3-Space™ Sensor

Miniature Attitude & Heading Reference System (AHRS) With Pedestrian Tracking (EEPTS™)

User's Manual

User's Manual Revision 1.1 for
TSS v3 Hardware Revisions 3.0 - 3.1z

Yost Labs
630 Second Street
Portsmouth, Ohio 45662

https://YostLabs.com/
Phone: 740-876-4936

Table of Contents

Preliminary Notice:

- All contents are subject to change.
- For questions, clarifications, technical support, and/or any other assistance, please reach out to us with an email to techsupport@yostlabs.com or information@yostlabs.com or fill out the form on our website at https://yostlabs.com/contact/.
- The legacy versions of the 3-Space™ Sensor Suite (1.1.9c/1.1.8/3.0r7/1.0.6) are not compatible with our 3.0 sensors. A new version of the Suite is under active development. Please contact us if you do not have a copy of this software.
- If you have a feature request, please let us know. We may be willing to add the feature, or the feature may already be added and is currently undocumented.
- If a feature is not working as documented, please reach out to us with your sensor's serial number, firmware revision, and brief summary of the interaction. We may have updated firmware for you, or the documentation may be out of date.

1 Overview of the 3-Space™ Sensor

1.1 Sensor Introduction

The 3-Space™ Sensor is a miniature, high-precision, high-reliability, Attitude and Heading Reference System (AHRS). The 3-Space™ family includes sensors with a variety of communication methods, including USB, wireless, serial, I2C, and SPI. The Attitude and Heading Reference System (AHRS) uses triaxial gyroscope, accelerometer, and magnetometer (compass) sensing components in conjunction with advanced on-board filtering and processing algorithms to determine orientation relative to an absolute reference orientation in real-time.

Orientation can be returned in absolute terms or relative to a designated reference orientation. The gradient descent calibration process and high update rates increase accuracy and greatly reduce and compensate for sensor error. The 3-Space™ Sensor system also utilizes a dynamic sensor confidence algorithm that ensures optimal accuracy and precision across a wide range of operating conditions.

Building upon the robust foundation of the 3-Space™ Sensor, Yost Labs has developed the Extended Embedded Pedestrian Tracking System (EEPTS™). The EEPTS™ represents an evolution of the pedestrian tracking system included in previous versions of the sensors, incorporating proprietary algorithms that synergize with the onboard orientation filter and components to deliver unparalleled tracking accuracy. This military-grade tracking solution is engineered to operate in environments where GPS is unreliable or unavailable.

The 3-Space™ Sensor unit features are accessible via a well-documented open communication protocol that allows access to all available sensor data and configuration parameters. Versatile commands allow access to raw sensor data, normalized sensor data, and filtered absolute and relative orientation outputs in multiple formats: quaternion, Euler angles (pitch/yaw/roll), rotation matrix, axis angle, & two-vector (forward/up).

1.2 Sensor Applications

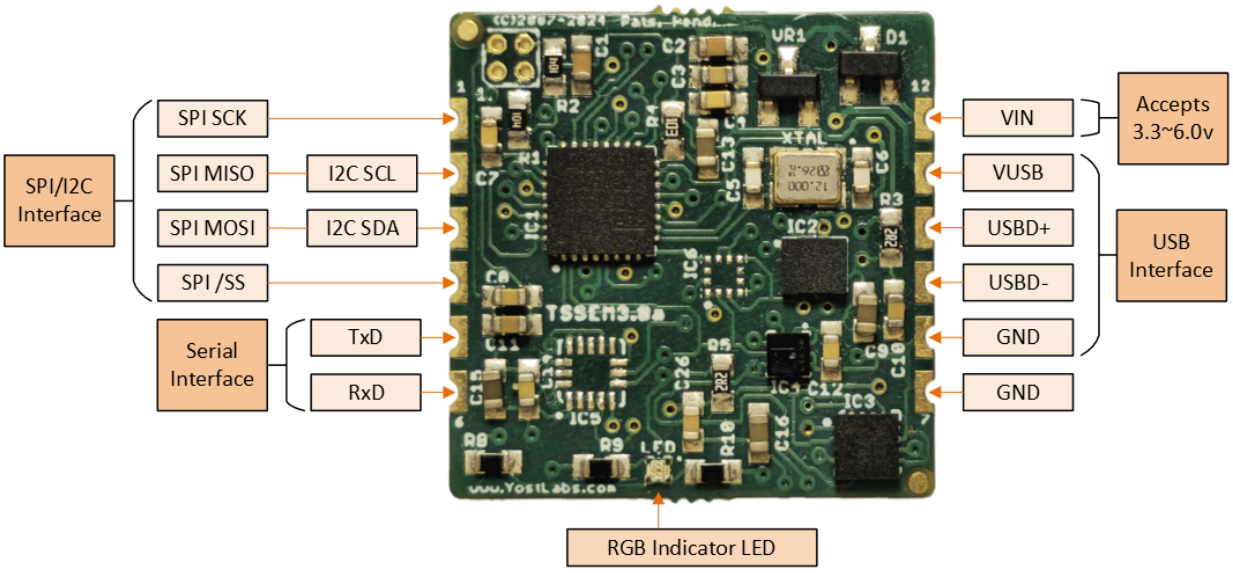
- Robotics
- Motion capture
- Positioning and stabilization
- Vibration analysis
- Inertial augmented localization
- Personnel / pedestrian navigation and tracking
- Unmanned air/land/water vehicle navigation
- Education and performing arts
- Healthcare monitoring
- Gaming and motion control
- Accessibility interfaces
- Virtual reality and immersive simulation

1.3 Sensor Types and Variations

Overview of Family Members

- Embedded Sensor
  - High performance embeddable sensor.
- Data Logger
  - Standalone data logging capabilities with BLE and GPS options.
- LX Embedded Sensor
  - Low cost embeddable sensor.

1.3.1 3-Space™ Embedded Sensor: TSS-EM3



1.3.1.1 Supported Communication

- USB
  - USB 2.0 Full Speed
- UART
  - Up to 4000000 baud (see section 3.5 for full list)
  - Data bits: 8
  - Parity: None
  - Stop bits: 1
- SPI
  - Up to 10 MHz
- I2C
  - Up to 3.4Mbps

1.3.1.2 Supported Features

- Small and light high-performance AHRS available at 23mm x 23mm x 2mm and only 1.3 grams
- Innovative military-grade pedestrian tracking system for use in GPS-denied environments
- Fast sensor update and filter rate allow use in real-time applications, including stabilization, virtual reality, real-time immersive simulation, and robotics
- Highly customizable orientation sensing with options such as tunable filtering, oversampling, and orientation error correction
- Advanced integrated orientation filtering allows sensor to automatically reduce the effects of sensor noise and sensor error
- Robust open protocol allows commands to be sent in human readable form (ASCII mode), or more quickly in machine readable form (binary mode)
- Orientation output format available in absolute or relative terms in multiple formats (quaternion, rotation matrix, axis angle, two-vector)
- Absolute or custom reference axes
- Access to raw sensor data
- Flexible communication options: SPI, USB 2.0, or asynchronous serial
- USB communication through a virtual COM port
- Castellated SMT edge pads provide secure SMT mounting and allow optional through-hole mounting
- Upgradeable firmware
- RGB status LED
- Programmable interrupt capability
- Development kit available
- RoHS Compliant
- +5v tolerant I/O signals

1.3.1.3 Embedded Sensor Specifications

General	
Part number	TSS-EM3 v3.0
Dimensions	23mm x 23mm x 2.2mm (0.9 x 0.9 x 0.086 in.)
Weight	1.3 grams (0.0458 oz)
Supply voltage	+3.3v ~ +6.0v
Power consumption[1]	30mA @ 5v
Communication interfaces	USB 2.0 FS, SPI, I2C, Asynchronous Serial
Filter update rate[2]	up to 2000 Hz with Extended Kalman AHRS up to 2000 Hz with QGRAD3™ AHRS up to 2000 Hz in IMU mode
Orientation output	absolute & relative quaternion, Euler angles, axis angle, rotation

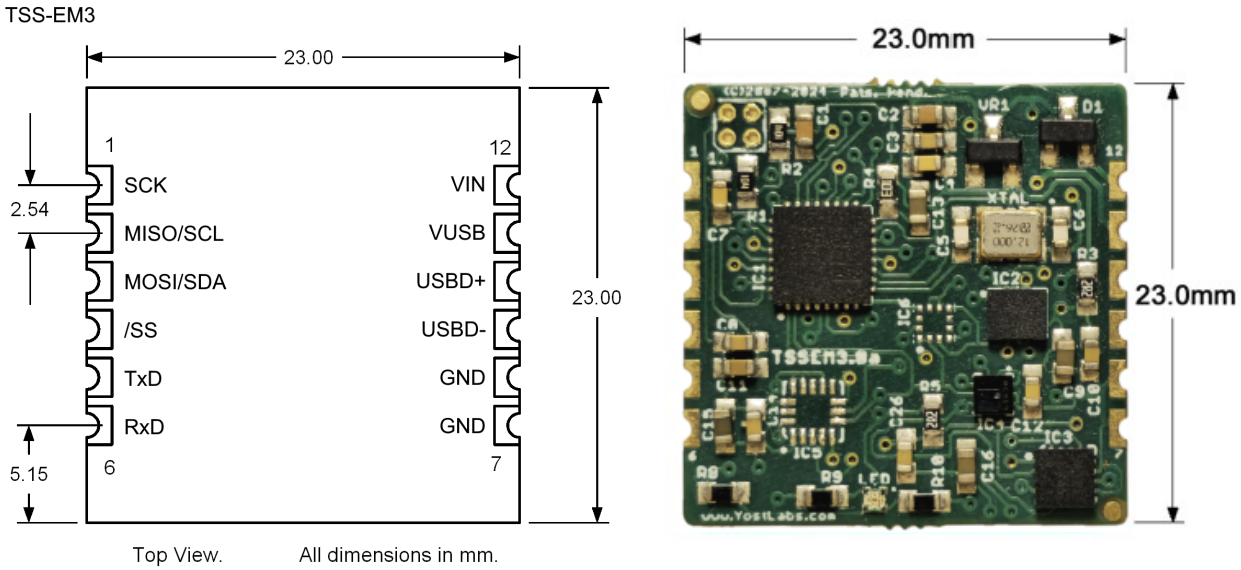
	matrix, two vector
Other output	corrected sensor data, normalized sensor data, temperature, pedestrian motion-path (longitude / latitude) offsets
SPI clock rate	10 MHz max
I2C clock rate	3.4 MHz max
Internal pull-up	50k Ω
Serial baud rate	4,800~4,000,000 selectable, default: 115,200
Shock survivability	5000g
Temperature range	-40°C ~ 85°C (-40°F ~ 185°F)
Sensor	
Orientation range	360º about all axes
Orientation accuracy[3]	±1º for dynamic conditions & all orientations
Orientation resolution	<0.08º
Orientation repeatability	0.085º for all orientations
Accelerometer scale	±2g / ±4g / ±8g / ±16g selectable (L), ±8g / ±16g / ±32g / ±64g selectable if included (M), ±100g / ±200g / ±400g selectable if included (H)
Accelerometer resolution	16-bit (L, M), 12-bit (H)
Accelerometer noise density	60µg/√Hz (L), 300µg/√Hz (M), 15mg/√Hz (H)
Accelerometer sensitivity	0.0000610352g/digit - 0.00048828g/digit (L), 0.000244140g/digit - 0.001953125g/digit (M), 0.049g/digit - 0.195g/digit (H)
Accelerometer temperature sensitivity	±0.01%/°C (L, M, H)
Gyroscope scale	±125/±250/±500/±1000/±2000/±4000 º/sec selectable
Gyroscope resolution	16-bit
Gyroscope noise density	0.005º/sec/√Hz
Gyroscope bias stability @ 25°C	1.5º/hr average for all axes
Gyroscope sensitivity	0.004375º/sec/digit for ±125º/sec, 0.140016º/sec/digit for ±4000º/sec
Gyroscope non-linearity	0.01% full-scale
Gyroscope temperature sensitivity	±0.005°/sec/C
Magnetometer scale	±8.0 Ga
Magnetometer resolution	18 bit
Magnetometer sensitivity	0.0625 mGa/digit

Magnetometer non-linearity	0.1% full-scale

1. Varies based on power Settings
2. Depends upon communication mode and filter mode.
3. Average value when calibrated.

1.3.1.4 Embedded Sensor Hardware Overview

The 3-Space™ Embedded v3 is packaged as a 23mm x 23mm x 2.2mm castellated edge SMT module. Alternatively, the module can be through-hole mounted by adding standard 0.1" header strips to the castellated edge pads.



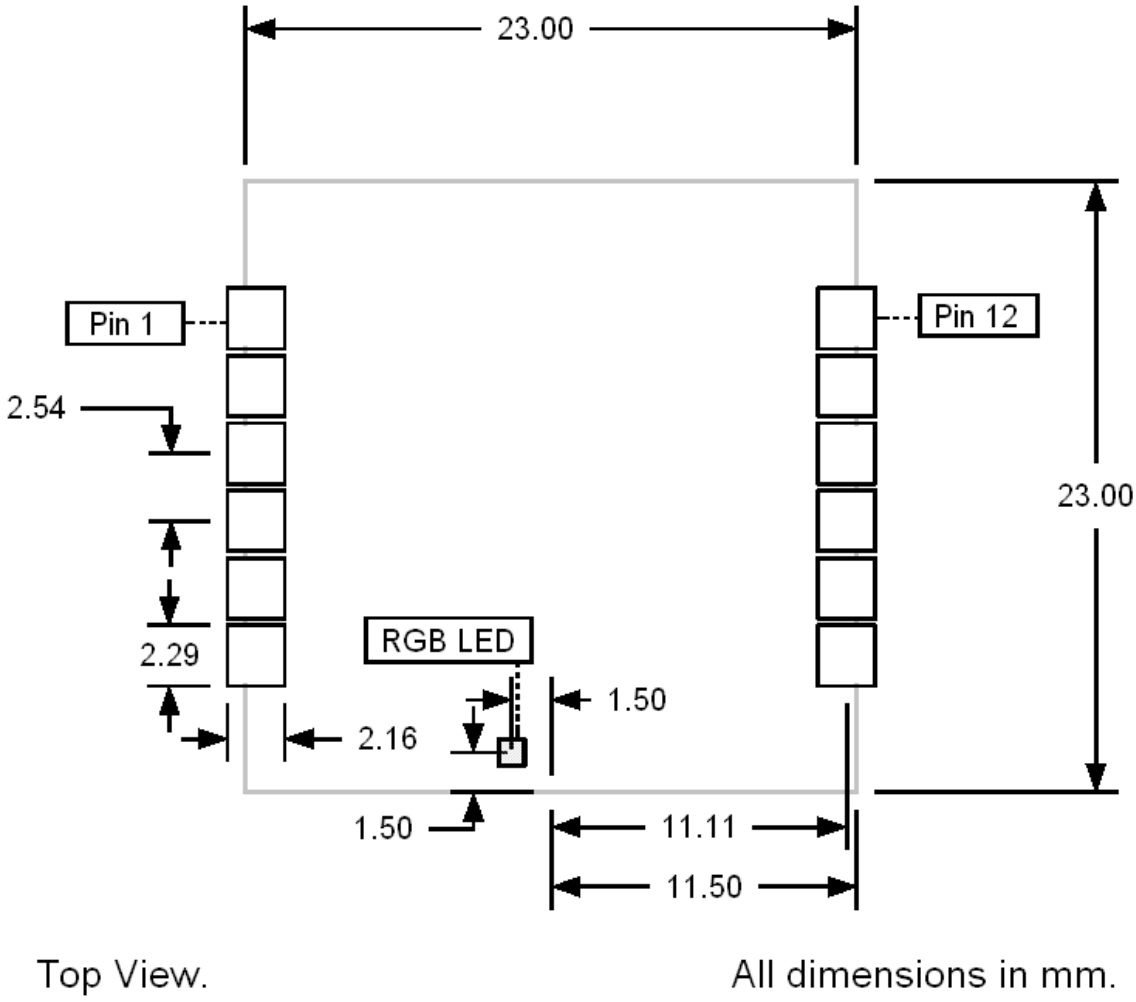
1.3.1.5 Embedded Sensor Pin Functions

Pad Number	Signal Name	Description
1	SCK	SPI Serial Clock. Input to Module.
2	MISO	SPI Master In Slave Out. Output from Module. Can be configured to act as filter update Interrupt.
2	SCL	I2C Serial Clock Pin. Input to Module.
3	MOSI	SPI Master Out Slave In. Input to Module.
3	SDA	I2C Serial Data Pin. Input to Module & Output from Module.
4	/SS	SPI Slave Select. Active Low Input to Module
5	TxD	UART Asynchronous Transmit Data. Output from Module. Can be configured to act as filter update Interrupt.
6	RxD	UART Asynchronous Receive Data. Input to Module
7	GND	Ground. Only one ground pad must be connected.
8	GND	Ground. Only one ground pad must be connected. Commonly connected to USB supply ground
9	USBD-	USB Data Minus. Only requires connection during USB mode use.
10	USBD+	USB Data Plus. Only requires connection during USB mode use
11	VUSB	+5v USB Power Supply Input . Only requires connection during USB mode use.
12	VIN	Voltage Input +3.3v ~ +6.0v. Only required when USB power is not being used.

1.3.1.6 Embedded Sensor PCB Layout

PCB layout can affect the functionality and performance of the 3-Space™ Embedded module. Please follow the suggested SMT footprint.

TSS-EM - Suggested SMT Footprint

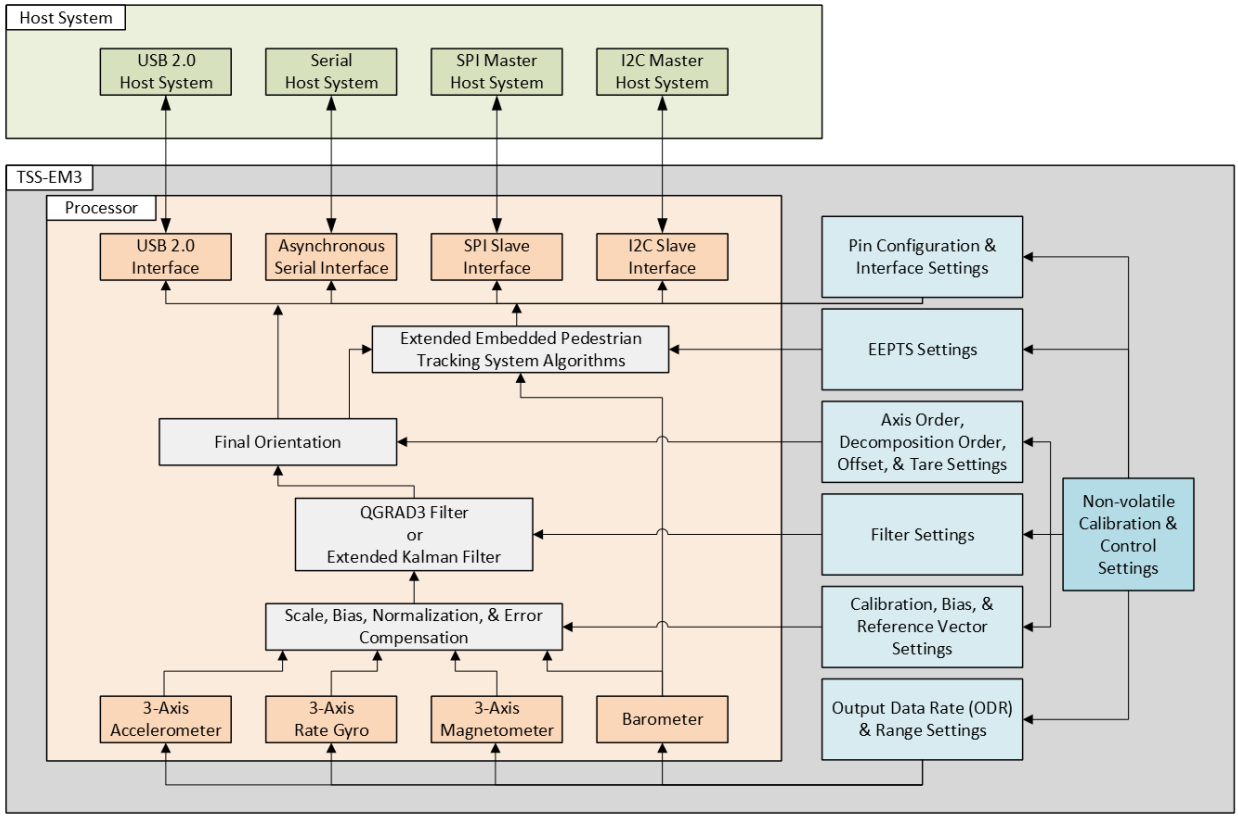


Additionally, please adhere to the following layout guidelines:

- Do not place magnetic components such as speakers and motors in close proximity to the module. The magnetic fields generated can adversely affect the performance of the magnetometer component.
- Do not place components containing ferrous metals in close proximity to the module. They may disturb Earth's magnetic fields and thus adversely affect the performance of the magnetometer component.
- Do not route high-current conductors or high-frequency digital signal lines in close proximity to the module. They may generate magnetic fields that adversely affect the performance of the magnetometer component.

- Do not reflow with the device on the bottom or side of a board. The module's components aren't glue-bonded to the module, so they may become dislodged if reflowed in non-upwards-facing orientations.
- Thoroughly test and characterize any PCB design that uses the module. Failure to test and characterize a system using the TSS-EM3 module may result in unforeseen performance consequences due to mechanical stresses, electrical characteristics, and component layout.

1.3.1.7 Embedded Sensor Block Diagram



1.3.2 3-Space™ Data Logger: TSS-DL3

1.3.2.1 Supported Communication

- USB-C
- BLE (On B models only)

1.3.2.2 Supported Features

- Small and light high-performance AHRS available at 42mm x 39mm x 14mm and only 20 grams
- Innovative military-grade pedestrian tracking system for use in GPS-denied environments
- Optional built-in GPS (On G models only)
- Fast sensor update and filter rate allow use in real-time applications, including stabilization, virtual reality, real-time immersive simulation, and robotics
- Highly customizable orientation sensing with options such as tunable filtering, oversampling, and orientation error correction
- Advanced integrated orientation filtering allows sensor to automatically reduce the effects of sensor noise and sensor error
- Robust open protocol allows commands to be sent in human readable form (ASCII mode), or more quickly in machine readable form (binary mode)
- Orientation output format available in absolute or relative terms in multiple formats (quaternion, rotation matrix, axis angle, two-vector)
- Absolute or custom reference axes
- Access to raw sensor data
- Flexible communication options: USB 2.0, or BLE
- USB communication through a virtual COM port

- Upgradeable firmware
- RGB status LED
- RoHS Compliant

1.3.2.3 Data Logger Sensor Specifications

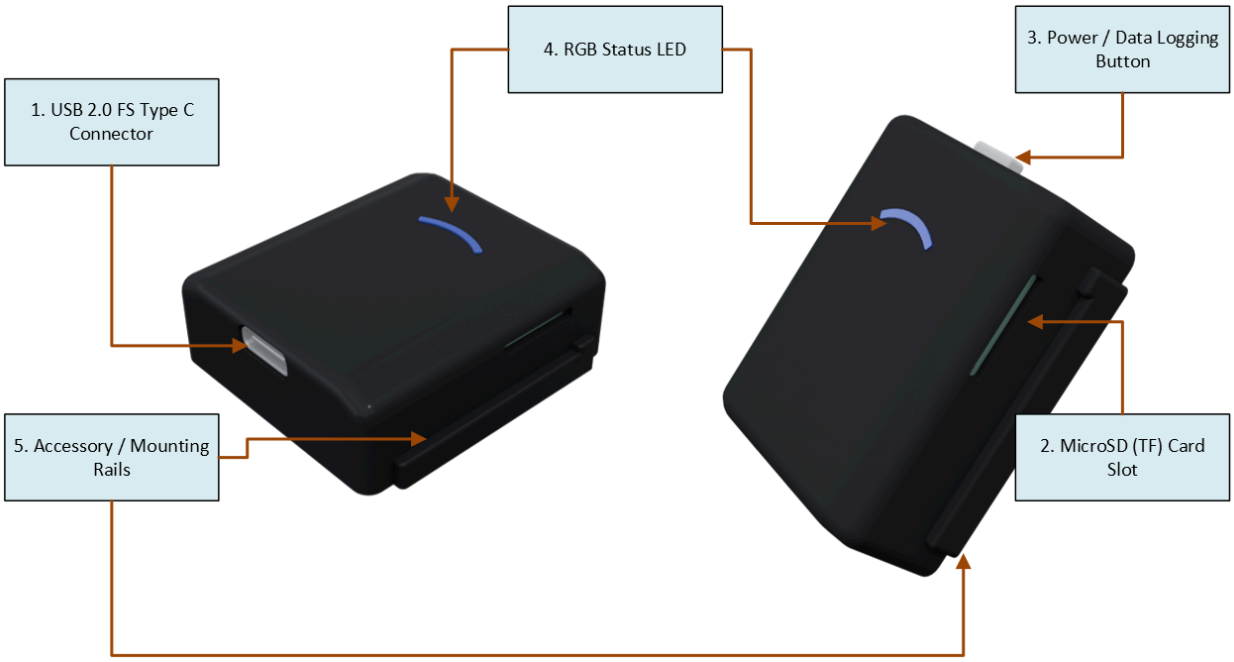
General	
Part number	TSS-DL3 v3.0
Dimensions	42mm x 39mm x 14-18.5mm (1.65 x 1.53 x 0.55-0.72 in.)
Weight	20-33 grams (0.7 - 1.16 oz)
Supply voltage	5v USB
Battery technology	Rechargeable Lithium-Polymer
Battery lifetime[1]	Up to 24 hours
Communication interfaces	USB 2.0 FS, BLE
Storage media	MicroSD card
Filter update rate[2]	up to 2000 Hz with Extended Kalman AHRS up to 2000 Hz with QGRAD3™ AHRS up to 2000 Hz in IMU mode
Orientation output	absolute & relative quaternion, Euler angles, axis angle, rotation matrix, two vector
Other output	corrected sensor data, normalized sensor data, temperature, pedestrian motion-path (longitude / latitude) offsets
Shock survivability	5000g
Temperature range	-40°C ~ 85°C (-40°F ~ 185°F)
Sensor	
Orientation range	360º about all axes
Orientation accuracy[3]	±1º for dynamic conditions & all orientations
Orientation resolution	<0.08º
Orientation repeatability	0.085º for all orientations
Accelerometer scale	±2g / ±4g / ±8g / ±16g selectable (L), ±8g / ±16g / ±32g / ±64g selectable if included (M), ±100g / ±200g / ±400g selectable if included (H)
Accelerometer resolution	16-bit (L, M), 12-bit (H)
Accelerometer noise density	60µg/√Hz (L), 300µg/√Hz (M), 15mg/√Hz (H)
Accelerometer sensitivity	0.0000610352g/digit - 0.00048828g/digit (L), 0.000244140g/digit - 0.001953125g/digit (M), 0.049g/digit - 0.195g/digit (H)
Accelerometer temperature sensitivity	±0.01%/°C (L, M, H)

