEM recorder (Anderson, et al 2017)

## Known Issues

***This document is in progress and may not reflect the current firmware***

1. The accelerometer channel BHZ is unable to be used when the other 4 channels are active.
2. Accelerometer channels BHN and BHE are not available

*These are being investigated and addressed in future firmware revisions and patches*

### Future Modifications

There are several future firmware modifications being considered as a result of user field experiences. These are:

1. Scheduled window recordings. This would allow the system to go into deep sleep and awaken at specific times.
2. Real time miniseed stream. This would allow a miniseed reader to pull data from the Sapphire in real-time.

# Overview

The Sapphire recorder is a simple nodal infrasound recorder designed for short campaigns.

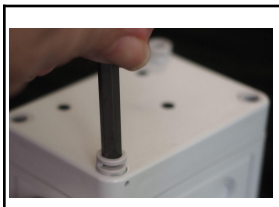
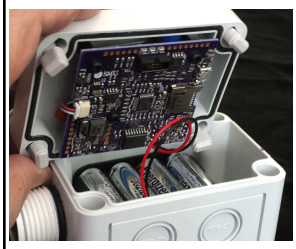
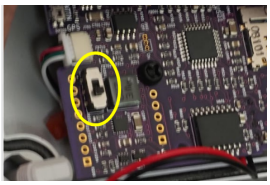
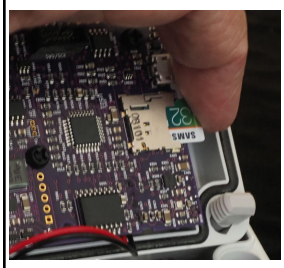
Originally intended as a low cost alternative to evaluate the coherence of microbaroms propagating in the stratosphere using 40 or more stations distributed over 200 km, it has been used to record more traditional 1-5 Hz infrasound as well as stronger signals above 10 Hz.

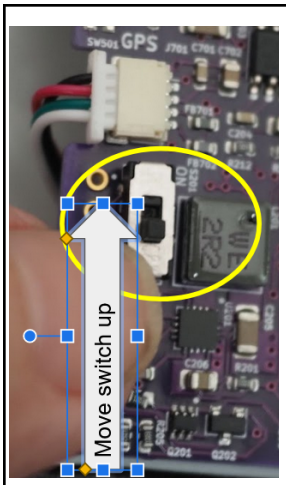
### Evolution from Similar Systems

The development of the Sapphire was prompted by the success described in Anderson's 2018 description of the GEM infrasound logger (doi: 10.1785/0220170067). Electing to call this version a Sapphire (another GEM) honors the original concept and design. Since the first SMU internal concept paper (in April 2019), there continue to be indications that similar such systems developed in other institutions. The success of nodal seismic recorders in the seismological academic community further indicates the utility of similar such systems for infrasound.

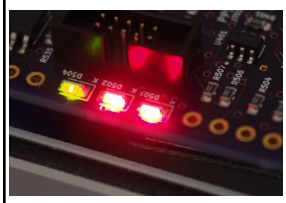
For this project, the design focused on constraining production costs, necessarily trading off performance, reliability, and flexibility in the units.

# Quick Start

| | |
|---|---|
|  | ## In the Lab<br>Using a flat blade screwdriver loosen the four plastic bolts. These are captive bolts and will stay connected to the lid. |
|  | Carefully open the lid to avoid pulling on the two wires connected to the battery pack. The wires should be long enough to allow the top to sit upside down on the table. |
|  | Set the switch in the off position. The handle will be away from the nearby white connector. |
|  | **Leave the SD card fully inserted if you do not wish to erase it**<br>If there is an inserted micro SD card, eject it by pushing it in slightly to activate the release. The card should spring out slightly. The card does not have to be completely removed |
|  | Install 4 fresh AA batteries. Use Eveready Lithium L91 batteries for deployments in cold temperatures or for maximum run time. If using alkaline, remove the batteries when the unit is not in use to avoid leakage.<br>Double check the polarity of the batteries as they are installed. |

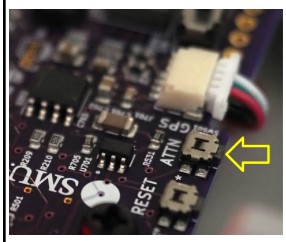



Turn on the power by moving the slide switch towards the white connector. This will start the Sapphire



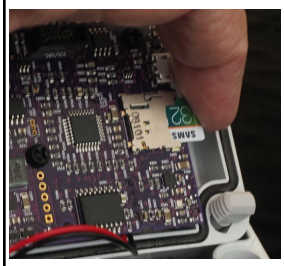
**If the SD card is fully inserted this step will be skipped**

Observe the LEDs. The LEDs will cycle through various combinations and after a few minutes will stop with the two red LEDs on steady.



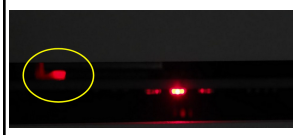

**SKIP this step if you do not wish to erase the SD card.**

Depress the ATTN button once. The button depresses from the side, not the top. One red LED will turn off and one will remain on when the button is released. Be careful not to press the reset button -- that would force a power cycle reset.



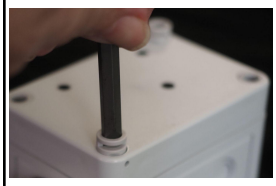
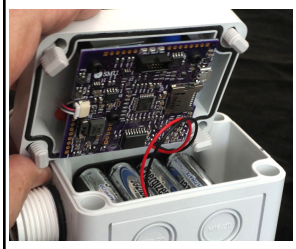
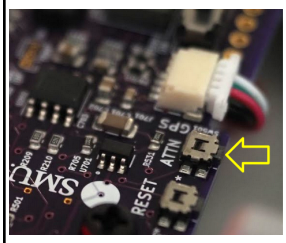
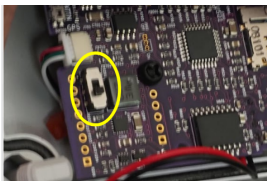
Insert the micro SD card. The LEDs will work through several cycles. Eventually, you should see the green LED blink once for every 16 samples recorded, the center red LED on indicating no GPS lock, and the end red LED blinking each time data is written to the SD card.

If the system stalls for several minutes with all LEDs on solid, there may have been a lock-up on a bus that will require a power cycle to clear.



## In the Field

Loosely place the top back and move outside where there is a clear view of the sky to allow the GPS to set the time. This should be done within a few minutes of turning power on since the GPS has a short timeout. On the top side of the circuit board (opposite the 3 LEDs), you may see a blue LED flashing once per second indicating the GPS is active. It will be later joined by a red

| | |
|---|---|
| | LED once a position fix is done. The center red LED will remain on until a GPS time lock has occurred.<br>*NOTE: If all three LEDs stay on for several minutes, a bus lockup has occurred during startup. Switch the power off for 30 seconds and restart the system.* |
|  | Carefully close the top, making sure not to pinch any protruding wires. The screws should be just firm enough to close the box and engage the gasket. |
| | Place the unit in an appropriate wind shielded location. If rain is expected during the deployment, tilt hose connection slightly downward to avoid ponding water on the top as this will interfere with GPS signals. |
| | Retrieve the unit when the experiment is complete. |
|  | ## In the Lab<br>Open the unit. This is best done inside to avoid dirt and water getting into the unit. |
|   | Depress and release the ATTN button. This will force the current file to be closed correctly. Failure to do this step can result in a loss of the last 15 minutes of data. |
|  | Turn the power off |
| | There are two options to recover the data. One is to simply extract the SD card and read it using a USB micro SD card reader. The other is to plug a micro USB cable into a computer and allow the computer to mount the drive directly. For large datasets, using a USB card reader is faster. |

| | |
|---|---|
|  | If the batteries are exhausted, low or won't be used in the next several weeks, remove and discard them prior to storing the unit. Batteries are inexpensive relative to most experiments. |

# Field Considerations

## Wind Noise

Typically, wind noise is the dominant interfering noise on infrasound observations. By far the biggest infrasound blinding noise is direct wind on the sensor. While the Sapphire is adapted for porous hose wind reducers, and these are particularly effective below a few Hz, the small size of the Sapphire means that there are other easy (although perhaps less effective) methods to reduce noise on the sensor.

Dense vegetation is an effective reducer and can be used alone or with very short segments of porous hose. Units should be placed on the ground to avoid wind driven motion.

The small size allows the units to be placed in sprinkler valve boxes, typically tied wrapped or taped to the top to provide the least obstructed view of GPS satellites and minimize damage from flooding of the valve box.

## Weather Resistance

The Sapphires are not fully weather sealed. When deployed the hose port is about ½" above the base and water less than ½" is not a concern. Water over 1¼" will enter the box, but should (if everything was sealed correctly not flood the battery compartment. A greater concern is blowing rain or rain splash.

To minimize the risk of loss due to water, we recommend sealing the units in a ziplock bag which will be transparent to infrasound.

Boards have not been conformal coated and systems are also sensitive to condensing humidity. In these conditions, using zip lock bags with 5 gm packets of indicating silica gel drying agents will provide good protection from condensation.

## GPS Considerations

The GPS is the dominant power consumer in the Sapphire and the time required for a GPS to lock to satellites is dependent upon the GPS signal strength and the number of satellites visible at any one time. For the GPS, the ideal placement is in the open with no additional materials above the lid. This often isn't practical (finding a no wind area with a clear sky view is difficult). In general plastic containers (if they aren't black) have little effect on the signals. Thin films of water, wet ground above the lid, and heavy wet vegetation obscure the signals and can even result in a complete failure to lock. Other containers, including those of black plastic should be tested.

Another GPS antenna concern is ponded water on top of the box. This can also block GPS signals. Leaving the units on a slight incline will minimize the possibility of ponding water.

If there is a concern about a particular installation for GPS, a test unit can be loaded with the GPS_Serial firmware and satellite SNR can be examined using UBLOX's U-Center software. This does require a field laptop and USB cable, but may provide valuable information for new kinds of installations.

