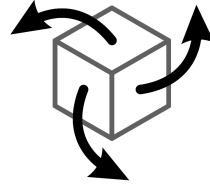


3-SPACE SENSOR

3-SPACE



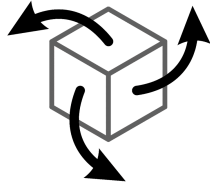
# 3-Space Sensor Data-Logging Miniature Attitude & Heading Reference System

## User's Manual

**YEI Technology**  
630 Second Street  
Portsmouth, Ohio 45662

[www.YeiTechnology.com](http://www.YeiTechnology.com)  
[www.3SpaceSensor.com](http://www.3SpaceSensor.com)





# 3-Space Sensor Data-Logging

Miniature Attitude & Heading  
Reference System

## User's Manual

**YEI Technology**  
630 Second Street  
Portsmouth, Ohio 45662

[www.YeiTechnology.com](http://www.YeiTechnology.com)  
[www.3SpaceSensor.com](http://www.3SpaceSensor.com)

Toll-Free: 888-395-9029  
Phone: 740-355-9029

# Table of Contents

1. Usage/Safety Considerations.....	1
1.1 Usage Conditions.....	1
1.2 Technical Support and Repairs.....	1
1.3 Battery Safety Considerations.....	2
2. Overview of the YEI 3-Space Sensor.....	3
2.1 Introduction.....	3
2.2 Applications.....	3
2.3 Hardware Overview.....	4
2.3 Features.....	5
2.5 Block Diagram of Sensor Operation.....	6
2.6 Specifications.....	7
2.7 Physical Dimensions.....	8
2.8 Axis Assignment.....	8
3. Description of the 3-Space Sensor.....	9
3.1 Orientation Estimation.....	9
3.1.1 Component Sensors.....	9
3.1.2 Scale, Bias, and Cross-Axis Effect.....	9
3.1.3 Reference Vectors.....	10
3.1.4 Kalman Filter.....	10
3.1.5 Reference Orientation/Taring.....	10
3.1.6 Other Estimation Parameters.....	10
3.2 Communication.....	11
3.3 Input Device Emulation.....	11
3.3.1 Axes and Buttons.....	11
3.3.2 Joystick.....	11
3.3.3 Mouse.....	11
3.4 Data-Logging.....	11
3.4.1 Mass Storage Device.....	11
3.4.2 SD Card Format and Directory Structure.....	12
3.4.3 Data-Logging.....	12
3.4.4 LED Capture Behavior.....	15
3.4.5 Real Time Clock.....	15
3.5 Sensor Settings.....	16
3.5.1 Committing Settings.....	16
3.5.2 Natural Axes.....	16
3.5.3 Sensor Settings and Defaults.....	16
3.5.4 Capture Settings and Defaults.....	17
4. 3-Space Sensor Usage/Protocol.....	18
4.1. Usage Overview.....	18
4.1.1 Protocol Overview.....	18
4.1.2 Computer Interfacing Overview.....	18
4.2. Protocol Packet Format.....	19
4.2.1 Binary Packet Format.....	19
4.2.2 ASCII Text Packet Format.....	20
4.3. Command Overview.....	21
4.3.1 Commands for Reading Filtered Sensor Data.....	21
4.3.2 Commands for Reading Normalized Sensor Data.....	21
4.3.3 Commands for Reading Raw Sensor Data.....	22
4.3.5 Commands for Reading Filter Parameters.....	23
4.3.6 Commands for Calibration.....	24
4.3.7 Battery Commands.....	24
4.3.8 Data-Logging Commands.....	24
4.3.9 General Commands.....	25
4.4. Command Details.....	26
Appendix.....	45
USB Connector.....	45
Hex / Decimal Conversion Chart.....	45

# 1. Usage/Safety Considerations

## 1.1 Usage Conditions

- Do not use the 3-Space Sensor in any system on which people's lives depend (life support, weapons, etc.)
- Because of its reliance on a compass, the 3-Space Sensor will not work properly near the earth's north or south pole.
- Because of its reliance on a compass and accelerometer, the 3-Space Sensor will not work properly in outer space or on planets with no magnetic field.
- 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.
- Because of its reliance on a compass, care should be taken when using the 3-Space Sensor near ferrous metal structures, magnetic fields, current carrying conductors, and should be kept about 6 inches away from any computer screens or towers.
- Since the Data-Logging 3-Space Sensor uses removable MicroSD media, it is the end-user's responsibility to ensure that storage media is compatible.
- The Data-Logging 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. See section 1.4 Battery Considerations for further information pertaining to battery safety.

## 1.2 Technical Support and Repairs

Limited Product Warranty: YEI warrants the media and hardware on which products are furnished to be free from defects in materials and workmanship under normal use for sixty (60) days from the date of delivery. No warranties exist for any misuse. YEI will repair or replace any defective product which is returned within this time period.

Product Support: YEI provides technical and user support via our toll-free number (888-395-9029) and via email (support@YostEngineering.com). Support is provided for the lifetime of the equipment. Requests for repairs should be made through the Support department. For damage occurring outside of the warranty period or provisions, customers will be provided with cost estimates prior to repairs being performed.

### **1.3 Battery Safety Considerations**

The Data-logging 3-Space Sensor contains a rechargeable lithium-polymer battery. Lithium-polymer batteries have high energy densities and can be dangerous if not used and cared for properly. The Data-Logging 3-space Sensor has been designed to include multiple levels of battery safety assurance. The Data-Logging 3-Space Sensor circuitry includes smart charging circuitry with thermal management to prevent over-charging the battery. The battery pack itself also includes protection circuitry to prevent over-charge, 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:

- Don't disassemble, crush, puncture, shred, or otherwise attempt to change the form of your battery.
- Don't attempt to change or modify the battery yourself. Contact YEI technical support for battery replacement or battery repair.
- Don't let the mobile device or battery come in contact with water or other liquids.
- Don't allow the battery to touch metal objects.
- Don't place the sensor unit near a heat source. Excessive heat can damage the sensor unit or the battery. High temperatures can cause the battery to swell, leak, or malfunction.
- Don't dry a wet or damp sensor unit with an appliance or heat source, such as a hair dryer or microwave oven.
- Don't drop the sensor unit. Dropping, especially on a hard surface, can potentially cause damage to the sensor unit or the battery.
- Discontinue use immediately and contact YEI technical support if the battery or sensor unit produce odors, emit smoke, exhibit swelling, produce excess heat, exhibit leaking.
- Dispose of Lithium-polymer batteries properly in accordance with local, state, and federal guidelines.

## 2. Overview of the YEI 3-Space Sensor

### 2.1 Introduction

The YEI 3-Space Sensor™ Data-Logging integrates a miniature, high-precision, high-reliability, Attitude and Heading Reference System (AHRS) with a MicroSD card interface and a rechargeable lithium-polymer battery solution into a single low-cost end-use-ready unit. The Attitude and Heading Reference System (AHRS) uses triaxial gyroscope, accelerometer, and compass sensors 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 proprietary multi-reference vector mode increases accuracy and greatly reduces and compensates for sensor error. The YEI 3-Space Sensor Data-Logging system also utilizes a dynamic sensor confidence algorithm that ensures optimal accuracy and precision across a wide range of operating conditions.

The YEI 3-Space Sensor™ Data-Logging unit features are accessible via a well-documented open communication protocol that allows access to all available sensor data and configuration parameters using a USB 2.0 interface, and configuration parameters are also accessible through configuration files on the SD card. Versatile commands allow access to raw sensor data, normalized sensor data, and filtered absolute and relative orientation outputs in multiple formats including: quaternion, Euler angles (pitch/roll/yaw), rotation matrix, axis angle, two vector(forward/up).

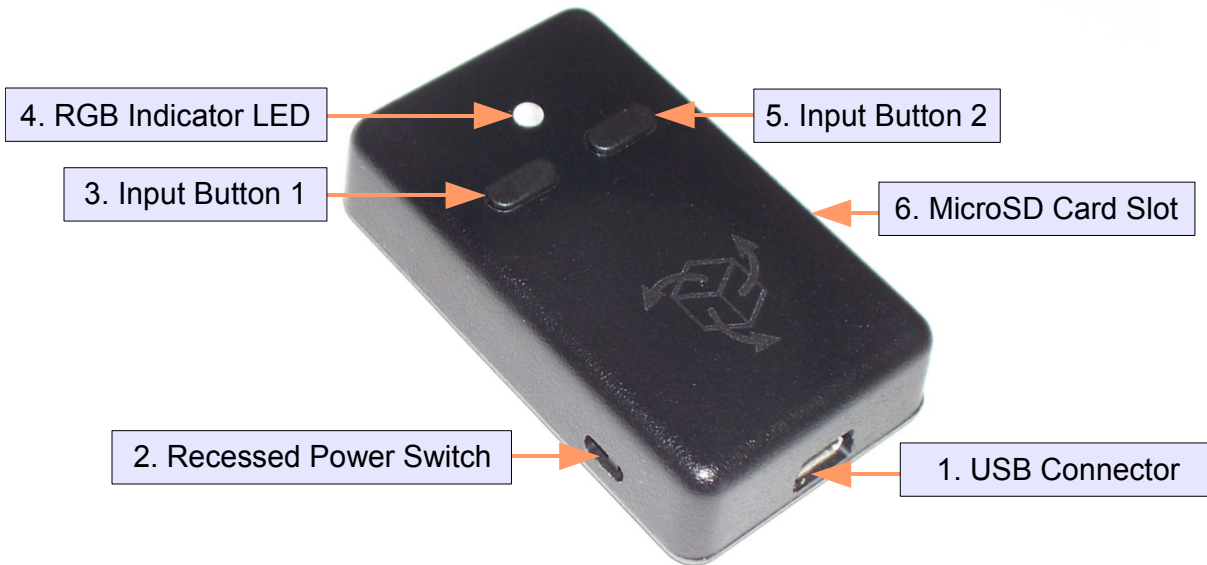
The YEI 3-Space Sensor™ Data-Logging grants access to its SD card on a host PC through a USB Mass Storage interface. Settings for how to gather data can be accessed through configuration files, and gathered data can be stored in a variety of formats onto the SD card. The SD card uses the FAT32 filesystem.

When used as a USB device, the 3-Space Sensor™ provides mouse emulation and joystick emulation modes that ease integration with existing applications.

### 2.2 Applications

- Robotics performance analysis
- Motion capture
- Information gathering
- Personnel / pedestrian tracking
- Unmanned air/land/water vehicle tracking
- Education and performing arts
- Healthcare monitoring
- Asset tracking
- Vibration analysis and monitoring
- Event detection and monitoring

## 2.3 Hardware Overview



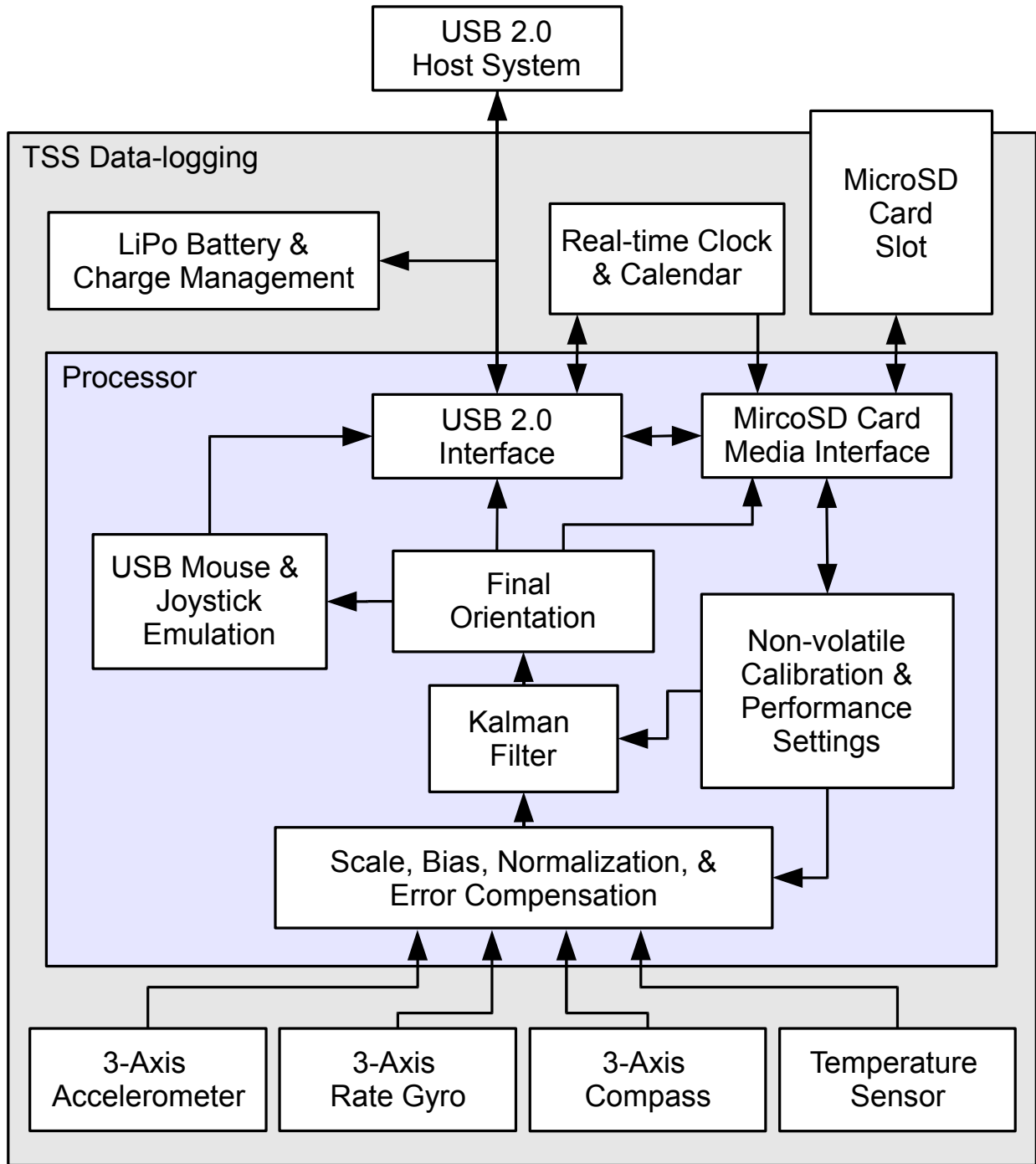
1. **USB Connector** – The 3-Space Sensor uses a 5-pin mini USB connector to connect to a computer via USB and to charge the internal battery. The USB connector provides for both power and communication signals.
2. **Recessed Power Switch** – The 3-Space Sensor can be switched on and off when powered from the internal battery by using the recessed power switch. When connected via USB, the unit is powered and the batteries will begin recharging regardless of the position of the recessed power switch.
3. **Input Button 1** – The 3-Space Sensor includes two input buttons that can be used in conjunction with the orientation sensing capabilities of the device. The inputs are especially useful when using the 3-Space Sensor as an input device such as in joystick emulation mode or mouse emulation mode.
4. **Indicator LED** – The 3-Space Sensor includes an RGB LED that can be used for visual status feedback.
5. **Input Button 2** – The 3-Space Sensor includes two input buttons that can be used in conjunction with the orientation sensing capabilities of the device. The inputs are especially useful when using the 3-Space Sensor as an input device such as in joystick emulation mode or mouse emulation mode.
6. **Recessed MicroSD Card Slot** – The 3-Space Sensor MicroSD media can be inserted or removed through the recessed slot. This slot is recessed to help prevent accidental card removal.