Gyroscope scale	±125/±250/±500/±1000/±2000/±4000 °/sec selectable
Gyroscope resolution	16-bit
Gyroscope noise density	0.005°/sec/√Hz
Gyroscope bias stability @ 25°C	1.5°/hr average for all axes
Gyroscope sensitivity	0.004375°/sec/digit for ±125°/sec, 0.140016°/sec/digit for ±4000°/sec
Gyroscope non-linearity	0.01% full-scale
Gyroscope temperature sensitivity	±0.005°/sec/C
Magnetometer scale	±8.0 Ga
Magnetometer resolution	18 bit
Magnetometer sensitivity	0.0625 mGa/digit
Magnetometer non-linearity	0.1% full-scale

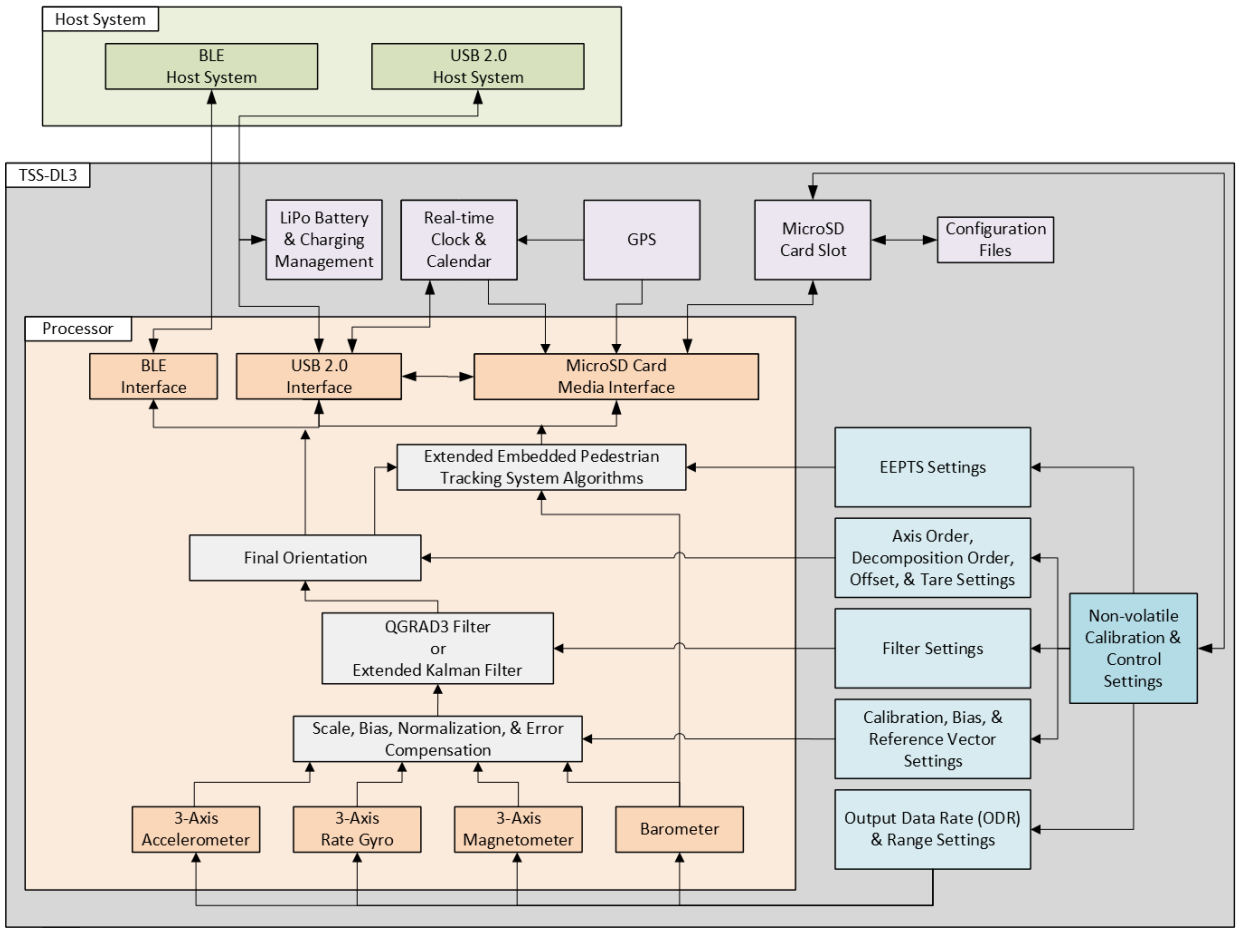
1. Varies based on power Settings
2. Depends upon communication mode and filter mode.
3. Average value when calibrated.

1.3.2.4 Data Logger Sensor Hardware Overview

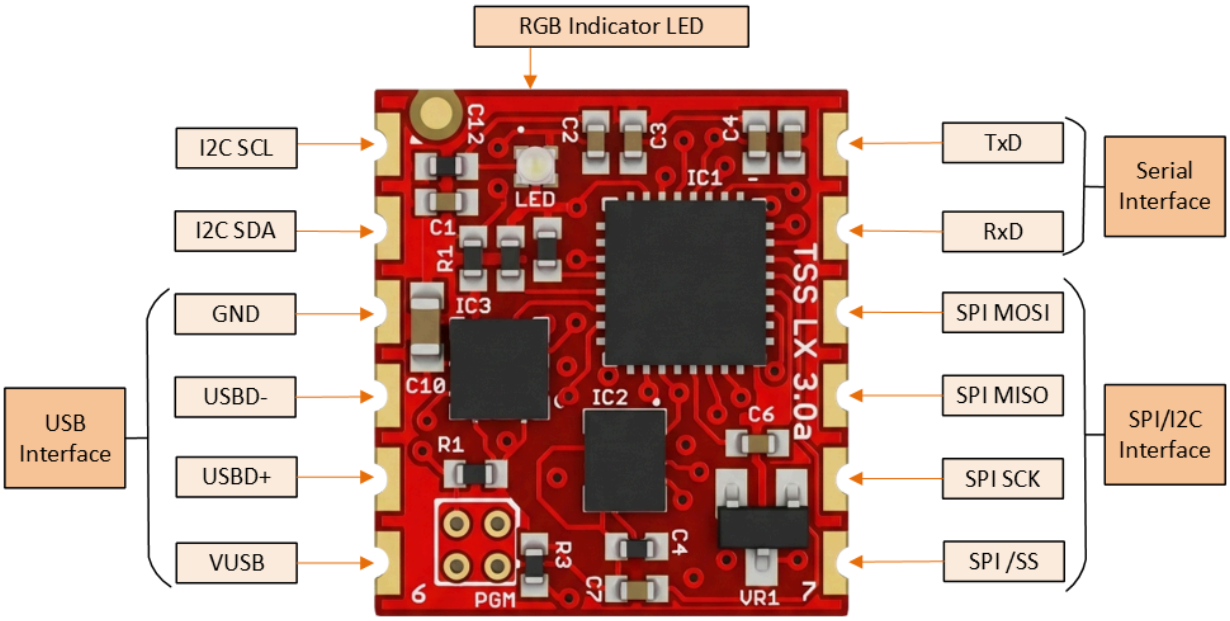
The 3-Space™ Data Logger v3 is 42mm x 39mm x 14mm cased sensor with a USB-C port for charging and communication.



1.3.2.5 Data Logger Sensor Block Diagram



1.3.3 3-Space™ LX Sensor: TSS-LX3



1.3.3.1 Supported Communication

- USB
  - USB 2.0 Full Speed
- UART
  - Up to 4000000 baud (see section 3.5 for full list)
  - Data bits: 8
  - Parity: None
  - Stop bits: 1
- SPI
  - Up to 10 MHz
- I2C
  - Up to 3.4Mbps

1.3.3.2 Supported Features

- Small and light high-performance AHRS available at 16mm x 15mm x 1.7mm and only 0.9 grams
- Fast sensor update and filter rate allow use in real-time applications, including stabilization, virtual reality, real-time immersive simulation, and robotics
- Highly customizable orientation sensing with options such as tunable filtering, oversampling, and orientation error correction
- Advanced integrated orientation filtering allows sensor to automatically reduce the effects of sensor noise and sensor error

- Robust open protocol allows commands to be sent in human readable form (ASCII mode), or more quickly in machine readable form (binary mode)
- Orientation output format available in absolute or relative terms in multiple formats (quaternion, rotation matrix, axis angle, two-vector)
- Absolute or custom reference axes
- Access to raw sensor data
- Flexible communication options: SPI, USB 2.0, or asynchronous serial
- USB communication through a virtual COM port
- Castellated SMT edge pads provide secure SMT mounting and allow optional through-hole mounting
- Upgradeable firmware
- RGB status LED
- Programmable interrupt capability
- Development kit available
- RoHS Compliant
- +5v tolerant I/O signals

1.3.3.3 LX Sensor Specifications

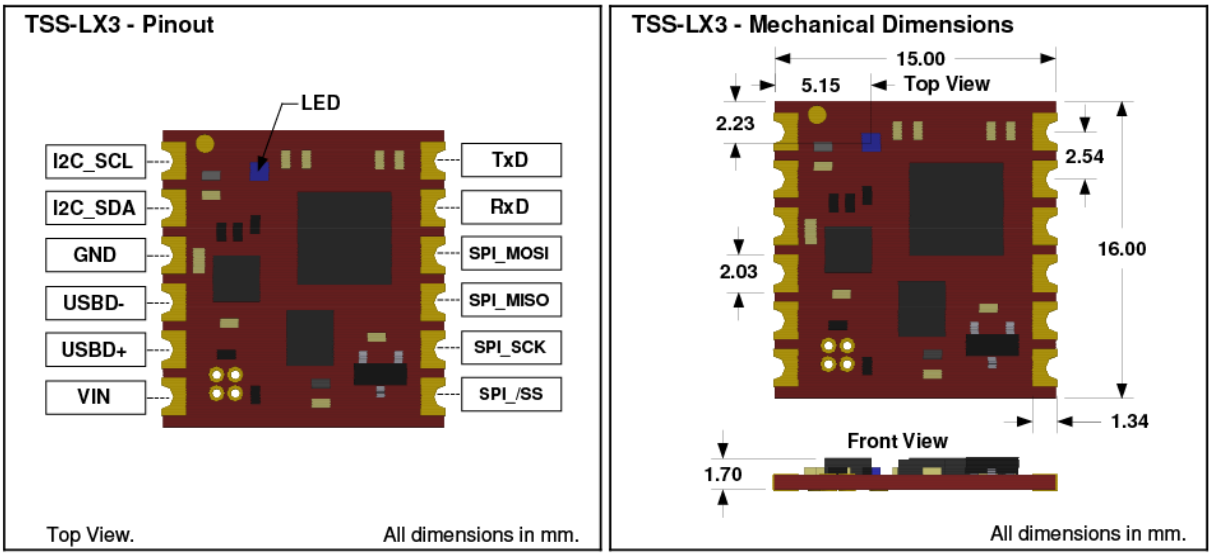
General	
Part number	TSS-LX3 v3.0
Dimensions	16mm x 15mm x 1.7mm (0.63 x 0.59 x 0.067 in.)
Weight	0.9 grams (0.032 oz)
Supply voltage	+3.3v ~ +6.0v
Power consumption[1]	Minimum: 10mA in low-power modes. Maximum: 55mA at full-performance
Communication interfaces	USB 2.0 FS, SPI, I2C, Asynchronous Serial
Filter update rate[2]	up to 2000 Hz with Extended Kalman AHRS up to 2000 Hz with QGRAD3™ AHRS up to 2000 Hz in IMU mode
Orientation output	absolute & relative quaternion, Euler angles, axis angle, rotation matrix, two vector
Other output	corrected sensor data, normalized sensor data, temperature
SPI clock rate	10 MHz max
I2C clock rate	3.4 MHz max
Internal pull-up	50k Ω
Serial baud rate	4,800~4,000,000 selectable, default: 115,200
Shock survivability	5000g
Temperature range	-40°C ~ 85°C (-40°F ~ 185°F)
Sensor	

Orientation range	360º about all axes
Orientation accuracy[3]	±1º for dynamic conditions & all orientations
Orientation resolution	<0.08º
Orientation repeatability	0.085º for all orientations
Accelerometer scale	±2g / ±4g / ±8g / ±16g selectable
Accelerometer resolution	16-bit
Accelerometer noise density	160µg/√Hz, 300µg/√Hz(M), 15mg/√Hz(H)
Accelerometer sensitivity	0.0000610352g/digit - 0.00048828g/digit
Accelerometer temperature sensitivity	±0.01%/°C
Gyroscope scale	±125/±250/±500/±1000/±2000/±4000 º/sec selectable
Gyroscope resolution	16-bit
Gyroscope noise density	0.005º/sec/√Hz
Gyroscope bias stability @ 25°C	1.5º/hr average for all axes
Gyroscope sensitivity	0.004375º/sec/digit for ±125º/sec, 0.140016º/sec/digit for ±4000º/sec
Gyroscope non-linearity	0.01% full-scale
Gyroscope temperature sensitivity	±0.007%/°C (-40°C to +85°C)
Magnetometer scale	±8.0 Ga
Magnetometer resolution	18 bit
Magnetometer sensitivity	0.0625 mGa/digit
Magnetometer non-linearity	0.1% full-scale

1. Varies based on power Settings
2. Depends upon communication mode and filter mode.
3. Average value when calibrated.

1.3.3.4 LX Sensor Hardware Overview

The 3-Space™ LX v3 is packaged as a 16mm x 15mm x 1.7mm castellated edge SMT module. Alternatively, the module can be through-hole mounted by adding standard 0.1" header strips to the castellated edge pads.

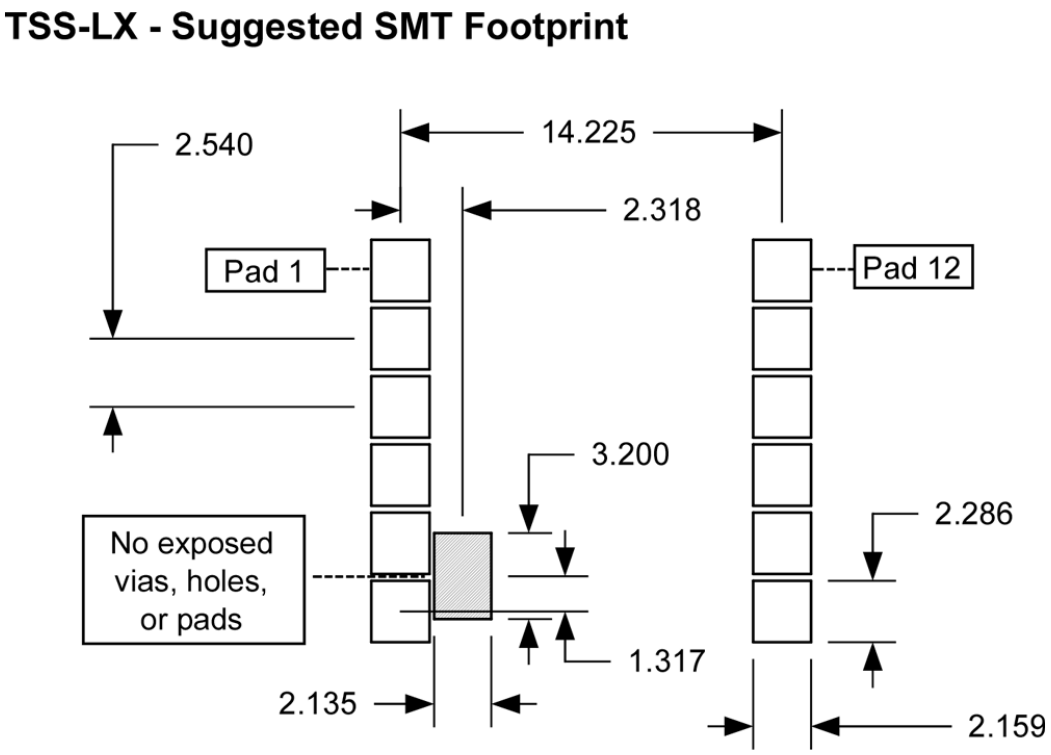


1.3.3.5 LX Sensor Pin Functions

Pad Number	Signal Name	Description
1	SCL	I2C Serial Clock Pin. Input to Module.
2	SDA	I2C Serial Data Pin. Input to Module & Output from Module.
3	GND	Ground. Only one ground pad must be connected. Commonly connected to USB supply ground
4	USBD-	USB Data Minus. Only requires connection during USB mode use.
5	USBD+	USB Data Plus. Only requires connection during USB mode use
6	VIN	+5v USB Power Supply Input . Only requires connection during USB mode use.
7	/SS	SPI Slave Select. Active Low Input to Module
8	SCK	SPI Serial Clock. Input to Module.
9	MISO	SPI Master In Slave Out. Output from Module. Can be configured to act as filter update Interrupt.
10	MOSI	SPI Master Out Slave In. Input to Module.
11	RxD	UART Asynchronous Receive Data. Input to Module
12	TxD	UART Asynchronous Transmit Data. Output from Module. Can be configured to act as filter update Interrupt.

1.3.3.6 LX Sensor PCB Layout

PCB layout can affect the functionality and performance of the 3-Space™ LX module. Please follow the suggested SMT footprint.

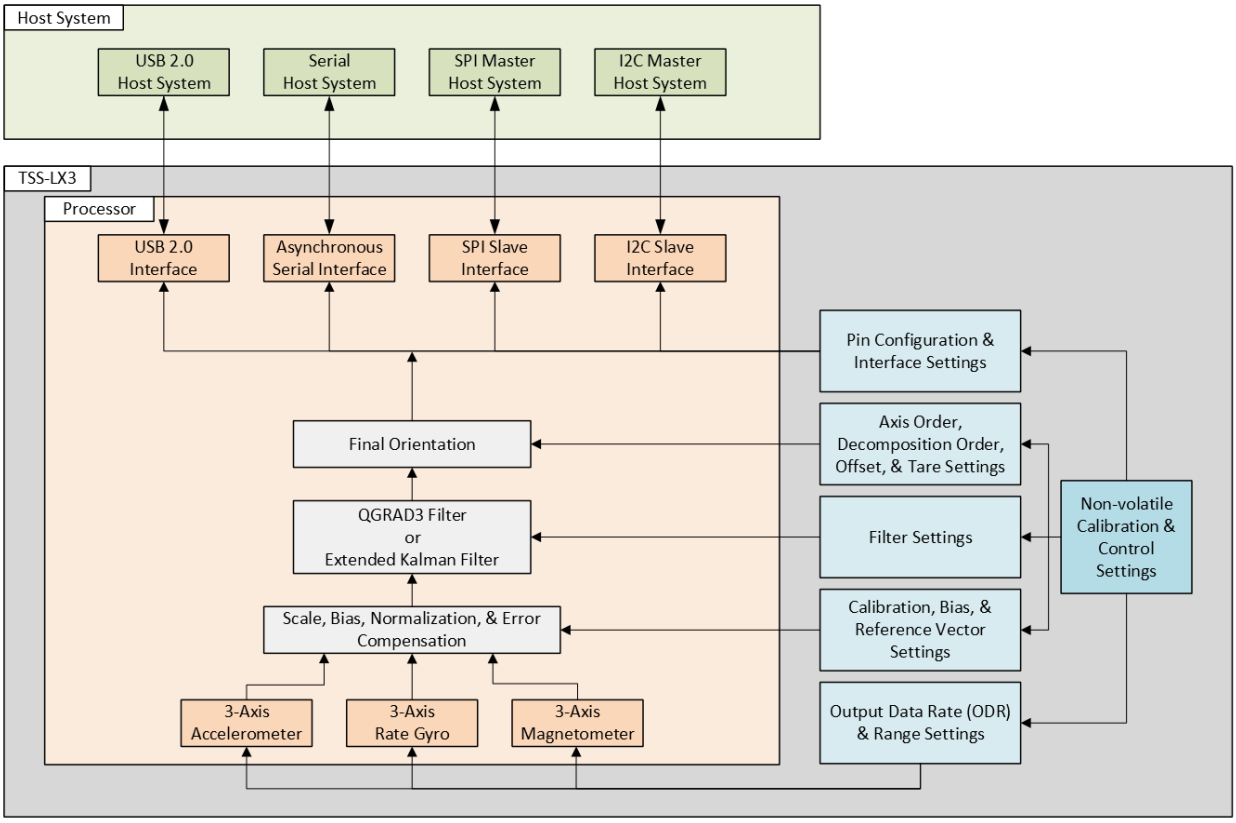


Additionally, please adhere to the following layout guidelines:

- Do not place magnetic components such as speakers and motors in close proximity to the module. The magnetic fields generated can adversely affect the performance of the magnetometer component.
- Do not place components containing ferrous metals in close proximity to the module. They may disturb Earth's magnetic fields and thus adversely affect the performance of the magnetometer component.
- Do not route high-current conductors or high-frequency digital signal lines in close proximity to the module. They may generate magnetic fields that adversely affect the performance of the magnetometer component.
- Do not reflow with the device on the bottom or side of a board. The module's components aren't glue-bonded to the module, so they may become dislodged if reflowed in non-upwards-facing orientations.

- Thoroughly test and characterize any PCB design that uses the module. Failure to test and characterize a system using the TSS-LX3 module may result in unforeseen performance consequences due to mechanical stresses, electrical characteristics, and component layout.

1.3.3.7 LX Sensor Block Diagram



1.4 Usage and Safety Considerations

1.4.1 Usage Considerations

General Considerations:

- Do not use the 3-Space™ Sensor in any system on which people's lives depend (life support, weapons, etc.).
- The 3-Space™ Sensor needs to be calibrated for proper performance. Please see the calibration section for additional information and instructions.
- Because the wireless 3-Space™ Sensor models use RF communication technology, communication failure modes should be carefully considered when designing a system that uses a wireless 3-Space™ Sensor.

- The wireless 3-Space™ Sensor is powered by a rechargeable lithium-polymer battery. Lithium-polymer batteries have high energy densities and can be dangerous if not used properly. Please see the Battery Safety Considerations section for additional details.

Because of the 3-Space™ Sensor's reliance on the accelerometer, please consider the following:

- Care should be taken when using the 3-Space™ Sensor in a car or other moving vehicle, as the disturbances caused by the vehicle's acceleration may cause the sensor to give inaccurate readings.
- The 3-Space™ Sensor will not work properly while in extended freefall, in outer space, or in other situations in which there is no discernable acceleration from gravity.
- The 3-Space™ Sensor will not work properly on astronomical bodies with a significantly different mass than Earth, or in situations in which the acceleration from gravity deviates too far from 1G.

Because of the 3-Space™ Sensor's reliance on the magnetometer, please consider the following:

- The 3-Space™ Sensor will not work properly near Earth's poles (both magnetic and geographic).
- The 3-Space™ Sensor will not work properly in places where Earth's magnetic field is heavily obstructed or is otherwise too weak to be detected (such as inside a metal container).
- The 3-Space™ Sensor will not work properly in places or situations in which magnetic or electromagnetic interference is so strong that Earth's magnetic field cannot be detected properly (such as near a powerful magnet).
- The 3-Space™ Sensor will not work properly in places or situations in which magnetic or electromagnetic interference constantly or intermittently changes (such as near an electric motor).
- The 3-Space™ Sensor will not work properly in outer space, or on astronomical bodies with no magnetic field.
- Care should be taken when using the 3-Space™ Sensor near ferrous metal structures, magnetic fields, and current carrying conductors. The sensor should be kept about 6 inches away from these objects, which include but are not limited to:
    - Electronic displays / monitors
    - Laptops / other computers
    - Cell phones / smart phones
    - Watches / smart watches / fitness devices

    - Desks (most desks contain unseen ferrous structural material)

  If the 3-Space™ Sensor is to be used near or attached to any of the above objects, the sensor will need to be recalibrated for optimal performance.

In situations where there are issues caused by the accelerometer, magnetometer, or gyroscope components, individual components can be manually disabled. This is not recommended, and the 3-Space™ Sensor will not work properly, but these incorrect orientation estimations may be preferable in certain situations. In these situations, please consider the following:

- In situations in which primary components are disabled, the orientation estimate will be inaccurate.
- In situations in which primary components are disabled, the degree to which the orientation estimate is inaccurate is unknown.
- If all accelerometers are disabled, drift will occur around the up axis.
- If all magnetometers are disabled, drift will occur around the north axis.
- If all accelerometers and all magnetometers are disabled, drift will occur around all axes.
- If all gyroscopes are disabled, orientation updates will be slower and less accurate.
- If all accelerometers, magnetometers, and gyroscopes are disabled, the sensor's orientation will not change unless updated manually.

1.4.2 Battery Safety Considerations

Many models in the 3-Space™ Sensor Family contain a rechargeable lithium-polymer battery. Lithium-polymer batteries have high energy densities and can be dangerous if not used and cared for properly. 3-Space™ Sensors have been designed to include multiple levels of battery safety assurance. The 3-Space™ Sensor design includes smart charging circuitry with thermal management to prevent overcharging the battery, and the battery pack itself includes protection circuitry to prevent overcharge, over-voltage, over-current, and over-discharge conditions. Most battery issues arise from improper handling of batteries, and particularly from the continued use of damaged batteries. As with any lithium-polymer battery-powered device, the following should be observed:

- Do not disassemble, crush, puncture, shred, or otherwise attempt to change the form of the battery.
- Do not attempt to change, replace, or modify the battery. Contact Yost Labs to arrange for battery replacement or battery repair.
- Do not allow the 3-Space™ Sensor or battery to be in contact with water.
- Do not allow the battery to touch metal or other electrically conductive objects (such as graphite).
- Do not place the sensor unit near a heat source. Excessive heat can damage the 3-Space™ Sensor or the battery. High temperatures can cause the battery to swell, leak, or malfunction.
- Do not attempt to use an appliance or heat source to dry a wet or damp 3-Space™ Sensor (such as a hair dryer, stove, or microwave oven).
- Do not drop the 3-Space™ Sensor. Dropping the sensor, especially on a hard surface, can potentially cause damage to the sensor or the battery.
- If the 3-Space™ Sensor or battery produces odors, emits smoke, exhibits swelling, produces excess heat, or exhibits leaking, discontinue use immediately. If it is safe to do so, remove sources of external power (unplug the USB cable) from the device and move the device to a well-ventilated area away from anything flammable. Move to a safe area and contact Yost Labs for additional instructions.
- If the 3-Space™ Sensor or battery produces flames or otherwise catches fire, move to a safe area and contact emergency services immediately.
- Dispose of Lithium-polymer batteries properly in accordance with local, state, and federal guidelines.

1.4.3 Technical Support and Repairs

For information on our warranty, repair, and return policies, please visit https://yostlabs.com/warranty/.

For immediate inquiries, please call +1 (740) 876-4936 between the hours of 9am and 5pm US Eastern Time.

To contact our custom development team to discuss customized hardware or software, please visit https://yostlabs.com/contact/custom-solutions/ and submit a message through the online form.

To contact customer support and/or technical support, please visit https://yostlabs.com/contact/. Feel free to send an email directly or submit a message through the online form. Please allow 1-2 business days for a response.