## Power Considerations

The GPS requires roughly 250mW while operating. Time to achieve a clock synchronization is dominated by the time to acquire GPS signals and is typically a few minutes. Power may be reduced by extending the GPS cycle times or free running without the GPS if clock drift can be tolerated (as may be the case in some deployments).

## Micro SD Card

The next most power hungry operation is writing to the SD card. SD cards vary in their power efficiency, with some cards requiring more than 3x the power of others. High speed and high capacity cards often require more power than their less expensive low speed varieties. Reducing the number of writes by reducing the sample rate, turning off unneeded channels, avoiding frequent disk synchronization will all help reduce the total power needed.

We recommend low speed, low capacity cards. An 8GB is sufficient for any deployment using 4 AA batteries.

*Data use for 30 day recording*

| | Data use in Gbytes | | | |
|---|---|---|---|---|
| | Sample Rate | | | |
| **Channels** | **16** | **32** | **64** | **128** |
| **1** | 0.2 | 0.3 | 0.7 | 1.4 |
| **2** | 0.3 | 0.7 | 1.4 | 2.8 |
| **3** | 0.5 | 1.0 | 2.1 | 4.2 |
| **4** | 0.7 | 1.4 | 2.8 | 5.6 |

| | | | | |
|---|---|---|---|---|
| **5** | 0.9 | 1.7 | 3.5 | 6.9 |

Total battery life depends on both the battery capacity (about 10,000mW-hrs) for the set of batteries and the peak power drains that have to be delivered. As batteries near exhaustion, it becomes difficult to support the GPS power drains without going to brown out conditions.

## Temperature

Some critical components in the Sapphire were specified for Industrial Temperature ranges, -40 °C to +85 °C**,** namely the temperature compensated oscillator and microcontroller. However, at the time of purchase, some components were only available in commercial ranges, 0 °C to +60 °C, and therefore operation in cold climates may need to be tested prior to deployments. For cold temperatures, lithium batteries, Eveready L91 should be used in place of alkaline.

The Sapphire, like all differentially based microbarographs, is sensitive to rapid temperature changes. Because the Sapphire differential gauge has a long period response, small changes in temperature due to direct sunlight may appear as long period noise. If long period signals are of concern, the units should not be placed in direct sunlight.

In general, it will be necessary to high pass filter the resultant infrasound BDF channel to eliminate the long period noise.

## LED Status during normal operation

The three LEDs on the board edge, Green (G), Center Red(C), and Edge Red(E), provide indications as to the current system state. However, they have to be assessed in context. A green LED at the start of the program indicates something different than it would after acquisition starts. The details also may change depending upon the exact firmware version and patch level.

*Table of common blink codes while the system is operating. This does not include some of the ephemeral codes during startup, nor the error codes (listed later in the document). LEDs are shown as position RRG. If the LED is on continuous it is shown as R,or G (depending upon color). If it is flashing an F is used. LEDs are shown in position with the green LED on the right.*

| | | |
|---|---|---|
| --G | Upon initial USB plugin to computer | USB disk emulator started. E indicates read/write activity. |
| R-G | Upon initial USB plugin to computer | USB disk emulator running. No activity. Waiting for user to move/edit/delete files or eject the drive |
| --- | Upon USB eject | USB disk emulator end |
| | | |

| | | |
|---|---|---|
| FF- | Startup: SD card ejected. Two lights flashing about once/second | Waiting for ATTN button to erase the SD card when it is inserted. Or insert card without ATTN if it should not be erased |
| F-- | Startup: SD card ejected | ATTN button has been pressed. SD card will be erased if it is inserted |
| CCC | Startup: cycle all combinations (1-7) | Time slot to push ATTN and clear NVRAM setting the configuration to default values |
| F-- | Startup: fast flashing red | Waiting for serial console to connect on USB |
| FFF | Startup: all flashing slowly | ALERT: Running a test version of the operating system. |
| R-- | During acquisition | Dumping a miniseed record to SD |
| --G | During acquisition | Sampling running. Will blink every 64th sample |
| -R- | During acquisition | If on steady it indicates no GPS lock. Short flash is used to indicate the processor has move to high power mode to handle some critical item |
| RRG | On steady for more than 1 minute | Processor hard fault. This could be a firmware bug or a hardware error. Power cycle may be needed to correct it |

## Field Startup

When the weather is cooperative without precipitation or excessive humidity, the field start procedure in the quick start guide is sufficient (ie opening the units in the field, switching on power, and observing if GPS is acquired. Under difficult field conditions, blowing dust, rain, snow, and generally miserable work conditions it is preferable to start the units in a vehicle or lab, seal them, and place them while they continue to take data.

Provided the units are not roughly handled, this indoor pre-start may give the most reliable start when working with large deployments since it simplifies the work in the field.

### Maximizing GPS efficiency

If starting units inside, a GPS signal is still needed to set the time. One can either start the units on a window sill with a partial view of the sky (avoid low e glass windows or windows with metallic films), or start the units inside and within a few minutes carry them outside to get an initial GPS lock.

If units are started inside without a GPS lock in the first few minutes, the units will attempt a second lock within the GPS cycle time, but the long search time (and failure) will deplete the batteries more than would otherwise happen.

The GPS included with the current Sapphires includes a battery backup that provides an accelerated startup under some conditions. When units have been off for an extended period (days) or are moved substantial distances (>100 km), the initial GPS lock may require more time than normal resulting in multiple failures to lock and increased power use upon the first locks. In addition, if the GPS is left on for an extended time, it will track additional satellites and will reduce the time for subsequent uses.

To efficiently use these features, the following steps are suggested:

1. Power the unit from a USB connection with the power switch off
2. Start the unit with the micro SD card ejected. This will leave the GPS on indefinitely until the SD card is inserted
3. Monitor the GPS LEDs until both the blue GPS LED and red GPS LED flash at once per second
4. Continue to leave the micro SD card ejected for an additional 15 minutes.
5. Insert the SD card.
6. Switch the power to on.
7. Remove the USB cable and close the unit

# Recording Format

The Sapphire dumps a file named CFG###.log each time it starts, overwriting any earlier version of the log file (where ### is the unit serial number). This file contains the current recording parameters in JSON format (see the section on Recording Parameters for more explanation). This file should be stored with the data in case there are questions about the settings for a particular experiment.

## Sample Format

Miniseed records are typically written as 32 bit floats with the conversion to Pa, hPa, ℃, or g already done. It is possible to request 32 bit ints instead. This will result in a slight power savings, but conversion will have to be done by the user. For the DLHR and accelerometer channel the conversion requires a relatively simple rescaling. For the BMP barometer channel, the conversion is more complex and requires calibration constants from each individual chip and scaling using a 12th order polynomial as described in Section 9 of the BMP388 datasheet. Calibration values are found in the LOG file.

## Timing

Provided the GPS has locked, miniseed records have the start time of the first sample of each block and do not need further correction. When the GPS has never locked, the Sapphire will start with the time 2022-01-01 00:00:00. Therefore nearly every experiment will have a segment of time series data starting with this time. If a later GPS cycle determines that the clock has drifted significantly, sampling will be stopped and reinitialized resulting in a small data gap.

For the LOG channel, the time in the miniSEED header represents the time that the particular block of log was dumped and should generally be ignored. There is a timestamp inside the JSON text which indicates the time associated with the log message.

## NaN and Flag samples

At times, due to conflicts with various threads running on the Sapphire, there may be a missed sample. In this case the Sapphire will insert a flag value (currently -128) to indicate a missing sample. This is done to retain the correct timing of succeeding samples. In processing, I typically replace -128 values with the median or mean of the surrounding samples.

If the sensors record an out-of-range (clipped) condition, NaN will be inserted into the sample stream.

## Channel Names

The Sapphire records in block multiplexed miniSEED format with the 6 possible channels.

*Sapphire channel names*

| Channel Name | Contents |
|---|---|
| BDF_00 | Differential pressure gauge. This is the traditional infrasound channel. Data is recorded in single precision float format and scaled to Pa. |
| BDE_00 | Barometer gauge. This is a microbarometer recorded in signal precision float format and scaled to hPa |
| BDK_00 | This is the temperature of the differential pressure gauge in Centigrade. Data is in single precision float format. |
| BDL_00 | This is the temperature recorded by the barometer in Centigrade. Data is in single precision float format. |
| BDZ_00 | This is the accelerometer output recorded in mg in signal precision float format. Most |

| | |
|---|---|
| | Sapphires omit this channel (accelerometer chips were difficult to source in 2021). |
| LOG_00 | This is a JSON formatted miniseed ascii block containing state-of-health information such as battery voltage and GPS location. It is not recorded at a fixed sample rate and generally has to be read outside of ObsPy |

## File Naming Convention

MiniSEED files are named as S###_yymmdd_Nnnn.msd where ### will be replaced by the Sapphire serial number. The yymmdd is the year, month, and day of the clock when the file was opened and nnn will be the file sequence starting with 001 for the first recorded file and incrementing each time. If the Sapphire is restarted file numbering will start with the next available number + 1 (miniSEED files are never overwritten). The file time associated with the file is the Sapphire time when the file was opened. If the GPS hasn't locked, this time will be 2022-01-01. Otherwise it should reflect the first time block in the file.