## 2.3 Features

The YEI 3-Space Sensor Data-Logging has many features that allow it to be a flexible all-in-one solution for your orientation sensing needs. Below are some of the key features:

- Small self-contained high-performance data-logging AHRS at 35mm x 60mm x 15mm and 28 grams
- Integrated Lithium-Polymer battery and charge control allows battery life of 5+ hours at full performance
- Fast sensor update and filter rate allow use in highly dynamic applications, including motion capture, performance & motion analysis, and navigation
- Highly customizable orientation sensing with options such as tunable filtering, oversampling, and orientation error correction
- Advanced integrated Kalman 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, or more quickly in machine readable form
- 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
- MicroSD card allows for data-logging applications, USB allows for real-time applications
- MicroSD card uses standard FAT32 file-system
- Flexible data logging configuration allows customization of logged data and allows event-based and time-based logging options
- Built-in clock/calendar provides for fully time-stamped event logging at high resolution
- USB communication through a virtual COM port
- Enumeration as USB mass-storage device makes access to logged data easy
- USB joystick/mouse emulation modes ease integration with existing applications
- Upgradeable firmware
- RGB status LED, two programmable input buttons
- Available in either hand-held or strap-down packaging
- RoHS compliant

## 2.5 Block Diagram of Sensor Operation

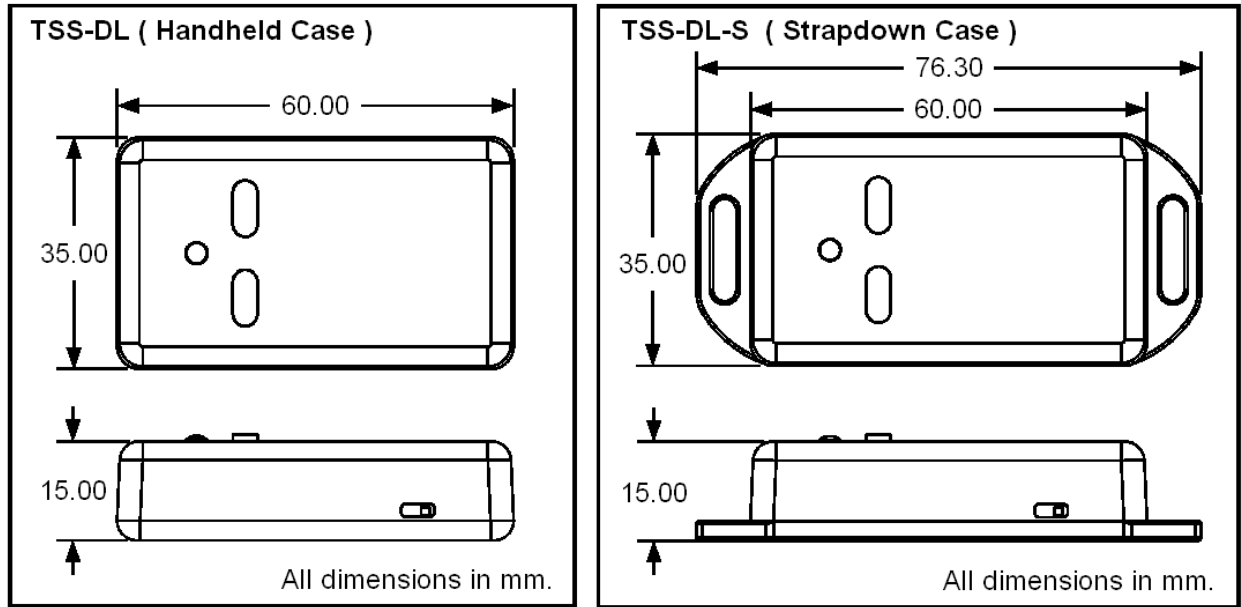


## 2.6 Specifications

General	
Part number	TSS-DL (Handheld Sensor Unit) TSS-DL-S (Strapdown Sensor Unit)
Dimensions	35mm x 60mm x 15mm (1.38 x 2.36 x 0.59 in.)
Weight	28 grams ( 0.98 oz )
Supply voltage	+5v USB
Battery technology	rechargeable Lithium-Polymer
Battery lifetime	5+ hours continuous use at full performance
Communication interfaces	USB 2.0 (Mass Storage & CDC ), MicroSD card configuration files.
Storage media	MicroSD card
Filter update rate	up to 200Hz with full functionality, up to 500Hz in IMU mode.
Orientation output	absolute & relative quaternion, Euler angles, axis angle, rotation matrix, two vector
Other output	raw sensor data, normalized sensor data, temperature, date/time.
Shock survivability	5000g
Temperature range	-40C ~ 85C ( -40F ~ 185F )
Processor	32-bit RISC running @ 60MHz
Sensor	
Orientation range	360° about all axes
Orientation accuracy	±2° for dynamic conditions & all orientations
Orientation resolution	<0.08°
Orientation repeatability	0.085° for all orientations
Accelerometer scale	±2g / ±4g / ±8g selectable
Accelerometer resolution	14 bit
Accelerometer noise density	99µg/√Hz
Accelerometer sensitivity	0.00024g/digit for ±2g range 0.00048g/digit for ±4g range 0.00096g/digit for ±8g range
Accelerometer temperature sensitivity	±0.008%/°C
Gyro scale	±250/±500/±2000 °/sec selectable
Gyro resolution	16 bit
Gyro noise density	0.03°/sec/√Hz
Gyro bias stability @ 25°C	11°/hr average for all axes
Gyro sensitivity	0.00875°/sec/digit for ±250°/sec 0.01750°/sec/digit for ±500°/sec 0.070°/sec/digit for ±2000°/sec
Gyro non-linearity	0.2% full-scale
Gyro temperature sensitivity	±0.016%/°C
Compass scale	±1.3 Ga default. Up to ±8.1 Ga available
Compass resolution	12 bit
Compass sensitivity	5 mGa/digit
Compass non-linearity	0.1% full-scale

\*Specifications subject to change

## 2.7 Physical Dimensions



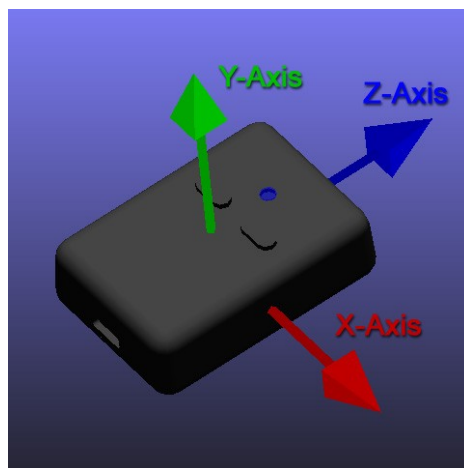
## 2.8 Axis Assignment

All YEI 3-Space Sensor product family members have re-mappable axis assignments and axis directions. This flexibility allows axis assignment and axis direction to match the desired end-use requirements.

The natural axes of the 3-Space Sensor are as follows:

- The positive X-axis points out of the right hand side of the sensor, which is the side that is facing right when the buttons face upward and plug faces towards you.
- The positive Y-axis points out of the top of the sensor, the side with the buttons.
- The positive Z-axis points out of the front of the sensor, the side opposite the plug.

The natural axes are illustrated in the diagram below



Bear in mind the difference between natural axes and the axes that are used in protocol data. While they are by default the same, they can be remapped so that, for example, data axis Y could contain data from natural axis X. This allows users to work with data in a reference frame they are familiar with.

## 3. Description of the 3-Space Sensor

### 3.1 Orientation Estimation

The primary purpose of the 3-Space Sensor is to 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.

#### 3.1.1 Component Sensors

The 3-Space Sensor estimates orientation by combining the data it gets from three types of sensors: a gyroscope, an accelerometer, and a compass. A few things you should know about each of these sensors:

- **Accelerometer:** This sensor measures the acceleration due to gravity, as well as any other accelerations that occur. Because of this, this sensor is at its best when the 3-Space Sensor is sitting still. Most jitter seen as the orientation of the sensor changes is due to shaking causing perturbations in the accelerometer readings. To account for this, by default, when the 3-Space Sensor is being moved, the gyroscope becomes more trusted (becomes a greater part of the orientation estimate), and the accelerometer becomes less trusted.
- **Gyroscope:** This sensor measures angular motion. It has no ability to give any absolute orientation information like the accelerometer or compass, and so is most useful for correcting the orientation during sensor motion. Its role during these times becomes vital, though, as the accelerometer readings can become unreliable during motion.
- **Compass:** This sensor measures magnetic direction. The readings from the compass and accelerometer are used together to form the absolute component of orientation, which is used to correct any short term changes the gyroscope makes. Its readings are much more stable than those of the accelerometer, but it can be adversely affected by any ferrous metal or magnetic objects. When the accelerometer is less trusted, the compass is treated in the same way so as to avoid updates to orientation based on partial absolute information.

#### 3.1.2 Scale, Bias, and Cross-Axis Effect

The readings taken from each component sensor are not in a readily usable form. The compass and accelerometer readings are not unit vectors, and the gyroscope readings aren't yet in radians per second. To convert them to these forms, scale and bias must be taken into account. Scale is how much larger the range of data read from the component sensor is than the range of data should be when it is converted. For example, if the compass were to give readings in the range of -500 to 500 on the x axis, but we would like it to be in the range of -1 to 1, the scale would be 500. Bias is how far the center of the data readings is from 0. If another compass read from -200 to 900 on the x axis, the bias would be 350, and the scale would be 550. The last parameter used in turning this component sensor data into usable data is cross-axis effect. This is the tendency for a little bit of data on one axis of a sensor to get mixed up with the other two. This is an effect experienced by the accelerometer and compass. There are 6 numbers for each of these, one to indicate how much each axis is affected by each other axis. Values for these are generally in the range of 1 to 10%. These parameters are applied in the following order:

- 1) Bias is subtracted from each axis
- 2) The three axes are treated as a vector and multiplied by a matrix representing scale and cross-axis parameters

Factory calibration provides default values for these parameters for the accelerometer and compass, and users should probably never need to change these values. To determine these parameters for the gyroscope, you must calibrate it. Read the Quick Start guide or the 3-Space Suite manual for more information on how to do this.

### 3.1.3 Reference Vectors

In order to get an absolute estimation of orientation from the accelerometer and compass, the sensor needs a reference vector for each to compare to the data read from it. The most obvious choice for these are the standard direction of gravity(down) and the standard direction of magnetic force(north), respectively. However, the sensor does provide several different modes for determining which reference vector to use:

- **Single Manual:** Uses 2 reference vectors it is given as the reference vectors for the accelerometer and compass.
- **Single Auto:** When the sensor powers on or is put into this mode, it calculates gravity and north and uses those calculated vectors as the reference vectors.
- **Single Auto Continual:** The same as Single Auto, but the calculation happens constantly. This can account for some shifts in magnetic force due to nearby objects or change of location, and also can help to cope with the instability of the accelerometer.
- **Multiple:** Uses a set of reference vectors from which the best are picked each cycle to form a single, final reference vector. This mode has the ability to compensate for certain errors in the orientation. In this mode the sensor will have a slightly slower update rate, but will provide greater accuracy. For information on how to set up this mode, see the Quick Start guide or the 3-Space Suite manual.

### 3.1.4 Kalman Filter

The component sensor data and reference vectors are fed into a Kalman filter, which uses statistical techniques to optimally combine the data into a final orientation reading.

### 3.1.5 Reference Orientation/Taring

Given the results of the Kalman filter, the sensor can make a good estimation of orientation, but it will likely be offset from the actual orientation of the device by a constant angle until it has been given a reference orientation. This reference orientation tells the sensor where you would like its zero orientation to be. The sensor will always consider the zero orientation to be the orientation in which the plug is facing towards you and top(the side with buttons on it) facing up. The sensor must be given a reference orientation that represents the orientation of the sensor when it is in the position in which you consider the plug to be towards you and the buttons up. The act of giving it this reference orientation to the sensor is called taring, just as some scales have a tare button which can be pressed to tell the scale that nothing is on it and it should read zero. For instructions on doing this, refer to the Quick Start guide or 3-Space Suite manual.

### 3.1.6 Other Estimation Parameters

The 3-Space Sensor offers a few other parameters to filter the orientation estimate. Please note that these only affect the final orientation and not the readings of individual component sensors.

- **Oversampling:** Oversampling causes the sensor to take extra readings from each of the component sensors and average them before using them to estimate orientation. This can reduce noise, but also causes each cycle to take longer proportional to how many extra samples are being taken.
- **Running Average:** The final orientation estimate can be put through a running average, which will make the estimate smoother at the cost of introducing a small delay between physical motion and the sensor's estimation of that motion.
- **Rho Values:** As mentioned earlier, by default the accelerometer and compass are trusted less than the gyros when the sensor is in motion. Rho values are the mechanism that handles the concept of trust. They involve parameters, one for the accelerometer and one for the compass, that indicate how much these component sensors are to be trusted relative to the gyroscope. A lower value for the parameter means more trust. The default mode for this is "confidence mode", where the rho value chooses between a minimum and maximum value based on how much the sensor is moving. The other option is to have a single, static rho value.

## 3.2 Communication

Obtaining data about orientation from the sensor or giving values for any of its settings is done through the sensor's communication protocol. The protocol can be used through the USB port. A complete description of how to use this protocol is given in section 4 of this document. Also, you may instead use the 3-Space Suite, which provides a graphical method to do the same. To learn how to use this, read the 3-Space Suite manual. In addition, data-logging options allow the sensor to be configured to log data in certain formats, with various criteria for beginning and ending a data-logging session. These options and working with the data-logging system is covered in section 3.4 of this document.