1.4.4 Regulatory Approval

1.4.4.1 United States FCC Approval

TSS-DL3 units can contain FCC ID: SH6MDBT50Q

This device complies with part 15 of the FCC Rules. Operation is subject to the following two: (1) This device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

1.4.4.2 Canada IC Approval

TSS-DL3 units can contain IC ID: 8017A-MDBT50Q

This device complies with Industry Canada license-exempt RSS Standard(s). Operation is subject to the following two conditions. (1) This device may not cause harmful interference. (2) This device must accept any interference received, including interference that may cause undesired operation.

Le présent appareil est conforme aux CNR d'Industrie Canada applicables aux appareils radio exempts de licence.L'exploitation est autorisée aux deux conditions suivantes: (1) l'appareil ne doit pas produire de brouillage, et (2) l'utilisateur de l'appareil doit accepter tout brouillage radioélectrique subi, même si le brouillage est susceptible d'en compromettre le fonctionnement.

1.4.4.3 European Approval

TSS-DL3 can contain a communication module that has been certified for use in European countries.

The following testing has been completed:
Test standard ETSI EN 300 328 V2.2.2:2019

2 Function Descriptions

2.1 Orientation Filter

The primary purpose of the 3-Space™ Sensor is to accurately estimate orientation. In order to understand how to handle this estimation and use it in a meaningful way, there are a few concepts about the sensor that should be understood. The following sections describe these concepts.

2.1.1 Primary Component Overview

The 3-Space™ Sensor estimates orientation by combining the data it gets from three types of sensing components: gyroscopes, accelerometers, and magnetometers. The gyroscopes primarily give accurate rotational data, and the accelerometers and magnetometers are used together to provide a global reference frame and correct errors over time. Most 3-Space™ Sensors contain one of each of these 3-axis components, but some off-the-shelf models contain 2 or more accelerometers, and custom designs support multiple of all primary components (and most other components).

All primary components have selectable range, ODR, and calibration parameters. Components are configurable via the sensor settings protocol. See section 3.2 Settings Protocol Overview.

For custom designs or requirements, please contact us via telephone or through our online form at https://yostlabs.com/contact/custom-solutions/.

2.1.1.1 Gyroscopes:

These components measure angular motion. The gyroscopes used in the 3-Space™ family of sensors are very accurate in the short term, and are excellent for measuring quick angular motions (rotations). In the long term, the orientation measured by the gyroscopes will slowly drift due to small amounts of measurement noise and process noise. Additionally, since gyroscopes measure motion relative to the sensor-body itself rather than relative to an external absolute reference-frame, they must be corrected by other components to achieve absolute orientation and to account for the long term drift.

- Gyroscope range capability is expressed in ±degrees per second (DPS, deg/sec, deg/s, °/s), with one full revolution around a single axis being equivalent to ±360°.

- Readings from the gyroscope are given in radians per second (rad/s, rad/sec, radians/sec), with one full revolution around a single axis being equivalent to 2π radians, or roughly 6.283185 radians.

2.1.1.2 Accelerometers:

These components measure acceleration. Acceleration as measured is the combined acceleration that is present from the static acceleration force due to gravity as well as other accelerations that occur from the 3-Space™ Sensor moving. For the orientation estimate calculation the accelerometers are used to determine the direction of gravity, which helps to provide a frame of reference for the gyroscopes' rotations. Because the accelerometers will pick up all accelerations experienced by the 3-Space™ Sensor, not just those from gravity, the orientation calculations are most accurate while the sensor is sitting still and is only experiencing gravitational acceleration. To reduce potential jitter in the orientation estimate from non-gravitational accelerations, when the 3-Space™ Sensor is being moved, the gyroscope readings become more favored in the orientation estimate, and the accelerometer readings become less favored.

- Accelerometer specifications and readings are expressed in terms of g-force (*g* or g_0), with 1*g* being equal to 9.80665m/s².
- A sensor with default settings when sitting at rest with the logo/LED facing upwards will report an average of +1.0g on the Y-axis.

The accelerometers’ output can also be used to detect dynamic motions due to movement, vibration, translation, etc.

2.1.1.3 Magnetometers:

These components measure magnetic direction. The primary use of the magnetometers in the orientation calculation is to determine the direction of magnetic north, which helps to provide a frame of reference for the gyroscopes' rotations. Of the primary sensing components, the magnetometer readings are the most resilient to physical vibrations and shakes, but are very sensitive to any changes in nearby magnetic fields.

The magnetometers will be adversely affected by any ferrous metal, magnetic objects, and other sources of electromagnetic interference. Additionally, Earth's magnetic field differs depending on geographical location. Because of these reasons, to achieve the most accurate orientation estimates, the magnetometers often require recalibration when moved to a different location, environment, or mounting point. The extent to which this recalibration is necessary depends on the level of accuracy required and should be evaluated on a case-by-case basis.

- The magnetometer specifications and readings are expressed in gauss (G, Ga).
- Readings from Earth's magnetic field are typically within 0.2 to 0.8 gauss.
- A sensor with default settings, still, and the power button facing magnetic north will report mostly positive readings on the z-axis. This will usually be slightly off from geographic north, with the sensor pointed partially into the ground.

2.1.2 Filter Overview

The 3-Space™ Sensor currently includes two different filter modes for providing orientation estimation. There is also the option to disable orientation filtering if orientation estimations are not required (IMU Mode).

2.1.2.1 Kalman Mode:

If the filter is set to Kalman Mode, the Extended Kalman Filter (EKF) sensor fusion algorithm will run. The EKF is industry standard and has been tested and proven to work effectively. Normalized sensor data and reference vectors are fed into the filter, which uses statistical techniques to optimally combine the data into a final orientation reading. The EKF can provide the highest-accuracy orientation fusion, but requires some setup for optimal performance.

2.1.2.2 QGRAD3 Mode:

If the filter is set to QGRAD3 Mode, the Yost Labs QGRAD3™ proprietary sensor fusion algorithm will run. The QGRAD3™ filter is designed to be computationally faster than the EKF with comparable accuracy. The QGRAD3™ filter requires less setup to use than the EKF, and so is the default filter setting on the 3-Space™ Sensor.

2.1.2.3 IMU Mode:

If the filter is set to IMU Mode, no orientation filters will run. The orientation estimation outputs, linear acceleration outputs, and EEPTS™ outputs will be unavailable. This mode has the lowest power consumption.

2.1.3 Calibration Overview

Each of the components produces outputs ain a “raw” output format that is not directly usable. As such, each sensing component on the 3-Space™ Sensor supports some form of calibration. Raw sensor component readings will often be different from the real world quantities they are measuring and thus must be converted and subsequently corrected for scale, cross-axis effect, and bias for optimal results.

For each of the three axes (x, y, and z), there is one value that is used to correct the scale, two values that are used to correct the cross-axis effect (one for each of the other two axes), and one value that is used to correct for the bias.

Scale is how much larger the range of data is than it should be, and needs to be corrected for on most accelerometers and magnetometers. For example, if an accelerometer were to give readings in the range of -4.4 to +4.4 on the y-axis, but we expect the accelerometer to operate in the range of ±4.0g on all axes, we would want to

set the scale value of the y-axis to 0.9 to ensure that the sensor readings correspond correctly to real world units.

The cross-axis effect is the tendency for a small portion of data on one axis of a sensor to affect the sensed values on the other two axes. Cross-axis effects are unavoidably present in all sensing elements but are generally greater in accelerometers and magnetometers. Values for the cross-axis effect are generally within the range of ±0.1 (less than 10% cross-axis effect on each neighboring axis).

Bias is how far the center of a data set is from 0, and needs to be corrected for on most accelerometers, magnetometers, and gyroscopes. This can be thought of as an offset that must be corrected for by translating the data-range up/down. For example, if a magnetometer were to give readings in the range of -8.3 to +7.7 on the x-axis, but we expect the magnetometer to operate in the range of -8.0 to +8.0 gauss on all axes, we would want to set the bias value of the x-axis to -0.3 to ensure that the sensor readings are properly centered at 0.

If another magnetometer, operating in the range of ±1G, read from -0.2 to +0.9 on the z-axis, the bias would be 0.35, and the scale would be ~1.82.

Each accelerometer, gyroscope, and magnetometer has a 9-value 3x3 calibration matrix and a 3-value calibration bias associated with it to be used for data correction. The calibration matrices represent the cross-axis and scale corrections for each of the three axes, and can be accessed with the following settings commands, where $\mathscr{X}$ is the ID of the component you wish to access:

calib_mat_accel$\mathscr{X}$; calib_mat_gyro$\mathscr{X}$; calib_mat_mag$\mathscr{X}$

The calibration bias values represent the bias corrections for each of the three axes, and can be accessed with the following settings commands, where $\mathscr{X}$ is the ID of the component you wish to access:

calib_bias_accel$\mathscr{X}$; calib_bias_gyro$\mathscr{X}$; calib_bias_mag$\mathscr{X}$

Please see the commands section for more information.

Corrected data is computed from the raw data using one of the two following formulas:

```
Corrected = Matrix * ( Raw + Bias )
Corrected = Matrix * ( Offset * Raw + Bias ) * Offset
```

The first formula is the simple form calibration formula that is used with default settings. The second formula is used if the setting 'axis_offset_enabled' is set to 1. While this is the default setting, the offset defaults to the identity and so the second formula simplifies to the first. The second formula only needs to be used if 'axis_offset_enabled' is 1 and the user has applied an offset as described in section 2.1.6

2.1.4 Magnetometer Reference Vector

The Extended Kalman Filter requires a 3-value magnetometer reference vector (MRV) to help determine where magnetic north is. Behind the scenes, the MRV is the reading on the magnetometer (x, y, z) when the sensor's orientation is the identity quaternion (x=0, y=0, z=0, w=1). The MRV can also be used to determine magnetic inclination / magnetic dip, and vice versa. The MRV is not used in QGRAD3™ mode or IMU mode.

2.1.4.1 Automatic MRV

Upon receiving power, the 3-Space™ Sensor will check the `filter_mref_mode` setting. With `filter_mref_mode` set to 1 (the default setting) and with no saved MRV, or with `filter_mref_mode` set to 2, the 3-Space™ Sensor will run the "Begin Passive Calibration" command to obtain an MRV for use with Kalman Mode / EKF.
With `filter_mref_mode` set to 0 (Manual Mode), the sensor is forced to use the MRV saved on the sensor.

With `filter_mref_mode` set to 3 (GPS Mode), the sensor will use GPS readings and the World Magnetic Model to determine the MRV, and will continually update. This mode will only work properly on sensor models with GPS modules built-in, but this functionality can be simulated with an external GPS and the `filter_mref_gps` setting (see the "Manual MRV" section).

For an overview of the "Begin Passive Calibration" command, please see the "Filter Setup" section.

To check the "filter_mref_mode" setting, send the following ASCII string to the sensor:

```
?filter_mref_mode\n
```

To change the "filter_mref_mode" setting, send the following ASCII string to the sensor, replacing $\mathscr{X}$ with 0, 1, 2, or 3 (the default setting is 1):

```
!filter_mref_mode=𝒳\n
```

2.1.4.2 Manual MRV

The current MRV can be queried with the setting:

```
?filter_mref\n
```

and can be updated manually with the setting

```
!filter_mref=x,y,z\n
```

where x, y, and z are floats corresponding to a magnetometer reading. When the MRV is obtained using the "Begin Passive Calibration" command, x will equal 0, y will be negative, and z will be positive. When setting the MRV manually, the same should also be true.

Updating the MRV will also update the magnetic dip.

The current magnetic dip / inclination can be queried with the setting

```
?filter_mref_dip\n
```

and can be updated manually with the setting

```
!filter_mref_dip=x\n
```

where x is a float corresponding to the magnetic dip angle in degrees. This angle will range between -90° (minimum at the South Magnetic Pole) to +90° (maximum at the North Magnetic Pole).

Updating the magnetic dip will also update the MRV.

The MRV / dip can also be set by supplying GPS coordinates with the setting:

```
!filter_mref_gps=x,y\n
```

where x is the latitude in decimal format and y is the longitude in decimal format.
This setting is set-only and cannot be queried.

2.1.5 Filter Setup

For any filter setting, you will need to perform the following to get the highest quality sensor data and orientation estimate outputs:

- Set gyroscope bias (Passive Calibration)
- Set accelerometer calibration matrix (Gradient Descent Calibration)
- Set accelerometer bias (Gradient Descent Calibration)
- Set magnetometer calibration matrix (Gradient Descent Calibration)
- Set magnetometer bias (Gradient Descent Calibration)

If you are using the Kalman Filter setting, you will additionally need to perform the following:

- Set magnetometer reference vector (Passive Calibration)

2.1.5.1 Filter Setup - Passive Calibration Command (Gyroscope, Magnetometer Reference Vector)

The "Begin Passive Calibration" command, 165(0xA5), is used to obtain the gyroscope bias and the magnetometer reference vector (MRV).

To obtain and set a gyroscope bias, first ensure that the 3-Space™ Sensor is at rest, and then send the following ASCII string to the sensor:

```
:165,1\n
```

The sensor must remain still until the "Get Passive Calibration Active" command, 166(0xA6), returns 0. To run this command, send the following ASCII string to the sensor:

```
:166\n
```

When the command returns 0, passive calibration is finished, and the gyroscope bias has been set. To check the bias, you can send the ASCII string

```
?valid_gyros\n
```

to determine the ID number of your gyroscope, and then

```
?calib_bias_gyroX\n
```

where X is replaced with the ID number of your gyroscope.
When you are finished, please remember to send the "Commit Settings" command (either `!commit\n` or `:225\n`) to save the settings to the sensor's non-volatile memory.

To obtain and set an MRV, first make sure that the sensor is in a magnetic environment that is as close as possible to how it will be used.

- If the sensor is going to be attached near or on any metallic object(s) or sources of magnetic interference, attach the sensor to that object now.
- If the sensor is going to be used mostly handheld or in another situation away from any sources of magnetic interference, ensure that the sensor is away from any ferrous/magnetic material or electromagnetic interference.
- Please see section 1.4.1 Usage Considerations for examples of objects that can and will cause interference.

Next, make sure the sensor is as still as possible while following the above considerations. Send the following ASCII string to the sensor:

```
:165,2\n
```

The sensor must remain as still as possible until the "Get Passive Calibration Active" command, 166(0xA6), returns 0, which usually takes less than 5 seconds. To run this command, send the following ASCII string to the sensor:

```
:166\n
```

When the command returns 0, passive calibration is finished, and the MRV has been set. Please see section 2.1.4.2 Manual MRV for instructions on how to check the MRV values.

When you are finished, please remember to send the "Commit Settings" command (either `!commit\n` or `:225\n`) to save the settings to the sensor's non-volatile memory.

To obtain a gyroscope bias and an MRV at the same time, send:

```
:165,3\n
```

This is only recommended if optimal conditions are met (the sensor can stay completely still for the gyroscope bias, and is in a proper magnetic environment for the MRV).

2.1.5.2 Filter Setup - Gradient Descent Calibration (Accelerometer, Magnetometer)

To determine optimal calibration matrices and biases for the accelerometer and magnetometer, we recommend using one of our Gradient Descent Calibration tools.

If possible, we recommend using our DearPyGui-based 3-Space™ Sensor Suite, available here: (please send us an email us at information@yostlabs.com or techsupport@yostlabs.com until it is released on the website)

A Python 3 implementation of a gradient descent algorithm for sensor calibration can be found here: https://yostlabs.com/wp-content/uploads/2024/11/Calibration-Scripts.zip
This can be used as-is to calibrate a sensor, though it lacks a graphical interface.

2.1.6 Tare and Offset

Many times it is desired to report the sensor’s orientation relative to a specific reference orientation rather than in a global reference frame. Alternatively, it may be necessary to mount the sensor in a system in an orientation such that the sensor’s reference frame doesn’t match the overall system’s reference frame. To account for these use-cases, the 3-Space™ Sensor products support a tare, a base tare, an offset, and a base offset.

A tare is used to change the reference/identity orientation of the sensor. The tare can also be thought of as setting the “tare” or “zero” orientation for the sensor. Alternatively, the tare orientation can be thought of as positioning an imaginary camera used to view the sensor. The tare is most useful when the use-case requires the orientation relative to a known “zero” orientation or when trying to match the orientation’s reference space to the viewer’s reference space.

In situations in which it is useful to view a sensor's orientation as if it were offset into another orientation, the Offset feature can be used. This is frequently used when mounting a sensor onto an object to allow its axes to match the object's axes regardless of mounting position/orientation.

The tare feature and offset feature can be used together. This is how the final orientation is calculated by our sensor firmware:

$$orientation_{final} = tare\_quat * orientation_{filtered} * offset\_quat$$

Essentially, setting the offset_quat is rotating the x, y, and z axes around the sensor into a more convenient orientation, and setting the tare_quat is rotating the camera/viewer around the sensor for a change in point of view.

2.1.6.1 Setting the Tare

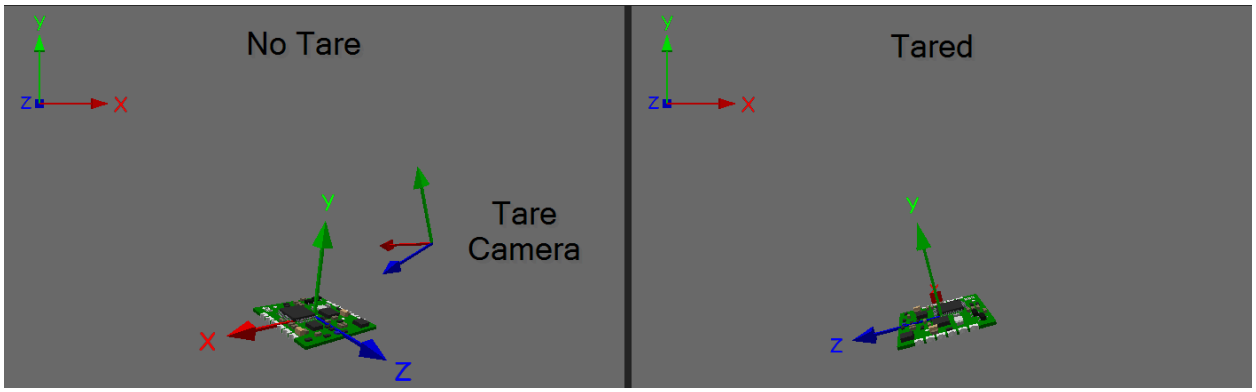
The sensor's orientation reading may differ from the actual orientation of the device by a constant angle until it has been given a reference orientation. In these situations, the tare command can be called to set the reference orientation, which tells the sensor where you would like the identity orientation to be.

The identity orientation refers to the orientation in which the orientation quaternion has x, y, and z values at 0.0, and the w value at 1.0 (an identity quaternion). We define our identity orientation as the orientation a properly calibrated sensor would be in when the positive z-axis is pointed towards magnetic north, and the positive y-axis is pointed straight up into the sky.

For reference, on a cased sensor with default settings, the positive z-axis points out from the power button, the negative z-axis points out from the USB port, the positive y-axis points out from the logo on the front of the sensor, and the negative y-axis points out from the serial number sticker on the back of the sensor.

The act of giving the sensor a new reference orientation is called taring. Keep in mind that taring a sensor and using the tared orientation does not change how the sensor will rotate along its local axes.

Setting a reference orientation via the tare is also the same as orbiting the viewer around the sensor. Here is an example of a sensor with and without the tare applied to visualize how the tare behaves.



2.1.6.2 Setting the Offset

There are many applications for which it will be necessary or convenient to mount the sensor at odd angles, but it may also be desired in these situations that orientations can be treated as though the sensor were mounted normally. For example, if the sensor were mounted on a sloped surface of a vehicle like a car hood, it would be helpful if the orientations could read as though the sensor was mounted in a way that more closely matched the overall orientation of the vehicle, which does not include that slope.

The offset quaternion can be used to deal with mounting differences. This offset allows the sensor to pretend it is mounted in any given orientation while being actually mounted in any other actual orientation.

It should be noted that while it may seem like the set axis directions command could be used for the same purpose, the offset orientation feature is the preferred way to deal with alternate mountings, as the axis directions mode has no way to account for a mounting that isn't a 90 degree based orientation away from the standard orientation.

When setting the offset, both the orientation and actual data axes are offset by the same amount. This means the offset also affects data such as accelerometer, magnetometer, and gyroscope data. This offset only gets applied to axes if the setting 'axis_offset_enabled' is set to 1 (this is the default), otherwise only the orientation is affected. Also, if an offset is applied and axis_offset is enabled, then the calibration formula slightly changes as described in section 2.1.3.

2.1.6.3 Examples: Offset and Base Offset

If you do not know your offset quaternion ahead of time, you can use the base offset to help set the final offset.

1. Place the sensor in an orientation aligned with the overall vehicle or device the sensor is being mounted on, or in the orientation that you would like the sensor to act like it is in.
    a. Typically this means align the natural Z (forward) axis of the sensor with the forward direction of your object, and the natural Y (up) axis of the sensor with the up direction of your object.
2. Call command 22(0x16), "Set Base Offset With Current Orientation", which sets the “base_offset” setting with the current filtered orientation.
    a. You can manually set the base offset with a settings command:
    `!base_offset=x,y,z,w\n`
    where x, y, z, and w are floats corresponding to the base orientation quaternion.
    b. You can check the current base offset with the setting command:
    `?base_offset\n`
    c. To reset the base offset, call command 20(0x14).
3. Mount the sensor onto the vehicle or device as you intend to for the end application.
4. Call command 19(0x13), which will set the offset based on the difference between the current orientation and the base offset. After this command is called, the sensor should now be acting as though it is in the orientation that "Set Base Offset With Current Orientation" was called in.
5. Make sure to commit the sensor settings to keep these changes.

2.1.6.4 Examples: Using Tare and Offset Together

In most situations, it is desirable to apply both an offset and a tare. The offset allows for the orientation to match the objects orientation regardless of mounting location, and the tare allows defining an identity orientation. There are many ways of configuring the tare and offset to get similar end results. The following outlines our general recommended procedure for setting both the offset and tare.

1. Align the natural Z (forward) axis of the sensor with the forward direction of your object, and the natural Y (up) axis of the sensor with the up direction of your object.
2. Call command 22(0x16), "Set Base Offset With Current Orientation"
3. Attach/mount the sensor to the other object.
4. Call command 19(0x13), "Set Offset With Current Orientation"

In the case that the object the sensor is attached to is easily movable, and the sensor's mounted orientation may vary, such as a sensor attached to a human with an elastic strap, the following procedure may be more convenient for long term use when calculating a new offset is frequently required.

1. Align the natural Z (forward) axis of the sensor with a direction you consider to be forward, such as North, or facing towards a specific wall. Align the natural Y (up) axis of the sensor with the direction you consider to be up, which in general is simply away from the ground.
    a. By default, the Base Offset is upright and facing Magnetic North, so if that is your desired reference location, steps 1-3 can be skipped.
2. Call command 22(0x16), "Set Base Offset With Current Orientation"
3. Commit this base offset. This base offset will now be used every time the sensor is going to have its offset set.
4. Mount the sensor
5. Position the object such that it is upright and facing towards your defined forward direction.
6. Call command 19(0x13), "Set Offset With Current Orientation"
7. To apply a new offset, start again from step 4.

Once an offset is set, a tare can be used to set the identity direction.

1. Set the Base Tare to the same value as the offset
    a. This is automatically done when the offset is set and the setting 'tare_auto_base' is set to 1, which is its default value, so this step can be ignored.
2. Position the object with the sensor mounted to it in its desired identity orientation.
3. Call command 96(0x60), "Set Tare With Current Orientation"

If the object can not be easily moved/positioned:

1. Reset the Base Tare '!base_tare=0,0,0,1\n'
2. Unmount the sensor

3. Align the natural Z (forward) axis of the sensor with your desired identity forward and the natural Y (up) axis of the sensor with your desired identity forward.
    a. Think of the sensor as a camera, and you are pointing the camera to set which direction is forward and up
4. Call command 96(0x60), "Set Tare With Current Orientation"
5. You can now safely remount the sensor.

*In general, it does not matter if tare is done before or after the offset, but we recommend this approach when following along with a visual representation of the sensor

2.1.7 Axis Management

Most data given by the sensor will be in relation to an X, Y, and Z axis. The axes are defined as an ordering of XYZ or E/W (east/west) U/D (up/down) and N/S (north/south). By changing the axis order, the reference space of the data can be changed to match your applications needs.

2.1.7.1 Natural Axes

With default sensor settings, the local natural axes will originate from the center of the sensor, and will emerge from the following places on the sensor:

X Positive/E East: From the right side of the sensor (red arrow).
- On a cased sensor, this will be your right-hand side as you look at the Yost Labs logo in its upright orientation.
- On an embedded sensor, this will be the edge with the castellations marked 1-6.

X Negative/W West: From the left side of the sensor.
- On a cased sensor, this will be your left-hand side as you look at the Yost Labs logo in its upright orientation.
- On an embedded sensor, this will be the edge with the castellations marked 7-12.

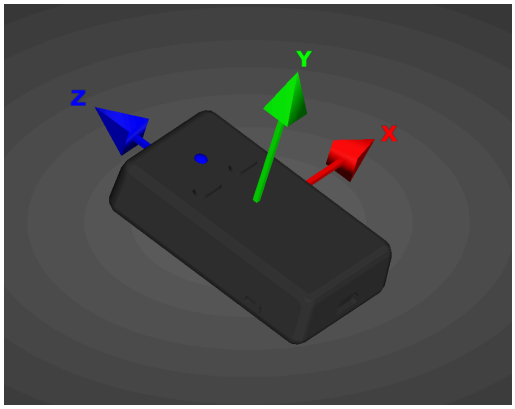
Y Positive/U Up: From the top of the sensor (green arrow).
- On a cased sensor, the top is the side where the Yost Labs logo is etched.
- On an embedded sensor, the top is the side where the serial number sticker is.

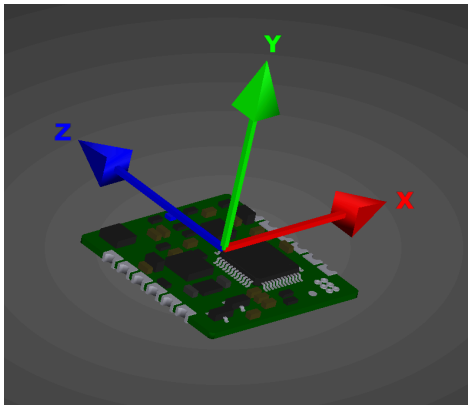
Y Negative/D Down: From the bottom of the sensor.
- On a cased sensor, the bottom is the side where the serial number is etched.
- On an embedded sensor, the bottom is the largest flat side.

Z/N North: From the front of the sensor (blue arrow).
- On a cased sensor, the front is where the power button is.
- On an embedded sensor, the front is the edge closest to the LED.

Z Negative/S South: From the back of the sensor.

- On a cased sensor, the back is where the USB port is.
- On an embedded sensor, the back is the edge farthest from the LED.





This is a left-handed coordinate system. It was chosen as the default because it is very common in game development, and therefore is familiar with many people. It can be described as XYZ (right, up, forward) or EUN (east, up, north).

2.1.7.2 Modifying Axis Order

If the natural axes do not match your preferred axis setup, the axes can be negated and swapped to the most convenient configuration for you.