| Name | Date modified | Type | Size |
|---|---|---|---|
| CFG502.log | 1/1/2022 12:00 AM | Text Document | 2 KB |
| S502_220101_N001.msd | 7/9/2022 2:45 AM | MSD File | 20 KB |
| S502_220101_N003.msd | 7/9/2022 2:15 PM | MSD File | 17 KB |
| S502_220101_N005.msd | 7/9/2022 4:46 PM | MSD File | 397 KB |
| S502_220709_N001.msd | 7/9/2022 2:20 PM | MSD File | 185 KB |
| S502_220709_N002.msd | 7/9/2022 2:30 PM | MSD File | 415 KB |
| S502_220709_N003.msd | 7/9/2022 2:40 PM | MSD File | 393 KB |
| S502_220709_N004.msd | 7/9/2022 2:50 PM | MSD File | 403 KB |
| S502_220709_N005.msd | 7/9/2022 3:00 PM | MSD File | 393 KB |
| S502_220709_N006.msd | 7/9/2022 3:10 PM | MSD File | 403 KB |
| S502_220709_N007.msd | 7/9/2022 3:14 PM | MSD File | 176 KB |
| S502_220709_N008.msd | 7/9/2022 3:14 PM | MSD File | 8 KB |
| S502_220709_N009.msd | 7/9/2022 3:16 PM | MSD File | 63 KB |
| S502_220709_N010.msd | | MSD File | 0 KB |
| S502_220709_N012.msd | 7/9/2022 4:50 PM | MSD File | 144 KB |
| S502_220709_N013.msd | 7/9/2022 5:00 PM | MSD File | 393 KB |
| S502_220709_N014.msd | 7/9/2022 5:10 PM | MSD File | 403 KB |
| S502_220709_N015.msd | 7/9/2022 5:20 PM | MSD File | 393 KB |
| S502_220709_N016.msd | 7/9/2022 5:30 PM | MSD File | 423 KB |
| S502_220709_N017.msd | 7/9/2022 5:32 PM | MSD File | 78 KB |

*Output data files. The S502_220101* files are usually the start of each recording session. This example shows 3 sessions. The last session would include files S502_220101_N005 and files S502_220709_N012-S502_220709_N017. Note that the skipped sequence numbers indicating a change in sessions*

## Data Recovery Workflow

Since the recording is native miniSEED, there are a variety of tools that can read the raw data directly. Usually the LOG channel has to have some special handling, but other channels are easily adapted.

*Tools for examining the miniseed data*

| System | Libraries and Commands |
|---|---|
| Python | ObsPy |
| Matlab | rdmseed[2] |
| Antelope | miniseed2days, miniseed2db |
| Geotool | Will read native file, but skipping LOG records |

## Python

Someone who knows ObsPy needs to add a short segment here on reading the Sapphire Data

## Matlab

Use the community function rdmseed at https://github.com/IPGP/mseed-matlab.

Comment out the 3 lines as shown below regarding WordOrder swaps

```
        case 4
                % --- decoding format: IEEE floating point
                D.EncodingFormatName = {'FLOAT32'};
                dd = fread(fid,ceil((D.DataRecordSize - D.OffsetBeginData)/4),'*float');
%               if xor(~WordOrder,le)
%                       dd = swapbytes(dd);
%               end
                D.d = dd(1:D.NumberSamples);
```

While in matlab, the command 'doc mseed' will list a detailed set of instructions for the function. Then a simple retrieval and plot would be

```
rdmseed('S098N04.msd','plot');
```

## Antelope

In Antelope, for efficiency it is convenient to demultiplex the input miniseed file and build miniseed day files for processing. All files from all Sapphires can be loaded into a single directory, typically called Sapphire_Raw. Then the sequence of commands for preprocessing and displaying the files is:

```
miniseed2days Sapphire_Raw
miniseed2db 2022 Sapphire_db
dbpick Sapphire_db
```

The above assumes that the year is 2022.

[2] A minor patch must be made to correctly handle floats.

### Geotool

Geotool is a general seismic trace display tool available to institutions with some connection to the CTBTO. If using geotool, open the file, specify miniseed format, and change the required extension from 'mseed' to 'msd'. The file should be able to be directly read into Geotool.

## Working with the LOG channel

The log channel is recorded in miniSEED ascii format and is embedded as a sequence of JSON strings. Although most miniseed libraries can read the ascii file, they often get confused since the miniseed header times are not evenly sampled. Therefore some work has to be done to make sense of the files. The simple PERL script below will strip the ascii out of the miniseed to produce a file that can be examined, printed, or processed with standard JSON libraries.

```
#!/bin/perl
#  extract LOG file block from miniseed and output in a readable format
#
#
#
$lineN = 0;
while ($ARGV = shift) {
       open(ARGV,$ARGV)||warn "Couldn't open $ARGV";
       while(read(ARGV,$buf,512)>0) {
       @log = unpack("x15 A3 x2 x10 S x24 a456",$buf);
       if ($log[0]=~/LOG/) {
                     @text = split(undef,$log[2]);
                     $line[$lineN] = join(//,@text[0..$log[1]-1]);
                     $lineN++;
       }
  }
  }
  $lines = join(//,@line);
  $lines =~ s/}}/}}\n/g;
  $lines =~ s/\000//g;
  print "$lines";
```

To use the file, copy the above lines to a file named 'extractLog.pl', then execute it as

```
extractLog.pl SN098*.msd
```

While in the directory with the miniSEED files.

### Log Variables

Samples of a log file are extracted below

*Sapphire Log file data segments*

| | |
|---|---|
| {"gps":{"Time":1640995200,"Lat":0,"Lon":0,"Height":0,"Nfix":0,"OnTime":71,"Slew":0,"SlewRate":0,"Aging":0,"i":01},"i":00} | GPS initializing, no time fix yet |

| | |
|---|---|
| {"power":{"Time":1640995201,"batVolt":5.70919,"current_mA":73,"power_mW":259.6,"busVoltage_V":3.696,"i":02},"i":00} | |
| {"gps":{"Time":1640995212,"Lat":328097526,"Lon":-967228401,"Height":116880,"Nfix":1,"OnTime":10,"Slew":0,"SlewRate":0,"Aging":0,"i":011},"i":00} | GPS has time fix and is now slewing the local clock |
| {"power":{"Time":1660182135,"batVolt":5.70919,"current_mA":66,"power_mW":321.6,"busVoltage_V":3.612,"i":067},"i":00}<br>{"gps":{"Time":1660182135,"Lat":328097510,"Lon":-967228102,"Height":1168901,"Nfix":4,"OnTime":82,"Slew":-1500,"SlewRate":66,"Aging":-3,"i":067},"i":00}<br>{"power":{"Time":1660182136,"batVolt":5.70919,"current_mA":11.8,"power_mW":321.6,"busVoltage_V":3.612,"i":068},"i":00} | Local time is locked to GPS and GPS is being switched off |

The above shows the JSON formatted log messages. Variables are explained in the following table.

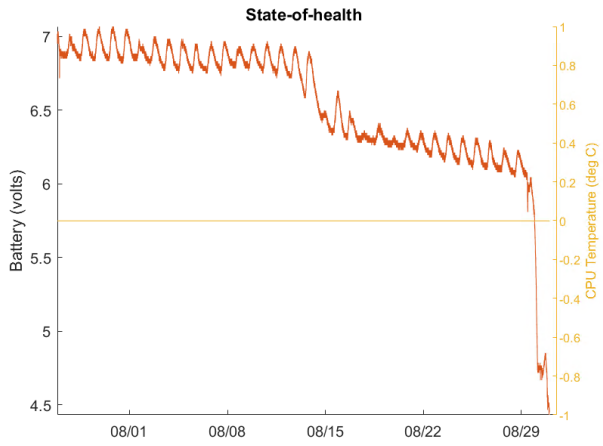
*JSON variable names and use*

| Variable Name | use |
|---|---|
| Time | POSIX time (seconds from 1970/1/1). This is the local time for this record. |
| batVolt | Battery voltage as measured by the microprocessor A2D. This is the value used when comparing battery voltage to cutoff levels |
| current_mA | Current in mA at the battery voltage level. Since the current is only measured when the processor is at high operation levels (not at idle), this only reflects peak power use. |
| busVoltage | Battery voltage measured at the power monitor chip which should be about the same as that at the microprocessor A2D |
| power_mW | Power as calculated by the power monitor chip |
| Lat | Latitude * 1e7 |
| Lon | Longitude * 1e7 |

| | |
|---|---|
| Height | Height in m * 1e3 |
| Nfix | Number of valid clock pulses at this time. This number is periodically reset. In general we require Nfix > 4 before believing the time |
| OnTime | Number of seconds the GPS module has been powered on |
| Slew | Current calculated slew in nS between the local clock and the GPS time |
| SlewRate | Current slew rate in nS/S of the local oscillator. |
| Aging | A number from -128 to +128 that is used to trim the local clock frequency. |
| i | Counter starting with 1 that increments for each of this type of message (one index for gps messages and another for power messages) |

### Monitoring Battery Voltage

*Battery voltage extracted from log file*




The LOG file records the battery voltage each time state-of-heath is recorded. The figure to the left shows a typical battery voltage for Lithium L91 AA batteries for a deployment with a single shot GPS, two channel recording at 128 SPS. The offset step around 8/14 is a characteristic of L91 batteries as they reach their 50% capacity. The diurnal ripples are the effect of temperature on the battery voltage.

## Recording Parameters

The Sapphire has a moderate set of recording parameters that may be adjusted by the user. When parameters are adjusted by means of a custom config file, they replace the default parameters held in non-volatile memory and will be used for all future acquisitions. This guide assumes that all parameters are set to their defaults. We recommend that if

parameters are changed from their defaults they either be set back to default at the end of the experiment, or a note is made for the next user of the changes.