## 3.3 Input Device Emulation

### 3.3.1 Axes and Buttons

The 3-Space Sensor has the ability to act as a joystick and/or mouse. Both of these are defined in the same way, as a collection of axes and buttons. Axes are input elements that can take on a range of values, whereas buttons can only either be on or off. On a joystick, the stick part would be represented as 2 axes, and all the physical buttons on it as buttons. The 3-Space Sensor has no physical joystick and only 2 physical buttons, so there are a number of options to use properties of the orientation data as axes and buttons. Each input device on the 3-Space Sensor has 2 axes and 8 buttons. For more information on setting these up, see the 3-Space Suite manual. All communication for these input devices is done through the standard USB HID(Human Interface Device) protocol.

### 3.3.2 Joystick

As far as a modern operating system is concerned, a joystick is any random collection of axes and buttons that isn't a mouse or keyboard. Joysticks are mostly used for games, but can also be used for simulation, robot controls, or other applications. The 3-Space Sensor, as a joystick, should appear just like any other joystick to an operating system that supports USB HID(which most do).

### 3.3.3 Mouse

When acting as a mouse, the 3-Space Sensor will take control of the system's mouse cursor, meaning if the mouse portion is not properly calibrated, using it could easily leave you in a situation in which you are unable to control the mouse cursor at all. In cases like this, unplugging the 3-Space Sensor will restore the mouse to normal operation, and unless the mouse enabled setting was saved to the sensor's memory, plugging it back in should restore normal operation. Using the default mouse settings, caution should be exercised in making sure the orientation estimate is properly calibrated before turning on the mouse. For help with this, see the Quick Start guide.

The mouse defaults to being in Absolute mode, which means that the data it gives is meant to represent a specific position on screen, rather than an offset from the last position. This can be changed to Relative mode, where the data represents an offset. In this mode, the data which would have indicated the edges of the screen in Absolute mode will now represent the mouse moving as quickly as it can in the direction of that edge of the screen. For more information, see command 251 in section 4.3.7, or the 3-Space Suite manual.

## 3.4 Data-Logging

### 3.4.1 Mass Storage Device

The Data-Logging 3-Space Sensor exposes the contents of its SD card to a computer by enumerating as a Mass Storage device in addition to a virtual COM port. Upon being connected to a computer through USB, the sensor will cease any current data-logging session and will cede control of the SD card to the computer, as both the computer and the sensor cannot write to the SD card without coming into conflict. No further data-logging can be done at this point until the computer no longer controls the SD card. Unplugging the sensor will return control of the SD card to it. Also, the sensor has a Mass Storage Off mode which will return control of the SD card to it even while attached to a computer. For more information on this, see commands 57 and 58 in section 4.

### 3.4.2 SD Card Format and Directory Structure

The Data-Logging 3-Space Sensor will attempt, upon power on or upon SD card insertion, to place the directory structure it requires on to the card. If this is unsuccessful for any reason, such as the card being in read only mode or the card having not been formatted yet, the sensor's LED pulse red twice quickly once a second. It will also do this if no SD card is inserted at all. This indicates that it is not ready to do data-logging. The SD card may be formatted in one of two ways: either let any SD card formatting utility (such as the one built in to Windows) format the card as a single FAT32 partition, or insert the unformatted card into the Data-Logging 3-Space Sensor and call command 59 (look in section 4 for details on this command). Also, the 3-Space Suite has an easy way to call this command in the Sensor Info window when connected to a Data-Logging sensor. Refer to the Data-Logging Quick Start guide for more information. When the card has been properly initialized by the sensor, the LED will turn solid blue. The directory structure the sensor sets up is as follows:

```

/ (Root Directory)
  /Data/
  /Config/

```

The **/Data/** directory is for holding the results of any data capture sessions. A new directory inside this will be created for each session, named according to when the capture started (For information on setting up the current time, see section 3.4.4).

This **/Config/** directory holds configuration files which can be modified to change the current settings of the sensor. It contains two files: **sensor.cfg** and **capture.cfg**. **sensor.cfg** allows access to many sensor settings that can also be modified through protocol commands, such as orientation averaging modes. **capture.cfg** allows access to settings having to do with how and how often data is gathered. See section 3.4.3 for information on the contents of **capture.cfg** and data-logging options in general. In order to change a setting in one of these files, simply find the name of the property you want to modify on the left of an equals sign, and then change the value for it on the right of the equals sign. If the value is textual (properties where this is the case will have a list of the possible values listed to the right of the assignment), be sure to enclose the text in quotes. Quotes are not necessary for numeric values.

### 3.4.3 Data-Logging

As described in section 3.4.2, upon the start of a data-logging session, a new directory will be created to hold the data, named after the time the session was started (given by the real time clock, see section 3.4.4).

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

#### **CaptureStartEvent**

The CaptureStartEvent property in the **capture.cfg** file selects options for starting a capture session. Possible values for the CaptureStartEvent property are:

- “on command”: This setting has no way of starting a capture session except through the calling of the begin data-logging session command, command 60. See section 4 for more information on this command. Because calling a command requires a USB connection which can communicate with the sensor, the sensor will have to be taken out of Mass Storage mode before this command is called. The command to turn off Mass Storage mode is command 58. Also note that regardless of start event, this command can be used to start a data-logging session.
- “on startup”: Whenever the sensor starts up, it will attempt to start logging data as soon as it can. After this one session, it will not attempt to start another session.
- “left button”, “right button”, and “both buttons”: A session will begin when the appropriate button or buttons are pressed. Note that if the stop condition is set to the same set or part of the same set of buttons, the buttons will need to be released before they will register as a stop condition.
- “motion”: A session will begin when the accelerometer detects that motion has risen to a certain level. This level is given in gs (gravity units) and can be set through the property CaptureStartEventMotionThreshold.

## CaptureStopEvent

The CaptureStopEvent property in the **capture.cfg** file selects options for stopping a capture session. Possible values for the CaptureStopEvent property are:

- “on command”: Like the same property for the start events, this means a session can only be stopped through a command. Use command 61 for ending a data-logging session. Also note that regardless of stop event, this command can be used to stop a data-logging session.
- “always”: This will always stop a data-logging session as soon as it starts. This is most useful in concert with the CapturePostStopGatherTime property, which gives a length of time after the stop of a session data should continue to be logged.
- “left button”, “right button”, and “both buttons”: Just as for start events, these will stop a session when the buttons are pressed.
- “motion stop”: This will stop a session when the motion falls below a certain threshold. This threshold, given in gs(gravity units), is indicated by the property CaptureStopEventMotionThreshold.
- “capture count”: This will stop a session after a certain number of samples have been taken. This number is given by the property CaptureStopEventCaptureCount.
- “capture duration”: This will stop a session after it has lasted for a certain amount of time, given in milliseconds by the property CaptureStopEventCaptureDuration.
- “period count”: This only has any effect when the CaptureStyle property is set to “periodic”. It will stop a capture after a certain number of capture periods, given by the property CaptureStopEventPeriodCount. Using this when in continuous mode will cause the session to never end.

## CaptureFormat

The CaptureFormat property specifies a format string similar to that required by the C function printf. It consists of a string of whatever characters are desired to show up in the data-logging file, in addition to a number of % delimited tokens which indicate data of a certain type. Options for characters which may follow the % are as follows:

- d: The current date.
- t: The current time.
- q: The tared orientation, in quaternion form.
- a: The tared orientation, in axis angle form.
- e: The tared orientation, in Euler angle form(given in pitch, yaw, roll order).
- m: The tared orientation, in rotation matrix form.
- g: The current calculated angular difference between readings, in quaternion form.
- uq: The untared orientation, in quaternion form.
- ua: The untared orientation, in axis angle form.
- ue: The untared orientation, in Euler angle form(given in pitch, yaw, roll order).
- um: The untared orientation, in rotation matrix form.
- ng: The latest normalized(scaled and biased) gyroscope reading.
- na: The latest normalized(scaled and biased) accelerometer reading.
- nc: The latest normalized(scaled and biased) compass reading.
- nt: The latest temperature reading, in degrees C.
- nb: The battery level percentage.
- rg: The raw(as it comes from the sensor) gyroscope reading.
- ra: The raw(as it comes from the sensor) accelerometer reading.
- rc: The raw(as it comes from the sensor) compass reading.
- %: An actual %.

Note that any non-data characters will only be included in the file in ASCII data-logging mode. For more information, see the property CaptureDataMode. In ASCII mode, data that is comprised of multiple values will be separated by commas.

### ***CaptureRate***

The CaptureRate property in the **capture.cfg** file determines the desired sampling period while logging is active. This property is specified in milliseconds.

### ***CaptureStyle***

The CaptureStyle property in the **capture.cfg** file determines if “continuous” or “periodic” data capture style is used. In continuous mode, data-logging will be enabled at all times during a data-logging session. In periodic mode, data-logging will start and stop during the course of a session. The properties that control this behavior are CaptureStylePeriodicCaptureTime, which determines how long it captures for before it stops, and CaptureStylePeriodicRestTime, which determines how long it is stopped before it starts capturing again.

### ***CapturePostStopGatherTime***

The CapturePostStopGatherTime property in the **capture.cfg** file specifies an amount of time, after a session has stopped, to continue gathering data. This value is specified in milliseconds.

### ***CaptureFileStub***

The CaptureFileStub property in the **capture.cfg** file specifies the base name of the data-logging file which resides in each session's directory, with a “.txt” appended to it if data is captured in ASCII mode, and “.dat” if in binary mode. With a CaptureFileMode of “new”, this stub will also have a number added on to it before the “.txt” or “.dat”.

### ***CaptureFileMode***

The CaptureFileMode property in the **capture.cfg** file specifies how files are used to capture data. This property is one of the following:

- “append”: New samples are added to the end of the single data log in the session directory.
- “replace”: In continuous mode, the same as append. In periodic mode, only data from the most recent period will appear.
- “new”: A new file will be made for each data capture period, and each period's data will be placed in this new file. Only one file will be made in continuous mode, named “<CaptureFileStub>1.txt” in ASCII mode or “<CaptureFileStub>1.dat” in binary.

### ***CaptureDataMode***

The CaptureDataMode property in the **capture.cfg** file specifies whether captured data is stored in the file as human readable ASCII or compact binary. This property is one of the following:

- “ascii”: Data is logged in a human readable form. Superfluous characters in the formatting string are placed in the data log.
- “binary”: Data is logged in a compact, non-human readable form. Superfluous characters are ignored.

### ***CaptureFileInfoHeader***

The CaptureFileInfoHeader property in the **capture.cfg** file determines whether a line should be written at the top of each capture file indicating the format of the data contained within. This property can be set to either 0 for off or 1 for on.

### 3.4.4 LED Capture Behavior

The RGB LED can be used as an indicator of the current state of the sensor. Below is a summary of the LED meanings:

**Solid Blue ( or solid currently set custom color setting ):** Power on and no data-logging session in progress.

**Yellow:** A data-logging session is in progress, but a sample is not currently being taken.

**Green:** The LED will emit a green flash and return to yellow when a data-logging sample is taken.

**Red ( double pulse ):** MicroSD media not present or file system error.

**Red ( Single Pulse when not plugged into USB ):** Battery low - battery life remaining is at or below 5%.

**Yellow ( Single Pulse when plugged into USB ):** Battery is actively charging.

**Green (Single Pulse when plugged into USB ):** Battery is fully-charged.

**Red ( Rapid Constant Pulse ):** Panic mode. Indicates corrupted sensor settings.

### 3.4.5 Real Time Clock

The Data-Logging 3-Space Sensor contains a real time clock chip which allows it to keep track of time. The clock chip uses a separate clock battery which maintains the time and clock settings. This internal clock battery is calculated to have a life of about 5 years. The clock chip must be given an initial time for it to report time properly in a desired time zone. This time can be given to the chip using command 62, and read back using command 63. See the entries for these commands in Section 4 for more details. In addition, the 3-Space Suite has an option for automatically setting the clock of the sensor to the clock of the host computer. Go to the Sensor Info window when connected to the Data-Logging sensor and there will be an option to do this. For more information, refer to the Data-Logging Quick Start Guide.

## 3.5 Sensor Settings

### 3.5.1 Committing Settings

Changes made to the 3-Space Sensor will not be saved unless they are committed. This allows you to make changes to the sensor and easily revert it to its previous state by resetting the chip. For instructions on how to commit your changes, see the Quick Start guide or 3-Space Suite manual. Any changes relating to the multiple reference vector mode are an exception to this rule, as all these changes are saved immediately.

### 3.5.2 Natural Axes

The natural axes of the 3-Space Sensor are as follows:

- The positive X-axis points out of the right hand side of the sensor, which is the side that is facing right when the buttons face upward and plug faces towards you.
- The positive Y-axis points out of the top of the sensor, the side with the buttons.
- The positive Z-axis points out of the front of the sensor, the side opposite the plug.

Bear in mind the difference between natural axes and the axes that are used in protocol data. While they are by default the same, they can be remapped so that, for example, data axis Y could contain data from natural axis X. This allows users to work with data in a reference frame they are familiar with.

See section 2.8 for a diagram illustrating the natural axes.