You can check and set the axis setup with the settings "axis_order" and "axis_order_c". With the default axis setup, you should see the following output:

```
?axis_order\n
axis_order=XYZ\n
?axis_order_c\n
axis_order_c=EUN\n
```

For axis_order_c, you can choose e/w for east/west, u/d for up/down, and n/s for north/south. These compass directions correlate to the direction the axis would represent when the sensor's untared orientation is its identity, which is considered upright and the default Z axis (front of the sensor) pointing North.

Changing the axis order does the following:

- Swaps the order data is given from the accel and mag
- Swaps the order data is given from the gyro
    - If the handedness changes, the values will also be negated
- Swaps the format of orientation to be represented in the new axis space
    - For a quaternion, this involves swapping the axis to match the new order, and then inverting the quaternion (negating the x, y, and z component) if there is a change in handedness.

Examples of how data would look when switching from XYZ to -YZX across different data types

Type	Original	New
Accel/Mag	1,2,3	-2,3,1
Gyro	1,2,3	2,-3,-1
Quaternion	0.654,0.261,-0.065,0.707	0.261,0.065,-0.654,0.707

To change the axis order, see these examples:

```
!axis_order=xyz\n
!axis_order=zxy\n
!axis_order=-x-y-z\n
!axis_order=-zy-x\n
!axis_order_c=eun\n
!axis_order_c=wds\n
!axis_order_c=dws\n
!axis_order_c=ned\n
```

If changing the axis order was successful, you should see the return "0,1\n".

Note: When changing axis order, if using XYZ, that XYZ refers to the default XYZ of the sensor, not the current XYZ. Anywhere else in this manual XYZ refers to the values returned from commands such that the first value is X, second value is Y, and third is Z regardless of the set axis order unless otherwise specified. For clarity, some instances may explicitly refer to the default axes as the "natural axes".

2.1.7.3 Axis Order with Tare & Offset

When applying a tare and offset as described in section 2.1.6 using the offset/tare with current orientation commands, their values are stored in regards to the axis order at the time the command was called. For this reason it is recommended to first set your desired axis_order before performing an offset or tare to avoid unintended results.

By using both the axis_order and offset with 'axis_offset_enabled=1' as described in section 2.1.6.2, the sensor can be configured to give data in the desired order and matched to your object's desired orientation. While the axis order remaps the sensor's default axes, the only way to actually change their physical relation to the sensor is via also applying an offset.

2.1.8 Euler Angles

The 3-Space™ Sensor can return the current orientation as represented in Euler Angles. The Euler Angle representation has multiple ways of being calculated. This section goes over how the Euler Angles are calculated, what they represent, how best

to set them up for your project, as well as the limitations of Euler Angles and how they may not behave as expected.

2.1.8.1 Euler Basics

Euler Angles represent an orientation by applying a sequence of rotations around specified axes in a specified order. There are an infinite number of Euler Angle sequences that represent the same orientation. For our implementation, all rotations will be in the range of [pi, -pi], [pi/2, -pi/2], or [0, pi] depending on the axis/decomposition order. We also allow all variations of Tait-Bryan and Proper Euler Angles.

Computational Idea

By default, the Euler Angles are computed intrinsically from the current quaternion. The core concept behind this is to rotate the unit vector for the last axis of rotation by the current quaternion. Then, using trigonometry, compute the angles used by the first 2 rotations to position the unit vector of the last axis at the computed spot. Finally, rotate any axis other than the last axis by the known quaternion, as well as the quaternion as a result of just the first 2 rotations, and compute the difference.

Application

Because Euler angles represent the final rotation as a sequence of rotations in a specified order, it is important to note that a change in a rotation around any axis may cause changes to the angles reported on other axes. The change will be dependent on both the set Euler Decomposition Order, as well as orientation of the sensor prior to the additional rotation.

2.1.8.2 Euler Decomposition Order

The decomposition order can be set using the 'euler_order' setting. This is set using a 3 to 4 character string specifying the order. The first 3 characters are the axes of rotation. The last character, which is optional, specifies intrinsic or extrinsic. If the last character is not supplied, it defaults to intrinsic.

Example:
`'euler_order=ZYXe'`

Valid Orders

Tait-Bryan Euler Angles	Proper Euler Angles
XYZ	XYX
XZY	XZX
YXZ	YXY
YZX	YZY
ZXY	ZXZ
ZYX	ZYZ

Note: Each of these can have an 'i' or 'e' added at the end to specify intrinsic or extrinsic respectively.
Note: When using Tait-Bryan Angles, we recommend having an axis that will not rotate 90 degrees or more as the middle axis in the euler order.

Reading Results

When Euler Angles are returned, they are returned in the same order as the decomposition order. So if the *euler_order=ZXY*, and the returned response is `-0.00573,-0.87815,-3.12452` then that means `-0.00573` is the rotation around the Z axis, `-0.87815` is the rotation around the X axis, and `-3.12452` is the rotation around the Y axis.

Intrinsic vs Extrinsic

When rotating via Euler Angles, for example around the X axis, the location of the other 2 axes will change based on the rotation. If the next rotation is applied based on the new location of the axes (the local axes), that is using Intrinsic Euler Angles. If the next rotation is instead applied using the old axis locations (the global axes), that is using Extrinsic Euler Angles.

Intrinsic Euler Angles and Extrinsic Euler Angles are actually opposites of each other. That is, rotating by a given order intrinsically is the same as rotating by the reverse order extrinsically. Therefore, order XYZi and ZYXe are actually equivalent but with the order the elements are returned swapped.

Tait-Bryan Angles

These are any *euler_order* in which all 3 axes are unique. When using Tait-Bryan Angles, the First and Third axis will have a range of [-pi,pi], while the second axis will have a range of [-pi/2,pi/2]. When the second axis approaches pi/2 or -pi/2, the other axes may experience jumps in values due to the singularity issue with Euler Angles often referred to as Gimbal lock.

Proper Angles

These are any *euler_order* in which the first and third axis are the same. When using Proper Angles, the First and Third axis will have a range of [-pi,pi], while the second axis will have a range of [0,pi], When the second axis reaches 0 or pi, the other axes may experience jumps in values due to the singularity issue with Euler Angles often referred to as Gimbal lock.

2.2 Secondary Component Overview

2.2.1 Barometer

The barometer component primarily measures atmospheric pressure and estimates elevation. Typically, elevation changes of ~10 centimeters can be detected by checking the pressure readings. The elevation is used to help in the global positioning estimate given by the Extended Embedded Pedestrian Tracking System.
Readings of atmospheric pressure from the barometer are given in millibar (mbar), with typical readings between 950 and 1050 mbar (depending on the temperature, weather, and elevation).
Readings of altitude from the barometer are given in meters above sea level.
This component is not included on the TSS-LX3, and may not be included on certain custom sensor models.

2.2.2 MicroSD Card Reader

The microSD card reader component is used to record data logging sessions onto the inserted microSD card. Most microSD cards are compatible, and several popular file system formats are supported (exFAT is preferred for optimal performance). Please see the data logging section for additional details.
This component is only included on sensor models that support data logging.

2.2.3 Real-Time Clock

The real-time clock component is used to keep track of the calendar date and time. The RTC is powered by a separate coin cell battery and will continue to function when the sensor is powered off.
This component is only included on sensor models that support data logging.

2.2.4 Bluetooth Module

The Bluetooth component is used to achieve wireless communication between the 3-Space™ Sensor and another Bluetooth capable device (minimum Bluetooth 4.0, recommended 5.3); examples include a laptop's included Bluetooth radio, or a USB Bluetooth antenna. The Bluetooth component supports connections on the 2.4GHz band using Bluetooth Low Energy. Communicating with the sensor over BLE usually requires a custom implementation or application; example scripts for implementing a BLE connection are available upon request.
This component is only included on sensor models that support wireless communication via Bluetooth.
Compatible Bluetooth devices are not sold by or provided by Yost Labs.

2.3 Data Streaming

The sensor features a streaming mode where it can be instructed to periodically send back the response from up to 16 commands at a set interval without any further communication from the host. This should be used for gathering bulk amounts of data, especially when sample timing is important. Using the base command protocol adds overhead of sending the command and parsing the request, which can lead to inconsistent timing. Data streaming will consistently gather samples with up to 1 microsecond accuracy and allow for far faster data rates (Up to 2000 Hz) than can be achieved by only using the base command protocol.

Before activating streaming mode, you must configure your streaming slots and your packet frequency. Streaming slots is a list of up to 16 different commands that will have their data streamed, while the packet frequency determines how often data is sent.

2.3.1 Base Configuration

2.3.1.1 Stream Slots

To set your streaming slots, use the "stream_slots" command. For example:

```
!stream_slots=37\n
!stream_slots=0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,32\n
```

`Note: When reading streaming slots via '?stream_slots\n', all 16 slots will be returned. 255 Represents an empty streaming slot.`

If a command requires an ID, such as 55 (Get Corrected Accelerometer Vector by ID), then the ID must be provided alongside the command number separated with a colon. For example, with an ID of 1:
`!stream_slots=55:1\n`

Most data commands are streamable.
The data commands include:

- 4.1 Orientation Commands
- 4.2 Normalized Data Commands
- 4.3 Corrected Data Commands
- 4.4 Other Data Commands
- From 4.6 EEPTS™ Commands, only 70, 71, 72
- 4.9.3 Battery Commands on models with batteries

- 4.9.4 GPS Data Commands on models with GPS modules

A full list of streamable commands can be obtained from the sensor by sending

`'?streamable_commands\n'`.

2.3.1.2 Stream Frequency

There are two settings that can be used to set the packet frequency. Setting one of these will also set the other, they are simply two different ways of interpreting the streaming frequency.

The "stream_hz" setting allows you to set the number of packets per second. The maximum value allowed is 2000 packets per second.

Example for 50 packets per second: `!stream_hz=50\n`
Example for 5 packets per minute: `!stream_hz=0.833333\n`

The "stream_interval" setting allows you to set the amount of time (in microseconds) between each packet. The minimum value is 500 microseconds.

Example for 1 second between packets: `!stream_interval=1000000\n`
Example for 1 millisecond between packets: `!stream_interval=1000\n`

Internally, the sensor uses the *stream_interval*, and the precision is limited to microsecond accuracy. Because of this, a set *stream_hz* may not be perfectly obtainable and be slightly modified. For example, setting *stream_hz* to 1500 will cause *stream_interval* to be set to 666, which means when *stream_hz* is read back, it will actually be 1501.501465. In any situation where the *stream_hz* is not perfectly obtainable, the actual set *stream_hz* will always be as close as possible but larger than requested. For most use cases, setting via *stream_hz* and having it automatically set the closest obtainable *stream_interval* is perfectly fine. However, for absolute precision and control it is recommended to use *stream_interval* and calculate the actual obtainable rates yourself.

2.3.2 Active Streaming

2.3.2.1 Starting Streaming

To activate streaming, send the `Start Streaming (0x55)` command. If the command is sent in Ascii mode, then the streaming response will be in Ascii. If the

command is set in Binary mode, then the streaming response will be in Binary. If the command is sent using one of the *Header Enabled Start Bytes*, then the header will be added to the output of the streaming response.

2.3.2.2 Stopping Streaming

To stop streaming, either wait for streaming to end based on the set *stream_duration* or *stream_count* as described in section 2.3.3, or send the `Stop Streaming (0x56)` command.

2.3.2.3 Stream Response

The format of each output while streaming will be identical to the output from the command `Get Streaming Batch (0x54)`. The header will be output first as normal, followed by the data from each cmd in *stream_slots* in the order they are set in *stream_slots*. When using Binary Mode, this is identical to the base Binary Protocol for responses.

When getting responses in Ascii Mode, the response is identical to the base Ascii Protocol except for one change. There will be a semi-colon used to separate each individual command's response from the next instead of a comma.

Note: When *header_echo* is enabled, the echo byte will be 84 (0x54) since the stream response is identical to the Get Streaming Batch command.

Example responses:

With '`stream_slots=0,39`' and '`header_timestamp=1;header_status=1`' with streaming at 500 Hz (Interval of 2000µs) for 3 samples with header enabled.

Color code:
Header Cmd#0 Cmd#39 Separator/Terminator

Ascii:
```
0,1553199;-0.200756,0.964716,0.122505,0.118378;-0.406006,0.914917,0.043823\r\n
0,1555197;-0.200764,0.964714,0.122509,0.118379;-0.401611,0.907471,0.039429\r\n
0,1557199;-0.200763,0.964713,0.122513,0.118380;-0.401978,0.895569,0.035400\r\n
```

Binary:
```
0x0 0x2f 0xb3 0x17 0x0 0xfb 0x92 0x4d 0xbe 0xa1 0xf7 0x76 0x3f 0xe7 0xe3 0xfa
0x3d 0x2a 0x70 0xf2 0x3d 0x5 0xe0 0xcf 0xbe 0x0 0x38 0x6a 0x3f 0xbf 0x7f 0x33
0x3d 0x0 0xfd 0xba 0x17 0x0 0x14 0x95 0x4d 0xbe 0x7f 0xf7 0x76 0x3f 0x0 0xe6
```

```
0xfa 0x3d 0xb0 0x70 0xf2 0x3d 0xf5 0x9f 0xcd 0xbe 0x5 0x50 0x68 0x3f 0x4e 0x80
0x21 0x3d 0x0 0xcf 0xc2 0x17 0x0 0xd1 0x94 0x4d 0xbe 0x6e 0xf7 0x76 0x3f 0x19
0xe8 0xfa 0x3d 0x37 0x71 0xf2 0x3d 0xf 0xd0 0xcd 0xbe 0x3 0x44 0x65 0x3f 0x97
0xff 0x10 0x3d
```

Note: Both the Ascii and Binary examples represent the exact same data results, just different formats.

2.3.3 Timing

If the sensor is not capable of keeping up with the set streaming speed, it will instead output data as fast as possible. Setting the *cpu_speed* setting to a higher value will typically solve this problem.

Another problem occurs when the rate at which data is generated exceeds the rate at which it can be transferred. For example, if using UART with a baud rate of 115200, (14,400 bytes per second) and you are attempting to stream 8 bytes worth of data at 2000 Hz (16,000 bytes per second), the data generation rate exceeds what the baud rate can handle and so this situation will occur. When this occurs, the sensor will skip packets when the communication interface is not ready to receive additional data. So in the example provided, it is expected that there is 500µs between each packet. However, if packets start getting dropped, there may sometimes be 1000µs, 1500µs or some higher multiple of 500µs between packets depending on how long it takes the data to transfer. The sensor will not generate a new data sample as soon as there is room to transfer data, it will continue to attempt to output data at the time intervals defined by the set rate, and simply skip outputting data for a given time frame if there is no room.

2.3.4 Additional Settings

The following are additional settings that can be used to modify streaming behavior, but are not required.

stream_delay

This setting will cause streaming to start a specified time in seconds after the *Start Streaming* command is sent.

Example to receive data immediately (default): `!stream_delay=0\n`
Example to receive data after 2.5 seconds: `!stream_delay=2.5\n`

stream_mode

This setting controls whether *stream_duration* or *stream_count* is used to determine when streaming ends.

Example to use *stream_duration* (default): `!stream_mode=0\n`
Example to use *stream_count*: `!stream_mode=1\n`

stream_duration

This setting controls how long data will be streamed before stopping when *stream_mode* is 0. Streaming will automatically end after the specified time in seconds. *stream_duration* is time actually spent outputting data, *stream_delay* is not counted towards part of time spent streaming.

Example to stream indefinitely: `!stream_duration=0\n`
Example to stream for 1 minute: `!stream_duration=60.0\n`

stream_count

This setting specifies how many data samples streaming will output before it stops streaming. This setting is only used if *stream_mode* is set to 1.

Example to stream for 100 samples: `!stream_count=100\n`
Example to stream for 20,000 samples: `!stream_count=20000\n`

2.4 Data Logging

The on-board data logging feature is available on 3-Space Sensor models that contain a microSD card reader with a properly formatted SD card inserted. These models will contain "Data Logger v3" in the model name and "DL3" in the SKU.

For ways to log data on other sensors, please see the 3-Space Sensor Suite manual for a GUI-based approach, or look at our script examples to see how to use our API to stream data and then save the data to a file.

2.4.1 Data Capture Options

The following sections describe the configuration properties that determine data-logging capture behavior.

2.4.1.1 Output File Settings

Inside each subdirectory will be the data file (Continuous Mode) or files (Periodic Mode) associated with the data logging session, and optionally a "settings.cfg" file.

The data files will be named after the "`log_base_filename`" setting (the default value is "data"), an incrementer starting at 0, and then a ".csv" (if Ascii Mode) or ".bin" (if Binary Mode) file extension. For example, the first filename with default settings would be "data0.csv". For more information on the difference between Ascii Mode and Binary Mode, please see section 3.1

If the setting "`log_output_settings`" is set to 1 (default), then there will additionally be a "settings.cfg" file that will contain a copy of the DL3's settings as they were when the data logging session began.

2.4.1.2 Log Slots Setting

The "`log_slots`" setting specifies which commands the DL3 will use to save data to the SD card. This settings operation is identical to the "`stream_slots`" setting used for data streaming in section 2.3.3. For more information look towards that section.

2.4.1.3 Log Hertz Setting

The "`log_hz`" setting specifies the number of samples per second the DL3 will attempt to write to the SD card. The actual number of samples per second can be lower if the SD card is not performant enough, or if too much data is trying to be written per logging cycle. The maximum possible value is 2000.

2.4.1.4 Continuous Mode and Periodic Mode

The "`log_style`" setting determines if the sensor is in Continuous Mode (0) or Periodic Mode (1).

- If set to 0, data will be logged continuously until the session ends.
- If set to 1, data will be logged for the number of seconds specified in the "`log_periodic_capture_time`" setting, and then will pause for the number of seconds specified in the "`log_periodic_rest_time`" setting (after which data logging will resume). This cycle will continue until the session ends.

If the DL3 is in Periodic Mode, the "`log_file_mode`" setting determines if each capture cycle will be in the same file (set to 0), or in separate files (set to 1).

2.4.1.5 Start Event Settings

The Start Event settings determine under what conditions a data logging session will begin. Data logging sessions can always be started with the "`Start Logging 60(0x3C)`" command. Additional start events can be set using the "`log_start_event`" setting. Any combination of additional start events can be set using the following integers 0-4, which correspond to the following:

ID	Name	Description
0	Button	The session will start when the power button is clicked.
1	Accel Threshold	The session will start when there is an accelerometer event such as an impact. Accelerometer vector lengths will be checked in a 100ms window, and if the difference between the minimum and maximum lengths in any given window exceeds the value specified in the "`log_start_motion_threshold`" setting, a data logging session will begin. A higher value means there must be a larger change in motion before the data logging session starts.
2	None	If the only set start event, sessions will *only* start with the "Start Logging" command.
3	GPS Fix	The session will start when the GPS gets a fix (on G models only).
4	On Power	The session will start as soon as the DL3 boots and is ready.

Example:
"`!log_start_event=0,3\n`" allows additionally starting via the button or when a GPS Fix is obtained.

2.4.1.6 Stop Event Settings

The Stop Event settings determine under what conditions a data logging session will end. Data logging sessions can always be stopped with the "`Stop Logging 61(0x3D)`" command. Additional stop events can be set using the "`log_stop_event`" setting. Any combination of additional stop events can be set using the following integers 0-5, which correspond to the following:

ID	Name	Description
0	Button	The session will stop when the power button is clicked.
1	Accel Threshold	The session will stop when there are no accelerometer events detected for a specified amount of time. Accelerometer vector lengths will be checked in a 100ms window, and if the difference between the minimum and maximum lengths is lower than the value specified in the "`log_stop_motion_threshold`" setting for the amount of time specified in the "`log_stop_motion_delay`" setting, the data logging session will end.
2	None	If the only set stop event, sessions will *only* stop with the "Stop Logging" command.
3	Duration	The session will stop when the amount of time that has passed while actively logging reaches the number of seconds specified in the "`log_stop_duration`" setting. In Periodic Mode, only capture time is counted; rest time is ignored.
4	Capture Count	The session will stop when the log contains the number of samples specified in the "`log_stop_count`" setting.
5	Period Count	The session will stop when the number of periodic capture sessions reaches the amount specified in the "`log_stop_period_count`" setting. This setting only works in Periodic Mode ("`log_style`" is set to 1).

Example:
"`!log_stop_event=1,3\n`" allows additionally stopping when readings fall below the accel threshold or after a specified duration.

2.4.1.7 Ascii Mode and Binary Mode

Data logging can be configured to be done in Ascii or Binary mode using the "`log_data_mode`" setting. If set to 1, logging will be done in Ascii Mode, if set to 2, logging will be done in Binary Mode. Note that unlike data streaming, Ascii/Binary Mode is not affected by sending the "`Start Logging`" command in Ascii or Binary.

In Ascii Mode, the log file/s will be created as ".csv" files (comma separated values). The first line of the file will contain the column names. The column names will be in the order of the commands set in the "log_slots" setting, and the names will be identical to the names returned by the "Get Streaming Command Label" (0x53) command.

In Binary Mode, the log file/s will be created as ".bin" files, and will contain the raw binary of the data returned by the commands registered in the "log_slots" setting. This is identical to saving the results of binary data streaming into a file. Unlike ascii mode, there are no labels indicating the type of data in the file.

2.4.1.8 Header Settings

The header can be included in the log files by setting "log_header_enabled" to 1. If enabled, the header will be included before each data sample. In ascii mode, this means additional columns will be added for the header fields that are enabled. Note that unlike with data streaming, the header is separated from the data by a comma instead of a semicolon since it is stored in a CSV.

2.4.1.9 Immediate Mode Settings

Immediate mode is a way to simultaneously data log and read the data being logged to a file at the same time. To enable this, set "log_immediate_output_enabled" to 1. The communication interface for sending out the data will be the last interface used to send data to the device.

When receiving data from the sensor via this method while logging, the output can be paused and resumed. This can be done using the command "Pause Log Streaming (0x57)" which when passed a 1, pauses the log streaming, and when passed 0, resumes it. When it resumes, it continues from where it was paused. This does not pause the logging session, only the data being sent.

Because the immediate streaming can be paused, and the communication interface may be slower than the SD card, the data being received is not necessarily received in real time. This is actually a variation of *File Streaming* as described in the *File Streaming/Bulk Output* section. This means the data logged to the file is sent as fast as possible over the communication interface while there is data available to send. So as long as the sensor is configured in a way it can keep up with the data logging, the data will be received in real time. If it can not keep up, the data is not in real time. If the streaming is paused and resumed, the sensor will stream data faster then it is being logged (if possible) until it catches up.

After logging is stopped, the data streaming will also stop, whether it finished sending all the data that was recorded in the file or not. To check if there is any data unread and to continue reading data, use command "Get Last Live Logging Location (0xAA)". This command returns the cursor index to continue reading from, and the file path the data was written to. With this information, you can open the logged file, set the cursor, and then continue reading data from the file using the File System commands, as described in the File System section of this manual.

Immediate Mode Header

Immediate mode logging has its own separate enableable header. This can be enabled by setting "log_immediate_output_header_enabled" to 1. This header can also be configured to be in Ascii Mode, Binary Mode, or match the mode of data logging. This is done using "log_immediate_output_header_mode" where 0 causes it to match "log_data_mode", 1 is Ascii Mode, and 2 is Binary Mode.

The immediate mode header is not logged to the data file. It is a header for the packet that is being streamed from the file. Its command echo byte, if enabled, will show as "Read Bytes (0xB1)". It is recommended to use the header when programmatically parsing the streamed data with the *data_len* field enabled. That way, the application can know exactly how many bytes to read each time, especially as it is possible to stop logging/pause streaming in the middle of a data packet. In general, the same rules and recommendations that apply to *File Streaming* also apply to live logging.

2.5 File System

3-Space Sensor models that contain the on-board data logging features, such as the "DL3", will also include an on-board file system. The file system is used to record data from data logging sessions, and can be accessed using the file system commands. The file system can also be accessed from a computer via the file explorer due to the sensor emulating a mass storage device. Note that if the sensor uses a removable form of media for its file system, such as a Micro SD card, the file system will not be accessible if the media is not present.

2.5.1 Mass Storage Device

The DL3 exposes the contents of its File System to a computer by enumerating as a USB Mass Storage Device when the sensor is plugged into the computer with a compatible USB Type-C cable (USB-C 2.0 or higher) and when the setting command "?fs_msc_enabled" returns "fs_msc_enabled=1", which is the default value for this

setting. This is in addition to the 3-Space Sensor Family's normal behavior of enumerating as a virtual COM port.

In this mode, when connected to a computer, the computer is in control of the File System, and the sensor will not be able to write any data to prevent corruption, so certain commands may error. In order to have the DL3 take back control of the File System, eject the drive per your computer's OS, and then the DL3 will be able to control the File System. To allow the computer to access the File System again after ejecting it, set "!fs_msc_enabled=1" (This must be done even if the value of "fs_msc_enabled" is already 1).

2.5.1.1 Auto Mass Storage Control

The setting called "`fs_msc_auto`" can be set to 1 to allow the DL3 to automatically take control of the File System when the DL3 requires it, and then give control back to the OS when no longer needed. Because of the way this is done, this is essentially the same as unplugging removable media from your computer without first ejecting the media, and thus can have the same problems when enabled.

Compatibility:

Windows - Starting in Windows 10 version 1809, the default media removal policy was changed to "Quick Removal", which prevents write caching and allows the media to be safely removed at any time without first ejecting. This allows the "sd_msc_auto" feature to safely work. If you have changed this setting, do not use this feature.

MacOS - Mac is NOT safe to use this feature, as it performs caching operations on the device itself even when not in active use. It is fairly likely to corrupt the File System if this setting is used with Mac.

Linux - Dependent on OS

General - If your OS has a way to disable write caching, allowing the ability to safely remove media at any time, such as the Windows Quick Removal setting, then "fs_msc_auto" can be safely used as long as that feature is enabled. Overall, the safest way to manage the File System is to not use this setting. This setting is purely for convenience but has strict limitations.

Note: On units with hardware revision YL-TSSDL3.0f and lower (prerelease units), SD card functionality will be disrupted on a power source change. This includes if the DL3 is running on battery power, and then is plugged in to charge. Any currently active data logging sessions will be halted and any unflushed data will be lost.

2.5.2 SD Card Format and Directory Structure

2.5.2.1 SD Card Format

To format or reformat the file system (WARNING: THIS WILL DELETE ALL DATA), you can send command 59 (0x3B) and the DL3 will format to an appropriate file system based on the type of media the sensor uses to support its filesystem. For example, an SD card will be formatted to an appropriate FAT-based File System. As this formatting process will remove all data on the media/sensor and cannot be reversed, please check for and back up any data currently on the media/sensor before issuing the format command.

After the formatting process completes, the file system will be blank with no files or folders. This is normal, as the DL3 sensor only creates files and folders as they are needed in response to user initiated commands. This includes committing settings and starting data logging.

The DATA and CONFIG folders will be created when the next data logging session starts. Additionally, the CONFIG folder will be created (if it does not exist yet) when settings are committed to the sensor.

NOTE: Because calling commit attempts to save files to the SD card, the commit command can fail if the sensor is not in control of the SD card at the time (this includes sending the "`!commit`" setting and also sending command 225(0xE1) over ascii or binary). In this event the debug system will also generate error messages if enabled. If sent via "`!commit`" the error response will be "8,0".