Almost all testing during firmware development was made with default parameters. There has been relatively little testing with parameters adjusted far from defaults. While we are interested in correcting any errors discovered for defaults that you regard as sensible, we recommend through testing before doing an experiment if you modify parameters. *You have been warned!*

## Examining the current configuration

The current configuration is recorded in the file CFG###.log where ### is the Sapphire serial number. This file is overwritten each time the Sapphire starts. It is a valid JSON structure. The file is constructed in the following manner:

1. The parameters are loaded from program constant defaults.
2. If NVram has defined parameters, those overwrite the defaults.
3. If there is a file named config.jsn on the SD card, and it has the correct JSON format, it is parsed and any redefined parameters are loaded into NVRAM
4. For a few parameters, the firmware probes the hardware to discover what sensors are connected, the size of the nvram, and the size of the SD card. These values are overwritten upon each execution.
5. The final parameters are written to the CFG###.log file

*An example CFG Log file is shown below*


```
{
  "version": 5,
  "revision": 3,
  "patch": 0,
  "acq": {
        "sampleRate": 128,
        "recordDLHR": true,
        "recordBMP": true,
        "recordDLHR_T": true,
        "recordBMP_T": true,
        "recordACC_Z": false,
        "format": 0,
        "missedSample": true,
        "reserveTime": 750
  },
  "gnss": {
        "mode": "cycled",
        "period_min": 20,
        "minOn_sec": 8,
        "searchPeriod_sec": 600,
        "requiredOn_sec": 4,
        "minVoltage": 4.400000095
  },
  "timing": {
        "sync": true,
        "jamsetThreshold_us": 500,
        "trimThreshold_us": 100,
        "requiredTrim_sec": 20,
        "requiredNDelta_sec": 10,
        "agingOffset": 0,
        "microOffset_us": 500,
        "latency_us": 175
```

```
  },
  "sdcard": {
        "syncTime_min": 2,
        "filePeriod_min": 10
  },
  "debug": {
        "msLogLevel": 0,
        "msLevel": 0,
        "verifyTiming": false,
        "usbOnTime_min": 60,
        "debugMask": 0
  },
  "status": {
        "sohPeriod": 1,
        "serialNo": 99,
        "watchdogTimeout_sec": 16,
        "batteryCutoff_volts": 4.099999905
  },
  "compiled": {
        "compileTime": 1660163965,
        "optionsFingerprint": 261367,
        "waitOnInput_sec": 20,
        "rtcPeriod_ms": 0,
        "rtcOffset_ms": 0,
        "sampleOffset_ms": 0
  },
  "probed": {
        "addrDLHR": 41,
        "addrBMP": 118,
        "addrACC": 0,
        "addrINA": 64,
        "addrDS3231": 104,
        "sizeFRAM": 8192,
        "sizeSD": 4
  }
}
```


The configuration parameters are explained below

| Parameter | Valid Values | Use |
|---|---|---|
| version | 5 | unused |
| revision | 2 or greater | If the version in NVRAM is different than the default version, NVRAM is completely cleared and all parameters revert to their defaults |
| patch | 0-999 | If the patch level in the program default is different than that in NVRAM, all of NVRAM is cleared and all parameters revert to their defaults |
| sampleRate | 16,32,64,**128** | Reduced sample rates have improved self noise and will extend battery life |
| recordDLHR | **true** | Record the infrasound BDF channel. Must be true for any recording to occur |
| recordBMP | **true**/false | Record the barometer as BDE channel |

| | | |
|---|---|---|
| recordDLHR_T | **true/**false | Record the temperature of the infrasound sensor |
| recordBMP_T | **true/**false | Record the temperature of the barometer |
| recordACC_Z | true**/false** | Record the accelerometer if present |
| format | **0/**1 | Format for miniseed data. Zero for single precision floats, 1 for 32 bit ints. |
| missedSample | **true/**false | Flag missed samples with -128 if true. If false, duplicate the prior sample to avoid introducing spikes that have to be removed in processing |
| reserveTime | **750** | Number of milliseconds to buffer data. This needs to be large enough that no samples are missed during SD card writes but small enough that there is room in memory for all channels. Use care if changing this and test throughly |
| Gnss mode | oneshot, **cycled**, continuous | If onshot, the time is set once at the beginning and then one time later to check for clock drift. If cycled, the GPS is checked every period. If continuous, the GPS is left on all the time. This last option increases the power to over 200mW but will result in a very closely locked time. |
| period_min | >3<br>**120** | Number of minutes between GPS attempts if in cycled mode |
| minOn_sec | >4<br>**8** | Minimum GPS on time to lock the local oscillator to the GPS time |
| searchPeriod_sec | >60<br>**600** | Maximum time for the GPS to stay on in one attempt. If there has been no lock in this many seconds, the GPS shuts down until the next cycle time |
| requiredOn_sec | >1<br>**4** | Minimum time for the GPS date/time to be accepted. |
| minVoltage | >2<br>**4.4** | When the supply voltage is below the GNSS minVoltage cutoff, the GPS is quenched. Data will continue to be recorded down to the cutoff voltage. |
| Timing sync | **true**/false | If true the local time is synced to GPS. If false, the local time is allowed to drift. This is sometimes used to rate the stability of the local oscillator |
| jamsetThreshold_us | >50<br>**500** | When a GPS lock occurs and the time offset is larger than the jamset threshold microseconds, sampling is stopped and the time is reset. For |

| | | |
|---|---|---|
| | | offsets under this threshold, the clock is slowly adjusted to the correct time. |
| trimThreshold_us | >50<br>**100** | For offsets less than the trim threshold, the clock is not adjusted at all. Offsets greater than the trim threshold require the clock to be slowly skewed to the correct time. Because the GPS is running during the skew, a small trim threshold results in higher power consumption. |
| requiredTime_sec | >2<br>**20** | The local oscillator is compared to the GPS time over the required time seconds to determine how to adjust the frequency. Longer times are more precise, but require more power. |
| requiredDelta_sec | >2<br>**10** | This sets the minimum skew time. There is also a hard coded maximum skew rate that handles larger skews |
| agingOffset | -128 - 128<br>**0** | This is the adjustment for the oscillator crystal aging. This number is only used until a GPS time lock is acquired and can safely be left at zero. |
| microOffset_us | **500** | A control to skew the 1PPS slightly so that sampling occurs slightly off the second. This stabilizes some of the multi-threading operations and reduces the self noise. This should not be changed. |
| latency_us | **175** | This is the latency in the clock set operation. It should not be changed. |
| sdcard syncTime | >2<br>**15** | This is the number of minutes between file directory updates. Effectively data is recorded in syncTime blocks. Hence resetting the system or turning off the system could lose this number of minutes of data. Lower numbers decrease battery life and can result in more missed samples. |
| filePeriod_min | >5<br>**240** | New files are created each filePeriod minutes and are aligned on the hour if an even number of hours are used in each file. Closing a file and opening a new one is a power expensive operation. |
| Debug msLogLevel | 0-32768 | Unused |
| msLevel | 0-32768 | Unused |
| verifyTiming | true/**false** | When set to true, and when the GPS is active, the polarity of one or more channels are inverted on the PPS pulse. |

| | | |
|---|---|---|
| usbOnTime_min | >2<br>**12** | This is the number of minutes the USB serial port is held active. If the system is plugged into a computer debug and status messages will be listed as the system runs. |
| debugMask | 0 | unused |
| Status sohPeriod | >0<br>**1** | Seconds between LOG messages. Practically, the log messages need to be frequent enough to capture the GPS time and position when it cycles on. |
| serailNo | 0-999 | The serial number is special in that it is retained during any NVRAM clear operations. |
| watchdogTimeout_sec | >4<br>**16** | If for some reason the system locks up the watchdog will force a reset after this many seconds |
| batteryCutoff_volts | >2<br>**4.0** | If during an SOH check the battery voltage drops below the cutoff voltage, the system does an immediate shutdown. This prevents disk errors during brown-out conditions |
| compiled compileTime | Program defined | This is the unix time of the firmware compilation and provide a second check on the firmware version in case someone forgot to change the patch level |
| optionsFingerprint | Program defined | This is a fingerprint of the compile time options that were used with the code. Combined with the compileTime, version, and revision, this gives a unique fingerprint as to the firmware code. |
| waitOnInput_sec | 0-86400<br>**20** | This is the number of seconds that the Sapphire will wait for a micro USB cable to be connected to a software serial port. During the wait, the end red LED will flash rapidly. At the end of the wait, the program will continue even if no cable is connected. |
| rtcPeriod_ms | 0 | unused |
| rtcOffset_ms | 0 | unused |
| sampleOffset_ms | 0 | Unused |
| probed | | These values are input by the firmware upon every execution. The values starting with addr are the bus address of the various sensors. If an address is zero, then the device does not exist on this |

| | | |
|---|---|---|
| | | Sapphire. |
| sizeSD | probed | This is the size of the current SD card in GB. |

## Changing the configuration

To change the configuration, allow the Sapphire to run for a short period so that it creates the CFG###.log file. Edit this file to reflect the needed changes. If you are not familiar with JSON format, you may wish to review the details at https://en.wikipedia.org/wiki/JSON. Only the items you wish to change need to be retained. The file should be saved on the micro SD card and named **config.jsn**. For the example below, we will change state-of-health log interval.

The file config.jsn is shown below

```
{
  "status": {
          "sohPeriod": 10
  }
}
```

Run the Sapphire a short time and then examine the output CFG###.log file to confirm the change. If the format of the config.jsn file was incorrect, it will be ignored and the CFG###.log file will show no change.

## Resetting the configuration to firmware defaults

It may be useful to reset the configuration to the firmware defaults. This is done by depressing the ATTN button during startup when the 3 LEDs start their sequence just before the end red LED starts a fast flashing to wait for a console connection. The button must be held down for at least two seconds and released before all three LEDs are illuminated. I typically push the button when the green and end red LEDs are on and the middle LED is off.

Confirm the configuration was set back to default by examining the CFG###.log file.

# Connecting to the micro USB

The micro USB connector provides 4 functions

1. It allows power to be provided from the USB port rather than batteries if the power switch is off. This allows certain functions to run without depleting the battery pack, or allows extended use should an external USB battery bank be used

2. It provides USB disk drive emulation during startup to allow configuration files to be quickly inspected or modified and allows small amounts of miniseed data to be downloaded to a computer without needing to extract the micro SD card.
3. It provides a serial connection during firmware startup that will provide diagnostic and state of health information to help isolate problems.
4. It provides access to the boot loader to allow firmware to be replaced or updated.