### 3.5.3 Sensor Settings and Defaults

Setting Name	Purpose	Default Value
Accelerometer Rho Value	Determine how trusted the accelerometer is	Confidence Mode, 5 to 100
Compass Rho Value	Determine how trusted the compass is	Confidence Mode, 5 to 100
Accelerometer Coefficients	Determines the scale, bias, and cross-axis parameters for the accelerometer	Factory calibrated
Compass Coefficients	Determines the scale, bias, and cross-axis parameters for the compass	Factory calibrated
Accelerometer Enabled	Determines whether the compass is enabled or not	TRUE
Compass Enabled	Determines whether the accelerometer is enabled or not	TRUE
Gyroscope Enabled	Determines whether the gyroscope is enabled or not	TRUE
Axis Directions	Determines what natural axis direction each data axis faces	+X, +Y, +Z
Sample Rate	Determines how many samples the sensor takes per cycle	1 from each component sensor
Running Average Percentage	Determines how heavy of a running average to run on the final orientation	0(no running average)
Desired Update Rate	Determines how long each cycle should take(ideally)	0 microseconds
Reference Mode	Determines how the accelerometer and compass reference vectors are determined	Single Auto
CPU Speed	Determines how fast the CPU will run	60 MHz
LED Color	Determines the RGB color of the LED	0,0,1(Blue)
Joystick Enabled	Determines whether the joystick is enabled or not	TRUE
Mouse Enabled	Determines whether the mouse is enabled or not	FALSE
Button Gyro Disable Length	Determines how many cycles the gyro is ignored after a button is pressed	5
Multi Reference Weight Power	Determines what power each multi reference vector weight is raised to	10
Multi Reference Cell Divisions	Determines how many cells the multi reference lookup table is divided into per axis	4
Multi Reference Nearby Vectors	Determines how many nearby vectors each multi reference lookup table cell stores	8

### 3.5.4 Capture Settings and Defaults

Setting Name	Purpose	Default Value
Capture Rate	Determine how often to capture data(in ms)	10
Capture Format	Determine what data to capture	“%d %t %q”(Date, time, and orientation as quaternion)
Start Event	Determine how a session is started	“left button”
Start Event Motion Threshold	Determine how much motion starts a session(in g)	0.5
Capture Style	Determine when to capture data	“continuous”
Periodic Capture Time	Determine how long to capture data in a period(in ms)	5000
Periodic Rest Time	Determine how long to rest in a period(in ms)	5000
Stop Event	Determine how a session is stopped	“right button”
Stop Event Motion Threshold	Determine how much motion stops a session(in g)	0.3
Stop Event Capture Count	Determine how many captures to perform before stop	100
Stop Event Capture Duration	Determine how long to capture before stop(in ms)	5000
Stop Event Period Count	Determine how many periods before stopping	1
Post Stop Gather Time	Determine how long to capture after stop	0
File Stub	Determine name of data log	“data”
File Mode	Determines how to put data in files	“append”
Data Mode	Determines how data is written	“ascii”
File Info Header Enabled	Determines whether or not to have a file header	1

## 4. 3-Space Sensor Usage/Protocol

### 4.1. Usage Overview

#### 4.1.1 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 and the checksum has been verified as correct(ASCII mode commands do not use checksums for convenience). Incomplete packets and packets with incorrect checksums 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.

Specific details of the 3-Space Sensor protocol and its control commands are discussed in the following pages.

#### 4.1.2 Computer Interfacing Overview

The Data-logging 3-Space Sensor enumerates as a composite USB device consisting of a USB mass storage device, a CDC serial communication device, and an HID mouse/keyboard device.

When interfacing with a computer, the 3-Space Sensor presents itself as a COM port, which provides an interface by which the serial communication the protocol requires may happen. The name of this COM port is specific to the operating system being used. It is possible to use multiple 3-Space Sensors on a single computer. Each will be assigned its own COM port.

The USB mass storage device allows access the configuration files that are used to determine sensor configuration and data-logging options. For more detail on these files refer to section 3.4 Data-Logging.

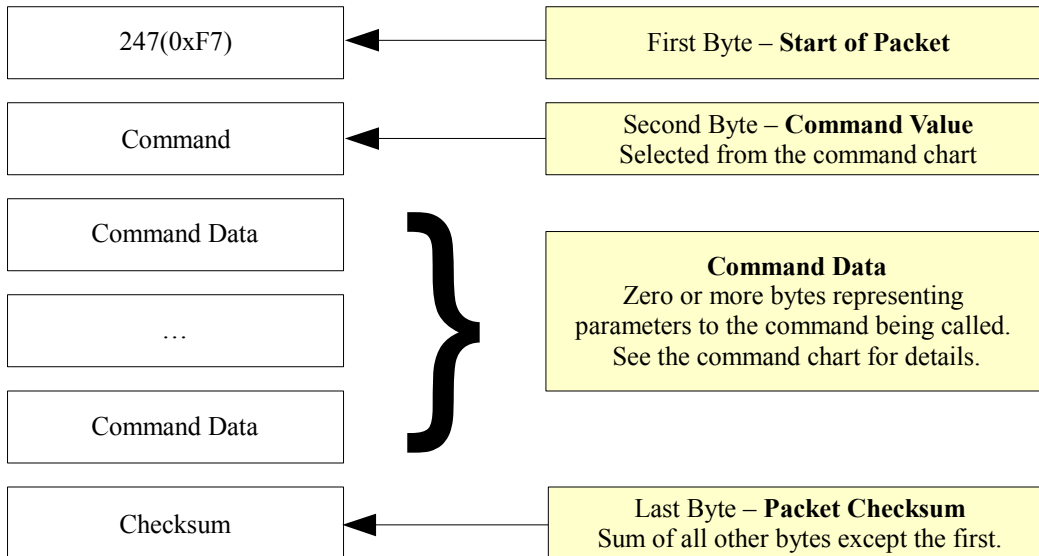
For more information on how to install the sensor software on a computer and begin using it, see the Quick Start guide.

## 4.2. Protocol Packet Format

### 4.2.1 Binary Packet Format

The binary packet size can be three or more bytes long, depending upon the nature of the command being sent to the controller. Each packet consists of an initial “**start of packet**” byte, followed by a “**command value**” specifier byte, followed by zero or more “**command data**” bytes, and terminated by a packet “**checksum value**” byte.

Each binary packet is at least 3 bytes in length and is formatted as shown in figure 1



**Figure 1 - Typical Binary Command Packet Format**

#### Binary Return Values:

When a 3 Space Sensor command is called in binary mode, any data it returns will also be in binary format. For example, if a floating point number is returned, it will be returned as its 4 byte binary representation.

For information on the floating point format, go here: [http://en.wikipedia.org/wiki/Single\\_precision\\_floating-point\\_format](http://en.wikipedia.org/wiki/Single_precision_floating-point_format)

Also keep in mind that integer and floating point values coming from the sensor are stored in big-endian format.

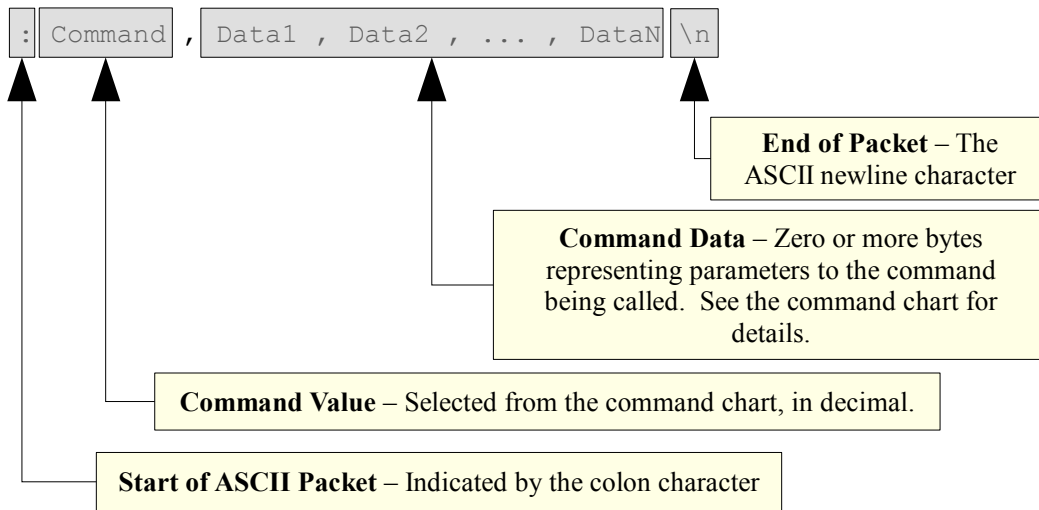
#### The Checksum Value:

The checksum is computed as an arithmetic summation of all of the characters in the packet (except the checksum value itself) modulus 256. This gives a resulting checksum in the range 0 to 255. The checksum for binary packets is transmitted as a single 8-bit byte value.

### 4.2.2 ASCII Text Packet Format

ASCII text command packets are similar to binary command packets, but are received as a single formatted line of text. Each text line consists of the following: an ASCII colon character followed by an integral command id in decimal, followed by a list of ASCII encoded floating-point command values, followed by a terminating newline character. The command id and command values are given in decimal. The ASCII encoded command values must be separated by an ASCII comma character or an ASCII space character. Thus, legal command characters are: the colon, the comma, the period, the digits 0 through 9, the minus sign, the new-line, the space, and the backspace. When a command calls for an integer or byte sized parameter, the floating point number given for that parameter will be interpreted as being the appropriate data type. For simplicity, the ASCII encoded commands follow the same format as the binary encoded commands, but ASCII text encodings of values are used rather than raw binary encodings.

Each ASCII packet is formatted as shown in figure 2.



**Figure 2 - Typical ASCII Command Packet Format**

Thus the ASCII packet consists of the the following characters:

- **:** – the ASCII colon character signifies the start of an ASCII text packet.
- **,** – the ASCII comma character acts as a value delimiter when multiple values are specified.
- **.** – the ASCII period character is used in floating point numbers.
- **0~9** – the ASCII digits are used to in integer and floating point values.
- **-** – the ASCII minus sign is used to indicate a negative number
- **\n** – the ASCII newline character is used to signify the end of an ASCII command packet.
- **\b** – the ASCII backspace character can be used to backup through the partially completed line to correct errors.

If a command is given in ASCII mode but does not have the right number of parameters, the entire command will be ignored.

**Sample ASCII commands:**

:0\n	Read orientation as a quaternion
:106,2\n	Set oversample rate to 2

**ASCII Return Values:**

All values are returned in ASCII text format when an ASCII-format command is issued. To read the return data, simply read data from the sensor until a Windows newline(a carriage return and a line feed) is encountered..

### 4.3. Command Overview

There are over 90 different command messages that are grouped numerically by function. Unused command message bytes are reserved for future expansion.

When looking at the following command message tables, note the following:

- The “Data Len” field indicates the number of additional data-bytes the command expects to follow the command-byte itself. This number doesn't include the Start of Packet, Command, or Checksum bytes. Thus, the total message size can be calculated by adding three bytes to the “Data Len” listed in the table.
- Likewise, the “Return Data Len” field indicates the number of data-bytes the command delivers back to the sender once the command has finished executing.
- Under “Return Data Details”, each command lists the sort of data which is being returned and next to this in parenthesis the form this data takes. For example, a quaternion is represented by 4 floating point numbers, so a command which returns a quaternion would list “Quaternion(float x4)” for its return data details.
- Command length information only applies to binary commands, as ascii commands can vary in length.
- For quaternions, data is always returned in x, y, z, w order.
- Euler angles are always returned in pitch, yaw, roll order.
- When calling commands in ASCII mode, there is no fixed byte length for the parameter data or return data, as the length depends on the ASCII encoding.

#### 4.3.1 Commands for Reading Filtered Sensor Data

These commands return sensor data which has been filtered using a Kalman filter. None of these commands take any parameters, they only return data.

Command	Description	Return Len	Return Data Details
0(0x00)	Read filtered, tared orientation(Quaternion)	16	Quaternion(float x4)
1(0x01)	Read filtered, tared orientation(Euler Angles)	12	Euler Angles(float x3)
2(0x02)	Read filtered, tared orientation(Rotation Matrix)	36	Rotation Matrix(float x9)
3(0x03)	Read filtered, tared orientation(Axis Angle)	16	Axis(float x3), Angle(float)
4(0x04)	Read filtered, tared orientation(Two Vector(Forward, Down))	24	Vector(float x3), Vector(float x3)
5(0x05)	Read filtered gyro rates	16	Quaternion(float x4)
6(0x06)	Read filtered, untared orientation (Quaternion)	16	Quaternion(float x4)
7(0x07)	Read filtered, untared orientation (Euler Angles)	12	Euler Angles(float x3)
8(0x08)	Read filtered, untared orientation (Rotation Matrix)	36	Rotation Matrix(float x9)
9(0x09)	Read filtered, untared orientation (Axis Angle)	16	Axis(float x3), Angle(float)
10(0x0A)	Read filtered, untared orientation (Two Vector(Forward, Down))	24	Vector(float x3), Vector(float x3)
11(0x0B)	Read filtered, tared forward and down vectors in sensor reference frame (Two Vector(Forward, Down))	24	Vector(float x3), Vector(float x3)
12(0x0C)	Read filtered, North, Earth vectors in sensor reference frame(Two Vector(North, Earth))	24	Vector(float x3), Vector(float x3)

#### 4.3.2 Commands for Reading Normalized Sensor Data

These commands return sensor data which has been converted from a raw form to a form that represents a real world quantity, but has not yet been used with the Kalman filter. None of these commands take any parameters, they only return data.

Command	Description	Return Len	Return Data Details
32(0x20)	Read all(gyro, accelerometer, compass)	36	Vector(float x3), Vector(float x3), Vector(float x3)
33(0x21)	Read gyros	12	Vector(float x3)
34(0x22)	Read accelerometer	12	Vector(float x3)
35(0x23)	Read compass	12	Vector(float x3)
36(0x24)	Read temperature C	4	float
37(0x25)	Read temperature F	4	float
38(0x26)	Read confidence factor	4	float

### 4.3.3 Commands for Reading Raw Sensor Data

These commands return sensor data just as it was when it was read from each sensor. None of these commands take any parameters, they only return data.

Command	Description	Return Len	Return Data Details	Data Len	Data Details
64(0x40)	Read all raw	36	Vector(float x3), Vector(float x3), Vector(float x3)	0	
65(0x41)	Read gyro raw	12	Vector(float x3)	1	Gyro range(byte: 0 for $\pm 250$ dps, 1 for $\pm 500$ dps, 2 for $\pm 2000$ dps )
66(0x42)	Read accelerometer raw	12	Vector(float x3)	1	Accel range (byte: 0 for $\pm 2$ g, 1 for $\pm 4$ g, 2 for $\pm 8$ g )
67(0x43)	Read compass raw	12	Vector(float x3)	1	Compass range (byte: 0 for $\pm 0.88$ Ga, 1 for $\pm 1.3$ Ga, 2 for $\pm 1.9$ Ga, 3 for $\pm 2.5$ Ga, 4 for $\pm 4.0$ Ga, 5 for $\pm 4.7$ Ga, 6 for $\pm 5.6$ Ga, 7 for $\pm 8.1$ Ga)

### 4.3.4 Commands for Setting Filter Parameters

These commands allow the configuration of parameters associated with the Kalman filter. Most of these commands take parameters, none return any data.