2.5.2.2 Directory Structure

The DL3 Sensor uses the following directory structure:

<Root Directory>:
- /CONFIG
  - /sensor.cfg
- /DATA
  - /<YYYY-MM-DD_HH-MM-SS> or <session-01>
    - /<data0.csv>
    - /<data1.csv>
    - /<data2.csv>
    - …
    - /settings.cfg
  - /<YYYY-MM-DD_HH-MM-SS> or <session-02>
    - …

/<YYYY-MM-DD_HH-MM-SS> or <session-03>

…

…

The CONFIG directory contains the sensor.cfg file. The settings contained in this file will be applied to the sensor when the sensor reboots, when "`!fs_cfg_load`" is sent, or when the sensor.cfg file is modified and saved. The file will be changed to match the sensor's committed settings when the "`!commit`" setting is sent.

WARNING: When modifying and saving the sensor.cfg file, all settings are reloaded, not just the modified ones. If the "fs_msc_enabled" in the file has a value of 0 (disabled), then upon saving the file, the mass storage will disable itself.

The DATA directory contains the results of any data capture sessions. A new subdirectory will be created for each data capture session. If the `log_folder_mode` setting is set to 0, then the first subdirectory will be named "session-" followed by an incrementer. If the "`log_folder_mode`" setting is set to 1, then the subdirectory will be named based on the date and time currently saved on the real-time clock component.

2.5.3 Media Formats

2.5.3.1 SD

When an SD card is detected, the DL3 Sensor will attempt to mount the SD card. If the SD card mounts successfully, the DL3's LED will blink cyan/teal once. If there is an error mounting the SD card, the DL3's LED will blink orange once. Most mounting issues will be because the SD card is unformatted, or is formatted with an incompatible file system. The DL3 primarily supports FAT, FAT32 and exFAT file systems.

To reformat the card, please see the File System Format section. When formatting via the format command, the file system type will be automatically determined based on the size of the inserted SD card.

2.5.4 File System Commands

The file system has multiple commands that can be used to browse directories, read files, and delete files from the system. This allows accessing data on the sensor when using a connection interface other than USB or when using a device that is not capable of enumerating the USB port as a Mass Storage Device. An example would be reading data stored on a file over a wireless connection such as bluetooth.

2.5.4.1 Paths

Multiple commands take and return paths as strings. These paths are relative to the current working directory of the file system. The current working directory defaults to the root directory and can be changed via the `"Change Directory" (0xAC)` command. When specifying a path, a forward slash '/' is used as the directory separator.

Path	Description
sensor.cfg	A file in the current directory
session-01/data0.csv	A file in another folder relative to the current directory
/	The root directory
/config	A folder in the root directory
.	The current directory
..	Parent directory of the current directory
../file.txt	A file in the parent directory of the current directory

WARNING: ".." does not work if the media is formatted as exFAT and will instead be treated as "." and stay in the current directory.

2.5.4.2 Navigating Directories

To browse files stored on the sensor, the `"Get Next Directory Item" (0xAB)` and `"Change Directory" (0xAC)` commands are used. By utilizing these commands together, the user can discover and navigate to all items stored by the file system.

Get Next Directory

Each call to this command will return a single item in the current directory. Upon reaching the last item, an empty identifier will be set to indicate the end has been reached. Once the end has been reached, the next call will loop back to the first item in the directory.

The response returns three values: Type_ID, Item Name, and Size. The Type_ID is used to indicate what type the returning item is and has the following possible responses:

Type_ID	Type
0x00	File
0x01	Directory
0x80	End Of Directory
0xFF	Error/Reload

The Item Name is a string that is the name of the File or Directory. In the case of End of Directory, the name will be an empty string. In the case of an error, the name is undefined. When an error occurs, it could be due to a change in directory structure or reloading the SD card. To recover from this condition, Change Directory back to the root directory.

The size is the size in bytes of the returned file. This field is only valid for file types and will be 0 for any other type.

Change Directory

The change directory command can be used to change the current working directory of the file system. This only affects the results of file system commands. When passing in a path to a command, the path should be relative to the current working directory. This means, given the Path "/data/session-01", that directory can be opened from the root directory like so:

1. `":0xAC,data/session-01\r\n"`
2. `":0xAC,data\r\n"` followed by `":0xAC,session-01\r\n"`

To return to the root directory, send `":0xAC,/\r\n"`. For more information on paths that can be used for changing directories, look at the above "Paths" section.

2.5.4.3 Reading Files

To read a file, the file must first be opened using the `"Open File" (0xAD)` command and then any of the available reading commands. Once done reading, the file should be closed via the `"Close File" (0xAE)` command. Only one file can be open at a time.

Opening/Closing Files

To open a file, simply pass the path of the file to the `"Open File" (0xAD)` command. To close the currently opened file, call the `"Close File" (0xAE)` command, no additional inputs are required.

Read Line

The `"Read Line" (0xB0)` command can be called to read data up until a newline character '\n' appears or a NULL character '\0'. The returned string will not include the newline character or carriage return characters. If the line has no data on it, an empty string will be returned.

If the line is too long (more than 4000 characters) the returned result will be truncated, but the file cursor will still advance past the found newline.

When the End of File is reached, a '\xFF' character will be returned to indicate the end of the file. Therefore, to read a whole file using this command, you can repeatedly call `"Read Line"` until the last byte received is 0xFF.

This command is meant to be used only on files that contain Ascii data.

Read Bytes

The `"Read Bytes" (0xB1)` command returns the next requested number of bytes of data from the file. The maximum amount of bytes that can be requested at once is 4000. If the number of bytes requested exceeds the remaining bytes in the file, the output after the end will be padded with 0xFF until the requested number of bytes have been output.

Cursor

Whenever a read command is called, the cursor in the file advances however many bytes are read, and the next read will start from that position in the file. The cursor can be manually set using the `"Set Cursor Index" (0xB3)` command.

To better know the current position in the file, the user can also send the `"Get Remaining File Size" (0xAF)` command. The value returned by this is the number of bytes until the end of the file from the current cursor position.

2.5.4.4 File Streaming/Bulk Output

File streaming is a method of reading a file as fast as possible. It is enabled by calling the `"Stream File" (0xB4)` command while a file is open. When called, the remaining size of the current file will be returned to notify the caller how many data bytes to expect.

Unlike regular streaming, the rate at which the data is output is not configurable. Rather, once you start file streaming, the `"Read Bytes" (0xB1)` command will be automatically output as fast as possible until End Of File is reached. The file will be output in chunks of 512 bytes, with the last chunk being the size of the remaining data.
Warning: Unlike sending the actual `"Read Bytes"` command, when using Ascii Mode, file streaming will NOT add the additional "\r\n" at the end of the data like directly calling the `"Read Bytes"` command does. This is done intentionally to make it so that when file streaming from a terminal in ascii mode without the header, the result is having the file directly printed to the terminal as is without any formatting breaks.

If file streaming is started with a header start byte, the header will be prepended to each output. When the header is enabled the Echo Byte in the header will be the same as the `"Read Bytes" (0xB1)` command.

It is highly recommended to do file streaming with the header enabled and with the header_length enabled when using binary mode. This allows the user to always know how much data is in a packet by reading the data length field in the header.

The `"Stop Streaming File" (0xB5)` command can be called at any time during file streaming to stop the output of data. Doing so will leave the file cursor where it left off, so calling the `"Stream File" (0xB4)` command again without first closing the file will result in file streaming continuing from where it left off.

File streaming will automatically end when the end of file is reached. The file will still remain open until the user closes it.

2.5.4.5 Modifying Files

Currently, the only operation available to modify files via the sensor commands is `"Delete File/Folder" (0xB2)`. When this command is called, the file at the path given to the command will be deleted. If the given path is a folder instead, the folder, along with all its contents, will be deleted.

2.6 Status LED

The 3-Space™ Sensor contains an RGB LED used to indicate the status of the device and events that occur. LED status is split into two types: static color states and blinking color events. Static colors are used to indicate a State, while blinks are used to indicate events.

2.6.1 Static Color States

A static color is the color shown by the LED when no events are occuring. When an event does occur, the event's color will blink, and then the led will return to the static color state. If multiple states are true at the same time, the color shown will be based on a priority system, with 0 being the highest priority.

State	Color	Priority	Notes
Idle (default)	Blue	4	Color can be changed via the "led_rgb" setting
Shut off from Battery	Red	0	
Logging	Purple	2	Due to the activity LED event happening every logged data packet, this may look like blinking purple and cyan.
Charging	Yellow	3	
Charged	Green	3	
GPS has fix	Orange	1	Enabled via the "gps_led" setting. Off by default.

2.6.2 Event Colors

An event is indicated by the LED blinking a color 1 or more times. It is possible for multiple events to occur sequentially, for example an Activity blink followed by a Debug blink.

Event	Color	Behavior
Activity Packet	Light Blue	When the sensor processes a command or is outputting streaming or logging packets, this event will trigger. This blink event is limited to 4 times per second.
Debug	Magenta	Blinks once every 3 seconds while a debug message is available. Can be enabled/disabled by the "debug_led" setting.
Passive Calibration	Clear/Off	Continuously blinks while Passive Calibration is being performed
Active	Purple	Continuously blinks while Active Calibration is being performed.

Calibration		
Missing Component	White	Blinks once per second if a component fails to be detected on startup.
SD Card Valid	Cyan	On SD card insert, blinks once if successfully initialized.
SD Card Failed	Orange	On SD card insert, blinks once if failed to initialize.
Failed to Log	Orange	Blinks once to indicate no stream slots or needs ejected, blinks twice to indicate a file system error.
Logging Dropped Packet	Orange	Blinks to indicate logging can not keep up with the requested rate. Limited to twice per second.
Battery Low	Red	Blinks to indicate battery is below 10%. Continues to blink once every 2.5 seconds.
Hardfault	Magenta	If a hard-fault occurs, the LED will alternate between Magenta and Black/Off. Can be enabled/disabled by the "debug_fault" setting. (Off by default)
Watchdog Timeout	Orange	If a watchdog timeout occurs, the LED will alternate between Orange and Black/Off. Can be enabled/disabled by the "debug_wdt" setting. (Off by default)

2.7 Power Management

The 3-Space™ Sensor contains a variety of settings that can be configured to reduce or increase power usage. This section describes the various settings that should be considered to better manage power usage.

2.7.1 Presets

To quickly save power without worrying about individual settings, there is the 'pm_mode' setting. This setting is a write only setting that takes an integer to set the power level of the sensor, with the value ranging from 0-MAX where MAX is different per sensor type. Most sensors will have MAX as 3. Lower values use less power, while higher values use more.

To set the power mode, send a command with the following structure:
`'!pm_mode={level}\n'`

The general presets are as follow, but may differ on some sensor types:

Level	Name	Properties

0	Low	CPU Speed 48 MHz, ODR 500 Hz
1	Normal	CPU Speed 96 MHz, ODR 1000 Hz
2	High	CPU Speed 144 MHz, ODR 1500 Hz
3	Turbo	CPU Speed 196 MHz, ODR 2000 Hz

2.7.2 Individual Power Settings

pm_idle_enabled

This setting allows the 3-Space™ Sensor to enter an idle mode when there is no processing required at the current time. It defaults to enabled. Generally this both saves power and increases performance and should not be changed.

cpu_speed

The speed the processor runs at in megahertz. Has a large impact on both the power drain and performance of the 3-Space™ Sensor. Supported speeds may differ per sensor. This setting requires a sensor restart to take effect. This setting does not need to be committed to persist.
This setting has a read only setting associated with it called 'cpu_speed_cur'. This read only setting gives the actual current speed of the processor in MHz. This only differs from 'cpu_speed' when the user changes 'cpu_speed' but has yet to restart the sensor.

ODR

The output data rate (ODR) of the individual Accel/Mag/Gyro/Baro components also affect the power consumption of the 3-Space™ Sensor. The higher the ODR, the higher the power consumption and vice versa. To minimize power consumption, it is recommended to set the ODR of the components to the minimum value required for your application.

Setting ODR can be done per component, component type, or all at once. The basic form of these ODR settings are as follows:

Key	R/W	Description
odr_all	W	Sets the ODR of all components as close to the supplied value as possible.

odr_{*type*}	W	Sets the ODR of all components of the specified *type* as close to the supplied value as possible.
odr_{type}{id}	R/W	Sets the ODR of the specified component as close to the supplied value as possible.

The minimum ODR is 1, and will result in the most power savings.

Streaming/Data Logging

Streaming and Data Logging increase power consumption based on how fast and how much data is being streamed per packet. There are no additional tips for managing power when using these events other than to use the lowest 'stream_hz'/'log_hz' setting you can and stream as little data as you can.
Note: Data logging only applies to sensor types that support it.

2.7.3 Device Specific Power Settings

GPS

The GPS uses a significant amount of power. The power consumption can be reduced by putting the GPS into standby mode when it is not needed. Doing so is one of the biggest contributors to reducing power usage on the 3-Space™ Sensor. To put the GPS into standby use the 'gps_standby' setting.
Example: `"!gps_standby=1\n"`

Also, be aware that the GPS uses more power when attempting to get a fix. Once the GPS has a fix, its power consumption will drop. However, even with a fix, putting the GPS into standby will still significantly reduce the power usage of the 3-Space™ Sensor.

2.7.4 Power Considerations

The following are components/events that can significantly affect the devices power usage but may not currently have a way to reduce the power consumption outside of simply not using those features.

Bluetooth

Devices with bluetooth capability may see a large jump in power usage when the sensor is sending/receiving messages quickly over a Bluetooth/BLE connection. This is most noticeable when wirelessly streaming data over a bluetooth connection.

SD

Devices with an SD card may see large jumps in power usage when the SD card is being initialized. This only occurs when the SD card is initially inserted into the sensor, the sensor is initially powered on while an SD card is already inserted, or when reformatting the SD card. Outside of this initialization process the power usage from the SD card is generally stable, however small power increases may be noticed when the SD card creates files, such as at the time data-logging is started, or when actively writing to a file, such as while actively data-logging.

2.8 Extended Embedded Pedestrian Tracking System: EEPTS™

Each sensor in the new 3-Space™ 3.0 Family, except for the LX, has the ability to run a pedestrian tracking algorithm designed to work in GPS-compromised environments. This technology, referred to as EEPTS™ (Extended Embedded Pedestrian Tracking System), uses inertial sensor data, AHRS data (orientation estimation and linear acceleration from our QGRAD3™ filter), and knowledge of human biomechanics to compute a step-by-step motion path of a human-worn or human-carried sensor.

For each step taken, the sensor logs multiple data items, such as the heading of travel, the displacements in travel, and the estimated longitude and latitude. For more information, see "Output format of `YL_EEPTS_OUTPUT_DATA`" below.

For best results, the EEPTS™ should be supplied with as many valid GPS coordinates as possible. This will allow the sensor to automatically adjust and calibrate step lengths on a per-user basis. In situations where this is not possible, it is recommended to supply at least the estimated starting coordinates. This will allow the sensor to use the WMM (World Magnetic Model) to correct for changes in Earth's magnetic field and to use geographic north instead of magnetic north.

2.8.1 General Use

2.8.1.1 Configuration

Configuration of the EEPTS system should generally be done through the use of 'presets' that have been defined via the settings protocol. These presets apply multiple EEPTS settings all at once for the configuration they represent. Some of these presets enable more capabilities than the others, but for the most accurate results, you should only enable features that are needed. For example, the 'pts_preset_motion' can enable Walking/Running (WR), or Walking/Running/Crawling (WRC). The WRC can do everything the WR can, however WR will generally be more accurate than WRC, but will not work with crawling at all.

Typical Setup:

1. Mount/attach the sensor to the person / worn equipment / carried equipment.
2. If you have previously edited any pedestrian tracking settings and would like to reset them now, send `!pts_default\n` to the sensor in ASCII mode.
3. Set your desired **motion preset** by sending the following setting via ASCII mode.
    a. If you know that there will be ***no*** crawling, send `!pts_preset_motion=0\n`
    b. If you know that there ***could*** be or ***will*** be crawling, send `!pts_preset_motion=1\n`
4. Set your desired **heading preset** by sending the following via ASCII mode.
    a. For orientation independent operation, send `!pts_preset_heading=0\n`
    b. For orientation dependent operation/tilt compensated, send `!pts_preset_heading=1\n`
5. Set your desired **hand preset** by sending the following setting via ASCII mode. (This must be set last as it may overwrite some settings done by **heading preset**)
    a. If the sensor will ***not*** be mounted near or carried in one hand, send `!pts_preset_hand=0\n`
    b. If the sensor ***could*** be mounted near or carried in one hand, send `!pts_preset_hand=1\n`
    c. If the sensor will ***only*** be mounted near or carried in one hand, send `!pts_preset_hand=2\n`
6. If **heading preset** was set to **1** skip to step 9
7. Have the person face north and stand how they will normally be standing.
    a. If the sensor is on worn equipment, the person should be wearing the equipment normally.

    b. If the sensor is on carried equipment, the person should be carrying the equipment how they will be normally carrying the equipment.
8. Use command `74(0x4A) EEPTS Auto Offset` to set the orientation offset.
    a. This will allow orientation-dependent functions such as Tilt Heading to work properly.
    b. To check the offset, send `?pts_offset_quat\n` to the sensor in ASCII mode.
9. If you would like to have these settings persist through power cycles, send `!commit\n` to the sensor in ASCII mode.

2.8.1.2 Running EEPTS

Once configuration is done, simply enable pedestrian tracking by using command `68(0x44) EEPTS Start`. At this point the sensor is automatically obtaining data and processing it through EEPTS to compute steps and heading. If you have an external GPS it is recommended to frequently call command `73 (0x49)` and pass in the latitude and longitude of the external GPS.

To receive the step data from EEPTS after it has been started, the commands `70 (0x46) EEPTS Get Oldest Step` and `71 (0x47) EEPTS Get Newest Step` can be used. The 10 most recent steps will be stored on the sensor and can be retrieved via these commands. Each command returns a YL_EEPTS_OUTPUT_DATA struct as defined in section 2.7.2. If there are no steps available in the internal buffer, then the last step that was received via either of these commands is repeated. Duplicates can be identified via the segment_count field of the output. Alternatively, the command `72 (0x48) EEPTS Get Available Step Count` can be sent to know how many steps are ready to be read out. All 3 of these commands are also streamable.

Once done, call command `69(0x45) EEPTS Stop` to stop the data collection and computation of steps and allow start to be called again to effectively reset the EEPTS system.

Note that when initially starting EEPTS and using the Orientation Independent heading mode, the heading may not initially match the user's actual heading until after they take enough steps that the sensor can confidently and accurately determine the user's walking direction. At this point, the system will self correct itself and appear stable.

2.8.1.3 Best Mounting Practices

For best results, attach to a tight-fitting chest harness, or place on or near the sternum. Alternatives are the waist/belt (minimize bouncing), back, shoulder. Mounting on arms, hands, legs, and feet may produce suboptimal results.

2.8.2 Data Output Structure

The commands `70 (0x46) EEPTS Get Oldest Step` and `71 (0x47) EEPTS Get Newest Step` output the data in the order of the following structure:

```
struct YL_EEPTS_OUTPUT_DATA {
	uint32_t segment_count;
	uint32_t timestamp;
	double estimated_gps_longitude;
	double estimated_gps_latitude;
	float estimated_gps_altitude;
	float estimated_heading_angle;
	float estimated_distance_travelled;
	float estimated_distance_x;
	float estimated_distance_y;
	float estimated_distance_z;
	uint8_t estimated_locomotion_mode;
	uint8_t estimated_receiver_location;
	float last_event_confidence;
	float overall_confidence;
};
```

Name	Description
segment_count	A segment is the same as a step. This is the step ID for this step.
timestamp	In microseconds the time of this step
estimated_gps_longitude	In decimal degrees (positive east, negative west)
estimated_gps_latitude	In decimal degrees (positive north, negative south)
estimated_gps_altitude	In meters
estimated_heading_angle	In degrees (0 north, 90 east, 180 south, 270 west)
estimated_distance_travelled	Total distance travelled in meters since starting EEPTS

estimated_distance_x	Meters travelled for just this step along positive east/negative west
estimated_distance_y	Meters travelled for just this step along positive north/negative south
estimated_distance_z	Meters travelled for just this step along positive up/negative down
estimated_locomotion_mode	Estimated motion mode. 0 - Idle 1 - Walking 2 - Jogging 3 - Running 4 - Crawling 5 - Unknown 6 - Other
estimated_receiver_location	Estimated sensor location. 0 - Unknown 1 - Chest 2 - Waist 3 - Back 4 - Hand 5 - Shoulder
last_event_confidence	Confidence in the last step between. Range 0-1.
overall_confidence	Average confidence of all steps. Range 0-1.

2.8.3 Additional Settings

Outside of the settings automatically handled by presets, there are also some additional settings the user may want to configure for their use case.

2.8.3.1 Magnetic Declination

When computing the heading, there may need to be a constant offset applied to the heading to align magnetic north with true north. This value is known as the magnetic declination. This value defaults to 0 and has two ways to be set:

1. `'pts_mag_declination={angle}\n'`
    a. Set this setting using an angle in degrees between 0 and 360
2. `'pts_auto_declination=1\n'`

a. Enabling auto declination allows EEPTS to automatically set the magnetic declination based on its current GPS location and the WMM.
b. After setting this, the user must supply at least one GPS point via command `73 (0x49) EEPTS Insert GPS` while EEPTS is running.

2.8.3.2 Distance Estimator Scalars

Distance estimation per step may change from person to person based on their walking gait. To help tune the distance per user, multipliers can be assigned to the computed distance based on the detected motion mode (Walking/Running/Crawling…). There are two primary ways to do this:

1. `'pts_estimator_scalars={idle},{walk},{jog},{run},{crawl},{unknown},{other}\n'`
    a. Each {} is a multiplier between 0.1 and 5 that is applied for that motion mode.
    b. Defaults to '1,1,1,1,1,1,1'
2. `'pts_auto_estimator_scalar_rate={update_rate}\n'`
    a. {update_rate} is a value from 0-25 where 0 is disabled
    b. If enabled, then EEPTS will automatically modify the estimator_scalars after {update_rate} consistent samples of a given motion type with GPS input.
        i. It simply takes the computed distance via GPS and compares it to the estimated distance to determine the appropriate multiplier
        ii. The larger the value of {update_rate}, the more accurate the new scalars will be, but also the longer it will take to modify the scalars.
    c. Requires constant GPS input. If no GPS data is being given, the scalars will not be automatically updated.

2.8.3.3 World Magnetic Model

The world magnetic model is used by EEPTS to automatically set settings, do live calibration, and increase accuracy by using nearby magnetic information. The world magnetic model is updated every 5 years and is stored as a list of coefficients that are then used to compute the magnetic information for a given location at a given time. For this reason, the user may need to supply the current date/time to the WMM for the highest accuracy, or update the WMM to a new version.

The time used by the WMM defaults to the version date of the sensor's firmware. If the sensor has a RTC (Real Time Clock) it will automatically use the date/time from this component, otherwise the user should set the date/time before starting EEPTS. To

manually set the date time, send `'pts_date={day},{month},{year}\n'` where day is 1-31, month is 1-12, and year is the full 4 digit year.

Because the WMM is updated every 5 years, you may need to manually update your WMM. This is only required if the result date of `'?pts_wmm_version\n'` is outdated by 5 years. To update the WMM, download the coefficient file (wmm.cof) from the NOAA (National Oceanic and Atmospheric Administration) and send `'pts_wmm_set="{file_contents}"\n'`. To verify that it was properly set, read back the version using the above mentioned setting. The WMM is not committed to the sensor, meaning this would be required after every power cycle. Alternatively, when the WMM is updated, download the newest version of Firmware for your sensor as Yost Labs will update the default WMM built into the firmware when it changes.

2.8.3.4 Debug

The EEPTS debug system works the same as described in section 3.3 Debug Handling, but with different settings. The EEPTS debug system uses the settings 'pts_debug_level' and '`pts_debug_module`'.

Levels:

```
Error = 0x1
Warning = 0x2
Info = 0x4
```

Modules:

```
PIPELINE = 0x1
SEGMENTER = 0x2
EEPTS = 0x4
CLASSIFIER = 0x8
MOTION_CLASSIFIER = 0x10
PROFILER = 0x20
HEADING = 0x40
ORIENT = 0x80
DISTANCE_ESTIMATOR = 0x100
LOCATION_CLASSIFIER = 0x200
WMM = 0x400
ASSERT = 0x800
DATA = 0x1000
```

3 Communication Protocol

3.1 Command Protocol Overview

The 3-Space™ Sensor receives messages from the controlling system in the form of sequences of serial communication bytes called packets. For ease of use and flexibility of operation, two methods of encoding commands are provided: binary and text. Binary encoding is more compact, more efficient, and easier to access programmatically. ASCII text encoding is more verbose and less efficient yet is easier to read and easier to access via a traditional terminal interface. Both binary and ASCII text encoding methods share an identical command structure and support the entire 3-Space™ command set.

The 3-Space™ Sensor buffers the incoming command stream and will only take an action once the entire packet has been received. Incomplete packets, and also packets with incorrect checksums (binary mode only), will be ignored. This allows the controlling system to send command data at leisure without loss of functionality. The command buffer will, however, be cleared whenever the 3- Space Sensor is either reset or powered off/on.

3.1.1 General Overview

The structure of a command is as follows:
{*start_byte*}{*command_number*}{$param_{0\text{-}n}$}{*terminator*}

The available start bytes and the format of the following command elements are different depending on if using ascii or binary protocol.

Command Numbers are in the range of 0-255, and parameters can be any data type as required for the given command. Commands require 0 or more parameters. If 0 parameters are required, the *command_number* is immediately followed by the terminator. Section 4 of this manual contains a table of all available commands, their required parameters, and their output data.

The response to a command will contain the data relevant to the given command. If the command was sent using ascii mode, the response will be in ascii, and vice versa. Additional information may be included at the beginning of the response based on if the used *start_byte* enables the *Header* and what parts of the *Header* are enabled via the settings.

3.1.2 Ascii Command Protocol

In this mode all data will be passed in using ascii characters. Data will be formatted as strings, such as "1.75". The *command_number* and *params* will be separated from each other via commas. Ascii mode also recognizes backspaces when building commands to allow undoing any typos in partially completed commands. Once a *terminator* is received the command will execute.

3.1.2.1 Format

Start Byte	Function
: (colon)	No header, standard start
; (semi-colon)	Header enabled

Terminator
\r (carriage return)
\n (new line / line feed)

Responses to ascii commands will also comma separate its returned values and will always terminate with '\r\n' (carriage return and new line). If the header is enabled, and the command also returns its own data, a semicolon will be used to separate the header from data.

3.1.2.2 Examples

Color code:

Start_Byte Cmd_Number Params Terminator

Command	Response
:0\n	-0.019029,0.077614,0.095470,-0.992220\r\n
:95,1000\n	No response
;95,1000\n	0,95\r\n
;39\n	0,39;-0.189819,0.968445,-0.028259\r\n

For the above commands, when the header is used, treating as if the *status* and *echo* fields of the header are enabled.

3.1.3 Binary Command Protocol

In this mode all data will be passed in binary format using little endian order. For example, if a float is being passed in, it is represented in its 4 byte binary representation. The *terminator* is replaced as a single byte checksum of the given command.