## Power Options

If the power switch is in the off position or the battery pack voltage is below 5 volts, power will be taken from the USB connector. Peak power spikes may be as high as 150mA which should be available from most USB connections without problems.

One can also run from an external USB power pack, however, for many power packs the Sapphire power drain is so small that the pack may switch the port off. In addition, the Sapphire will have no way of knowing that a pack is near exhaustion and therefore may be switched off with little warning. More sophisticated power options are described later in the document, but external power USB may be of use in some conditions.

## Disk Drive Emulation

If the Sapphire is off and is then plugged into a computer, it will begin USB disk drive emulation. Depending upon the amount of data on the micro SD card, it may take a few minutes to initialize. While in disk drive mode the green LED will remain on. The end red LED will be on when there is pending activity, such as the initial read, or if files are changed or erased.

Disk drive emulation ends when the drive is ejected, either with a command on the attached computer (preferred) or when the ATTN button on the Sapphire is depressed and released. Normal initialization for acquisition begins at that point.

While it is possible to transfer files using the USB connection, this will be much slower than extracting the SD card and using a SD card reader. The USB drive emulator is most convenient for editing a config.jsn file, for deleting old files, for examining the CFG log file or for downloading one or two data files.

## Serial Monitor Mode

When in normal acquisition, the USB port has a serial port definition and emits a number of program state, diagnostic, and state-of-health messages. These may be viewed with serial monitor programs. The Arduino IDE is one such package that includes the necessary serial port drivers and monitor software.

## Bootloader Mode

If the reset button is depressed twice with about ½ second between presses, the Sapphire will go to bootloader mode. The end red LED will slowly dim and brighten and the GACBOOT disk drive will be available to the connected computer.

# Loading new firmware

Loading new firmware into the Sapphire requires a computer with a micro-USB cable and a firmware update in *.uf2 format. With the power switch in the off position, plug the usb cable into the Sapphire connector, then press the RESET switch twice with about ½ second between presses. Too long or too short of a time will cause the boot process to abort. If the timing is correct, the edge red LED will stay on in a slowly wavering glow. If you elect to abort the load, simply press the reset button once.

Once in boot mode, the Sapphire will appear as a USB disk drive named GACBOOT. There will be 3 files present.

INFO_UF2.TXT simplify identifies the current boot loader

INDEX.HTM points to this Sapphire reference page.

CURRENT.UF2 is a copy of the currently running firmware. You can copy this file to your local computer if you want to save a copy of what is currently running (often useful if you find out you need to revert to an older version).

To load new firmware, simply copy the desired *.uf2 file onto the GACBOOT disk, usually by dragging the file over to the disk.

## Updating the boot loader

The boot loader rarely needs to be upgraded. When it is necessary, simply drag the update-bootloader*.uf2 file on the GACBOOT disk. Note that doing so may erase the acquisition firmware and require an acquisition firmware reload as well.

## Updating Sapphire Acquisition firmware

Sapphire software release packages may be found at
https://github.com/ChrisHayward-dev/SapphireReleases.git

You should use the most current version UNLESS you've discovered errors and need to revert or you are deploying an array or replicating an experiment and wish to maintain a consistent firmware release.

## Using Test Firmware

Included in the releases are various utilities and diagnostic programs that we've found useful from time to time. Notes accompanying them are brief and these will generally only be useful for advanced users that are diagnosing problems. They are included for completeness.

### Test_DHLR

The DHLR sensor is the differential microbarograph sensor that measures the pressure between one port and a backing volume. In some cases the backing volume has an additional unplanned leak or the backinig volume needle has become clogged. This program simply dumps the differential readings to the serial port. We use it with the Arduino Serial Plot function to look at real time data as adjustments are made to the backing volume plumbing.

### GPS_Serial

The GPS_Serial firmware is a two-way Serial bridge from the USB serial port to the GPS serial port. It allows use of the Ublox U-Center software to examine and operate the GPS receiver. This can be used if one suspects that there has been an antenna failure of the GPS.

## Instrument Characteristics

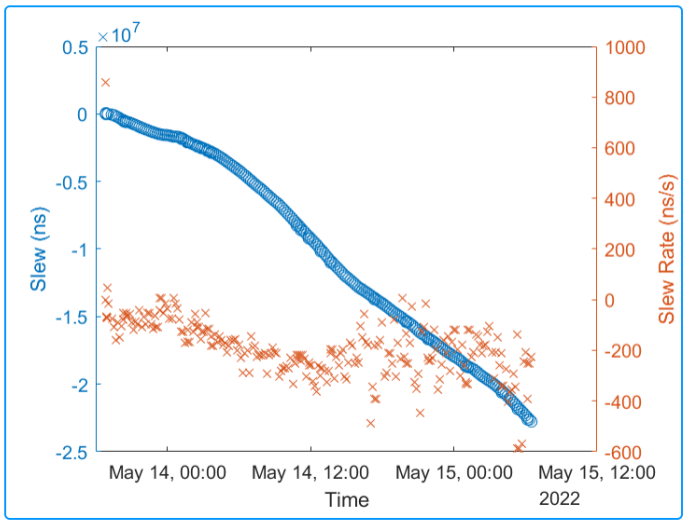
### Clock drift

The local oscillator has a manufacturer rated clock drift of 2ppm (or 2 uSec for every second). Provided temperatures do not shift radically from one GPS fix to the next, we usually trim to better than 1ppm. Thus, the maximum drift between GPS fixes is given by the following table.

*GPS drive estimates assuming 1ppm clock stability*

| **Maximum Drift in Samples** | | | |
|---|---|---|---|
| | GPS Period (min) | | |
| Sample Rate | **60** | **120** | **240** |
| **16** | 0.12 | 0.23 | 0.46 |
| **32** | 0.23 | 0.46 | 0.92 |
| **64** | 0.46 | 0.92 | 1.84 |
| **128** | 0.92 | 1.84 | 3.69 |

If the experiment is such that highly accurate sample time is not needed (for example a network interested in microbarom signals), maximum run times will be attained with the GPS run a signal time, at startup, and then left off for the duration of the acquisition.




*Observed clock drift on a sample system. The uncorrected clock drift resulted in about 15mS/day, about 0.2 ppm stability, 10x better than the nominal estimate. Actual use in the field will depend upon daily temperature variations. Clock stability can be improved by insulating the enclosure with simple materials.*

## Power consumption

Estimating Sapphire total run time until the power is exhausted is prone to large errors because much of the power budget depends upon how long it takes the GPS to acquire a lock and this depends not only on the specifics of the antenna on a particular unit (some units have better antennas than others), but also on the placement of the Sapphire and on weather conditions.

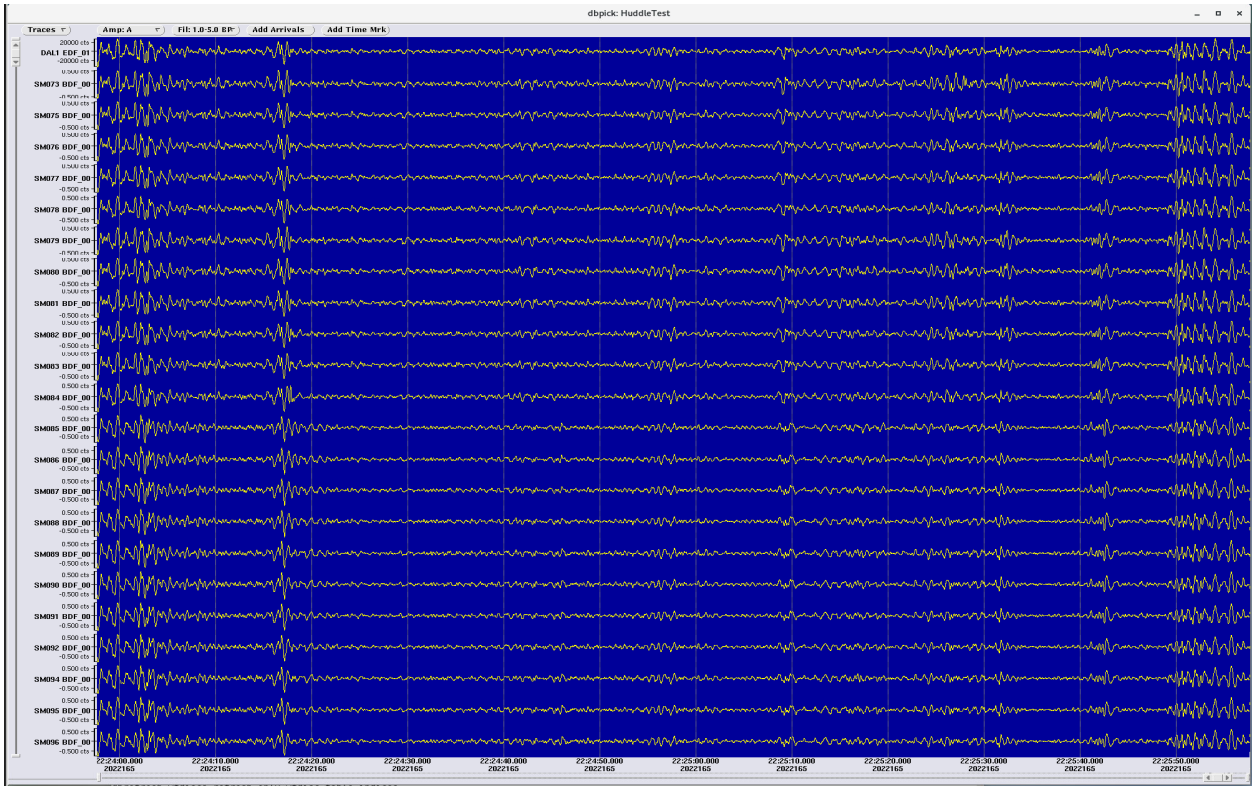
*Estimated power and run time durations for various configurations*

| | | Power for acquisition only (no GPS) | | Days w/ GPS one-shot | Days at GPS cycle times | | |
|---|---|---|---|---|---|---|---|
| Channels | Sample Rate | mA avg @5V | mW | | 60 | 120 | 240 |
| int32 1 | 16 | 1.8 | 9 | 58.3 | 25.30 | 35.29 | 43.98 |
| float 1 | 16 | 1.99 | 9.95 | 52.7 | 24.19 | 33.18 | 40.74 |
| 2 | 16 | 2.34 | 11.7 | 44.8 | 22.39 | 29.87 | 35.87 |
| 3 | 16 | 2.36 | 11.8 | 44.5 | 22.29 | 29.70 | 35.62 |
| 4 | 16 | 2.15 | 10.75 | 48.8 | 23.33 | 31.58 | 38.36 |
| 4 | 32 | 1.81 | 9.05 | 58.0 | 25.24 | 35.18 | 43.80 |
| 4 | 64 | 2.74 | 13.7 | 38.3 | 20.63 | 26.82 | 31.56 |
| 1 | 128 | 3.13 | 15.65 | 33.5 | 19.16 | 24.39 | 28.24 |
| 2 | 128 | 3.89 | 19.45 | 27.0 | 16.83 | 20.73 | 23.45 |
| 3 | 128 | 4.04 | 20.2 | 26.0 | 16.43 | 20.13 | 22.69 |