Command	Description	Data Len	Data Details
96(0x60)	Tare with current orientation	0	
97(0x61)	Tare with quaternion	16	Quaternion(float x4)
98(0x62)	Tare with rotation matrix	36	Rotation Matrix(float x9)
99(0x63)	Set static rho mode(Accelerometer)	4	Rho value(float)
100(0x64)	Set confidence rho mode(Accelerometer)	8	Min rho value(float), Max rho value(float)
101(0x65)	Set static rho mode(Compass)	4	Rho value(float)
102(0x66)	Set confidence rho mode(Compass)	8	Min rho value(float), Max rho value(float)
103(0x67)	Set desired update rate	4	Update rate in microseconds(int)
104(0x68)	Set multi reference vectors with current orientation	0	
105(0x69)	Set reference vector mode	1	Mode(byte, 0 for single static, 1 for single auto, 2 for single auto continuous, 3 for multi)
106(0x6a)	Set oversample rate	1	Rate(1 for off, 2+ for number of samples per frame)
107(0x6b)	Enable/disable gyros	1	Mode(byte, 0 for disabled, 1 for enabled)
108(0x6c)	Enable/disable accelerometer	1	Mode(byte, 0 for disabled, 1 for enabled)
109(0x6d)	Enable/disable compass	1	Mode(byte, 0 for disabled, 1 for enabled)
110(0x6e)	Reset multi reference vectors to zero	0	
111(0x6f)	Set multi reference resolution	2	Resolution(byte x2, number of cell divisions, number of nearby vectors)
112(0x70)	Set compass multi reference vector	13	Index(byte), Vector(float x3)
113(0x71)	Set compass multi reference check vector	13	Index(byte), Vector(float x3)
114(0x72)	Set accel multi reference vector	13	Index(byte), Vector(float x3)
115(0x73)	Set accel multi reference check vector	13	Index(byte), Vector(float x3)
116(0x74)	Set axis directions	1	Axis direction byte
117(0x75)	Set running average percent	4	Percent(float)
118(0x76)	Set compass reference vector	12	Vector(float x3)
119(0x77)	Set accelerometer reference vector	12	Vector(float x3)
120(0x78)	Reset Kalman filter	0	
121(0x79)	Set accelerometer range	1	Accel range(byte, 0 for 2G, 0x10 for 4G, 0x30 for 8G)
122(0x7a)	Set multi reference weight power	4	Weight power(float)
123(0x7b)	Enable / disable filter	1	Mode(byte, 0 for disabled, 1 for enabled)
124(0x7c)	Set running average mode	1	Mode(byte, 0 for normal, 1 for confidence)
125(0x7d)	Set gyroscope range	1	Gyro range mode(byte, 0 for 250 dps, 1 for 500 dps, 2 for 2000 dps)
126(0x7e)	Set compass range	1	Compass range mode(byte, see command details for options)

### 4.3.5 Commands for Reading Filter Parameters

These commands allow the reading of parameters associated with the Kalman filter. All these commands return data, and accept no parameters.

Command	Description	Return Len	Return Data Details	Data Len	Data Details
128(0x80)	Read tare orientation(Quaternion)	16	Quaternion(float x4)	0	
129(0x81)	Read tare orientation(Rotation Matrix)	36	Rotation Matrix(float x9)	0	
130(0x82)	Read rho data(Accelerometer)	9	Rho mode(byte, 1 for confidence, 0 for static), min/static rho(float), (max rho(float))	0	
131(0x83)	Read rho data(Compass)	9	Rho mode(byte, 1 for confidence, 0 for static), min/static rho(float), (max rho(float))	0	
132(0x84)	Read current update rate	4	Update rate in microseconds(int)	0	
133(0x85)	Read compass reference vector	12	Vector(float x3)	0	
134(0x86)	Read accelerometer reference vector	12	Vector(float x3)	0	
135(0x87)	Read reference vector mode	1	Mode(byte, 0 for single static, 1 for single auto, 2 for single auto continuous, 3 for multi)	0	
136(0x88)	Read compass multi reference vector	12	Vector(float x3)	1	Index
137(0x89)	Read compass multi reference check vector	12	Vector(float x3)	1	Index
138(0x8a)	Read accel multi reference vector	12	Vector(float x3)	1	Index
139(0x8b)	Read accel multi reference check vector	12	Vector(float x3)	1	Index
140(0x8c)	Read gyro enabled state	1	Mode(byte, 0 for disabled, 1 for enabled)	0	
141(0x8d)	Read accelerometer enabled state	1	Mode(byte, 0 for disabled, 1 for enabled)	0	
142(0x8e)	Read compass enabled state	1	Mode(byte, 0 for disabled, 1 for enabled)	0	
143(0x8f)	Read axis directions	1	Axis direction byte	0	
144(0x90)	Read oversample rate	1	Rate(1 for off, 2+ for number of samples per frame)	0	
145(0x91)	Read running average percent	4	Percent(float)	0	
146(0x92)	Read desired update rate	4	Update rate in microseconds(int)	0	
147(0x93)	Read Kalman filter's covariance matrix	36	Covariance Matrix(float x9)	0	
148(0x94)	Read accelerometer range	1	Accel range(byte, 0 for 2G, 0x10 for 4G, 0x30 for 8G)	0	
149(0x95)	Read multi reference weight power	4	Weight power(float)	0	
150(0x96)	Read multi reference resolution	2	Resolution(byte x2, number of cell divisions, number of nearby vectors)	0	
151(0x97)	Read number of multi reference cells	4	Number of cells(int)	0	
152(0x98)	Read filter enable state	1	Mode(byte, 0 for disabled, 1 for enabled)	0	
153(0x99)	Read running average mode	1	Mode(byte, 0 for normal, 1 for confidence)	0	
154(0x9a)	Read gyroscope range	1	Gyro range mode(byte, 0 for 250 dps, 1 for 500 dps, 2 for 2000 dps)	0	
155(0x9b)	Read compass range	1	Gyro range mode(byte, see command details for options)	0	

### 4.3.6 Commands for Calibration

These commands allow the configuration and reading of calibration parameters and enabling of calibration modes.

Command	Description	Return Len	Return Data Details	Data Len	Data Details
160 (0xa0)	Set compass calibration parameters	0		48	Bias(float x3), Matrix(float x9)
161 (0xa1)	Set accelerometer calibration parameters	0		48	Bias(float x3), Matrix(float x9)
162 (0xa2)	Read compass calibration parameters	48	Bias(float x3), Matrix(float x9)	0	
163 (0xa3)	Read accelerometer calibration parameters	48	Bias(float x3), Matrix(float x9)	0	
164 (0xa4)	Read gyro calibration parameters	24	Bias(float x3), High range bias(float x3)	0	
165 (0xa5)	Begin gyro auto-calibration	0		0	
166(0xa6)	Set gyro calibration parameters	0		24	Bias(float x3), High range bias(float x3)
167(0xa7)	Set lookup-table vertex value	0		15	Value type(byte, 0 for compass, 1 for accelerometer), Index(short), Value(floatx3)
168(0xa8)	Read lookup-table vertex value	12	Value(floatx3)	3	Value type(byte, 0 for compass, 1 for accelerometer), index(short)

### 4.3.7 Battery Commands

These commands are specific to features of battery-powered sensors.

Command	Description	Return Data Len	Return Data Details	Data Len	Data Details
201(0xc9)	Read battery voltage	4	Voltage (float)	0	
202(0xca)	Read battery percent remaining	2	Percent (short)	0	
203(0xcb)	Read battery status	1	Status (byte)	0	

### 4.3.8 Data-Logging Commands

These commands are specific to features of data-logging sensors.

Command	Description	Return Data Len	Return Data Details	Data Len	Data Details
57(0x39)	Turn on mass storage mode	0		0	
58(0x3A)	Turn off mass storage mode	0		0	
59(0x3B)	Format and initialize SD card	0		0	
60(0x3C)	Begin data-logging session	0		0	
61(0x3D)	End data-logging session	0		0	
62(0x3E)	Set clock values	0		6	Month(byte), Day(byte), Year(byte), Hour(byte), Minute(byte), Second(byte)
63(0x3F)	Read clock values	6	Month(byte), Day(byte), Year(byte), Hour(byte), Minute(byte), Second(byte)	0	

### 4.3.9 General Commands

These commands are for the configuration of the sensor as a whole as opposed to configuration of the filter or sensors.

Command	Description	Return Data Len	Return Data Details	Data Len	Data Details
223(0xdf)	Read version extended	16	Version number(string)	0	
224(0xe0)	Restore factory settings	0		0	
225(0xe1)	Commit Settings	0		0	
226(0xe2)	Software reset	0		0	
227(0xe3)	Enable watchdog timer	0		4	Timeout rate in microseconds(int)
228(0xe4)	Disable watchdog timer	0		0	
229(0xe5)	Enter firmware update mode	0		0	
230(0xe6)	Get version	12	Version number(string)	0	
233(0xe9)	Set USB mode	0		1	Mode(byte, 2 for Unique, 1 for FTDI, 0 for CDC)
234(0xea)	Get USB mode	1	Mode(byte, 2 for Unique, 1 for FTDI, 0 for CDC)	0	
235(0xeb)	Set clock speed	0		4	Clock speed(int, Hz)
236(0xec)	Get clock speed	4	Clock speed(int, Hz)	0	
237(0xed)	Get serial number	4	Serial number(int)	0	
238(0xee)	Set LED color	0		12	Red(float), Green(float), Blue(float)
239(0xef)	Get LED color	12	Red(float), Green(float), Blue(float)	0	
240(0xf0)	Enable/Disable joystick	0		1	Mode(byte, 0 for disabled, 1 for enabled)
241(0xf1)	Enable/Disable mouse	0		1	Mode(byte, 0 for disabled, 1 for enabled)
242(0xf2)	Read joystick enabled state	1	Mode(byte, 0 for disabled, 1 for enabled)	0	
243(0xf3)	Read mouse enabled state	1	Mode(byte, 0 for disabled, 1 for enabled)	0	
244(0xf4)	Set control mode	0		3	Control class(byte), control index(byte), handler index(byte)
245(0xf5)	Set control data	0		7	Control class(byte), control index(byte), data point index(byte), data point(float)
246(0xf6)	Read control mode	1	Handler index(byte)	2	Control class(byte), control index(byte)
247(0xf7)	Read control data	4	Data point(float)	3	Control class(byte), control index(byte), data point index(byte)
248(0xf8)	Set button gyro disable length	0		1	Length in frames(byte)
249(0xf9)	Read button gyro disable length	1	Length in frames(byte)	0	
250(0xfa)	Read button state	1	Button state(byte, 1 bit per button)	0	
251(0xfb)	Set mouse absolute/relative	0		1	Mode(0 for absolute, 1 for relative)
252(0xfc)	Read mouse absolute/relative	1	Mode(0 for absolute, 1 for relative)	0	
253(0xfd)	Set joystick and mouse present/removed	0		2	Mode(byte for each, 0 for removed, 1 for present)
254(0xfe)	Read joystick and mouse present/removed	2	Mode(byte for each, 0 for removed, 1 for present)	0	

## 4.4. Command Details

In the tables below you'll find a description of each of the 3-Space Sensor commands and a brief explanation of how and where each command would be used.

<b>Function:</b>	Read filtered, tared orientation(Quaternion)
<b>Command Value:</b>	0 (0x00)
<b>Return Data Bytes:</b>	16
<b>Return Data Format:</b>	Quaternion(float x4)
<b>Description:</b>	Returns the final orientation as determined by the Kalman filter, relative to the tare orientation. This version returns the data as a quaternion.

<b>Function:</b>	Read filtered, tared orientation(Euler Angles)
<b>Command Value:</b>	1 (0x01)
<b>Return Data Bytes:</b>	12
<b>Return Data Format:</b>	Euler Angles(float x3)
<b>Description:</b>	Returns the final orientation as determined by the Kalman filter, relative to the tare orientation. This version returns the data as Euler angles.

<b>Function:</b>	Read filtered, tared orientation(Rotation Matrix)
<b>Command Value:</b>	2 (0x02)
<b>Return Data Bytes:</b>	36
<b>Return Data Format:</b>	Rotation Matrix(float x9)
<b>Description:</b>	Returns the final orientation as determined by the Kalman filter, relative to the tare orientation. This version returns the data as a quaternion.

<b>Function:</b>	Read filtered, tared orientation(Euler Angles)
<b>Command Value:</b>	3 (0x03)
<b>Return Data Bytes:</b>	16
<b>Return Data Format:</b>	Axis(float x3), Angle(float)
<b>Description:</b>	Returns the final orientation as determined by the Kalman filter, relative to the tare orientation. This version returns the data as an axis and an angle.

<b>Function:</b>	Read filtered, tared orientation(Two Vector(Forward, Down))
<b>Command Value:</b>	4 (0x04)
<b>Return Data Bytes:</b>	24
<b>Return Data Format:</b>	Vector(float x3), Vector(float x3)
<b>Description:</b>	Returns the final orientation as determined by the Kalman filter, relative to the tare orientation. This version returns the data as the forward and down vectors of the coordinate system defined by the orientation.

<b>Function:</b>	Read filtered gyro rates
<b>Command Value:</b>	5 (0x05)
<b>Return Data Bytes:</b>	12
<b>Return Data Format:</b>	Vector(float x3)
<b>Description:</b>	Returns the gyro rates as determined by taking the difference between the current orientation and the previous one.

## User's Manual

<b>Function:</b>	Read filtered, untared orientation(Quaternion)
<b>Command Value:</b>	6 (0x06)
<b>Return Data Bytes:</b>	16
<b>Return Data Format:</b>	Quaternion(float x4)
<b>Description:</b>	Returns the final orientation as determined by the Kalman filter, relative to the reference vectors. This version returns the data as a quaternion.

<b>Function:</b>	Read filtered, untared orientation(Euler Angles)
<b>Command Value:</b>	7 (0x07)
<b>Return Data Bytes:</b>	12
<b>Return Data Format:</b>	Euler Angles(float x3)
<b>Description:</b>	Returns the final orientation as determined by the Kalman filter, relative to the reference vectors. This version returns the data as Euler angles.

<b>Function:</b>	Read filtered, untared orientation(Rotation Matrix)
<b>Command Value:</b>	8 (0x08)
<b>Return Data Bytes:</b>	36
<b>Return Data Format:</b>	Rotation Matrix(float x9)
<b>Description:</b>	Returns the final orientation as determined by the Kalman filter, relative to the reference vectors. This version returns the data as a quaternion.

<b>Function:</b>	Read filtered, untared orientation(Euler Angles)
<b>Command Value:</b>	9 (0x09)
<b>Return Data Bytes:</b>	16
<b>Return Data Format:</b>	Axis(float x3), Angle(float)
<b>Description:</b>	Returns the final orientation as determined by the Kalman filter, relative to the reference vectors. This version returns the data as an axis and an angle.

<b>Function:</b>	Read filtered, untared orientation(Two Vector(Forward, Down))
<b>Command Value:</b>	10 (0x0A)
<b>Return Data Bytes:</b>	24
<b>Return Data Format:</b>	Vector(float x3), Vector(float x3)
<b>Description:</b>	Returns the final orientation as determined by the Kalman filter, relative to the reference vectors. This version returns the data as the forward and down vectors of the coordinate system defined by the orientation.

