3-Space Sensor

3-Space Sensor LX

Ultra-Miniature Attitude & Heading Reference System

With

Pedestrian Tracking

User's Manual

Yost Labs
630 Second Street
Portsmouth, Ohio 45662

www.yostlabs.com

3-Space Sensor LX

Ultra-Miniature Attitude & Heading Reference System
With
Pedestrian Tracking

User's Manual

Yost Labs
630 Second Street
Portsmouth, Ohio 45662

www.yostlabs.com

Phone: 740-876-4936

Table of Contents

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.
- The Yost Labs 3-Space LX module contains components that are sensitive to electro-static-discharge. Care should be taken when handling the module.
- PCB layout can affect the performance of the 3-Space LX module. Placing magnetic components, ferrous metal containing components, high-current conductors, motors, solenoids, actuators, and high-frequency digital signal lines should be avoided in proximity during PCB layout.

1.2 Technical Support and Repairs

Yost Labs provides technical and user support via phone at 740-876-4936 and via email (support@yostlabs.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.

2. Overview of the Yost Labs 3-Space Sensor

2.1 Introduction

The Yost Labs 3-Space Sensor™ LX is an ultra-miniature, high-precision, high-reliability, low-cost, low-power consuming SMT Attitude and Heading Reference System (AHRS) which 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. Additional algorithms support pedestrian tracking algorithms that, can compute and store an estimated motion path of a walking individual carrying or wearing the sensor.

Orientation can be returned in absolute terms or relative to a designated reference orientation. The proprietary dynamic stability algorithms increase accuracy and greatly reduce and compensate for sensor error. The Yost Labs 3-Space Sensor LX system also utilizes a dynamic sensor confidence algorithm that ensures optimal accuracy and precision across a wide range of operating conditions.

Motion-path data can be return as a series of steps that include direction, displacement, along with a confidence metric that can be used to characterize the accuracy of the estimated motion-path.

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

The 3-Space Sensor LX module also offers a range of communication interface options which include SPI, I2C, USB 2.0, and asynchronous serial.

2.2 Applications

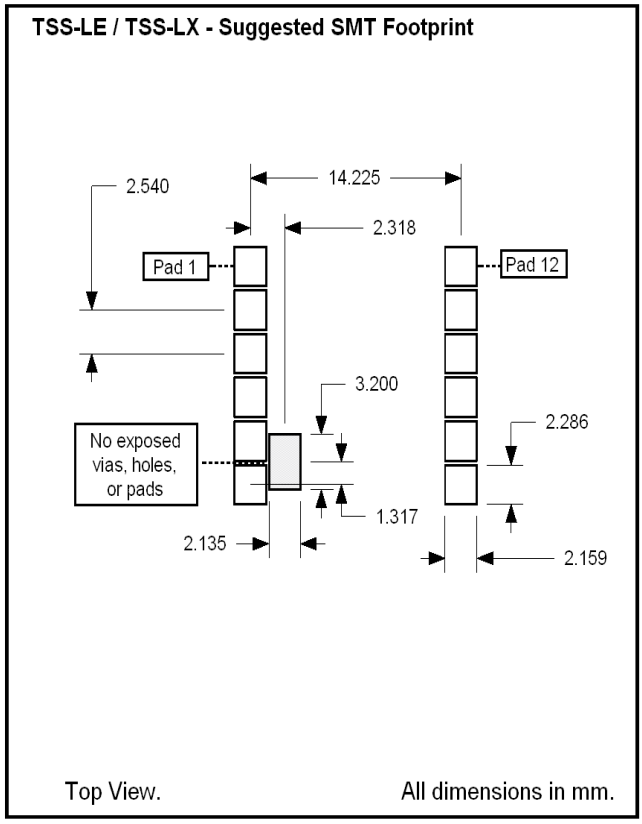
- Personnel / Pedestrian navigation and tracking
- Robotics
- Motion capture
- Positioning and stabilization
- Vibration analysis
- Inertial augmented localization
- Unmanned air/land/water vehicle navigation
- Education and performing arts
- Healthcare monitoring
- Gaming and motion control
- Accessibility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