To a smaller extent, the power will depend upon the aggregate sample rate (number of channels times the channel sample rate). Some estimates are given below for an average GPS lock time of 7 minutes and a battery cutoff at 90% of the maximum capacity. Lowest power results from recording in int32 format at 16SPS with only the BDF channel and no GPS, resulting in an estimated run time of 58 days.

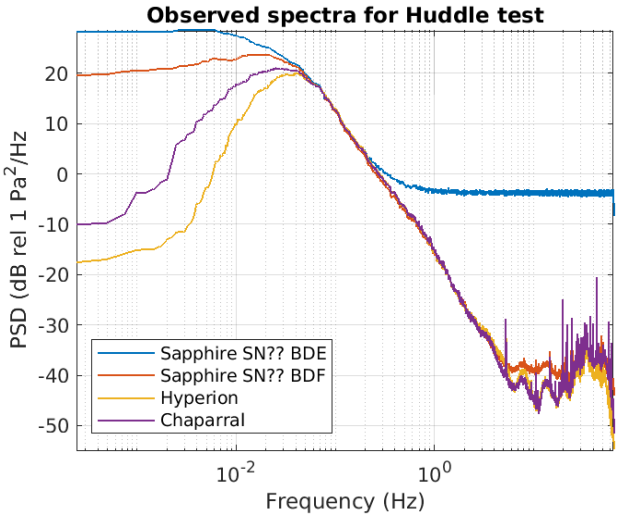
## Instrument Characteristics

Vendor data sheets provide a limiting description of the two primary sensors. The barometer is a BMP388 or BMP390 depending upon the Sapphire. The differential pressure sensor is a DLHR F50D gauge. Both sensors include internal factory calibration values. However, once the systems are integrated into the Sapphire, additional noise factors and characteristics may come into play. For this reason, we suggest huddle testing units before and after any critical data acquisition experiments.




*The above figure shows a typical huddle test of the DLHR sensors for most of the units in one box. A reference trace from a Hyperion sensor is shown as the first trace (labeled DAL1). Traces have been bandpass filtered 1-5 Hz.*

The nominal pressure range from the DLHR F50D is ±125Pa (0.5 in water). Pressures beyond this range result in a NaN output. Pressures beyond ±25kPa may permanently shift the calibration of the unit and pressures beyond ±75kPa may burst the diaphragm.

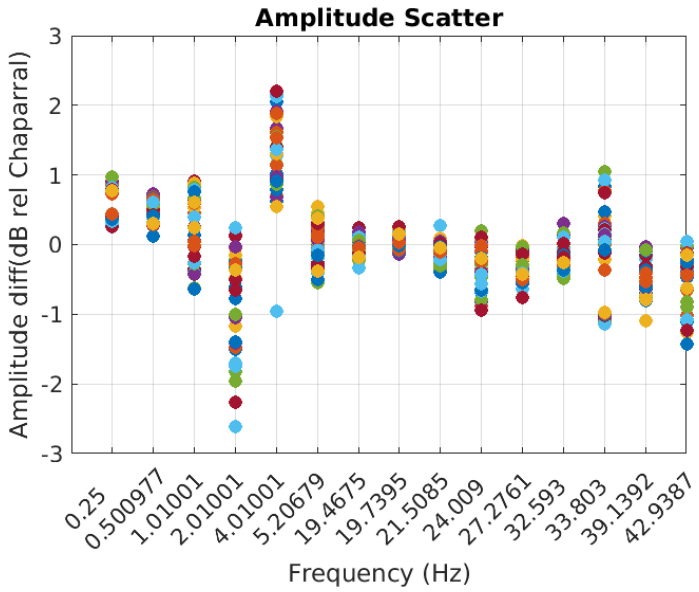



*Sample spectra for one Sapphire compared to a Hyperion and Chaparral*

For consistency among experiments, we typically declare one sensor (in this case a Hyperion) as a reference and compare all units against that in a typical operating environment (in this case a continually operating campus infrasound site). If such a site is not easily accessible, the internal BMP388 barometer can be compared to the DLHR sensor in the 0.01 to 0.1 Hz band.

The observed spectra for one example Sapphire is shown below. The blue curve is the spectra for the BDE, BMP388, barometer channel. The system self noise begins to dominate the spectra at frequencies above 0.1 Hz. Below 0.1 Hz to about 50 seconds, the BDE and BDF channels should match. Above 0.1 Hz the Hyperion and Chaparral spectra are identical and overlay the Sapphire up to about 3 Hz. At this site, beyond 3Hz Sapphire instrument noise level interferes with low amplitude areas, but the Sapphire does resolve most of the line spectra that are above the self noise of about -40dB rel 1 Pa2/Hz. DHLR self noise depends upon the sample rate and low rates reduce sensor self noise.

## Amplitude scatter

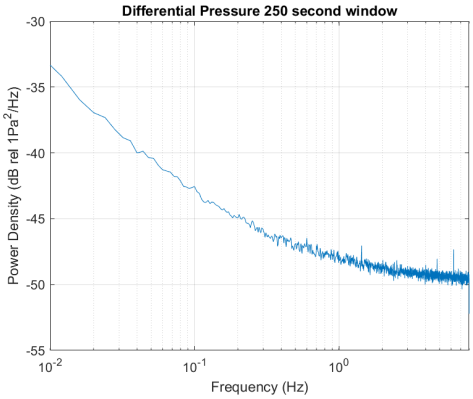


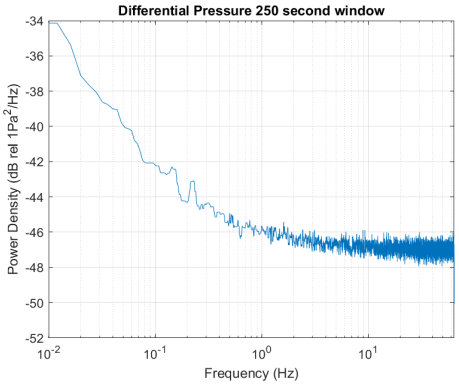

Thirty-eight Sapphire units were compared with the Chaparral at specific frequencies (the arbutrary 0.25 to 4.0 Hz and the 10 highest power peaks for line spectra in the 5-60Hz band. For good SNR peaks (those above 5 Hz), measured amplitudes are within +/- 1dB. For the arbitrary frequencies in low SNR conditions, differing noise cause spectral amplitudes to scatter more broadly, up to 3dB. Below 0.5 Hz,

units are within 1dB of each other and this is the expected range for any signals with good SNR.

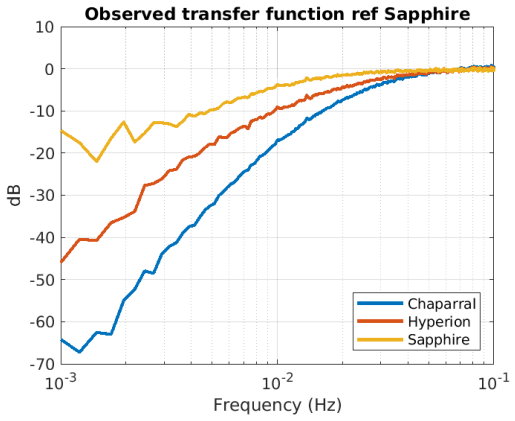
## DLHR Response

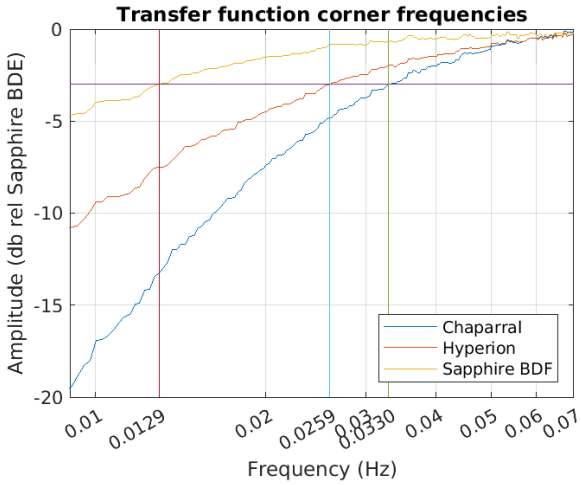
Noise levels for the DLHR sensor are dependent upon the sample rate, the ambient temperature stability, and the system power mode. The best noise performance requires the processor to run in low power mode. This only occurs when the USB conneciton is off..