<b>Function:</b>	Read filtered, tared forward and down vectors in sensor reference frame (Two Vector(Forward, Down))
<b>Command Value:</b>	11 (0x0B)
<b>Return Data Bytes:</b>	24
<b>Return Data Format:</b>	Vector(float x3), Vector(float x3)
<b>Description:</b>	Returns the forward and down vectors as determined by the Kalman filter, relative to the tare orientation. This version returns the data as the forward and down vectors of the coordinate system defined by the sensor reference frame.

<b>Function:</b>	Read filtered, North, Earth vectors in sensor reference frame(Two Vector(North, Earth))
<b>Command Value:</b>	12(0x0C)
<b>Return Data Bytes:</b>	24
<b>Return Data Format:</b>	Vector(float x3), Vector(float x3)
<b>Description:</b>	Returns the North and Earth vectors as determined by the Kalman filter, relative to the global sensor references. Returned vectors are in the coordinate system defined by the sensor reference frame.

## User's Manual

<b>Function:</b>	Read all(gyro, accelerometer, compass)
<b>Command Value:</b>	32 (0x20)
<b>Return Data Bytes:</b>	36
<b>Return Data Format:</b>	Vector(float x3), Vector(float x3), Vector(float x3)
<b>Description:</b>	Returns the normalized gyro, accelerometer, and compass values.

<b>Function:</b>	Read gyros
<b>Command Value:</b>	33 (0x21)
<b>Return Data Bytes:</b>	12
<b>Return Data Format:</b>	Vector(float x3)
<b>Description:</b>	Returns the normalized gyro rates.

<b>Function:</b>	Read accelerometers
<b>Command Value:</b>	34 (0x22)
<b>Return Data Bytes:</b>	12
<b>Return Data Format:</b>	Vector(float x3)
<b>Description:</b>	Returns the normalized gravity vector.

<b>Function:</b>	Read compass
<b>Command Value:</b>	35 (0x23)
<b>Return Data Bytes:</b>	12
<b>Return Data Format:</b>	Vector(float x3)
<b>Description:</b>	Returns the normalized north vector.

<b>Function:</b>	Read temperature C
<b>Command Value:</b>	36 (0x24)
<b>Return Data Bytes:</b>	4
<b>Return Data Format:</b>	float
<b>Description:</b>	Returns the temperature of the sensor in Celsius

<b>Function:</b>	Read temperature F
<b>Command Value:</b>	37(0x25)
<b>Return Data Bytes:</b>	4
<b>Return Data Format:</b>	float
<b>Description:</b>	Returns the temperature of the sensor in Fahrenheit

<b>Function:</b>	Read confidence factor
<b>Command Value:</b>	38 (0x26)
<b>Return Data Bytes:</b>	4
<b>Return Data Format:</b>	float
<b>Description:</b>	Returns a value indicating how much the compass and accelerometer are to be trusted to be unit vectors pointing in the proper directions at the moment. This value ranges from 0 to 1, with 1 being fully trusted and 0 being not trusted at all. This will change based on if the sensor is being moved and if it is near a strong magnetic field.

## User's Manual

<b>Function:</b>	Read all raw(gyro, accelerometer, compass)
<b>Command Value:</b>	64 (0x40)
<b>Return Data Bytes:</b>	36
<b>Return Data Format:</b>	Vector(float x3), Vector(float x3), Vector(float x3)
<b>Description:</b>	Returns the raw gyro, accelerometer, and compass values. Values are according to the currently selected range for each respective sensor.

<b>Function:</b>	Read gyro raw
<b>Command Value:</b>	65 (0x41)
<b>Data Bytes:</b>	1
<b>Data Format:</b>	Gyro range ( Byte, 0 for $\pm 250$ dps, 1 for $\pm 500$ dps, 2 for $\pm 2000$ dps )
<b>Return Data Bytes:</b>	12
<b>Return Data Format:</b>	Vector(float x3)
<b>Description:</b>	Returns the raw gyro values in the specified measurement range.

<b>Function:</b>	Read accelerometer raw
<b>Command Value:</b>	66 (0x42)
<b>Data Bytes:</b>	1
<b>Data Format:</b>	Accelerometer range ( Byte, 0 for $\pm 2$ g, 1 for $\pm 4$ g, 2 for $\pm 8$ g )
<b>Return Data Bytes:</b>	12
<b>Return Data Format:</b>	Vector(float x3)
<b>Description:</b>	Returns the raw accelerometer values in the specified measurement range.

<b>Function:</b>	Read compass raw
<b>Command Value:</b>	67 (0x43)
<b>Data Bytes:</b>	1
<b>Data Format:</b>	Compass range (byte: 0 for $\pm 0.88$ Ga, 1 for $\pm 1.3$ Ga, 2 for $\pm 1.9$ Ga, 3 for $\pm 2.5$ Ga, 4 for $\pm 4.0$ Ga, 5 for $\pm 4.7$ Ga, 6 for $\pm 5.6$ Ga, 7 for $\pm 8.1$ Ga)
<b>Return Data Bytes:</b>	12
<b>Return Data Format:</b>	Vector(float x3)
<b>Description:</b>	Returns the raw compass values in the specified measurement range.

<b>Function:</b>	Tare with current orientation
<b>Command Value:</b>	96 (0x60)
<b>Description:</b>	Sets current filtered orientation as the tare orientation.

<b>Function:</b>	Tare with quaternion
<b>Command Value:</b>	97 (0x61)
<b>Data Bytes:</b>	16
<b>Data Format:</b>	Quaternion(float x4)
<b>Description:</b>	Sets the tare orientation to the orientation specified by the given quaternion.

<b>Function:</b>	Tare with rotation matrix
<b>Command Value:</b>	98 (0x62)
<b>Data Bytes:</b>	36
<b>Data Format:</b>	Rotation Matrix(float x9)
<b>Description:</b>	Sets the tare orientation to the orientation specified by the given rotation matrix.

## User's Manual

<b>Function:</b>	Set static rho mode(Accelerometer)
<b>Command Value:</b>	99 (0x63)
<b>Data Bytes:</b>	4
<b>Data Format:</b>	Rho value(float)
<b>Description:</b>	Sets the rho mode for the accelerometer to static, using the given value as rho.

<b>Function:</b>	Set confidence rho mode(Accelerometer)
<b>Command Value:</b>	100 (0x64)
<b>Data Bytes:</b>	8
<b>Data Format:</b>	Min rho value(float), Max rho value(float)
<b>Description:</b>	Sets the rho mode for the accelerometer to confidence, using the given values as the min and the max. The confidence factor will be used in real time to interpolate between these to determine what rho value is used.

<b>Function:</b>	Set static rho mode(Compass)
<b>Command Value:</b>	101 (0x65)
<b>Data Bytes:</b>	4
<b>Data Format:</b>	Rho value(float)
<b>Description:</b>	Sets the rho mode for the compass to static, using the given value as rho.

<b>Function:</b>	Set confidence rho mode(Compass)
<b>Command Value:</b>	102 (0x66)
<b>Data Bytes:</b>	8
<b>Data Format:</b>	Min rho value(float), Max rho value(float)
<b>Description:</b>	Sets the rho mode for the compass to confidence, using the given values as the min and the max. The confidence factor will be used in real time to interpolate between these to determine what rho value is used.

<b>Function:</b>	Set desired update rate
<b>Command Value:</b>	103 (0x67)
<b>Data Bytes:</b>	4
<b>Data Format:</b>	Update rate in microseconds(int)
<b>Description:</b>	Sets the target update rate to the given rate. If the filter takes less time to update during a given frame than this rate specifies, the frame will be padded out to take the specified amount of time. If the filter takes more time to update than this rate, this rate will be ignored.

<b>Function:</b>	Set multi reference vectors with current orientation
<b>Command Value:</b>	104 (0x68)
<b>Data Bytes:</b>	0
<b>Description:</b>	Uses the current tared orientation to set up the reference vector for the nearest orthogonal orientation.

## User's Manual

<b>Function</b>	Set reference vector mode
<b>Command Value:</b>	105 (0x69)
<b>Data Bytes:</b>	1
<b>Data Format:</b>	Mode(byte, 0 for single static, 1 for single auto, 2 for single auto continuous, 3 for multi)
<b>Description:</b>	<p>Sets reference vector mode.</p> <ul style="list-style-type: none"> <li>-Single static mode uses a reference vector for the compass and another reference vector for the accelerometer at all times.</li> <li>-Single auto mode uses (0,-1,0) as the reference vector for the accelerometer at all times and uses the current average angle between the accelerometer and compass to calculate the compass reference vector. After that this mode acts like single static mode.</li> <li>-Single auto continuous mode uses (0,-1,0) as the reference vector for the accelerometer at all times and uses the average angle between the accelerometer and compass to constantly redetermine the compass reference vector.</li> <li>-Multi mode uses a collection of reference vectors for the compass and accelerometer, and selects which ones to use before each step of the filter.</li> </ul>

<b>Function:</b>	Set oversample rate
<b>Command Value:</b>	106 (0x6a)
<b>Data Bytes:</b>	1
<b>Data Format:</b>	Rate(1 for off, 2+ for number of samples per frame)
<b>Description:</b>	Sets the number of times to sample data for each run of the filter.

<b>Function:</b>	Enable/disable gyros
<b>Command Value:</b>	107 (0x6b)
<b>Data Bytes:</b>	1
<b>Data Format:</b>	Mode(byte, 0 for disabled, 1 for enabled)
<b>Description:</b>	Enables or disables the gyros(will be replaced with a still gyro reading if disabled).

<b>Function:</b>	Enable/disable accelerometer
<b>Command Value:</b>	108 (0x6c)
<b>Data Bytes:</b>	1
<b>Data Format:</b>	Mode(byte, 0 for disabled, 1 for enabled)
<b>Description:</b>	Enables or disables the accelerometer(This filter will not use this data if disabled).

<b>Function:</b>	Enable/disable compass
<b>Command Value:</b>	109 (0x6d)
<b>Data Bytes:</b>	1
<b>Data Format:</b>	Mode(byte, 0 for disabled, 1 for enabled)
<b>Description:</b>	Enables or disables the compass(This filter will not use this data if disabled).

<b>Function:</b>	Reset multi reference vectors to zero
<b>Command Value:</b>	110 (0x6e)
<b>Data Bytes:</b>	0
<b>Description:</b>	Resets all reference vectors in the multi reference scheme to zero.

## User's Manual

<b>Function:</b>	Set multi reference resolution
<b>Command Value:</b>	111 (0x6f)
<b>Data Bytes:</b>	2
<b>Data Format:</b>	Mode(byte x2, number of cell divisions, number of nearby vectors)
<b>Description:</b>	<p>Sets number of cell divisions and number of nearby vectors per cell for the multi reference lookup table.</p> <p>By default, the multiple reference vector mode only deals with orientations obtainable by successive rotations of 90 degrees about any of the three axes. This command can adjust it to accept orientations obtainable by 45 degree rotations. For 90 degrees, give a "number of cell divisions" of 4, and for 45 give 8. In addition, the number of vectors near to any given orientation which the scheme will check can be adjusted as well. If this value is 0, only the nearest orientation will be checked. The largest this value can be is 32. Also note that the larger this value is, the longer the multiple reference vector mode will take to run each cycle.</p>

<b>Function:</b>	Set compass multi-reference vector
<b>Command Value:</b>	112 (0x70)
<b>Data Bytes:</b>	13
<b>Data Format:</b>	Index(byte), Vector(float x3)
<b>Description:</b>	Directly set compass multi reference vector at the specified index to the specified vector.

<b>Function:</b>	Set compass multi-reference check vector
<b>Command Value:</b>	113 (0x71)
<b>Data Bytes:</b>	13
<b>Data Format:</b>	Index(byte), Vector(float x3)
<b>Description:</b>	Directly set compass multi reference check vector at the specified index to the specified vector.

<b>Function:</b>	Set accel multi-reference vector
<b>Command Value:</b>	114 (0x72)
<b>Data Bytes:</b>	13
<b>Data Format:</b>	Index(byte), Vector(float x3)
<b>Description:</b>	Directly set accel multi reference vector at the specified index to the specified vector.

<b>Function:</b>	Set accel multi-reference check vector
<b>Command Value:</b>	115 (0x73)
<b>Data Bytes:</b>	13
<b>Data Format:</b>	Index(byte), Vector(float x3)
<b>Description:</b>	Directly set accel multi reference check vector at the specified index to the specified vector.

<b>Function:</b>	Set axis directions
<b>Command Value:</b>	116 (0x74)
<b>Data Bytes:</b>	1
<b>Data Format:</b>	Axis direction byte

<b>Description:</b>	<p>Set which directions each of the natural axes of the sensor point. The lower 3 bits are used to specify which axis each of the natural axes will read as:                  000: XYZ(standard operation)                  001: XZY                  002: YXZ                  003: YZX                  004: ZXY                  005: ZYX                  (For example, using ZXY means that whatever value appears as X on the natural axes will now be the Z component of any new data)                  The 3 bits above those are used to indicate which axes should be reversed. If it is cleared, the axis is pointing in the positive direction, and if it is set the axis is flipped.                  (Note: these are applied to the axes <i>after</i> the above conversion takes place)                  Bit 4: Positive/Negative Z ( third resulting component )                  Bit 5: Positive/Negative Y ( second resulting component )                  Bit 6: Positive/Negative X ( first resulting component )</p>
---------------------	---

<b>Function:</b>	Set running average percent
<b>Command Value:</b>	117 (0x75)
<b>Data Bytes:</b>	4
<b>Data Format:</b>	Percent(float)
<b>Description:</b>	<p>Sets what percentage of running average to use on the sensor's orientation. This is computed as follows:  <math>total\_orient = total\_orient * percent</math>  <math>total\_orient = total\_orient + current\_orient * (1 - percent)</math>  <math>current\_orient = total\_orient</math>                  If the percentage is 0, the running average will be shut off completely.</p>

<b>Function:</b>	Set compass reference vector
<b>Command Value:</b>	118 (0x76)
<b>Data Bytes:</b>	12
<b>Data Format:</b>	Vector(float x3)
<b>Description:</b>	Sets the static compass reference vector

<b>Function:</b>	Set accelerometer reference vector
<b>Command Value:</b>	119 (0x77)
<b>Data Bytes:</b>	12
<b>Data Format:</b>	Vector(float x3)
<b>Description:</b>	Sets the static accelerometer reference vector

<b>Function:</b>	Reset Kalman filter
<b>Command Value:</b>	120 (0x78)
<b>Data Bytes:</b>	0
<b>Data Format:</b>	None
<b>Description:</b>	Resets Kalman filter's covariance and state matrices

<b>Function:</b>	Set accelerometer range
<b>Command Value:</b>	121 (0x79)
<b>Data Bytes:</b>	1
<b>Data Format:</b>	Accel range(byte, 0 for ±2g, 0x10 for ±4g, 0x30 for ±8g)
<b>Description:</b>	Set which range of accelerometer data to use. Higher ranges can detect and report larger accelerations, but are not as accurate for smaller ones.