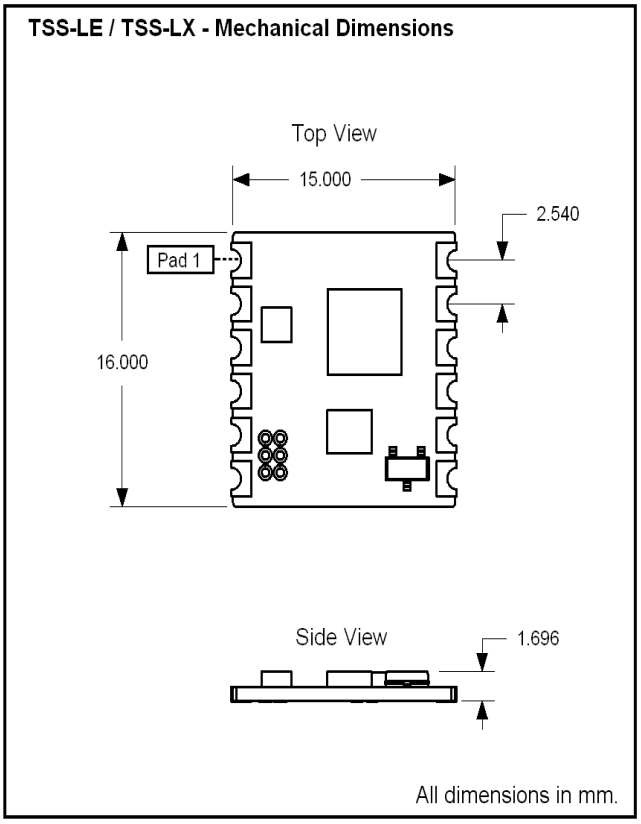
2.3 Hardware Overview

The Yost Labs 3-Space LX is packaged as a 16mmx15mmx1.696mm castellated edge SMT module. Alternatively, the module can be through-hole mounted by adding standard 0.1" header strips to the castellated edge pads.

2.3.1 PCB Layout

PCB layout should follow follow the suggested SMT footprint below.





Additionally, since PCB layout can affect the performance of the 3-Space LX module observe the following layout guidelines:

- Do not place un-tented pads, vias, or holes beneath the restricted area in the diagram.
- Do not place magnetic components such as speakers and motors in close proximity to the module since the magnetic fields generated can adversely affect the performance of the compass module.
- Do not place components containing ferrous metals in close proximity to the module since they may disturb earth's magnetic fields and thus adversely affect the performance of the compass module.
- Do not route high-current conductors or high-frequency digital signal lines in close proximity to the module since they may generate magnetic fields that may adversely affect the performance of the compass module.
- Do not reflow with the device on the bottom of a board. Since the module's components aren't glue-bonded to the module they may become dis-lodged if reflowed in non-up-facing orientations.
- Thoroughly test and characterize any PCB design that uses the module. Failure to test and characterize a system using the TSS-LX module may result in unforeseen performance consequences due to layout.