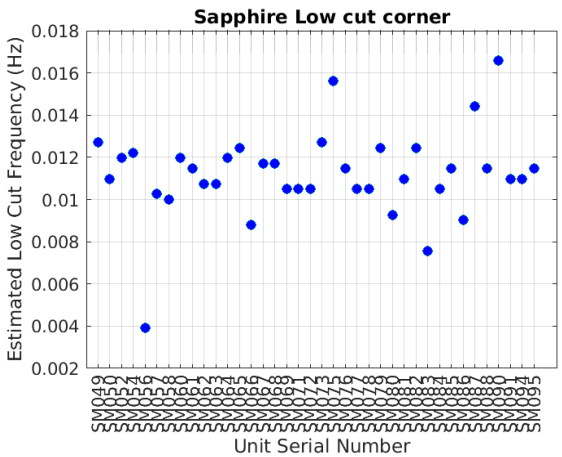



At 16SPS, the high frequency instrument noise is roughly -50 dB rel to 1 Pa^2/Hz. At 128 SPS the noise is slightly higher..



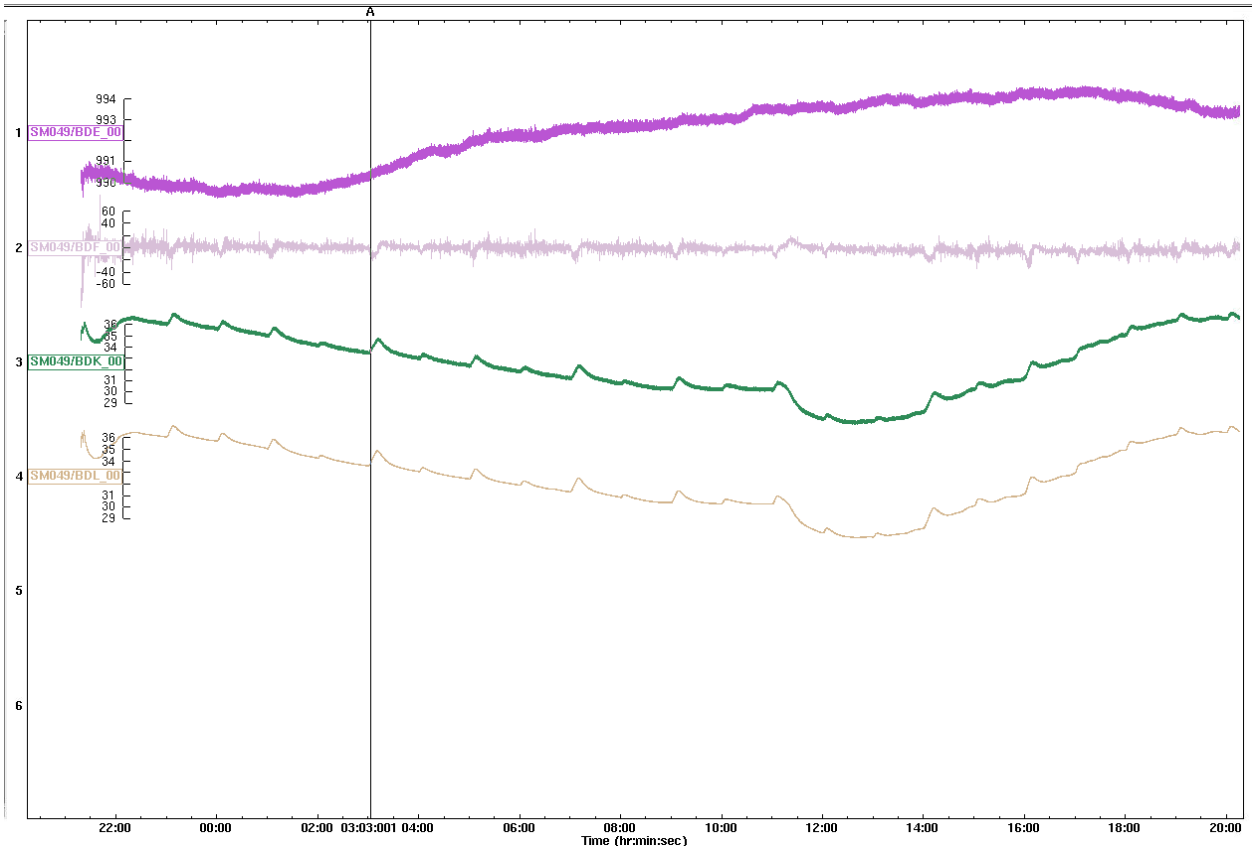




The frequency response of the DLHR differential sensor can be determined by comparing it with the BMP388 barometer for frequencies below 0.1 Hz. The figure shows the Chaparral Model 2.3 and Hyperion 3000 sensors for comparison. The first low cut corner is determined as the -3dB point. Because the low cut corner is dependent on details in the leak capillary and this results in some scatter at the corner.

The DLHR sensor and backing volume are sensitive to operation of the GPS module, most

likely a temperature related effect. Fortunately most temperature changes are slow and outside the band of interest to infrasound research. This appears as a long period pulse on the BDF channel. It can be recognized as a GPS induced noise by comparison to the BDF barometer channel which will not show the same feature. In addition, it will occur only during GPS activity as shown on the LOG channel and usually the temperature effect will be recorded on the BDK and BDL channels (if they are being recorded).




A 0.01 Hz low cut filter will eliminate this noise. Alternatively one can use the BDE channel for observations below 0.1 Hz.

### Temperature Sensors

Both the BMP388 and DLHR sensors include temperature sensors that are used in part to temperature compensate pressure readings. The BMP388 (BDL) channel seems to be a lower noise measurement.

The BDK channel (DLHR) temperature is noisier with a typical ±0.05 ℃ noise while the BDL channel (BMP388) is lower noise with a typical ±0.003 ℃ noise level.

# Modifications

## Firmware Supported Modifications

The Sapphire fully supports a few minor hardware modifications. These have been useful during testing and may have some use for advanced users who want to experiment with different deployments.

### External Power

The Sapphire uses a buck regulated power supply capable of input voltages between 3.5 and 15 volts. A standard 12-volt battery may be connected in place of the AA battery pack to provide extended run times. This may require drilling a hole in the case for the power connector. In such use, if a hose connection is in use, the connector should be made airtight such that it does not introduce an unwanted acoustic connection into the Sapphire.

### External power switch

The PCB includes a 3 pin connection at J502 that will support a second power switch. Shorting the left two pins of J502 will force the power supply off. Adding a magnetic reed switch would allow one to quickly turn off or on the Sapphire without opening the box. This may be of interest in difficult deployments.

### Plug in sensors

The Sapphire supports a plug in LIS3DH accelerometer on J504 to record vertical acceleration on a 5th channel. It will also support a plug in BMP388 or BMP390 sensor on J506. Both of these sensors are also supported on the STEMMA connector at J507.

### External GPS

The current Sapphire version uses a Beitian 220 GPS, a module designed for small drones. Other Ublox modules may be used and connected via J703. This will allow modules with PPS outputs to be easily integrated.

## Hardware Supported Modifications

The Sapphire includes several accessible hardware bus connections that are available for future firmware or modifications. Some may be of interest to advanced uses.

### Plug in I2C modules

The STEMMA connector at J507 connects to the I2C bus. Since this bus is shared with the other sensors, care must be used to avoid interfering with normal sampling.

### Analog Microphone A2D

The J504 connector may be reconfigured to support an analog microphone A2D. This has been prototyped, but not produced. It is designed to support higher frequency sampling at lower noise levels.

### External high capacity FRAM

Connector J501 supports Adafruit high capacity FRAM on the SPI bus. This is for future expansion.

## Troubleshooting

The Sapphire is a relatively simple device, so most problems are either simply solved or impossible to fix in the field. For that reason, it's recommended that when an experiment is planned, there are a few systems kept in reserve in case one or more fail to start. If everything works, these extra systems can be deployed in parallel at critical sites, or used for secondary sites.

| Symptom | Cause |
|---|---|
| When power cycled no LEDs ever turn on | Check battery condition. The system should operate when the battery pack voltage is above 4.5 volts. Install fresh batteries, verify that each cell is firmly seated and that all polarities are correct. Also check the battery pack cable. Sometimes these have been crimped between the top and bottom cutting the conductors inside the insulation. |
| End red LED rapid flashing during startup | The system is awaiting a USB console connection. This is normal and will eventually timeout. |
| All three LEDs on continuously for over a minute | Bus lockup. Turn the unit off for at least 30 seconds before powering back on |
| Middle red LED on during acquisition | GPS hasn't acquired a lock. If it has been more than 15 minutes, the GPS has given up and will not try again for 2 hours. Try moving the unit to a new location and restarting by pushing the reset button.<br><br>If this has occurred after several hours of acquisition, but was ok at first, this indicates that at the last attempt, GPS was not found. The data is likely ok, but there may be some clock drift depending upon how long the GPS was unsuccessful. |
| All three lights flash 4 times, then a light pattern flashes. This repeats once every few seconds. | This is a runtime error. The system has shut down completely. The pattern is an octal code with the green LED the low |

| | |
|---|---|
| | order bit (green = 1, middle red=2, end red=4. A zero is 4 flashes of all the lights.<br><br>Read the code, and consult the table in this document for more information on the error.<br><br>The most common non-firmware errors are a low battery or a SD card write error.<br><br>Firmware errors should be passed along to us (hayward@smu.edu) for investigation. If you have modified the recording parameters using a custom config file, you may want to revert back to default parameters and verify the error is not related to a bad choice of recording parameters. |

## LED Error Codes

Error codes reflect only firmware version 5.3.0. If you have a different firmware version, codes may be slightly different as source code lines move around when new sections are inserted.