Start Byte	Function
0xF7	No header, standard start
0xF9	Header enabled

3.1.3.1 Terminator / Checksum

In binary mode, the terminator is a single byte arithmetic checksum of the *command_number* and each byte in the *params* of the command, otherwise known as the sum of each individual byte in the command (besides the *start_byte*) modulus 256.

Responses to binary commands will contain the data in binary format packed, and in the order specified in the command table in section 4.

3.1.3.2 Examples

Color code:
Start_Byte Cmd_Number Params Terminator/Checksum

Command	Response
0xF7 0x00 0x00	0xB5 0xE2 0x9B 0xBC 0x17 0xF4 0x9E 0x3D 0xC6 0x85 0xC3 0x3D 0x21 0x02 0x7E 0xbF
0xF7 0x5F 0x08 0x3E 0x00 0x00 0x00 0x00 0x00 0x00 0xA5	No Response
0xF9 0x5F 0x08 0x3E 0x00 0x00 0x00 0x00 0x00 0x00 0xA5	0x00 0x5F
0xF9 0x27 0x27	0x00 0x27 0xE9 0x5F 0x42 0xBE 0x03 0xEC 0x77 0x3F 0x6B 0x7F 0xE7 0xBC

For the above commands, when the header is used, treating as if the *status* and *echo* fields of the header are enabled.
Note: These are identical to the examples in the ascii section, just with the command and response formatted for binary mode.

3.1.4 Response Header

The 3-Space Sensor is capable of returning additional data that can be prepended to all command responses, referred to as the *Header*. This capability is managed via the Response Header Bitfield, which can be configured using the "`header`" setting, or with the following aliases:

Bit #	Alias	Description	Data Type
0	header_status	0 = Success, Non-zero = Error	S8
1	header_timestamp	Microseconds since power on	U32
2	header_echo	Last issued command number	U8
3	header_checksum	Additive data checksum	U8
4	header_serial	Sensor's short serial number	U32
5	header_length	length of data bytes/chars	U16

Note: header_length and header_checksum do not include the header itself in their calculation.

Example: "`!header=63\n`" would enable all options.

Example: "`!header_timestamp=1\n`" would enable the timestamp bit.

Example: "`!header_checksum=0\n`" would disable the checksum bit.

Each bit in the field, if enabled, corresponds to a different piece of information that will be output prior to the expected response data. For a detailed explanation of each header option, please see the System Settings section.

When the header is prepended to the returned data, the header is output in the following order: *status* > *timestamp* > *echo* > *checksum* > *serial* > *length*. This means that *status* is always returned first if enabled, and *length* is always returned last if enabled.

See section 3.1.2.2 for Ascii examples and section 3.1.3.2 for Binary examples.

3.2 Settings Protocol Overview

For configuration the 3-Space™ Sensor uses a separate Key Value Pair protocol to write and read settings. The keys used for settings are case insensitive. Some settings also provide a way of calling commands via a key with no attached value.

3.2.1 Writing Settings

Writing a setting has the following format:

'!{key}={value}\n'

key is a case insensitive key name and *value* is an ascii representation of the data that key takes. For example '!range_accel3=8\n' will set accel3 to have a range of 8g.

Some settings may require multiple values to be passed to them, in that case the additional values are comma separated. The format is:

'!{key}={value$_0$},{value$_1$},...,{value$_n$}\n'

An example of this type of setting would be '!calib_bias_accel3=0.01,-0.02,0.035\n' which sets the calibration bias for accel3 using the given 3 component vector.

3.2.1.1 Writing Multiple Settings (Bulk Write)

Multiple settings can be written at the same time without having to send multiple writes by putting a semicolon ';' between sequential key value pairs. Therefore the full format for writing settings is the following:

'!{key$_0$}={value$_0$};{key$_1$}={value$_1$};...;{key$_n$}={value$_n$}\n'

An example of this would be '!header=0;header_status=1;header_timestamp=1\n' which resets the header and then enables the status and timestamp portion of the header.

The settings are written in the order they are given when doing this. If any of the settings fails to write due to a format error, the bulk write will immediately stop and any key value pairs after the one that failed will not be processed.

Warning: Because settings are fully buffered before they are executed, there is a max size setting string that can be sent at any given time. This limit is 2048 characters. If this size is exceeded, a warning debug message will be generated and the contents of the buffer will be discarded.

3.2.1.2 Response

After writing any number of settings, an ascii response of two integers will be sent back with a success/error response code, and the number of successful writes. The format is:

```
'{error_code},{num_successful_writes}\r\n'
```

For example, if writing 3 settings at once succeeds, the response will be '0,3\n'. However, if an error occurs, the *num_successful_writes* can be used to determine what key errored. If sending '!header=0;invalid_key=7;stream_hz=500\n', the response would be '2,1\r\n' which indicates an error due to an invalid key and 1 successful write, meaning the second key failed.

Common Error Codes:

Code	Meaning
0	Success
1	Generic Error
2	Invalid Key/Readonly
3	Invalid Value

Do note that some keys may be readonly and not have an associated write. In this situation, they will report as an invalid key when attempting to write.

3.2.1.3 Command Keys

Some keys are what are referred to as command keys. Command keys do not take any values, they simply execute some form of behavior. To call a command key, you use the write settings protocol and simply leave off the '=' and value.

For example, '!commit\n' will cause the sensor to commit all of its settings, or '!default\n' will cause the sensor to revert to default settings.

Command keys can be intermixed in bulk writes just like any regular setting, for example:

```
'!default;header_timestamp=2;commit\n'
```

This will restore settings to default, enable the timestamp in the header, and then commit all settings to the sensor.

3.2.2 Reading Settings

Reading keys has the following format:

```
'?{key}\n'
```

key is a case insensitive key name. The response will be in the form of:

```
'{key}={value}\r\n'
```

where *key* is the key name and *value* is an ascii representation of the data stored in that key. If a key is invalid, the response will instead be:

```
'<KEY_ERROR>\r\n'
```

3.2.2.1 Reading Multiple Settings (Bulk Read)

Multiple settings can be read at the same time without having to send multiple reads by putting a semicolon ';' between sequential keys. Therefore the full format for reading settings is the following:

$$\text{'}\{key_0\};\{key_1\};...,\{key_n\}\backslash n\text{'}$$

The response will be identical to the singular read except with semicolons separating the different key value pairs. The response will also be in the same order as the requested read.

For example:

```
'?header;invalid_key;cpu_speed\n'
```

may return

```
'header=0;<KEY_ERROR>;cpu_speed=96000000\r\n'
```

It is known that *invalid_key* is the key that failed because the <*KEY_ERROR*> is at the same index as *invalid_key*.

Do note that some keys are write only and will return a <*KEY_ERROR*> when attempting to read them. Write only keys are typically alias settings that are used to configure other settings either via a computation or via a preset.

3.2.2.2 Querying Settings

Readable settings can be queried via a query string. By using a query string, any key that includes the query string as a substring will be output in the same way a *Bulk Read* is done. The format for a query string is:

```
'?{{query_string}}\n'
```

For example:

```
'?{odr}\n'
```

Will return all settings with the substring 'odr' as part of the key which would look like:

```
'odr_accel3=1000;odr_accel4=1000;odr_gyro2=1000;odr_mag0=50
0;odr_baro0=75\r\n'
```

A query string is treated just like any other key and so can be used in the *Bulk Read* protocol as well. Just ensure the query string is surrounded in curly brackets '{}'.

3.2.2.3 Aggregate Settings

Some keys provide the ability to retrieve multiple settings all at once with just one key. These keys are retrieved the same way as any ordinary key, just they will output multiple key value pairs as if doing a *Bulk Read*. These keys are naturally readonly.

Example Aggregate Settings:

Key	Description
all	Returns all readable key value pairs
settings	Returns all unique key value pairs that are both readable and writable. Excludes duplicate aliases.
pts_settings	Returns all EEPTS settings.

3.2.2.4 Reapplying Settings

The response format of reading settings is exactly the same as the format used to write settings. What this means is that the result of a read can be saved off and then sent back to the sensor using the write settings protocol to reapply all the read settings. If doing this, just make sure not to cross the 2048 character limit as described in section 3.2.1.1. Reading settings can output any size of a string, so it is possible that the result of a read would need to be split into smaller sections before sending back into the write to ensure it fits in the character limit.

3.2.3 Alias settings

Alias settings are settings that have more then one key that can read/write the same value. An example of this are the *header* keys. The 'header' key is a bitfield that represents what components of the header are enabled. However, there are also keys such as 'header_status', 'header_timestamp', 'header_echo'... These header keys are aliases that can be used to write and read the individual bits of the 'header' setting.

It is also common for write-only keys to be alias settings. These types of settings provide a way to configure other settings either via a computation or a preset. For instance, 'pm_mode' is a setting that allows assigning a power management profile. It sets various settings such as ODR and 'cpu_speed' but is not itself readable as it is a preset rather than its own setting.

3.2.4 Data Types

The data used to write settings and the results from the read follow the same general format for different data types. Some data types also have multiple ways of inputting them. This is a list of the expected types.

Type	Description	Example
Unsigned	A positive integer. Can be input in	27 or 0x1B or 0b11011

Integer	decimal, hexadecimal, or binary notation.	
Signed Integer	A positive or negative integer.	-5
Float	A decimal value. No scientific notation.	0.8999 or .8999
String	Any combination of ascii characters.	This is a string
List	A list can contain any number of other data types, they are simply comma separated from each other.	1,-2,0.567,string

3.2.4.1 String Escape Characters

Because strings can contain any characters, including newlines and commas, a string may conflict with the ascii protocol. This could occur due to a comma in the string being treated as a list separator, or a newline in the string being treated as the terminator character. To avoid these situations, escape characters have been added in the form of quotations "" and backslash '\'.

Putting a backslash '\' before a character forces the character that comes after the backslash to be treated as part of the string. So `'My\,String\,'` will be treated as the string `"My,String,"` and neither of those commas will be considered data separators. In order to have a literal backslash in the string, you would need to send `'\\'` which would evaluate to `"\"`.

Alternatively, anything within quotes is automatically considered a string. Therefore, "My,String," will properly handle treating the commas as part of the string without having to put a backslash in front of them. However, this does mean that to have a quotation mark as part of the string, the quotation must always be escaped by a backslash. For example: `"Yost Labs Once Said \"Quote\"!"`

In general, when supplying a string as data in ascii mode, it is always a good idea to surround it in quotations. Also make sure to put a backslash before any literal quotations in the string or any literal backslashes in the string.

WARNING: Because escape characters can also escape newlines and carriage returns, which are used by the sensor to determine the incoming command/settings has been completed when in ascii mode, the user must ensure they properly close quotes otherwise the sensor will never be able to finalize the command until the quote is closed.

3.3 Debug Handling

The 3-Space™ Sensor has built-in support for various debug messages to help notify the user of any errors that may occur, as well as status information about the sensor. These messages could occur due to improper usage, such as calling a command with invalid settings, or unexpected events, such as an SD card filling up completely or failing to write. Debug messages are able to be programmatically enabled, disabled, and filtered via the settings protocol.

A debug message has the following form:
{timestamp_us} Level: {debug_level)} Module: {debug_module} {msg}
Here is an example:
159297473 Level: 0x4 Module: 0x1 Battery status changed: 3

3.3.1 Reading Debug Messages

When a debug message is generated the sensor's led will blink purple once every 3 seconds to indicate that there are available debug messages to be read out. To read out debug messages, there are two commands:

- `126 (0x7E) Get Debug Message Count`
- `127 (0x7F) Get Debug Message`

The first command returns the number of available debug messages to read out. The second command gets the actual debug message from the sensor as a string. If there is no available debug message and `Get Debug Message` is called, the sensor will instead return an empty string indicated by a NULL byte (0x00).

The buffer used to store debug messages before the user calls commands to read them out does have a limited size. This size is based on the length of the message rather than message count, so the upper limit of number of messages is based on the size of what messages are being generated as well. If the debug buffer is full and a new message is generated, the oldest messages will be removed to make room for the new message.

3.3.1.1 Debug Mode

There are two methods of reading out debug messages. The above mentioned method is the primary and default way debug messages are generated and read out. It is referred to as the *Buffered Debug Mode*. The other debug mode is *Immediate Mode*. In immediate mode, when a debug message is generated, it is instantly output over the current communication interface. It is generally preferable to use *Buffered Debug Mode* when programmatically communicating with the sensor to avoid any spontaneous messages that may cause misinterpretation of data, and *Immediate Mode* if actively using it from a terminal window to be able to instantly see debug messages in response to any action taken.

To change the debug mode, send the following setting:

- `'!debug_mode=0\n'` `Buffered Debug Mode`
- `'!debug_mode=1\n'` `Immediate Debug Mode`

When transitioning from *Buffered Debug Mode* to *Immediate Debug Mode* any messages currently buffered will be immediately output. Therefore, setting *debug_mode* to *Immediate* and then setting it back to *Buffered* can be an effective way to instantly receive all debug messages without having to manually call the `Get Debug Message` command multiple times.

3.3.2 Filtering Messages

To control what debug messages are generated, there are two configurable settings: 'debug_level' and 'debug_module'. Each of these settings is a bitfield that allows enabling/disabling different levels and modules. Every debug message has a level and module associated with it, as can be seen in the example debug message. For a debug message to be generated, both its level and module must be enabled.

3.3.2.1 Debug Levels

The following debug levels are available:

Name	Value	Description	Example Situation
Error	0x01	Indicates an unexpected error has occurred.	Components failed to be detected on startup.
Warning	0x02	Indicates an event that may affect data accuracy.	Failing to achieve the set streaming rate.

Info	0x04	General event information.	Charging status changed.

Default: 0x01 - Only errors enabled.

3.3.2.2 Debug Modules

The following debug modules are available:

Name	Value
Main	0x01
Assert	0x02
Flash	0x04
Settings	0x08
Timer	0x10
SPI	0x20
Gyro	0x40
Accel	0x80
Mag	0x100
Components	0x200
QGRAD3	0x400
PWM	0x800
LED	0x1000
Baro	0x2000
EEPTS	0x4000

Name	Value
Initialization	0x8000
Data Management System	0x10000
Protocol	0x20000
Commands	0x40000
Stream	0x80000
Interrupt	0x100000
RTC (Real Time Clock)	0x200000
Auto Calibration	0x400000
WMM (World Magnetic Model)	0x800000
Orientation	0x1000000
Bluetooth	0x2000000
Button	0x4000000
SD	0x8000000
GPS	0x10000000

Default: `0x0FFFFFF`

3.4 USB

When connecting a 3-Space™ Sensor to a computer via USB, it appears as a COM port. This COM port provides a serial interface for communication between the host and the sensor unit using protocol messages. The specific name of the COM port depends on the operating system being used. If you have multiple 3-Space™ Sensors connected to a single computer, each sensor will be assigned its own unique COM port.

The easiest way to find out which COM port belongs to a certain sensor is to take note of what COM port appears when that sensor is plugged in(provided the drivers have been installed on that computer already. Otherwise, find out what COM port appears once driver installation has finished.) Additionally, each sensor can be identified programmatically by reading the serial number of each attached sensor

3.5 UART

The 3-Space™ Sensor Uart uses the following settings for communication:

- Data bits: 8
- Parity: None
- Stop bits: 1

The supported baud rates are as follows:

- 4000000
- 2000000
- 921600
- 460800
- 230400
- 115200 (Default)
- 57600
- 38400
- 19200
- 9600
- 4800

3.6 Transactional Protocol

The transactional protocol is an additional wrapper around the base ascii/binary protocol used to help facilitate communication when using synchronous communication interfaces such as SPI and I2C. This is necessary to allow the master device to know

when valid data is available and understand which bytes are the response to a given command.

This protocol is referred to as the transactional protocol because it splits the communication into separate transaction types consisting of:

1. Sending
    a. Sending a command using the base ascii/binary protocol
2. Requesting
    a. Request the data generated by a sent command to be output
3. Response
    a. Reading the response of a request

Additional information about each interface that follows the transactional protocol will follow the overview of the transactional protocol.

3.6.1 Transactional Start Byte

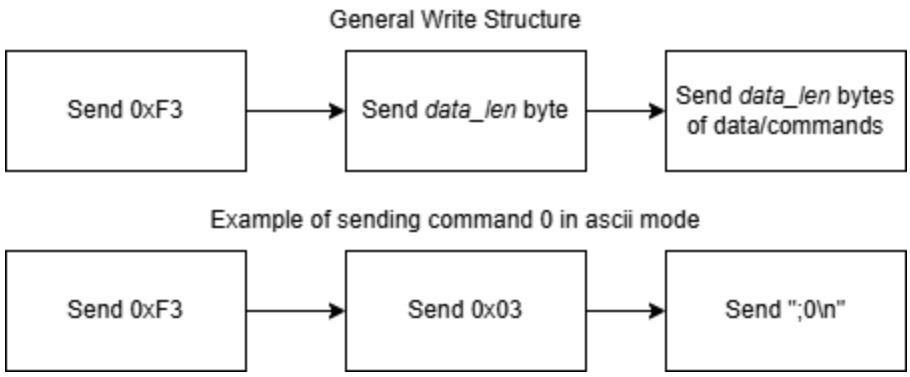
Each Transaction starts via sending one of the possible *Transactional Start Bytes*. Any bytes other than these will be ignored. These bytes are used to start both reading and writing operations. Aside from the transactional start byte and any additional metadata bytes that may be required, the data to be sent and read is identical to the base ascii/binary command protocol.

The possible *Transactional Start Bytes* are as follow:

- 0xF0 - Request Status
- 0xF1 - Read Data
- 0xF2 - Read Data with Max Size
    - Following bytes specifies max amount of data to read
- 0xF3 - Write Data
    - Following byte specifies amount of data to write

3.6.2 Sending Commands

To send commands/data to the 3-Space™ Sensor, the user must start a *write* transaction by sending the *Write Byte* 0xF3. Immediately following the *Write Byte* the user must then send a byte with a value between 0-255 that indicates the amount of data the user is about to write to the device. To write more than 255 bytes multiple send commands can be chained by repeating the send process.



3.6.3 Read Requests

To retrieve data from the 3-Space™ Sensor after a command is sent, the user must first request data using the *Read Start Bytes* that are specific to the transactional protocol as described in section 3.7.1.

After any of these *Read Start Bytes* are sent, the transactional protocol will transition to the *Response* state and will begin outputting data to the user. During this time, the sensor will not accept additional commands/start bytes until the response is fully read out. After the response is fully read out, it will transition back to the *Send* state.

3.6.3 Read Response

After a *Read Request* is received, the 3-Space™ Sensor will send a response based on the request type. In order to read the response to a request, a new transaction must be started. This could take the form of a duplicate start condition for example. More information about this will be in the individual communication interface sections such as I2C and SPI. Once a response is fully read out, the protocol will transition back to the *Send* state.

3.6.3.1 Response Types

The *Request Status (0xF0)* start byte causes the sensor to output only the 2 byte status header.

The *Read Data (0xF1)* start byte causes the sensor to output the 2 byte status header, followed by up to 255 bytes of data. The amount of data output after the header is the value of the second byte (data length byte) in the header. The data output will match the format of the base ascii/binary protocol.

The *Read Data with Max Size (0xF2)* start byte is identical to the *Read Data* start byte with the addition of requiring the user to send another byte indicating how much data

the user wants to read. This additional byte is not the amount that will be sent back, but rather the upper limit. For example, if 100 bytes are available to be read, but the user sends 50 as the second byte, then only 50 bytes will be sent back. However, if there are only 30 bytes available and the user sends 50 as the second byte, then only 30 bytes will be sent back. In either scenario, the second byte (data length byte) in the response header will indicate the actual amount of data sent back. The data output will match the format of the base ascii/binary protocol.

3.6.3.2 Status Header

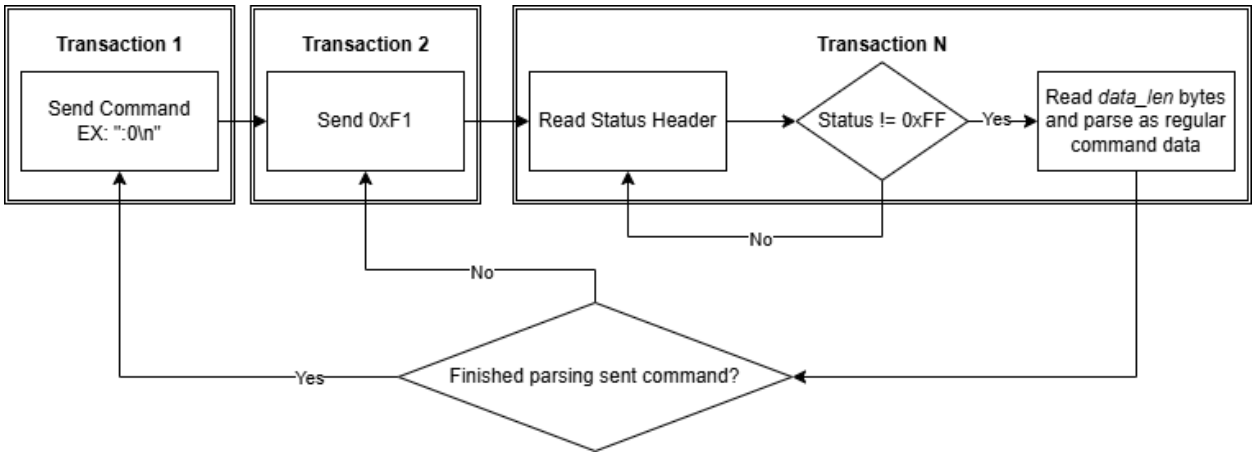
The status header is always sent in response to a request. It is 2 unsigned bytes where the first byte is a status indicator, and the second byte is the amount of data available to read (Does not include this 2 byte header).

The status byte indicates the current operation of the protocol and can be one of the following values:

- 0x00 - Idle/Complete
    - No base command is being actively processed/previous command finished processing.
- 0x01 - Gathering
    - Has received a protocol start byte and is currently gathering a command from the user
- 0x02 - Processing
    - A command has been received and its results are currently being computed and prepared for output
- 0xFF - No Response/Not Loaded

When no response data is loaded, meaning either a Read Request was never sent or all the data from the last read request was received, the sensor will continuously respond with 0xFF to reads. It is possible to attempt to read a response before the response has been loaded, in which case the status byte of the header will read as 0xFF. If this occurs, the user should start a new transaction and attempt to read the response again (It is not necessary to send the request byte again).

3.6.4 General Procedure



Note: Transaction 1 & 2 can be combined. They are separated to show they don't need to be together.

3.6.5 Speed Improvements

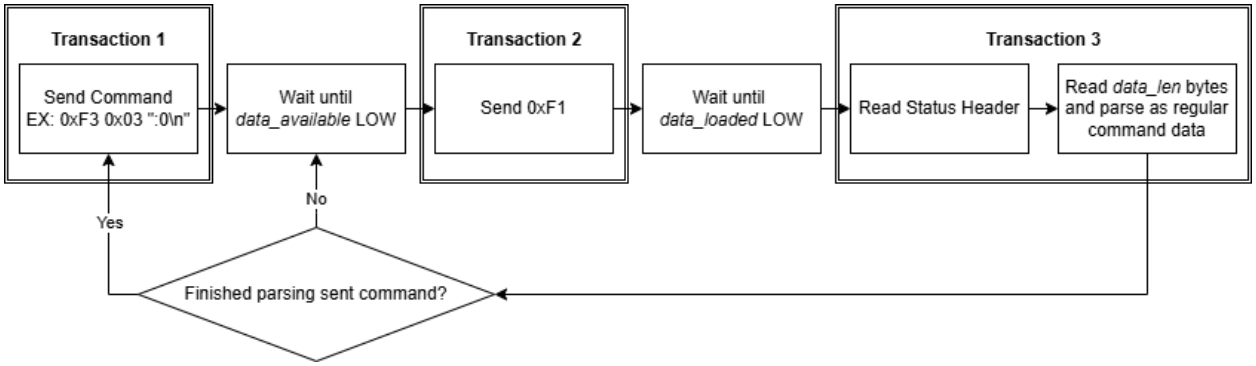
It is possible to send a read request before the previously sent command has finished processing. However, any request to read has a non-zero amount of processing time associated with it regardless of if there is data available at that time or not. Because of this, any request to read before the previous command has been fully processed by the 3-Space™ Sensor will effectively delay the amount of time it takes to finish processing the command. This can create a loop where constantly polling responses can take significantly longer to be received, and in extreme cases effectively lock up the sensor.
Note: This section is meant as a way of optimizing speed, and is generally not required. Also, the following methods will only improve speed if you are currently experiencing read requests that return a data length of zero.

3.6.5.1 IRQ Mode

To avoid requesting data before it is available, the UART Tx/Rx pins can be configured to act as interrupt pins to signal when data is *available*/*loaded*. To enable this functionality the user must send "!pin_mode0=8\n". Doing so will result in the following configuration:

- Uart TX -> *data_available*
    - LOW when there is data available to be sent in response to a request
- Uart RX -> *data_loaded*
    - LOW when the response to a request is ready to be sent. This is the same as the status byte in the request response header not being 0xFF.

The following procedure is the preferred way of communicating when using IRQ pins.



When IRQ mode is enabled, it is not required to use the pins, they are optional to help improve read speed. If limited on available IO pins, it is recommended to only use the *data_available* pin as that is the primary speed increase. If doing so, continue to check the status byte of the header as shown in the *General Procedure* diagram instead of waiting for *data_loaded* to go high.

If using the *data_loaded* pin, it is recommended to wait until *data_loaded* goes back high before sending another *Read Request* such as 0xF1.

3.6.5.2 Delay

To avoid requesting data before it is available, it may be optimal to place a delay in between sending the command and requesting a response. The length of this delay should match the amount of time it takes the sensor to process the command for best results. However, not all commands have the same processing time, and changing the sensor's power settings, such as CPU speed, can also affect the processing time. Therefore, experimentation may be required to find the best time for each command. It is preferable to use IRQ Mode if possible.

3.7 SPI

The 3-Space™ Sensor SPI interface works with the transactional protocol. A transaction in SPI is considered any data sent/read during a single instance of selecting and releasing the SS (slave select) line.

3.7.1 Pin Information

Name	Info

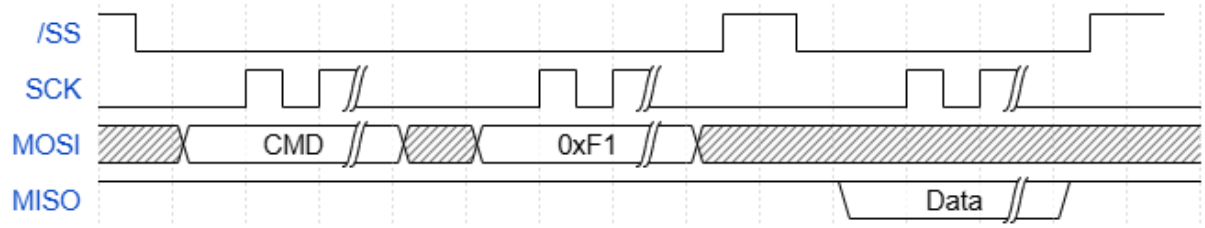
SCK	Internal Pull-up
MISO	
MOSI	Internal Pull-up
/SS	Internal Pull-up, Active Low

3.7.2 Configuration

4 Wire SPI
Up to 10 MHZ
SPI Mode 0 (CPOL=0, CPHA=0)
    SCK idles low
    Bit set on falling edge of SCK and read on rising edge of SCK
Sends data MSB (Most Significant Bit) first

By default, the SPI pins of the 3-Space™ Sensor are enabled. To change the behavior of the SPI pins, or restore SPI functionality, look at the "pin_mode1" setting.
Warning: If the SS pin is actively pulled low when the 3-Space™ Sensor initially powers on, the SPI pins will instead default to being I2C pins. The pin does have an internal pull-up, so if disconnected, will default to SPI.