## User's Manual

<b>Function:</b>	Set multi reference weight power
<b>Command Value:</b>	122 (0x7a)
<b>Data Bytes:</b>	4
<b>Data Format:</b>	Weight power(float)
<b>Description:</b>	Set weighting power for multi reference vector weights. Multi reference vector weights are all raised to the weight power before they are summed and used in the calculation for the final reference vector. Setting this value nearer to 0 will cause the reference vectors to overlap more, and setting it nearer to infinity will cause the reference vectors to influence a smaller set of orientations.

<b>Function:</b>	Enable/disable filter
<b>Command Value:</b>	123 (0x7b)
<b>Data Bytes:</b>	1
<b>Data Format:</b>	Mode (byte, 0 for disabled, 1 for full kalman, 2 for gyroless-filtered orientation, 3 for fast-filtered )
<b>Description:</b>	Used to disable the orientation filter or set the orientation filter mode. Changing this parameter can be useful for tuning filter-performance versus orientation-update rates. When using the sensor as an IMU, it will improve performance to disable the orientation filter. When using as a AHRS, where orientation is needed, full-Kalman, gyroless-filtered, or fast-filtered should be used.

<b>Function:</b>	Set running-average mode
<b>Command Value:</b>	124 (0x7c)
<b>Data Bytes:</b>	1
<b>Data Format:</b>	Mode (byte, 0 for normal, 1 for confidence)
<b>Description:</b>	Selects the mode that the running-average method uses. Normal uses a static running-average. Confidence uses a running average that changes dynamically based upon the confidence factor.

<b>Function:</b>	Set gyroscope range
<b>Command Value:</b>	125 (0x7d)
<b>Data Bytes:</b>	1
<b>Data Format:</b>	Gyro range(byte: 0 for $\pm 250$ dps, 1 for $\pm 500$ dps, 2 for $\pm 2000$ dps )
<b>Description:</b>	Sets the measurement range used by the gyroscope sensor.

<b>Function:</b>	Set compass range
<b>Command Value:</b>	126 (0x7e)
<b>Data Bytes:</b>	1
<b>Data Format:</b>	Compass range (byte: 0 for $\pm 0.88$ Ga, 1 for $\pm 1.3$ Ga, 2 for $\pm 1.9$ Ga, 3 for $\pm 2.5$ Ga, 4 for $\pm 4.0$ Ga, 5 for $\pm 4.7$ Ga, 6 for $\pm 5.6$ Ga, 7 for $\pm 8.1$ Ga)
<b>Description:</b>	Sets the measurement range used by the compass sensor.

<b>Function:</b>	Read tare orientation(Quaternion)
<b>Command Value:</b>	128 (0x80)
<b>Return Data Bytes:</b>	16
<b>Return Data Format:</b>	Quaternion(float x4)
<b>Description:</b>	Reads the tare orientation as a quaternion.

<b>Function:</b>	Read tare orientation(Rotation Matrix)
<b>Command Value:</b>	129 (0x81)
<b>Return Data Bytes:</b>	36
<b>Return Data Format:</b>	Rotation Matrix(float x9)
<b>Description:</b>	Reads the tare orientation as a rotation matrix.

## User's Manual

<b>Function:</b>	Read rho data(Accelerometer)
<b>Command Value:</b>	130 (0x82)
<b>Return Data Bytes:</b>	9
<b>Return Data Format:</b>	Rho mode(byte, 1 for confidence, 0 for static), min/static rho(float), (max rho(float))
<b>Description:</b>	Reads the rho mode and rho data for the accelerometer.

<b>Function:</b>	Read rho data(Compass)
<b>Command Value:</b>	131 (0x83)
<b>Return Data Bytes:</b>	9
<b>Return Data Format:</b>	Rho mode(byte, 1 for confidence, 0 for static), min/static rho(float), (max rho(float))
<b>Description:</b>	Reads the rho mode and rho data for the compass.

<b>Function:</b>	Read current update rate
<b>Command Value:</b>	132 (0x84)
<b>Return Data Bytes:</b>	4
<b>Return Data Format:</b>	Update rate in microseconds(int)
<b>Description:</b>	Reads the amount of time the last frame took.

<b>Function:</b>	Read compass reference vector
<b>Command Value:</b>	133 (0x85)
<b>Return Data Bytes:</b>	12
<b>Return Data Format:</b>	Vector(float x3)
<b>Description:</b>	Reads the single mode compass reference vector.

<b>Function:</b>	Read accelerometer reference vector
<b>Command Value:</b>	134 (0x86)
<b>Return Data Bytes:</b>	12
<b>Return Data Format:</b>	Vector(float x3)
<b>Description:</b>	Reads the single mode accelerometer reference vector.

<b>Function:</b>	Read reference vector mode
<b>Command Value:</b>	135 (0x87)
<b>Return Data Bytes:</b>	1
<b>Return Data Format:</b>	Mode(byte, 0 for single static, 1 for single auto, 2 for single auto continuous, 3 for multi)
<b>Description:</b>	Reads the current reference vector mode. See command 105 for details.

<b>Function:</b>	Read compass multi reference vector
<b>Command Value:</b>	136 (0x88)
<b>Data Bytes:</b>	1
<b>Data Format:</b>	Index
<b>Return Data Bytes:</b>	12
<b>Return Data Format:</b>	Vector(float x3)
<b>Description:</b>	Reads the multi mode compass reference vector at index <Index>.

<b>Function:</b>	Read compass multi reference check vector
<b>Command Value:</b>	137 (0x89)
<b>Data Bytes:</b>	1
<b>Data Format:</b>	Index
<b>Return Data Bytes:</b>	12
<b>Return Data Format:</b>	Vector(float x3)
<b>Description:</b>	Reads the multi mode compass reference check vector at index <Index>.

<b>Function:</b>	Read accelerometer multi reference vector
<b>Command Value:</b>	138 (0x8a)
<b>Data Bytes:</b>	1
<b>Data Format:</b>	Index
<b>Return Data Bytes:</b>	12
<b>Return Data Format:</b>	Vector(float x3)
<b>Description:</b>	Reads the multi mode accelerometer reference vector at index <Index>.

<b>Function:</b>	Read accelerometer multi reference check vector
<b>Command Value:</b>	139 (0x8b)
<b>Data Bytes:</b>	1
<b>Data Format:</b>	Index
<b>Return Data Bytes:</b>	12
<b>Return Data Format:</b>	Vector(float x3)
<b>Description:</b>	Reads the multi mode accelerometer reference check vector at index <Index>.

<b>Function:</b>	Read gyro enabled state
<b>Command Value:</b>	140 (0x8c)
<b>Return Data Bytes:</b>	1
<b>Return Data Format:</b>	Mode(byte, 0 for disabled, 1 for enabled)
<b>Description:</b>	Reads the enabled state of the gyros.

<b>Function:</b>	Read accelerometer enabled state
<b>Command Value:</b>	141 (0x8d)
<b>Return Data Bytes:</b>	1
<b>Return Data Format:</b>	Mode(byte, 0 for disabled, 1 for enabled)
<b>Description:</b>	Reads the enabled state of the accelerometer.

<b>Function:</b>	Read compass enabled state
<b>Command Value:</b>	142 (0x8e)
<b>Return Data Bytes:</b>	1
<b>Return Data Format:</b>	Mode(byte, 0 for disabled, 1 for enabled)
<b>Description:</b>	Reads the enabled state of the compass.

<b>Function:</b>	Read axis directions
<b>Command Value:</b>	143 (0x8f)
<b>Return Data Bytes:</b>	1
<b>Return Data Format:</b>	Axis direction byte.
<b>Description:</b>	Reads the axis direction byte. For its meaning, see command 116.

<b>Function:</b>	Read oversample rate
<b>Command Value:</b>	144 (0x90)
<b>Return Data Bytes:</b>	1
<b>Return Data Format:</b>	Rate(1 for off, 2+ for number of samples per frame)
<b>Description:</b>	Reads the current oversample rate.

<b>Function:</b>	Read running average percent
<b>Command Value:</b>	145 (0x91)
<b>Return Data Bytes:</b>	4
<b>Return Data Format:</b>	Percent(float)
<b>Description:</b>	Reads the current running average percent. For its meaning, see command 117.

<b>Function:</b>	Read desired update rate
<b>Command Value:</b>	146 (0x92)
<b>Return Data Bytes:</b>	4
<b>Return Data Format:</b>	Update rate in microseconds(int)
<b>Description:</b>	Reads the current desired update rate. For more information on this, see command 103.

<b>Function:</b>	Read Kalman filter's covariance matrix
<b>Command Value:</b>	147 (0x93)
<b>Return Data Bytes:</b>	4
<b>Return Data Format:</b>	Covariance Matrix(float x9)
<b>Description:</b>	Read Kalman filter's covariance matrix.

<b>Function:</b>	Read accelerometer range
<b>Command Value:</b>	148 (0x94)
<b>Return Data Bytes:</b>	1
<b>Return Data Format:</b>	Accel range(byte, 0 for 2G, 0x10 for 4G, 0x30 for 8G)
<b>Description:</b>	Read accelerometer sensitivity range.

<b>Function:</b>	Read multi reference weight power
<b>Command Value:</b>	149 (0x95)
<b>Return Data Bytes:</b>	4
<b>Return Data Format:</b>	Weight power(float)
<b>Description:</b>	Read weighting power for multi reference vector weights.

<b>Function:</b>	Read multi reference resolution
<b>Command Value:</b>	150 (0x96)
<b>Return Data Bytes:</b>	2
<b>Return Data Format:</b>	Resolution(byte x2, number of cell divisions, number of nearby vectors)
<b>Description:</b>	Reads number of cell divisions and number of nearby vectors per cell for the multi reference lookup table. See command 111 for more information.

## User's Manual

<b>Function:</b>	Read number of multi reference cells
<b>Command Value:</b>	151 (0x97)
<b>Return Data Bytes:</b>	4
<b>Return Data Format:</b>	Number of cells(int)
<b>Description:</b>	Reads total number of multi reference cells.

<b>Function:</b>	Read filter enabled state
<b>Command Value:</b>	152 (0x98)
<b>Return Data Bytes:</b>	1
<b>Return Data Format:</b>	Mode (byte, 0 for disabled, 1 for full kalman, 2 for gyroless-filtered orientation, 3 for fast-filtered )
<b>Description:</b>	Reads the value of the filter mode enabled.

<b>Function:</b>	Read running-average mode
<b>Command Value:</b>	153 (0x99)
<b>Return Data Bytes:</b>	1
<b>Return Data Format:</b>	Mode (byte, 0 for normal, 1 for confidence)
<b>Description:</b>	Reads the selected mode for the running-average.

<b>Function:</b>	Read gyroscope range
<b>Command Value:</b>	154 (0x9a)
<b>Return Data Bytes:</b>	1
<b>Return Data Format:</b>	Gyro range(byte: 0 for $\pm 250$ dps, 1 for $\pm 500$ dps, 2 for $\pm 2000$ dps )
<b>Description:</b>	Reads the current measurement range setting used by the gyroscope sensor.

<b>Function:</b>	Set compass range
<b>Command Value:</b>	155 (0x9b)
<b>Return Data Bytes:</b>	1
<b>Return Data Format:</b>	Compass range (byte: 0 for $\pm 0.88$ Ga, 1 for $\pm 1.3$ Ga, 2 for $\pm 1.9$ Ga, 3 for $\pm 2.5$ Ga, 4 for $\pm 4.0$ Ga, 5 for $\pm 4.7$ Ga, 6 for $\pm 5.6$ Ga, 7 for $\pm 8.1$ Ga)
<b>Description:</b>	Reads the current measurement range setting used by the compass sensor.

<b>Function:</b>	Set compass calibration parameters
<b>Command Value:</b>	160 (0xA0)
<b>Data Bytes:</b>	48
<b>Data Format:</b>	Bias(float x3), Matrix(float x9)
<b>Description:</b>	Sets the compass calibration parameters to the given values. These consist of a bias which is applied to the raw data as a translation, and a matrix by which the value is multiplied by.

<b>Function:</b>	Set accelerometer calibration parameters
<b>Command Value:</b>	161 (0xA1)
<b>Data Bytes:</b>	48
<b>Data Format:</b>	Bias(float x3), Matrix(float x9)
<b>Description:</b>	Sets the accelerometer calibration parameters to the given values. These consist of a bias which is applied to the raw data as a translation, and a matrix by which the value is multiplied by.

## User's Manual

<b>Function:</b>	Read compass calibration parameters
<b>Command Value:</b>	162 (0xA2)
<b>Return Data Bytes:</b>	48
<b>Return Data Format:</b>	Bias(float x3), Matrix(float x9)
<b>Description:</b>	Reads the compass calibration parameters.

<b>Function:</b>	Read accelerometer calibration parameters
<b>Command Value:</b>	163 (0xA3)
<b>Return Data Bytes:</b>	48
<b>Return Data Format:</b>	Bias(float x3), Matrix(float x9)
<b>Description:</b>	Reads the accelerometer calibration parameters.

<b>Function:</b>	Read gyro calibration parameters
<b>Command Value:</b>	164 (0xA4)
<b>Return Data Bytes:</b>	24
<b>Return Data Format:</b>	Bias(float x3), High range bias(float x3)
<b>Description:</b>	Reads the gyroscope calibration parameters.

<b>Function:</b>	Begin gyro auto-calibration
<b>Command Value:</b>	165 (0xA5)
<b>Description:</b>	Puts the sensor in gyro calibration mode. It will collect a few frames of data from the gyro to determine its bias. It will return to normal operation after this or if the sensor is reset.

<b>Function:</b>	Set accelerometer calibration parameters
<b>Command Value:</b>	166 (0xA6)
<b>Data Bytes:</b>	24
<b>Data Format:</b>	Bias(float x3), High range bias(float x3)
<b>Description:</b>	Sets the gyroscope calibration parameters to the given values. These consist of a bias for each gyro mode which will be applied to the data from the appropriate mode as a translation.