Error codes will blink repeatedly on the LEDs. You can force an error by starting the unit and letting it run for a few minutes, then extracting the SD card while the unit runs (this may corrupt the file system requiring that the card be reformatted). The unit will flash an error code about SD card errors as found in the following tables

Codes are octal numbers. A zero is represented by 4 quick flashes with all LEDs. Other digits are the LEDs as positions. For example a single green LED is 1, the middle red LED is 2, and the edge red LED is 4. Digits 5-7 are represented by multiple LEDs,

| LED | DIGIT |
|---|---|
| FFF | 0 |
| --G | 1 |
| -R- | 2 |
| -RG | 3 |
| R-- | 4 |

| | |
|---|---|
| R-G | 5 |
| RR- | 6 |
| RRG | 7 |

| Octal | Line | Source File | Error |
|---|---|---|---|
| 0011 | 0009 | ms_lib.cpp | |
| 0014 | 0012 | SM_diag.h | |
| 0017 | 0015 | SM_diag.h | |
| 0020 | 0016 | SM_diag.h | |
| 0022 | 0018 | diag.h | |
| 0025 | 0021 | diag.h | |
| 0031 | 0025 | SM_miniseed_lib.h | |
| 0032 | 0026 | SDFatDemo.ino | No SD Card! |
| 0035 | 0029 | SDFatDemo.ino | Unable to open SD card |
| 0051 | 0041 | SM_sdcard.cpp | Unable to init the SD Card!! |
| 0055 | 0045 | SM_sdcard.cpp | readInfo failed |
| 0061 | 0049 | SM_sdcard.cpp | readCID failed |
| 0063 | 0051 | Mini_Logger5.00.ino | Unable to parse JSON version override |
| 0074 | 0060 | SM_miniseed2.ino | SD Init failed with 0x0000 0 |
| 0074 | 0060 | demo_miniseed2.ino | SD Init failed with 0x0000 0 |
| 0100 | 0064 | AAMiniLogger.h | |
| 0100 | 0064 | SM_miniseed2.ino | Can't find valid FAT partition. |
| 0100 | 0064 | demo_miniseed2.ino | Can't find valid FAT partition. |
| 0103 | 0067 | AAMiniLogger.h | |
| 0105 | 0069 | AAMiniLogger.h | |
| 0111 | 0073 | SM_miniseed2.ino | Unable to open file! |
| 0111 | 0073 | demo_miniseed2.ino | Unable to open file! |
| 0117 | 0079 | SM_sdcard.cpp | readInfo failed |
| 0123 | 0083 | SM_sdcard.cpp | readCID failed |
| 0126 | 0086 | SM_sdcard.cpp | Unable to sd.begin |
| 0126 | 0086 | SM_sdcard.cpp | readInfo failed |
| 0132 | 0090 | SM_sdcard.cpp | readCID failed |
| 0137 | 0095 | SM_sdcard.cpp | SD card file:  unable to open for write |
| 0137 | 0095 | Threads.ino | LOG overflow on DLHR header |
| 0141 | 0097 | Threads.ino | LOG overflow on BMP header |
| 0142 | 0098 | SM_usbdrive.cpp | Unable to init SD card! |
| 0143 | 0099 | Threads.ino | LOG overflow on ACC header |
| 0150 | 0104 | SM_sdcard.cpp | Unable to sd.begin |
| 0160 | 0112 | AAMiniLogger.h | |
| 0160 | 0112 | SM_sdcard.cpp | SD card file:  unable to open for read |
| 0161 | 0113 | SM_sdcard.cpp | Unable to init the SD Card!! |
| 0165 | 0117 | SM_sdcard.cpp | Unable to run sd.begin |
| 0166 | 0118 | AAMiniLogger.h | |
| 0167 | 0119 | AAMiniLogger.h | |

| | | | |
|---|---|---|---|
| 0170 | 0120 | SM_sdcard.cpp | Unable to init the SD Card!! |
| 0171 | 0121 | AAMiniLogger.h | |
| 0172 | 0122 | AAMiniLogger.h | |
| 0172 | 0122 | Sensors.ino | IN219 bus hung |
| 0174 | 0124 | AAMiniLogger.h | |
| 0174 | 0124 | SM_sdcard.cpp | Unable to run sd.begin |
| 0201 | 0129 | Sensors.ino | IN219 bus hung |
| 0204 | 0132 | SM_sdcard.cpp | Card Format Failed! |
| 0207 | 0135 | Sensors.ino | DLHR Not detected! |
| 0213 | 0139 | SM_sdcard.cpp | Card Format Failed! |
| 0217 | 0143 | Sensors.ino | Unknown Sample Rate |
| 0226 | 0150 | SM_sdcard.cpp | erase failed for blocks 0 to 0 |
| 0226 | 0150 | Sensors.ino | DLHR Not detected! |
| 0235 | 0157 | SM_sdcard.cpp | erase failed for blocks 0 to 0 |
| 0236 | 0158 | Sensors.ino | Unknown Sample Rate |
| 0262 | 0178 | Mini_Logger5.00.ino | Inconsistent patch level in SM_config defaults!!! |
| 0265 | 0181 | Mini_Logger5.00.ino | Unable to parse JSON version override |
| 0271 | 0185 | Mini_Logger5.3.ino | Inconsistent patch level in SM_config defaults!!! |
| 0274 | 0188 | Mini_Logger5.3.ino | Unable to parse JSON version override |
| 0313 | 0203 | SM_sdcard.cpp | Unable to sd.begin |
| 0321 | 0209 | SM_sdcard.cpp | Unable to remote file!! SD card format problem? |
| 0326 | 0214 | SM_sdcard.cpp | SD card file:  unable to open for write |
| 0327 | 0215 | ms_lib.cpp | Buffer overfloat for 0 char |
| 0337 | 0223 | SM_sdcard.cpp | Unable to sd.begin |
| 0340 | 0224 | ms_lib.cpp | Buffer overflow for 0 char |
| 0347 | 0231 | SM_sdcard.cpp | SD card file:  unable to open for read |
| 0353 | 0235 | ms_lib.cpp | Buffer overflow for 0 char |
| 0355 | 0237 | SM_sdcard.cpp | Unable to sd.begin |
| 0362 | 0242 | Sensors.ino | Unknown Sample Rate |
| 0363 | 0243 | SM_sdcard.cpp | Unable to remote file!! SD card format problem? |
| 0364 | 0244 | Sensors.ino | Error accessing BMP sensor at 0x00 |
| 0370 | 0248 | SM_sdcard.cpp | SD card file:  unable to open for write |
| 0376 | 0254 | ms_lib.cpp | Buffer overflow for 0 char |
| 0401 | 0257 | SM_sdcard.cpp | Unable to sd.begin |
| 0402 | 0258 | Sensors.ino | Unknown Sample Rate |
| 0402 | 0258 | Threads.ino | Seamphore recording timeout |
| 0404 | 0260 | Sensors.ino | Error accessing BMP sensor at 0x00 |
| 0411 | 0265 | SM_sdcard.cpp | SD card file:  unable to open for read |
| 0411 | 0265 | ms_lib.cpp | Buffer overflow for 0 char |
| 0424 | 0276 | ms_lib.cpp | Buffer overflow for 0 char |
| 0437 | 0287 | ms_lib.cpp | Buffer overflow for 0 char |
| 0450 | 0296 | Threads.ino | Seamphore recording timeout |
| 0453 | 0299 | ms_lib.cpp | Buffer overflow for 0 char |
| 0474 | 0316 | SM_miniseed_lib.cpp | Unable to write header |
| 0520 | 0336 | Sensors.ino | Unknown Sample Rate |
| 0522 | 0338 | SM_miniseed_lib.cpp | Unable to write sample 0 of |
| 0524 | 0340 | Sensors.ino | Error accessing LIS2DH12 |
| 0532 | 0346 | Mini_Logger5.00.ino | Unable to reset I2C bus!! |
| 0537 | 0351 | Threads.ino | Recording Task is Blocked! |

| | | | |
|---|---|---|---|
| 0540 | 0352 | Sensors.ino | Unknown Sample Rate |
| 0544 | 0356 | Sensors.ino | Error accessing LIS2DH12 |
| 0600 | 0384 | Mini_Logger5.00.ino | cd.format MUST be MS_FLOAT and match mseed constructor! |
| 0602 | 0386 | Threads.ino | Low Voltage cutoff |
| 0603 | 0387 | Threads.ino | Low Voltage cutoff |
| 0612 | 0394 | Threads.ino | Recording Task is Blocked! |
| 0620 | 0400 | SM_ublox.cpp | Restart Message! |
| 0621 | 0401 | Mini_Logger5.3.ino | Unable to reset I2C bus!! |
| 0622 | 0402 | Threads.ino | Recording Task is Blocked! |
| 0646 | 0422 | Mini_Logger5.3.ino | Unable to reset I2C bus!! |
| 0660 | 0432 | Threads.ino | Low Voltage cutoff |
| 0661 | 0433 | Threads.ino | Low Voltage cutoff |
| 0662 | 0434 | Threads.ino | Low Voltage Cutoff |
| 0671 | 0441 | Threads.ino | Sample Timeout |
| 0706 | 0454 | Mini_Logger5.3.ino | cd.format MUST be MS_FLOAT and match mseed constructor! |
| 0725 | 0469 | Threads.ino | Overrun 0 |
| 0731 | 0473 | Threads.ino | Less than 1 sample |
| 0735 | 0477 | Threads.ino | Sample Timeout |
| 0754 | 0492 | Threads.ino | Sample Timeout |
| 1014 | 0524 | Threads.ino | Sample Overrun! (0) |
| 1016 | 0526 | Threads.ino | |
| 1023 | 0531 | Threads.ino | Overrun 0 |
| 1032 | 0538 | Threads.ino | Less than 1 sample |
| 1037 | 0543 | Threads.ino | Sample Timeout |
| 1143 | 0611 | Threads.ino | Sample Overrun! (0 of 0) @ 0(RTCsampleCount) |
| 1145 | 0613 | Threads.ino | |

## Drawings and Schematics

These are available as separate PDF files

### Enclosure Drawings

### Circuit schematics

### PCB Layout

### PCB Copper