2.3.2 Pin Functions

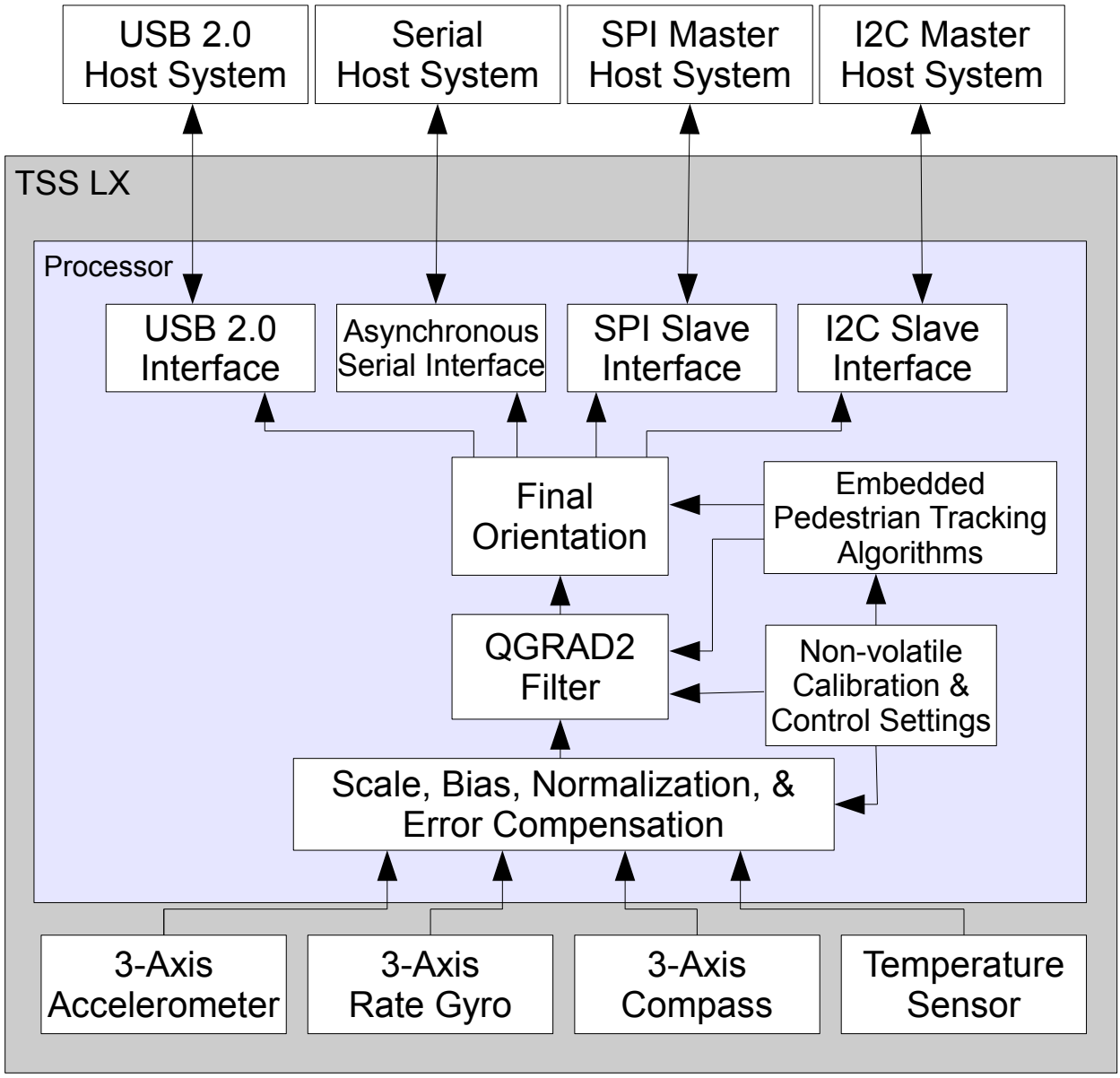
Pad Number	Signal Name	Description
1	SCL	I2C Serial Clock. Input to Module.
2	SDA	I2C Data Line. Input to / Output from Module.
3	GND1	Ground.
4	USBD-	USB Data Minus. Only requires connection during USB mode use.
5	USBD+	USB Data Plus. Only requires connection during USB mode use.
6	VIN	Voltage Input +3.3v ~ +6.0v.
7	/SS	SPI Slave Select. Active Low Input to Module.
8	MOSI	SPI Master Out Slave In. Input to Module.
9	MISO	SPI Master In Slave Out. Output from Module.
10	SCK	SPI Serial Clock. Input to Module.
11	RXD	UART Asynchronous Receive Data. Input to Module.
12	TXD	UART Asynchronous Transmit Data. Output from Module.