<b>Function:</b>	Read battery voltage
<b>Command Value:</b>	201 (0xc9)
<b>Data Bytes:</b>	4
<b>Data Format:</b>	Voltage (float)
<b>Description:</b>	Reads the current battery voltage as read by the battery voltage monitoring circuit.

<b>Function:</b>	Read battery remaining percentage
<b>Command Value:</b>	202 (0xca)
<b>Data Bytes:</b>	2
<b>Data Format:</b>	Percentage (short)
<b>Description:</b>	Reads the calculated current battery remaining.

<b>Function:</b>	Read battery status
<b>Command Value:</b>	203 (0xcb)
<b>Data Bytes:</b>	1
<b>Data Format:</b>	Status (byte)
<b>Description:</b>	Reads status byte indicating the state of the battery. 1 represents fully charged, 2 represents charging.

## User's Manual

<b>Function:</b>	Turn on mass storage mode
<b>Command Value:</b>	57(0x39)
<b>Description:</b>	Places the sensor in mass storage mode, allowing the host computer to modify the SD card, and preventing all data-logging.

<b>Function:</b>	Turn off mass storage mode
<b>Command Value:</b>	58(0x3A)
<b>Description:</b>	Takes the sensor out of mass storage mode, taking control of the SD card away from the host computer and allowing data-logging to happen.

<b>Function:</b>	Format and initialize SD card
<b>Command Value:</b>	59(0x3B)
<b>Description:</b>	Formats the SD card using FAT32, and places the required directory structure on the card. This command takes about 15 seconds for a 2 GB card, during which no other communication can occur with the sensor.

<b>Function:</b>	Begin data-logging session
<b>Command Value:</b>	60(0x3C)
<b>Description:</b>	Begins a data-logging session using the current settings.

<b>Function:</b>	End data-logging session
<b>Command Value:</b>	61(0x3D)
<b>Description:</b>	Ends the current data-logging session, if one is ongoing.

<b>Function:</b>	Set clock values
<b>Command Value:</b>	62(0x3E)
<b>Data Bytes:</b>	6
<b>Data Format:</b>	Month(byte), Day(byte), Year(byte), Hour(byte), Minute(byte), Second(byte)
<b>Description:</b>	Sets the current time value of the real time clock.

<b>Function:</b>	Read clock values
<b>Command Value:</b>	63(0x3F)
<b>Return Data Bytes:</b>	6
<b>Return Data Format:</b>	Month(byte), Day(byte), Year(byte), Hour(byte), Minute(byte), Second(byte)
<b>Description:</b>	Reads the current time value from the real time clock.

<b>Function:</b>	Read version extended
<b>Command Value:</b>	223 (0xDF)
<b>Return Data Bytes:</b>	16
<b>Return Data Format:</b>	Version(string)
<b>Description:</b>	Reads the extended version string.

<b>Function:</b>	Restore factory settings
<b>Command Value:</b>	224 (0xE0)
<b>Description:</b>	Restores all settings to factory settings. The settings are not committed to non-volatile memory by this command, so the commit settings command will have to be used if this is desired.

<b>Function:</b>	Commit settings
<b>Command Value:</b>	225 (0xE1)
<b>Description:</b>	Commits settings to non-volatile memory. This will cause them to persist even if the sensor is reset.

<b>Function:</b>	Software Reset
<b>Command Value:</b>	226 (0xE2)
<b>Description:</b>	Resets the sensor.

<b>Function:</b>	Enable watchdog timer
<b>Command Value:</b>	227 (0xE3)
<b>Data Bytes:</b>	4
<b>Data Format:</b>	Timeout rate in microseconds(int)
<b>Description:</b>	Enables the watchdog timer with the given timeout rate. If a frame takes more than this amount of time, the sensor will automatically reset. This is useful for dealing with sensor hangs or crashes, as the sensor would reset and continue normal operation.

<b>Function:</b>	Disable watchdog timer
<b>Command Value:</b>	228 (0xE4)
<b>Description:</b>	Disables the watchdog timer.

<b>Function:</b>	Enter firmware update mode
<b>Command Value:</b>	229 (0xE5)
<b>Description:</b>	Puts the sensor into firmware update mode. This will cease normal operation until the firmware update mode is instructed to return the sensor to normal operation.

<b>Function:</b>	Get version
<b>Command Value:</b>	230 (0xE6)
<b>Return Data Bytes:</b>	12
<b>Return Data Format:</b>	Version(string)
<b>Description:</b>	Reads the version identifier of the sensor firmware. This will be "TSSUSB", followed by 6 digits representing the date the firmware version was created.

<b>Function:</b>	Set USB mode
<b>Command Value:</b>	233 (0xE9)
<b>Data Bytes:</b>	1
<b>Data Format:</b>	Mode(byte, 2 for Unique, 1 for FTDI, 0 for CDC)
<b>Description:</b>	Sets the communication mode for USB. All modes present a COM port with which to communicate with the USB device. FTDI and CDC each present a regular numbered port, whereas Unique presents a port named YEI_<serial number>. The sensor will change modes immediately.

<b>Function:</b>	Get USB mode
<b>Command Value:</b>	234 (0xEA)
<b>Return Data Bytes:</b>	1
<b>Return Data Format:</b>	Mode(byte, 2 for Unique, 1 for FTDI, 0 for CDC)
<b>Description:</b>	Reads the communication mode for USB, as described in the previous command.

<b>Function:</b>	Set clock speed
<b>Command Value:</b>	235 (0xEB)
<b>Data Bytes:</b>	4
<b>Data Format:</b>	Clock speed(int, Hz)
<b>Description:</b>	Sets the clock speed to the given value. Available settings are: 15Mhz, 30Mhz, 60Mhz. This setting does not need to be committed, but does not take effect until the sensor is reset.

## User's Manual

<b>Function:</b>	Get clock speed
<b>Command Value:</b>	236 (0xEC)
<b>Return Data Bytes:</b>	4
<b>Return Data Format:</b>	Clock speed(int, Hz)
<b>Description:</b>	Reads the clock speed of the sensor's MCU. Possible settings are: 15Mhz, 30Mhz, 60Mhz.

<b>Function:</b>	Get serial number
<b>Command Value:</b>	237 (0xED)
<b>Return Data Bytes:</b>	4
<b>Return Data Format:</b>	Serial number(int)
<b>Description:</b>	Reads the sensor's serial number.

<b>Function:</b>	Set LED color
<b>Command Value:</b>	238 (0xEE)
<b>Data Bytes:</b>	12
<b>Data Format:</b>	Red(float), Green(float), Blue(float)
<b>Description:</b>	Sets the color of the LED on the sensor to the given RGB color.

<b>Function:</b>	Get LED color
<b>Command Value:</b>	239 (0xEF)
<b>Return Data Bytes:</b>	12
<b>Return Data Format:</b>	Red(float), Green(float), Blue(float)
<b>Description:</b>	Reads the current color of the sensor's LED, in RGB format.

<b>Function:</b>	Enable/Disable joystick
<b>Command Value:</b>	240 (0xF0)
<b>Data Bytes:</b>	1
<b>Data Format:</b>	Mode(byte, 0 for disabled, 1 for enabled)
<b>Description:</b>	Turns the data feed to the joystick on and off. If disabled, the sensor will still enumerate as a joystick, but the joystick will not function. This allows normal communication to occur at a faster rate.

<b>Function:</b>	Enable/Disable mouse
<b>Command Value:</b>	241 (0xF1)
<b>Data Bytes:</b>	1
<b>Data Format:</b>	Mode(byte, 0 for disabled, 1 for enabled)
<b>Description:</b>	Turns the data feed to the mouse on and off. If disabled, the sensor will still enumerate as a mouse, but the mouse will not function. This allows normal communication to occur at a faster rate.

<b>Function:</b>	Read joystick enabled state
<b>Command Value:</b>	242 (0xF2)
<b>Return Data Bytes:</b>	1
<b>Return Data Format:</b>	Mode(byte, 0 for disabled, 1 for enabled)
<b>Description:</b>	Read whether or not the joystick is enabled.

<b>Function:</b>	Read mouse enabled state
<b>Command Value:</b>	243 (0xf3)
<b>Return Data Bytes:</b>	1
<b>Return Data Format:</b>	Mode(byte, 0 for disabled, 1 for enabled)
<b>Description:</b>	Read whether or not the mouse is enabled.

<b>Function:</b>	Set control mode
<b>Command Value:</b>	244 (0xf4)
<b>Data Bytes:</b>	3
<b>Data Format:</b>	Control class(byte), control index(byte), handler index(byte)
<b>Description:</b>	<p>Sets the operation mode for one of the controls. The control classes are:</p> <p>Joystick Axis: 0                      Joystick Button: 1                      Mouse Axis: 2                      Mouse Button: 3</p> <p>There are 2 axes and 8 buttons on the joystick and mouse. The index(0 based) selects which of these you would like to modify. The handler index selects which handler you want to take care of this control. The indices are:</p> <p>Turn off this control: 255</p> <p>Axes:                      Global Axis: 0                      Screen Point: 1</p> <p>Buttons:                      Hardware Button: 0                      Orientation Button: 1                      Shake Button: 2</p> <p>For more information on these, see the section on Control Customization.</p>

<b>Function:</b>	Set control data
<b>Command Value:</b>	245 (0xf5)
<b>Data Bytes:</b>	7
<b>Data Format:</b>	Control class(byte), control index(byte), data point index(byte), data point(float)
<b>Description:</b>	Sets parameters for the specified control's operation mode. The control classes and indices are the same as described in command 244. Each mode can have up to 10 data points associated with it. How many should be set and what they should be set to is entirely based on which mode is being used, so you should reference the section for that mode in the Control Customization section.

<b>Function:</b>	Read control mode
<b>Command Value:</b>	246 (0xf6)
<b>Data Bytes:</b>	2
<b>Data Format:</b>	Control class(byte), control index(byte)
<b>Return Data Bytes:</b>	1
<b>Return Data Format:</b>	Handler index(byte)
<b>Description:</b>	Read the index of this control's mode. The control classes and indices are the same as described in command 244.

<b>Function:</b>	Read control data
<b>Command Value:</b>	247 (0xf7)
<b>Data Bytes:</b>	3
<b>Data Format:</b>	Control class(byte), control index(byte), data point index(byte)
<b>Return Data Bytes:</b>	4
<b>Return Data Format:</b>	Data point(float)
<b>Description:</b>	Read the value of a certain parameter of the specified control's operation mode. The control classes and indices are the same as described in command 244.

## User's Manual

<b>Function:</b>	Set button gyro disable length
<b>Command Value:</b>	248 (0xf8)
<b>Data Bytes:</b>	1
<b>Data Format:</b>	Length in frames(byte)
<b>Description:</b>	Determines how long, in frames, the gyros should be disabled after one of the physical buttons on the sensor is pressed. A setting of 0 means they wont be disabled at all. This setting helps to alleviate gyro disturbances caused by the buttons causing small shockwaves in the sensor.

<b>Function:</b>	Read button gyro disable length
<b>Command Value:</b>	249 (0xf9)
<b>Return Data Bytes:</b>	1
<b>Return Data Format:</b>	Length in frames(byte)
<b>Description:</b>	Reads the current button gyro disable length.

<b>Function:</b>	Read button state
<b>Command Value:</b>	250 (0xfa)
<b>Return Data Bytes:</b>	1
<b>Return Data Format:</b>	Button state(byte, 1 bit per button)
<b>Description:</b>	Reads the current state of the sensor's physical buttons.

<b>Function:</b>	Set mouse absolute/relative
<b>Command Value:</b>	251 (0xfb)
<b>Data Bytes:</b>	1
<b>Data Format:</b>	Mode(0 for absolute, 1 for relative)
<b>Description:</b>	Puts the mouse in absolute or relative mode. Please note that this change does not take effect immediately, and the sensor must be reset before the mouse will enter this mode.

<b>Function:</b>	Read mouse absolute/relative
<b>Command Value:</b>	252 (0xfc)
<b>Return Data Bytes:</b>	1
<b>Return Data Format:</b>	Mode(0 for absolute, 1 for relative)
<b>Description:</b>	Reads the current mouse absolute/relative state. Note that if the sensor has not been reset since it has been put in this state, the mouse will not reflect this change yet, even though this command will.

<b>Function:</b>	Set joystick and mouse present/removed
<b>Command Value:</b>	253 (0xfd)
<b>Data Bytes:</b>	2
<b>Data Format:</b>	Mode(byte for each, 0 for removed, 1 for present)
<b>Description:</b>	Sets whether the joystick and mouse are present or removed. If removed, they will not show up as devices on the target system at all. For these changes to take effect, the sensor driver may need to be reinstalled.

<b>Function:</b>	Read joystick and mouse present/removed
<b>Command Value:</b>	254 (0xfe)
<b>Return Data Bytes:</b>	2
<b>Return Data Format:</b>	Mode(byte for each, 0 for removed, 1 for present)
<b>Description:</b>	Reads the joystick and mouse present/removed state. See command 253 for more detail.

# Appendix

## USB Connector

The 3-Space Sensor has a 5-pin USB Type-B jack and can be connected via a standard 5-pin mini USB cable.

## Hex / Decimal Conversion Chart

		<i>Second Hexadecimal digit</i>															
		<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>	<i>7</i>	<i>8</i>	<i>9</i>	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>	<i>F</i>
First Hexadecimal Digit	<i>0</i>	000	001	002	003	004	005	006	007	008	009	010	011	012	013	014	015
	<i>1</i>	016	017	018	019	020	021	022	023	024	025	026	027	028	029	030	031
	<i>2</i>	032	033	034	035	036	037	038	039	040	041	042	043	044	045	046	047
	<i>3</i>	048	049	050	051	052	053	054	055	056	057	058	059	060	061	062	063
	<i>4</i>	064	065	066	067	068	069	070	071	072	073	074	075	076	077	078	079
	<i>5</i>	080	081	082	083	084	085	086	087	088	089	090	091	092	093	094	095
	<i>6</i>	096	097	098	099	100	101	102	103	104	105	106	107	108	109	110	111
	<i>7</i>	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
	<i>8</i>	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
	<i>9</i>	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
	<i>A</i>	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
	<i>B</i>	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
	<i>C</i>	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
	<i>D</i>	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
	<i>E</i>	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
	<i>F</i>	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

**Notes:**

Serial Number: \_\_\_\_\_





**YEI Technology**  
630 Second Street  
Portsmouth, Ohio 45662

Toll-Free: 888-395-9029  
Phone: 740-355-9029

[www.YeiTechnology.com](http://www.YeiTechnology.com)  
[www.3SpaceSensor.com](http://www.3SpaceSensor.com)