3.7.3 Example Transaction



Take special note of the repeated SS start/stop patterns in between the 0xF1 *Read Byte* and reading out the data. This repetition is required whenever transitioning from writing to reading. Also, the *Data* field includes the response header as described in the Transactional Protocol.

While not necessary, it is also recommended to keep the MOSI line high while reading data from MISO.

3.7.4 Clock Speed

The 3-Space™ Sensor SPI interface supports clock rates up to 10MHz. However, depending on the CPU speed of the sensor, the maximum rate may be limited. The following are suggestions based on set CPU speed:

CPU Speed	Recommended	Maximum SPI Clock Rate
48MHz	1MHz	2.5MHz
96MHz (default)	2MHz	5MHz
144MHz	3.5MHz	8MHz
192 MHz	5MHz	10MHz

The "Recommended" column contains clock speeds that when used at that CPU Speed should result in minimal delay due to 0 length data reads as described in section 3.7.5. The actual suggested maximum clock rate has its own column. If increasing the clock up to the maximum speed, we recommend using one of the methods listed in section 3.7.5 to reduce 0 length data reads.

3.8 I2C

The 3-Space™ Sensor I2C interface works with the transactional protocol. The I2C communication interface shares the same pins as the SPI interface. To use I2C, the pins must be configured via sending "!pin_mode1=3\n", or by pulling the SS pin low before the 3-Space™ Sensor initializes. For more pin configuration information refer to the *settings* table.

3.8.1 Pin Information

Name	Info
SCL	Internal Pull-up
SDA	Internal Pull-up

While these pins have internal pull-ups enabled by the 3-Space™ Sensor, external pull-ups are still required for a clean signal and communicating at faster speeds.

3.8.2 Configuration

By default, the I2C pins of the 3-Space™ Sensor are not enabled. I2C can be enabled in the following ways:

- Connect the SS pin to GND when powering on.
- Send "!pin_mode1=3\n". For more information about the *pin_mode* settings refer to the *settings* table.

Uses a 7-Bit device address. Defaults to 0x77.
The device address can be configured via the *"i2c_addr"* setting. It takes a value in the range 1-127 (0x01-0x7F) as the new address. The new address will not take effect until the I2C bus is reinitialized. This is done by either setting *"pin_mode1"* to I2C from a value other than I2C, or by restarting the sensor. This allows the address to be changed via I2C, but the change will not take effect until the sensor is restarted. The setting also must be committed before the restart occurs.

Here is a list of addresses that are exceptions and not allowed:

- 0x01-0x08
- 0x0C
- 0x28
- 0x37
- 0x61
- 0x78-0x7F

3.8.3 Example Transaction

Sending a command:

Start	I2C Address (default 0x77)							R/W	ACK	Write Byte	Data_Len	Command	Stop
S	1	1	1	0	1	1	1	W	0	0xF3	EX: 3	EX: ":0\n"	P

Reading response:

Start	I2C Address (default 0x77)							R/W	ACK	Read Byte	Repeat Start	I2C Address (default 0x77)							R/W	ACK
S	1	1	1	0	1	1	1	W	0	0xF1	Sr	1	1	1	0	1	1	1	R	0

Status Byte	ACK	Data_Len Byte	ACK	Data Byte	ACK

Data Byte	NACK	Stop

While the above diagram uses a *Repeated Start* when reading a response, this is not required. It can also be a Stop followed by a Start.

This is the same as using the Transactional Protocol as described, just with I2C Writes and Reads being used instead. Because of this, all reading does not need to be done in the same transaction. However, all reading should be done before performing another write as that is required by the transactional protocol. An example of this would be to send F1, then read the status and data_len byte. If the data_len byte is 20, you could read 10 bytes, send a stop condition, and then at a later point send the I2C address with the Read bit and continue reading the last 10 bytes from the previous request.

3.8.4 Clock Speed

The 3-Space™ Sensor I2C interface supports the following speeds:

- Standard mode (100 kbps)
- Fast mode (400 kbps)
- Fast mode plus (1 Mbps)
- High speed mode (3.4Mbps)

At any CPU Speed the Fast Mode Plus (1 Mbps) is fully supported. The max achievable speed, up to 3.4Mbps will vary depending on current power/processor settings.

4 Command List

4.1 Orientation Commands

Command	Command Name	Description	Input Length	Input	Output Length	Output
0 (0x00)	Get Tared Orientation	Returns the filtered and tared orientation estimate in quaternion form. Note: The imaginary vector is represented with XYZ and the real scalar is represented with W.	0		16	Quaternion XYZW (Float x4)
1 (0x01)	Get Tared Orientation As	Returns the filtered and tared orientation	0		12	Euler Angles XYZ (Float

	Euler Angles	estimate in Euler angle form.				x3)
2 (0x02)	Get Tared Orientation As Rotation Matrix	Returns the filtered and tared orientation estimate in rotation matrix form. See the calibration section for more details.	0		36	Rotation Matrix: r0 r1 r2 r3 r4 r5 r6 r7 r8 (Float x9)
3 (0x03)	Get Tared Orientation As Axis Angles	Returns the filtered and tared orientation estimate in axis-angle form. Note: The angle is given in radians.	0		16	Axis XYZ (Float x3) Angle (Float)
4 (0x04)	Get Tared Orientation As Two Vector	Returns the filtered and tared orientation estimate in two-vector form, where the first vector refers to forward and the second refers to down. This represents the direction the physical local forward/down (Natural Z and -Y Axis) vectors of the sensor are pointing in global space	0		24	Forward Vector XYZ (Float x3) Down Vector XYZ (Float x3)
5 (0x05)	Get Difference Quaternion	Returns the difference between the measured orientation from last frame and this frame. Note: The imaginary vector is represented with XYZ and the real scalar is represented with W.	0		16	Quaternion XYZW (Float x4)
6 (0x06)	Get Untared Orientation	Returns the filtered and untared orientation estimate in quaternion form. Note: The imaginary vector is represented with XYZ and the real scalar is represented with W.	0		16	Quaternion XYZW (Float x4)

7 (0x07)	Get Untared Orientation As Euler Angles	Returns the filtered and untared orientation estimate in Euler angle form.	0		12	Euler Angles XYZ (Float x3)
8 (0x08)	Get Untared Orientation As Rotation Matrix	Returns the filtered and untared orientation estimate in rotation matrix form. See the calibration section for more details.	0		36	Rotation Matrix: r0 r1 r2 r3 r4 r5 r6 r7 r8 (Float x9)
9 (0x09)	Get Untared Orientation As Axis Angles	Returns the filtered untared orientation estimate in axis-angle form	0		16	Axis XYZ (Float x3) Angle (Float)
10 (0x0A)	Get Untared Orientation As Two Vector	Returns the filtered and untared orientation estimate in two-vector form, where the first vector refers to forward and the second refers to down. This represents the direction the physical local forward/down (Natural Z and -Y Axis) vectors of the sensor are pointing in global space	0		24	North Vector XYZ (Float x3) Gravity Vector XYZ (Float x3)
11 (0x0B)	Get Tared Two Vector In Sensor Frame	Returns the filtered, tared orientation estimate in two vector form, where the first vector refers to forward and the second refers to down. These vectors are given in the sensor reference frame and not the global reference frame. These vectors represent the local axis of the sensor that is currently pointing towards the globally defined forward and down vectors (default north and gravity, modified by tare).	0		24	North Vector XYZ (Float x3) Gravity Vector XYZ (Float x3)

12 (0x0C)	Get Untared Two Vector In Sensor Frame	Returns the filtered, untared orientation estimate in two vector form, where the first vector refers to north and the second refers to down/gravity. These vectors are given in the sensor reference frame and not the global reference frame. These vectors represent the local axis of the sensor that is currently pointing towards the globally defined north and down/gravity vectors.	0		24	North Vector XYZ (Float x3) Gravity Vector XYZ (Float x3)

4.2 Normalized Data Commands

Command	Command Name	Description	Input Length	Input	Output Length	Output
32 (0x20)	Get All Normalized Component Sensor Data	Returns the Normalized Gyro Rate Vector, Normalized Accelerometer Vector, and Normalized Magnetometer Vector. All normalized vectors are the corrected vectors changed to be unit-length, and have no magnitude data associated with them.	0		36	Direction of Rotation XYZ (Float x3) Direction of Gravity XYZ (Float x3) Direction of Mag. North XYZ (Float x3)
33 (0x21)	Get Normalized Gyro Rate Vector	Returns the gyro rate vector, with calibrations applied, with unit-vector length, for axes XYZ. Note: This unit-length vector shows a relative level of rotation to all other directions and has no magnitude data. Warning: The normalized	0		12	Direction of Rotation XYZ (Float x3)

		gyro rate vector will give erratic results when the magnitude of the corrected gyro rate vector is near zero. Consider checking the magnitude of the corrected gyro rate vector and throw out results under a certain threshold, or consider using the Get Corrected Gyro Rate Vector command instead.				
34 (0x22)	Get Normalized Accelerometer Vector	Returns the accelerometer vector, with calibrations applied, with unit-vector length, for axes XYZ. Note: This unit-length vector shows the direction of acceleration and has no magnitude data.	0		12	Direction of Gravity XYZ (Float x3)
35 (0x23)	Get Normalized Magnetometer Vector	Returns the magnetometer vector, with calibrations applied, with unit-vector length, for axes XYZ. Note: This unit-length vector shows the direction of magnetic north and has no magnitude data.	0		12	Direction of Mag. North XYZ (Float x3)
51 (0x33)	Get Normalized Gyro Rate By ID	Pass the ID number of the gyroscope to query. Returns the Normalized Gyro Rate Vector for axes XYZ.	1	Gyro ID Number (U8)	12	Direction of Rotation XYZ (Float x3)
52 (0x34)	Get Normalized Accelerometer Vector By ID	Pass the ID number of the accelerometer to query. Returns the Normalized Accelerometer Vector for axes XYZ.	1	Accel ID Number (U8)	12	Direction of Gravity XYZ (Float x3)

53 (0x35)	Get Normalized Magnetometer Vector By ID	Pass the ID number of the magnetometer to query. Returns the Normalized Magnetometer Vector for axes XYZ.	1	Mag ID Number (U8)	12	Direction of Mag. North XYZ (Float x3)

4.3 Corrected Data Commands

Command	Command Name	Description	Input Length	Input	Output Length	Output
37 (0x25)	Get All Corrected Component Sensor Data	Returns the Corrected Gyro Rate Vector (radians/sec) for axes XYZ, the Corrected Accelerometer Vector (g) for axes XYZ, and the Corrected Magnetometer Vector (gauss) for axes XYZ. Note that this acceleration will include the static component of acceleration due to gravity.	0		36	C. Gyro Vec XYZ (Float x3) C. Accel Vec XYZ (Float x3) C. Mag Vec XYZ (Float x3)
38 (0x26)	Get Corrected Gyro Rate Vector	Returns the gyro rate vector with calibrations applied, in units of radians/sec, for axes XYZ.	0		12	C. Gyro Vec X (Float) C. Gyro Vec Y (Float) C. Gyro Vec Z (Float)
39 (0x27)	Get Corrected Accelerometer Vector	Returns the accelerometer vector with calibrations applied, in units of g, for axes XYZ. Note that this acceleration will include the static component of acceleration due to gravity.	0		12	C. Accel Vec X (Float) C. Accel Vec Y (Float) C. Accel Vec Z (Float)

40 (0x28)	Get Corrected Magnetometer Vector	Returns the magnetometer vector with calibrations applied, in units of gauss, for axes XYZ.	0		12	C. Mag Vec X (Float) C. Mag Vec Y (Float) C. Mag Vec Z (Float)
41 (0x29)	Get Corrected Global Linear Acceleration	Returns the Corrected Global Linear Acceleration of the device, in units of g, using the current axis_order(defaults to axes East, Up, North). Note that this is the overall acceleration that has been orientation compensated, uses the untared orientation, and does not include the acceleration due to gravity.	0		12	C. Gled. LinAccel East (Float) C. G. LinAccel Up (Float) C. G. LinAccel North (Float)
42 (0x2A)	Get Corrected Local Linear Acceleration	Returns the Corrected Local Linear Acceleration, in units of g, for axes XYZ. Note that this acceleration does not include the acceleration due to gravity.	0		12	C. L. LinAccel X (Float) C. L. LinAccel Y (Float) C. L. LinAccel Z (Float)
48 (0x30)	Correct Raw Gyro Rate Vector	Converts the supplied raw data gyroscope vector to its corrected data representation. Pass the Raw Gyro Rate Vector in units of radians/sec, for axes XYZ. Returns the Corrected Gyro Rate Vector in units of radians/sec, for axes XYZ.	13	Raw Gyro Vec X (Float) Raw Gyro Vec Y (Float) Raw Gyro Vec Z (Float) Gyro ID Number (U8)	12	C. Gyro Vec X (Float) C. Gyro Vec Y (Float) C. Gyro Vec Z (Float)
49 (0x31)	Correct Raw Accelerometer Vector	Converts the supplied raw data accelerometer vector to its corrected data representation.	13	Raw Accel Vec X (Float) Raw Accel Vec Y (Float) Raw Accel Vec	12	C. Accel Vec X (Float) C. Accel Vec Y (Float) C. Accel Vec

		Pass the Raw Acceleration Vector, in units of g, for axes XYZ. Returns the Corrected Accelerometer Vector, in units of g, for axes XYZ.		Z (Float) Accel ID Number (U8)		Z (Float)
50 (0x32)	Correct Raw Magnetometer Vector	Converts the supplied raw data magnetometer vector to its corrected data representation. Pass the Raw Magnetometer Vector, in units of gauss, for axes XYZ. Returns the Corrected Magnetometer Vector, in units of gauss, for axes XYZ.	13	Raw Mag Vec X (Float) Raw Mag Vec Y (Float) Raw Mag Vec Z (Float) Mag ID Number (U8)	12	C. Mag Vec X (Float) C. Mag Vec Y (Float) C. Mag Vec Z (Float)
54 (0x36)	Get Corrected Gyro Rate By ID	Pass the ID number of the gyroscope to query. Returns the Corrected Gyro Rate Vector, in units of radians/sec, for axes XYZ.	1	Gyro ID Number (U8)	12	C. Gyro Vec X (Float) C. Gyro Vec Y (Float) C. Gyro Vec Z (Float)
55 (0x37)	Get Corrected Accelerometer Vector By ID	Pass the ID number of the accelerometer to query. Returns the Corrected Acceleration Vector, in units of g, for axes XYZ. Note that this acceleration will include the static component of acceleration due to gravity.	1	Accel ID Number (U8)	12	C. Accel Vec X (Float) C. Accel Vec Y (Float) C. Accel Vec Z (Float)
56 (0x38)	Get Corrected Magnetometer Vector By ID	Pass the ID number of the magnetometer to query. Returns the Corrected Magnetometer Vector, in	1	Mag ID Number (U8)	12	C. Mag Vec X (Float) C. Mag Vec Y (Float) C. Mag Vec Z (Float)

		units of gauss, for axes XYZ.				

4.4 Other Data Commands

Command	Command Name	Description	Input Length	Input	Output Length	Output
13 (0x0D)	Get Primary Barometer Pressure	Returns the air pressure in millibar	0		4	Air Pressure (Float)
14 (0x0E)	Get Primary Barometer Altitude	Returns the altitude calculated from the pressure in meters	0		4	Altitude (Float)
15 (0x0F)	Get Barometer Altitude By ID	Returns the altitude calculated from the pressure in meters	1	ID (U8)	4	Altitude (Float)
16 (0x10)	Get Barometer Pressure By ID	Returns the air pressure in millibar	1	ID (U8)	4	Air Pressure (Float)
43 (0x2B)	Get Temperature Celsius	Returns the temperature of the sensor in Celsius.	0		4	Temperature (Float)
44 (0x2C)	Get Temperature Fahrenheit	Returns the temperature of the sensor in Fahrenheit	0		4	Temperature (Float)
45 (0x2D)	Get Motionless Confidence Factor	Returns a value indicating how much the sensor is being moved at the moment. This value will return 1 if the sensor is completely stationary, and will return 0 if it is in motion. This command can also return values in between indicating how much motion the sensor is experiencing.	0		4	Confidence Factor (Float)
250 (0xFA)	Get Button State	Returns 1 if button is pressed, 0 if not. On an embedded, this is a bitfield where each bit represents a different button.	0		1	State (U8)

4.5 Raw Data Commands

Command	Command Name	Description	Input Length	Input	Output Length	Output
65 (0x41)	Get Raw Gyro Rate By ID	Returns the raw gyro rate vector without any additional postprocessing.	1	ID (U8)	12	Gyro Rate in units of radians/sec (Floatx3)
66 (0x42)	Get Raw Accelerometer Vector By ID	Returns the raw acceleration vector without any additional postprocessing.	1	ID (U8)	12	Acceleration Vector in units of g (Floatx3)
67 (0x43)	Get Raw Magnetometer Vector By ID	Returns the raw magnetometer vector without any additional postprocessing	1	ID (U8)	12	Magnetometer Vector in units of gauss (Floatx3)

4.6 EEPTS™ Commands

Command	Command Name	Description	Input Length	Input	Output Length	Output
68 (0x44)	EEPTS Start	Starts the EEPTS™ system	0		0	
69 (0x45)	EEPTS Stop	Stops the EEPTS™ system	0		0	
70 (0x46)	EEPTS Get Oldest Step	Gets the oldest saved YL_EEPTS_OUTPUT_DATA from the EEPTS™ system	0		58	Step (YL_EEPTS_OUTPUT_DATA)
71 (0x47)	EEPTS Get Newest Step	Gets the most recent YL_EEPTS_OUTPUT_DATA from the EEPTS™ system	0		58	Step (YL_EEPTS_OUTPUT_DATA)
72 (0x48)	EEPTS Get Available Step Count	Returns the number of steps available to	0		1	Count (U8)

		read out				
73 (0x49)	EEPTS Insert GPS	Feeds latitude and longitude to the EEPTS™ system to enable it to use an external GPS. Also useful for priming the system.	16	Latitude (Double), Longitude (Double)	0	
74 (0x4A)	EEPTS Auto Offset	This should be called while the device is mounted in its permanent location on the body and the user is facing north. Allows orientation dependent functions such as Tilt Heading to work in any preset orientation	0		0	

4.7 Streaming Commands

Command	Command Name	Description	Input Length	Input	Output Length	Output
83 (0x53)	Streaming Get Command Label	Returns a string label associated with the data values of the requested command	1	Command Number (U8)	Varies	CSV Label (String)
84 (0x54)	Streaming Get Packet	Return a single packet of streaming data using the current slot configuration.	0		Varies	Streaming Packet
85 (0x55)	Streaming Start	Start a streaming session using the current slot and timing configuration	0		0	
86 (0x56)	Streaming Stop	Stop the current streaming session	0		0	

4.8 System Commands

Command	Command Name	Description	Input Length	Input	Output Length	Output
94 (0x5E)	Get Timestamp	Get the current internal timestamp	0		8	Timestamp (U64)
95 (0x5F)	Set Timestamp	Set the current internal timestamp to the specified value	8	Timestamp (U64)		
225 (0xE1)	Commit Settings	Commits all current sensor settings to non-volatile flash memory, which will persist after the sensor is powered off.	0		0	
226 (0xE2)	Software Reset	Resets the sensor.	0		0	
229 (0xE5)	Enter Bootloader	Places the sensor into a special mode that allows firmware upgrades. This will cease normal operation until the firmware update mode is instructed to return the sensor to normal operation.	0		0	
238 (0xEE)	Get Current LED Color	Returns the current color of the LED as its RGB value in the range of 0.0 to 1.0	0		12	Red (Float), Green (Float), Blue (Float)
126 (0x7E)	Get Debug Message Count	Gets the number of debug messages currently buffered	0		2	Count (U16)
127 (0x7F)	Get Debug Message	Gets the oldest debug message	0		Varies	Debug Message (String)
128 (0x80)	Self Test				4	Result (U32)
165 (0xA5)	Begin Passive Calibration	Performs auto-gyroscope calibration and/or obtains a magnetometer reference vector based on the provided bitfield. Sensor must	1	Mode Bitfield (U8)	0	

		remain still while samples are taken. The gyroscope bias will be automatically placed into the bias portion of the gyroscope calibration coefficient list. Bitfield: 0x1 - Enable Gyro Auto Calibration 0x2 - Enable Mag Ref Auto Calibration				
166 (0xA6)	Get Passive Calibration Active	Returns a bitfield representing what passive auto calibrations are currently active. If the bit is 1, it is active. Bitfield: 0x1 - Gyro Auto Calibration Active 0x2 - Mag Ref Auto Calibration Active	0		1	Active State (U8)
167 (0xA5)	Begin Active Calibration	Performs active magnetometer bias calibration. Sensor should be rotated spherically while samples are taken. The magnetometer bias will be automatically placed into the bias portion of the magnetometer calibration coefficient list.	0		0	
168 (0xA6)	Get Active Calibration Active	1 If "Active Calibration" is active, else 0	0		1	Active State (U8)
120 (0x78)	Reset Filter	Resets the state of the currently selected filter	0		0	
19 (0x13)	Set Offset With Current Orientation	Sets the offset orientation to be the same as the current filtered orientation.	0		0	
20 (0x14)	Reset Base Offset	Sets the base offset to an identity	0		0	

		quaternion.				
22 (0x16)	Set Base Offset With Current Orientation	Sets the base offset orientation to be the same as the current filtered orientation.	0		0	
96 (0x60)	Set Tare With Current Orientation	Sets the tare orientation to be the same as the current filtered orientation.	0		0	
97 (0x61)	Set Base Tare With Current Orientation	Sets the base tare orientation to be the same as the current filtered orientation.	0		0	

4.9 Product Specific Commands

4.9.1 Embedded Commands

Command	Command Name	Description	Input Length	Input	Output Length	Output
31 (0x1F)	Get Interrupt Status	Read the current interrupt status. This value will be 1 if the filter has updated since the last time the value was read or 0 otherwise.	0		1	Status (U8)

4.9.2 Data Logger Commands

Command	Command Name	Description	Input Length	Input	Output Length	Output
57 (0x39)	Mass Storage Controller Enable	Enables the Mass Storage Controller, allowing the device to show in the file explorer over USB.	0		0	
58 (0x3A)	Mass Storage Controller Disable	Disables the Mass Storage Controller, preventing the device from showing in the file explorer over USB.	0		0	
59 (0x38)	Format SD Card	Format SD card using fat32 and default settings. This will erase all data on the SD card.	0		0	
60 (0x3C)	Logging Start	Starts data logging using the current configuration	0		0	
61 (0x3D)	Logging Stop	Stops data logging and saves out current log file.	0		0	

62 (0x3E)	Set Clock Values	Sets the current time on the onboard real-time clock.	7	Year (U16), Month (U8), Day (U8), Hour (U8), Minute (U8), Second (U8)	0	
63 (0x3F)	Get Clock Values	Returns the current time as read by the onboard real-time clock.	0		7	Year (U16), Month (U8), Day (U8), Hour (U8), Minute (U8), Second (U8)
93 (0x5D)	Get Clock Values String	Returns the current time as read by the onboard real-time clock in ISO format as {YYYY}-{MM}-{DD}T{HH}:{MM}:{SS}.{SSS}			Varies	Date Time (String)
64 (0x40)	Logging Get Status	Returns the current status of the logging system. 1 if Logging, 0 if not Logging.	0		1	Status (U8)
87 (0x57)	Logging Pause Streaming	Pauses the streaming output caused when logging with log_immediate_output enabled. Set to 1 to pause, 0 to unpause.	1	Pause Enabled (U8)	0	
170 (0xAA)	Logging Get Last Live Location	Returns the last location that was sent during the most recent live logging session. Cursor Index indicates how far into the file the live logging ended, can be used in command 179. File Path is the full path to the logged file.	0		Varies	Curser Index(U64), File Path(String)
171 (0xAB)	Fs Get Next Directory Item	Returns the next item in the current active directory. An empty string will be returned at the end of a directory. After the end of the directory is reached calling this command again will wrap back to the beginning of the	0		Varies	Type_ID(U8), Item Name(String), Size(U64)

		directory.				
172 (0xAC)	Fs Change Directory	Enter the directory provided. Note: if the volume is formatted as exFAT, double dot name ".." cannot follow the parent directory and will be treated as "." and stay there.	Varies	Filename (String)	0	
173 (0xAD)	Fs Open File	Open the provided file. Must be done before any read commands.	Varies	Filename (String)	0	
174 (0xAE)	Close File	Close the currently opened file.	0		0	
175 (0xAF)	File Get Remaining Size	Returns the remaining amount of bytes in a file based on the amount already read.	0		8	Bytes Remaining (U64)
176 (0xB0)	File Read Line	Read file until a new line is found and returns the line. Not recommended for binary files. If the last character in the returned line is '\xFF' the EOF was reached.	0		Varies	Line (String)
177 (0xB1)	File Read Bytes	Reads provided number of bytes. If the number of bytes left in the file is less than the provided number to read the remaining bytes will be populated with 0xFF.	2	Number to Read (U16)	Varies	Read Bytes (String)
178 (0xB2)	Fs Delete File Or Folder	Deletes the given file. If a folder is given instead, recursively deletes the folder and everything in it.	Varies	Filename (String)	0	
179 (0xB3)	File Set Cursor Index	Sets the curser position in the currently opened file.	8	Index (U64)	0	
180 (0xB4)	File Streaming Start	Start streaming opened file data from current curser index. First	0		Varies	Data Length (U64)

		returns the number of bytes to be read. Stream will be broken into 512-byte chunks with the header sent between chunks if enabled.				
181 (0xB5)	File Streaming Stop	Stops file streaming immediately. Can resume from stop location by calling 180 again.	0		0	

4.9.3 Battery Commands

Command	Command Name	Description	Input Length	Input	Output Length	Output
200 (0xC8)	Battery Get Current	Read the Current Draw of the battery in mA. This value will be negative when discharging and positive when charging. If the battery is fully charged and the sensor is powered via a source other then the battery, this will read 0.	0		2	Battery current draw in milliamps (S16)
201 (0xC9)	Battery Get Voltage	Read the current battery level in volts. Note that this value will read as slightly higher than it actually is if it is read via a USB connection.	0		4	Battery level in voltage (Float)
202 (0xCA)	Battery Get Percent	Read the current battery lifetime as a percentage of the total. Note that this value will read as slightly higher than it actually is if it is read via a USB connection.	0		1	Battery level as percent (U8)
203 (0xCB)	Battery Get Status	Returns a value indicating the current status of the battery, which can be a 3 to indicate that the battery is currently not charging, a 2 to indicate that the battery is charging and thus plugged in, or a 1 to indicate that the sensor is fully charged.	0		1	Battery charge status (U8)