2.4 Features

The Yost Labs 3-Space Sensor LX 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:

- Smallest and lightest high-performance AHRS available at 16mmx15mmx1.696mm and less than 1 gram
- Fast sensor update and filter rate allow use in real-time applications, including stabilization, virtual reality, real-time immersive simulation, and robotics
- Highly customizable orientation sensing with options such as tunable filtering, oversampling, and orientation error correction
- Advanced integrated orientation filtering allows sensor to automatically reduce the effects of sensor noise and sensor error
- Integrated pedestrian tracking algorithms allow the extraction of detailed motion-path information.
- 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)
- Absolute or custom reference axes
- Access to raw sensor data
- Flexible communication options: SPI, I2C, USB 2.0, or asynchronous serial
- USB communication through a virtual COM port
- Castellated SMT edge pads provide secure SMT mounting and allow optional through-hole mounting
- Upgrade-able firmware
- Development kit available
- RoHS Compliant
- +5v tolerant I/O signals

2.5 Block Diagram of Sensor Operation



2.6 Specifications

General	
Part number	TSS-LX
Dimensions	16mm x 15mm x 1.696mm (0.63 x 0.59 x 0.067 in.)
Weight	>1 gram
Supply voltage	+3.3v ~ +6.0v
Power consumption	22mA @ 5v
Communication interfaces	USB 2.0, SPI, I2C, Asynchronous Serial
Filter update rate	Up to 1750Hz with full functionality
Orientation output	absolute & relative quaternion, Euler angles, axis angle, rotation matrix
Other output	raw sensor data, normalized sensor data, temperature, pedestrian motion path data
SPI clock rate	2.5 MHz max
I2C clock rate	300 kHz max
Serial baud rate	1,200~921,600 selectable, default: 115,200
Shock survivability	5000g
Temperature range	-40C ~ 85C (-40F ~ 185F)
Processor	32-bit RISC running @ 80MHz
Sensor	
Orientation range	360º about all axes
Orientation accuracy	±2º for dynamic conditions & all orientations
Orientation resolution	<0.08º
Orientation repeatability	0.085º for all orientations
Accelerometer scale	±2g / ±4g / ±8g / ±16g selectable
Accelerometer resolution	14 bit
Accelerometer noise density	90μg/√Hz for ±2g & ±4g range 110μg/√Hz for ±8g range 180μg/√Hz for ±16g range
Accelerometer sensitivity	0.000061g/LSB for ±2g range 0.000122g/LSB for ±4g range 0.000244g/LSB for ±8g range 0.000488g/LSB for ±16g range
Accelerometer temperature sensitivity	±1%/°C
Gyro scale	±125/±245/±500/±1000/±2000 º/sec selectable
Gyro resolution	16 bit
Gyro noise density	0.007º/sec/√Hz
Gyro bias stability @ 25°C	2.5º/hr average for all axes
Gyro sensitivity	0.004375º/sec/digit for ±125º/sec 0.00875º/sec/digit for ±245º/sec 0.01750º/sec/digit for ±500º/sec 0.03500º/sec/digit for ±1000º/sec 0.07000º/sec/digit for ±2000º/sec
Gyro non-linearity	0.2% full-scale
Gyro temperature sensitivity	±1.5%/°C
Compass scale	±49.152 Ga default.
Compass resolution	16 bit
Compass sensitivity	1.5 mGa/digit
Compass non-linearity	0.1% full-scale

2.7 Electrical Characteristics

2.7.1 Absolute Maximum Ratings*

Operating Temperature ….....	-40C ~ 85C (-40F ~ 185F)
Storage Temperature ….....	-60C ~ 150C (-76F ~ 302F)
Supply Voltage on VIN Pin with respect to Ground	-0.3v ~ 6.5v
Supply Voltage on VUSB Pin with respect to Ground	-0.3v ~ 6.5v
Voltage on I/O Pins with respect to Ground ….....	-0.3v ~ 5.5v
Current Sink/Source from I/O pins ….....	-4mA ~ +4mA

* NOTICE: Stresses beyond those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may adversely affect device reliability.

2.7.2 DC Characteristics

The following characteristics are applicable to the operating temperature range: TA = -40°C to 85°C

Symbol	Parameter	Min.	Typ.	Max.	Units
V_{IN}	Operating Supply Voltage on VIN pin	3.2	3.3	6.0	V
V_{USB}	Operating Supply Voltage on VUSB pin	3.8	5.0	6.0	V
V_{IL}	Input Low-level Voltage	-0.3		+0.8	V
V_{IH}	Input High-level Voltage	2.0		5.5	V
V_{OL}	Output Low-level Voltage			0.4	V
V_{OH}	Output High-level Voltage	2.6			V
I_{OL}	Output Low-level Current			-4	mA
I_{OH}	Output High-level Current			4	mA
C_{IN}	Input Capacitance			7	pF
I_{ACTLX}	Active Current Consumption for TSS-LX		22		mA

2.7.3 USB Characteristics

The on-chip USB interface complies with the Universal Serial Bus (USB) v2.0 standard. All AC parameters related to these buffers can be found within the USB 2.0 electrical specifications.

2.7.4 Asynchronous Serial Characteristics

The on-chip Asynchronous Serial interface is compatible with UARTs available on most micro-controllers. The device utilizes a minimum-wire configuration consisting of two communication wires: a TxD serial output and an RxD serial input. The Serial interface drives the TxD line at 3v logic-levels and the RxD input is 2.0~5.5v tolerant. Also note that since logic-level serial is voltage-based, the two connected systems must share a common ground reference.

For connection to alternate communication interfaces such as RS232, RS422, RS485, MIL-STD-188, EIA/TIA-562, and SpaceWire, additional external interface drivers may be added.

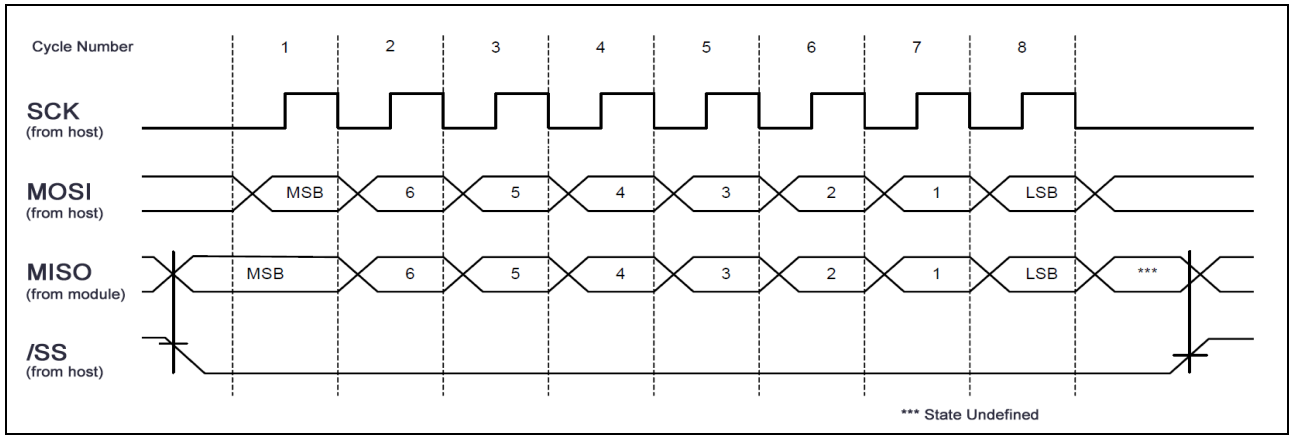
The Asynchronous Serial uses 8N1 (8 data bits, no parity, 1 stop bit) format and supports the following standard baud rates: 1200, 2400, 4800, 9600, 19200, 28800, 38400, 57600, 115200, 230400, 460800, 921600.