4.9.4 GPS Data Commands

Command	Command Name	Description	Input Length	Input	Output Length	Output
214 (0xD6)	GPS Get Is Active	Returns a 1 if the GPS has received updated values within the last 2 seconds.	0		1	Active (U8)
215 (0xD7)	GPS Get Latitude and Longitude	Returns the last valid Lat Long reading from the onboard GPS	0		16	Latitude (Double), Longitude (Double)
216 (0xD8)	GPS Get Altitude	Returns the last valid Altitude reading from the onboard GPS	0		4	Altitude (Float)
217 (0xD9)	GPS Get Fix Status	Returns current GPS fix state Follows standard GGA formatting	0		1	Fix Status (U8)
218 (0xDA)	GPS Get HDOP	Returns the last valid Horizontal Dilution of Precision reading from the onboard GPS	0		4	HDOP (Float)
219 (0xDB)	GPS Get Satellites	Returns the number of satellites seen by the onboard GPS	0		1	Satellites (U8)
220 (0xDC)	GPS Reset	Performs a restart on the GPS. 0: Hot Reset 1: Warm Reset 2: Cold Reset 3: Full Reset	1	Reset Type (U8)	0	

5 Settings List

5.1 System Settings

Settings Key	Description	Permission	Data Type	Alias
default	Resets all settings to factory defaults when written. Must call commit for settings to persist over power cycle.	W	None	
commit	Commits all current sensor settings to non-volatile flash memory, which will persist after the sensor is powered off.	W	None	
reboot	Resets the sensor.	W	None	
all	Returns all writable settings.	R	Aggregate	
settings	Returns all writable settings but excludes aliases.	R	Aggregate	
serial_number	Returns the sensors full serial number.	R	U64	
timestamp	Timestamp in microseconds from sensor power.	R/W	U64	
led_mode	Allows finer-grained control over the sensor LED. 0: Dynamic - the LED updates based on the status of the device 1: Static - the LED only shows the color specified by the led_rgb setting	R/W	U8	
led_rgb	Sets the idle color of the LED on the sensor to the specified RGB color.	R/W	Float Float Float	
version_firmware	Returns a string indicating the	R	String	

	current firmware version.			
version_hardware	Returns a string indicating the hardware version of the sensor.	R	String	
update_rate_sensor	Reads the amount of time taken by the sensors main loop in microseconds.	R	U32	
header	Configures the response header for data returned over a wired connection. The only parameter is a bitfield that determines which data is prepended to all data responses. The following bits are used: 0x1: (1 byte) Success/Failure, with non-zero values representing failure. 0x2: (4 bytes) Timestamp, in microseconds. 0x4: (1 byte) Command echo—outputs the called command. 0x8: (1 byte) Additive checksum over returned data, but not including response header. 0x20: (4 bytes) Serial number. For the full 8 byte serial number the serial_number setting must be read. 0x40: (2 byte) Data length, returns the length of the requested data, not including response header.	R/W	U8	header_status, header_timestamp, header_echo, header_checksum, header_serial, header_length

header_status	Alias setting to allow for easy setting of the status field in the header. Status: Success/Failure, with non-zero values representing failure.	R/W	U8	header
header_timestamp	Alias setting to allow for easy setting of the timestamp field in the header. Timestamp: time since sensor start in microseconds.	R/W	U8	header
header_echo	Alias setting to allow for easy setting of the command echo field in the header. Command Echo: Outputs the called command.	R/W	U8	header
header_checksum	Alias setting to allow for easy setting of the Checksum field in the header. Checksum: Additive checksum over returned data, but not including response header.	R/W	U8	header
header_serial	Alias setting to allow for easy setting of the Serial Number field in the header.	R/W	U8	header
header_length	Alias setting to allow for easy setting of the Data Length field in the header. Data Length: Returns the length of the requested data, not including response	R/W	U8	header

	header.			
valid_commands	Returns a comma separated list of valid command numbers.	R	String	

5.2 Power Management Settings

Settings Key	Description	Permission	Data Type	Alias
cpu_speed	The speed that the sensors main processor is set to run at. Valid speeds: 48MHz, 96MHz, 144MHz, 192MHz Requires sensor restart to take effect. Does not require commit to persist across power cycles.	R/W	U32	
cpu_speed_cur	The current speed of the sensor's main processor in MHz.	R	U32	
pm_mode	Sets processor speed and component data rates based on presets. 0: Low Power - CPU = 48MHz, Max ODR = 500Hz 1: Normal - CPU = 96MHz, Max ODR = 1000Hz 2: High Performance - CPU = 144MHz, Max ODR = 1500Hz 3: Turbo - CPU = 192MHz, Max ODR = 2000Hz	W	U8	

pm_idle_enabled	Enables CPU power saving during idle time. 1: Enabled (Default) 0: Disabled	R/W	U8	

5.3 Streaming Settings

Settings Key	Description	Permission	Data Type	Alias
stream_slots	Configures data output slots for streaming mode. Accepts a list of up to 16 commands (comma separated U8). Every streaming iteration each command will be executed in the order they were entered. The output data will be put in the order they were called. Unfilled slots will be filled with 255 (0xFF) for no command.	R/W	String	
stream_interval	Time in microseconds between each streaming output. Values lower than 500 will be clamped to 500.	R/W	U64	stream_hz
stream_hz	Output rate when streaming in hertz. Max value of 2000. Will clamp to closest microsecond value.	R/W	Float	stream_interval
stream_duration	Length of time to stream in seconds. Duration of 0 will stream until stopped.	R/W	Float	

stream_delay	Length of time in seconds the sensor will wait after recieving the start streaming command to return the first data packet.	R/W	Float	
stream_mode	Controls how stream duration is handled. 0: Duration - controlled via the stream_duration setting 1: Count - controlled via the stream_count setting	R/W	U8	
stream_count	A value greater than 1. Streaming will end after this many samples are output.	R/W	U64	
streamable_commands	A comma separated list of all available streaming commands	R	String	

5.4 Debug Settings

Settings Key	Description	Permission	Data Type	Alias
debug_level	Bitfield representing the currently active debug levels. 0x1: Error 0x2: Warning 0x4: Information Default Value: 0x1	R/W	U32	
debug_module	Bitfield representing the currently enabled debug module. For full list of available modules see the Debug Section of the user	R/W	U32	

	manual.			
debug_mode	0: Buffered - All debug messages will be buffered until Command 127 is called. 1: Immediate - All debug messages will be immediately output when generated. Will also immediately output any messages currently buffered when this mode is set.	R/W	U8	
debug_led	Enable Debug LED event when debug messages are available.	R/W	U8	
debug_fault	When set to 1, if a hard fault occurs the sensor will enter a debug state.	R/W	U8	
debug_wdt	When set to 1, if a WDT occurs the sensor will enter a debug state.	R/W	U8	

5.5 EEPTS™ Settings

Settings Key	Description	Permission	Data Type	Alias
pts_offset_quat	Offset quaternion used when using the orientation based heading method to set which direction is North relative to the sensor's mounting.	R/W	Float Float Float Float	
pts_default	Resets all of the EEPTS settings to their default value	W	None	
pts_settings	Returns a semicolon separated list of all EEPTS settings	R	Aggregate	

	and their values.			
pts_preset_hand	Sets multiple EEPTS settings to optimize speed and accuracy for the given situation: 0. Hand Disabled - The sensor will not be in the hand 1. Hand Enabled - The sensor may be in the hand 2. Hand only - The sensor will only be in the hand	W	U8	
pts_preset_motion	Sets multiple EEPTS settings to optimize for speed and accuracy for the given situation: 0 - Walk/Run enabled 1 - Walk/Run/Crawl enabled	W	U8	
pts_preset_heading	Sets multiple EEPTS settings to configure the optimal settings for different heading methods: 0 - Orientation Independent Heading 1 - Orientation Dependent Heading	W	U8	

5.6 Component Settings

Settings Key	Description	Permission	Data Type	Alias
valid_accels	Returns list of detected Accelerometers.	R	String	
valid_gyros	Returns list of detected Gyroscopes.	R	String	
valid_mags	Returns list of detected	R	String	

	Magnetometers.			
valid_baros	Returns list of detected Barometers.	R	String	
valid_components	Returns list of all detected components.	R	String	
primary_accel	Currently selected component(s) to use for the primary accel data output commands.	R/W	String	
primary_gyro	Currently selected component(s) to use for the primary gyro data output commands.	R/W	String	
primary_mag	Currently selected component(s) to use for the primary mag data output commands.	R/W	String	
primary_sensor_rfade	On sensors with multiple of the same component type, this value adjusts the threshold percentage for when one component will fade to the other. Must be between 0.0 and 1.0, inclusive. Example: 16G and 100G accelerometer, will start fading to higher range sensor at readings of 14.4G and 90G. Default value: 0.1	R/W	Float	
mag_bias_mode	Determines the behavior of the Auto Mag Bias calculator: 0: Manual - Magnetometer bias is only manually set. 1: Auto Single -	R/W	U8	

	Magnetometer bias will be automatically calibrated once on startup or when this is set. 2: Auto Continuous - Magnetometer bias will be continuously automatically updated.			
odr_all	Sets the ODR (output data rate / sample rate) of all aliased (accel, gyro, mag, baro) components. Each component will attempt to set the ODR as close as possible. If it is not possible to reach the desired ODR, the ODR will be set to the maximum valid ODR that component supports.	W	U32	odr_accel, odr_gyro, odr_mag, odr_baro
odr_accel	Sets the ODR (output data rate / sample rate) of all accelerometers.	W	U32	
odr_gyro	Sets the ODR (output data rate / sample rate) of all gyroscopes.	W	U32	
odr_mag	Sets the ODR (output data rate / sample rate) of all magnetometers.	W	U32	
odr_baro	Sets the ODR (output data rate / sample rate) of all barometers.	W	U32	
accel_enabled	Removes the accelerometer component from the QGRAD3 orientation calculation.	R/W	U8	

	The accelerometer component stays active and data can still be read with the appropriate commands.			
gyro_enabled	Removes the gyroscope component from the QGRAD3 orientation calculation. The gyroscope component stays active and data can still be read with the appropriate commands.	R/W	U8	
mag_enabled	Removes the magnetometer component from the QGRAD3 orientation calculation. The magnetometer component stays active and data can still be read with the appropriate commands.	R/W	U8	
calib_mat_accel$\mathscr{X}$	The matrix correction for accelerometer $\mathscr{X}$. Replace the $\mathscr{X}$ in the command with the number that represents the desired accelerometer.	R/W	Float Float Float Float Float Float Float Float Float	
calib_bias_accel$\mathscr{X}$	The bias correction for accelerometer $\mathscr{X}$. Replace the $\mathscr{X}$ in the command with the number that represents the desired accelerometer.	R/W	Float Float Float	
range_accel$\mathscr{X}$	The range for accelerometer $\mathscr{X}$ in	R/W	U16	

	(g). Replace the 𝒳 in the command with the number that represents the desired accelerometer.			
valid_ranges_accel𝒳	A list of valid ranges for accelerometer 𝒳. Replace the 𝒳 in the command with the number that represents the desired accelerometer.	R	String	
oversample_accel𝒳	The oversample for accelerometer 𝒳. Readings from the accel will only update after this many samples are taken, and its value will be the average of all those samples. Replace the 𝒳 in the command with the number that represents the desired accelerometer.	R/W	U16	
running_avg_accel𝒳	The smoothing factor for accelerometer 𝒳 in range of 0 to 1. Replace the 𝒳 in the command with the number that represents the desired accelerometer.	R/W	Float	
odr_accel𝒳	Sets the ODR (output data rate / sample rate) for accelerometer 𝒳. Replace the 𝒳 in the command with the number that represents the desired accelerometer.	R/W	U32	

update_rate_accel𝒳	Gets the current observed update rate for accelerometer 𝒳. Replace the 𝒳 in the command with the number that represents the desired accelerometer.	R	Float	
noise_profile_accel 𝒳	Gets the reported noise density, variance, bandwidth, sensitivity, and true ODR for accelerometer 𝒳. Replace the 𝒳 in the command with the number that respresents the desired accelerometer.	R	Float Float Float Float U32	
calib_mat_gyro𝒳	The matrix correction for gyroscope 𝒳. Replace the 𝒳 in the command with the number that represents the desired gyroscope.	R/W	Float Float Float Float Float Float Float Float Float	
calib_bias_gyro𝒳	The bias correction for gyroscope 𝒳. Replace the 𝒳 in the command with the number that represents the desired gyroscope.	R/W	Float Float Float	
range_gyro𝒳	The range for gyroscope 𝒳 in DPS (degrees per second). Replace the 𝒳 in the command with the number that represents the desired gyroscope.	R/W	U16	
valid_ranges_gyro𝒳	A list of valid ranges for gyroscope 𝒳.	R	String	

	Replace the 𝒳 in the command with the number that represents the desired gyroscope.			
oversample_gyro𝒳	The oversample for gyroscope 𝒳. Readings from the gyroscope will only update after this many samples are taken, and its value will be the average of all those samples. Replace the 𝒳 in the command with the number that represents the desired gyroscope.	R/W	U16	
running_avg_gyro𝒳	The smoothing factor for gyroscope 𝒳 in range of 0 to 1. Replace the 𝒳 in the command with the number that represents the desired gyroscope.	R/W	Float	
odr_gyro𝒳	Sets the ODR (output data rate / sample rate) for gyroscope 𝒳. Replace the 𝒳 in the command with the number that represents the desired gyroscope.	R/W	U32	
update_rate_gyro𝒳	Gets the current observed update rate for gyroscope 𝒳. Replace the 𝒳 in the command with the number that represents the desired gyroscope.	R	Float	
noise_profile_gyro𝒳	Gets the reported noise density, variance, bandwidth,	R	Float Float Float Float U32	

	sensitivity, and true ODR for gyroscope 𝒳. Replace the 𝒳 in the command with the number that respresents the desired gyroscope.			
calib_mat_mag𝒳	The matrix correction for magnetometer 𝒳. Replace the 𝒳 in the command with the number that represents the desired magnetometer.	R/W	Float Float Float Float Float Float Float Float Float	
calib_bias_mag𝒳	The bias correction for magnetometer 𝒳. Replace the 𝒳 in the command with the number that represents the desired magnetometer.	R/W	Float Float Float	
range_mag𝒳	The range for magnetometer 𝒳 in G (gauss). Replace the 𝒳 in the command with the number that represents the desired magnetometer.	R/W	U16	
valid_ranges_mag𝒳	A list of valid ranges for magnetometer 𝒳. Replace the 𝒳 in the command with the number that represents the desired magnetometer.	R	String	
oversample_mag𝒳	The oversample for magnetometer 𝒳. Readings from the magnetometer will only update after this many samples are taken, and its	R/W	U16	

	value will be the average of all those samples. Replace the 𝒳 in the command with the number that represents the desired magnetometer.			
running_avg_mag𝒳	The smoothing factor for magnetometer 𝒳 in range of 0 to 1. Replace the 𝒳 in the command with the number that represents the desired magnetometer.	R/W	Float	
odr_mag𝒳	Sets the ODR (output data rate / sample rate) for magnetometer 𝒳. Replace the 𝒳 in the command with the number that represents the desired magnetometer.	R/W	U32	
update_rate_mag𝒳	Gets the current observed update rate for mag 𝒳. Replace the 𝒳 in the command with the number that represents the desired mag.	R	Float	
noise_profile_mag𝒳	Gets the reported noise density, variance, bandwidth, sensitivity, and true ODR for mag 𝒳. Replace the 𝒳 in the command with the number that respresents the desired mag.	R	Float Float Float Float U32	
calib_bias_baro𝒳	An offset applied to the barometer 𝒳.	R/W	Float	

	Replace the $\mathscr{X}$ in the command with the number that represents the desired barometer.			

5.7 Filter Settings

Settings Key	Description	Permission	Data Type	Alias
axis_order	Configures how the axes map to the physical sensor using XYZ terminology. Example values: XYZ ZYX Y-XZ View the "Axis Management" section of the user manual for more details.	R/W	String	
axis_order_c	Configures how the axes map to the physical sensor using compass direction terminology. Example values: NED ENU DSW View the "Axis Management" section of the user manual for details.	R/W	String	
axis_offset_enabled	If enabled, applied offset will affect the data axes as well instead of only the orientation.	R/W	U8	
euler_order	The decomposition	R/W	String	

	order used when retrieving orientation as Euler Angles.			
update_rate_filter	Reads the amount of time taken by the last filter update step in microseconds.	R	U32	
update_rate_sms	Reads the amount of time taken by the last component manager update in microseconds.	R	U32	
offset	The offset quaternion used in the computation of the tared orientation. View the "Tare and Offset" section of the user manual for more details.	R/W	Float Float Float Float	
base_offset	The reference offset quaternion used when sending the "Offset with Current Orientation" command. View the "Tare and Offset" section for more details.	R/W	Float Float Float Float	
tare_quat	The tare quaternion used in the computation of the tared orientation. View the "Tare and Offset" section for more details.	R/W	Float Float Float Float	
tare_auto_base	If enabled, the base_tare will automatically be set when offsetting the sensor with the current orientation so that taring results in the identity orientation. View	R/W	U8	

	the "Tare and Offset" section for more details.			
base_tare	The reference tare quaternion used when sending the "Tare with Current Orientation" command. View the "Tare and Offset" section for more details.	R/W	Float Float Float Float	
tare_mat	The tare value used in the computation of the tared orientation, but in matrix form. View the "Tare and Offset" section for more details.	R/W	Float Float Float Float Float Float Float Float Float	
running_avg_orient	A smoothing factor in the range 0 to 1.	R/W	Float	
filter_mode	The method of orientation calculation: 0: IMU Mode 1: QGrad3 2: Extended Kalman Filter	R/W	U8	
filter_mref_mode	Sets the method for which the sensor will obtain the magnetic reference vector (MRV): 0: Manual - Will only use the MRV currently stored in settings. 1: Semi-Auto - Will use the mrv set by user if it exists, else will attempt to automatically get a new MRV on startup. 2: Auto - Will always attempt to automatically get a new MRV on startup.	R/W	U8	

	3: GPS - Will get MRV via GPS reading and WMM (world magnetic model), WIP.			
filter_mref	The magnetic reference as a vector. This should be the expected reading of the magnetometer when the sensor is facing north.	R/W	Float Float Float	
filter_mref_gps	Sets the magnetic reference vector via the WMM (world magnetic model) at the given GPS location.	W	Double Double	filter_mref
filter_mref_dip	Sets the magnetic reference vector via the expected magnetic dip.	R/W	Float	filter_mref

5.8 Product Specific Settings

5.8.1 Embedded Settings

Settings Key	Description	Permission	Data Type	Alias
pin_mode0	Sets the function of pins 5 and 6. Mode: 1 - Uart (default) 4 - Orientation Level Interrupt (pin 5) 5 - Orientation Pulse Interrupt (pin 5) 7 - Button (Pin 5 and 6 become inputs, get status with the Get Button State command) 8 - Transaction Interrupts (See the "Transaction	R/W	U8	

	Protocol" "IRQ Mode" section for more information)			
pin_mode1	Sets the function of pins 2 and 3. Mode: 2 - SPI (default) 3 - I2C 4 - Orientation Level Interrupt (pin 2) 5 - Orientation Pulse Interrupt (pin 2) 7 - Button (Pin 2 and 3 become inputs, get status with the Get Button State command)	R/W	U8	
uart_baudrate	Sets the baudrate of the sensors UART interface. Change will take effect immediately after the response is sent. Valid baudrates: 4800, 9600, 19200, 38400, 57600, 115200, 230400, 480800, 921600, 2000000, 4000000	R/W	U32	
i2c_addr	Sets the sensors address used for communicating over I2C. Will not be applied until I2C/Sensor is reinitialized. For a list of valid addresses look at the I2C section of the manual.	R/W	U8	

5.8.2 Data Logger Settings

Settings Key	Description	Permission	Data Type	Alias

power_hold_time	Time in seconds that the power button needs to be held for the sensor to turn off. Set to -1 to disable power off via holding the button.	R/W	Float	
power_hold_state	If 0, the sensor will power off when not connected to a wired connection. If 1, the sensor will stay powered via battery when not connected to a wired connection. This setting is not commitable. If the sensor was powered via the button, will default to 1, else its initial value is determined by power_initial_hold_state.	R/W	U8	
power_initial_hold_state	The initial value power_hold_state gets set to when the sensor initializes.	R/W	U8	
fs_cfg_load	Applies the settings contained in the <Root>:/CONFIG/sensor.cfg file.	W		
fs_msc_enabled	Enable or disable the mass storage controller.	R/W	U8	
fs_msc_auto	View the "Auto Mass Storage Control" section for details.	R/W	U8	

log_slots	Configures data output slots for logging. Accepts a list of up to 16 commands (comma separated U8). Every log iteration each command will be executed in the order they were entered. The output data will be put in the order they were called. Unfilled slots will be filled with 255 (0xFF) for no command.	R/W	String	
log_interval	Set the time between data logging packets in microseconds.	R/W	U64	
log_hz	Set the data logging rate in samples per second (Hz).	R/W	Float	
log_start_event	Enum for the logging start event. 0- Button 1- Accel Event Threshold 2- None/Command Only 3- On GPS Fix 4- On Power Pass in any combination as a comma separated list.	R/W	String	
log_start_motion_threshold	Threshold value used for the Accel Event Threshold start event. Logging starts when the change in acceleration (g) magnitude within a 0.1 second window exceeds this value.	R/W	Float	

	In general, smaller values will require less motion to start, and larger values will require more motion to start.			
log_stop_event	Enum for the logging stop event. 0- Button 1- Accel Event Threshold 2- Command Only 3- Duration 4- Capture Count 5- Period Count Pass in any combination as a comma separated list.	R/W	String	
log_stop_motion_threshold	Threshold value used for the Accel Event Threshold stop event. Logging stops when the change in acceleration magnitude within a 0.1 second window is continously beneath this value for the time specified by log_stop_motion_delay. In general, larger values will stop more easily while experiencing larger motion, and smaller values will require the sensor to be more motionless to stop.	R/W	Float	
log_stop_motion_delay	Time in seconds the change in acceleration magnitude within a	R/W	Float	

	0.1 second window needs to be below the motion threshold for the stop event to trigger.			
log_stop_count	Value used for the Capture Count stop event. Logging stops when this number of samples have been gathered.	R/W	U64	
log_stop_duration	Value used for the Duration stop event. Logging stops when this duration, in seconds, has elapsed.	R/W	Float	
log_stop_period_count	Value used for the Period Count stop event. Logging stops when this number of periods have been logged.	R/W	U32	
log_style	Enum for determining the method of data logging. 0: Continous - Single logging period for each capture session. 1: Periodic - Multiple logging periods for a capture session.	R/W	U8	
log_periodic_capture_time	Amount of time each period will log while in periodic mode.	R/W	Float	
log_periodic_rest_time	Time to wait between each capture period.	R/W	Float	

log_base_filename	Base filename used for each data file. The final filename will be "{log_base_filename}{N}.[csv|bin]" where N is an incrementing file count and csv or bin is based on log_data_mode. N starts at 0 during a session for all logging modes, and will increase for each additional file when log_file_mode is set to "New" instead of "Append".	R/W	String	
log_file_mode	Enum for how new data is logged during logging sessions in periodic mode. 0: Append - Append data from subsequent perioids to a single capture session file. 1: New - Create a new log file for every period during the logging session.	R/W	U8	
log_data_mode	Enum for determining ascii or binary logging. 1: Ascii 2: Binary	R/W	U8	
log_output_settings	If set to 1 a setting file will be created at the start of a capture session to capture the settings used during that capture session.	R/W	U8	

log_header_enabled	If set to 1, the response header will be included in the log results.	R/W	U8	
log_folder_mode	0: Session - Name folders based on an auto incrementing session index. 1: Date Time - Name folders based on current date time read from RTC.	R/W	U8	
log_immediate_output	If set to 1, the data being logged will be streamed out at the same time using the file streaming protocol.	R/W	U8	
log_immediate_output_header_enabled	If set to 1, the header will be prepended to each file stream packet while log_immediate_output is enabled.	R/W	U8	
log_immediate_output_header_mode	Controls what protocol mode the immediate_output_header uses. 0: Match - uses the mode specified by the log_data_mode setting 1: Ascii 2: Binary	R/W	U8	
rtc_year	The current year of the RTC	R/W	U16	rtc_datetime
rtc_month	The current month of the RTC (1-12)	R/W	U8	rtc_datetime
rtc_day	The current day of the RTC (1-31)	R/W	U8	rtc_datetime
rtc_hour	The current hour of the RTC (0-23)	R/W	U8	rtc_datetime
rtc_minute	The current minute of the RTC (0-59)	R/W	U8	rtc_datetime
rtc_second	The current second of the RTC (0-59)	R/W	U8	rtc_datetime

rtc_datetime	Allows retrieving and setting all the current RTC values.	R/W	U16 U8 U8 U8 U8 U8	rtc_year, rtc_month, rtc_day, rtc_hour, rtc_minute, rtc_second

5.8.3 BLE Settings

Settings Key	Description	Permission	Data Type	Alias
ble_name	Advertised name of BLE enabled sensors. Max size 19 characters. Requires reboot for name to apply.	R/W	String	
ble_connected	Returns the connection status of the BLE module. 0: Disconnected 1: Connected -1: Not available	R	S8	
ble_disconnect	Disconnects from whatever BLE device is connected to the sensor.	W	None	

5.8.4 GPS Settings

Settings Key	Description	Permission	Data Type	Alias
gps_standby	If set to 1, the GPS will be in standby mode. If set to 0, the GPS will be active. Defaults to 0.	R/W	U8	
gps_led	If set to 1, the led will appear green while the GPS has a fix.	R/W	U8	

5.8.5 Battery Settings

Settings Key	Description	Permission	Data Type	Alias

bat_chg_rate	Typical charge rate in mA. Note that the set charge rate may differ from the actual charge rate to some degree. If different, a debug message will be generated when this is set.	R/W	U16	
bat_cold_threshold	The charge rate as a percentage of typical (0.0-1.0) and the temperature it occurs at in degrees celsius.	R/W	Float Float	
bat_warm_threshold	The charge rate as a percentage of typical (0.0-1.0) and the temperature it occurs at in degrees celsius.	R/W	Float Float	
bat_hot_threshold	The charge rate as a percentage of typical (0.0-1.0) and the temperature it occurs at in degrees celsius.	R/W	Float Float	
bat_offset_threshold	When going from hot to warm or warm to normal battery states, the temperature must fall this far below the threshold for the state change to occur. This is to prevent situations where the sensor flickers between states due to the sensor heating up while charging and then cooling when charging is shut off repeatedly, causing the temperature to border a threshold.	R/W	Float	

bat_mah	The onboard batteries max capacity in mAh. Returns 0 if unknown.	R	U16	

6 Additional Resources

6.1 Application Notes

The 3-Space Sensor Suite for 3.x sensors can be downloaded off of our github here: https://github.com/YostLabs/3SpaceSuite-Python/releases
The 3-Space Python3 API for 3.x sensors is available in our library, which is available here: https://pypi.org/project/yostlabs/
Source code for the 3-Space Python3 API for 3.x sensors is available on our github here: https://github.com/YostLabs/3SpacePythonPackage