The factory default baud rate is 115200.

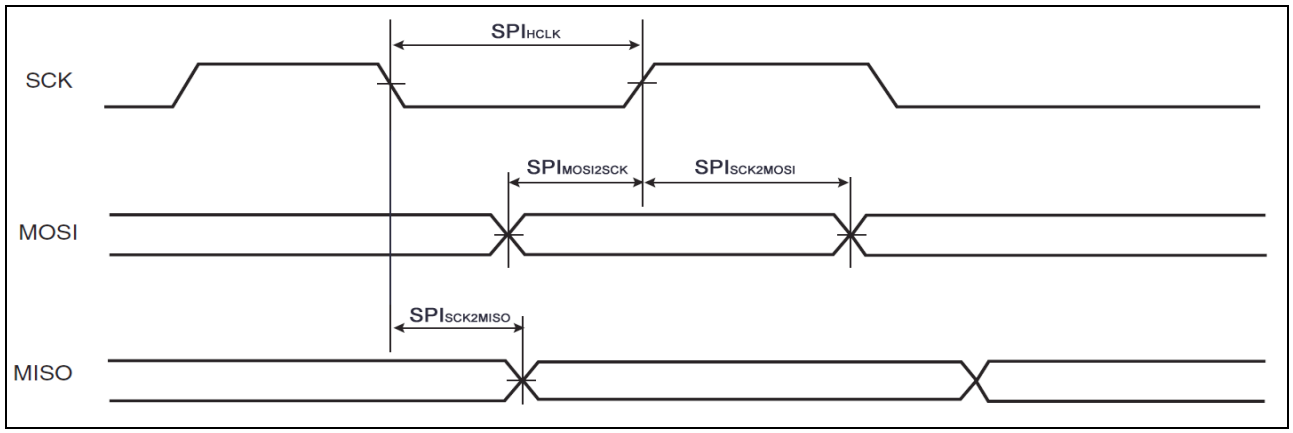
2.7.5 SPI Characteristics

The Serial Peripheral Interface or SPI is a full-duplex synchronous serial communication standard that is commonly supported on many micro-controllers and embedded systems.

The SPI interface is implemented as an SPI mode 0 slave device. This means that the SPI clock polarity is 0 (CPOL=0) and the SPI clock phase is 0 (CPHA=0). Bytes are transferred one bit at a time with the MSB being transferred first. The on-board SPI interface has been tested at speeds up to 2.5MHz. The diagram below illustrates a single complete SPI byte transfer.



The diagram and parameter table below illustrates additional timing requirements and limits of the SPI interface:



Symbol	Parameter	Min.	Max.	Units
SPI_{HCLK}	SPI Clock Cycle Period / 2	30		ns
$SPI_{SCK2MISO}$	SPI SCK falling to MISO Delay		6	ns
$SPI_{MOSI2SCK}$	SPI MOSI Setup time before SPI SCK rises	0		ns
$SPI_{SCK2MOSI}$	SPI MOSI Hold time after SPI SCK rises	1.5		ns

2.7.6 I2C Characteristics

The Inter-Integrated Circuit or I2C is a synchronous serial communication standard that is commonly supported on many micro-controllers and embedded systems.

The I2C interface is implemented as a slave device. The clock and data line should be pulled high until communication begins. Communication begins with a START condition (ST) of a HIGH to LOW transition on the data line while the SCL line is held HIGH. The I2C interface requires an address (ADD) of 0xEE for the slave device. Bytes are transferred one bit at a time with the MSB being transferred first. Both slave and master acknowledge signals (SAK and MAK) are required depending on the type communication. Communication ends with a No Master Acknowledge (NMAK) and a STOP condition (SP) of a LOW to HIGH transition on the SDA line while the SCL line is HIGH. The on-board I2C interface has been tested at speeds up to 300 kHz. The diagrams below illustrates complete I2C transfers.

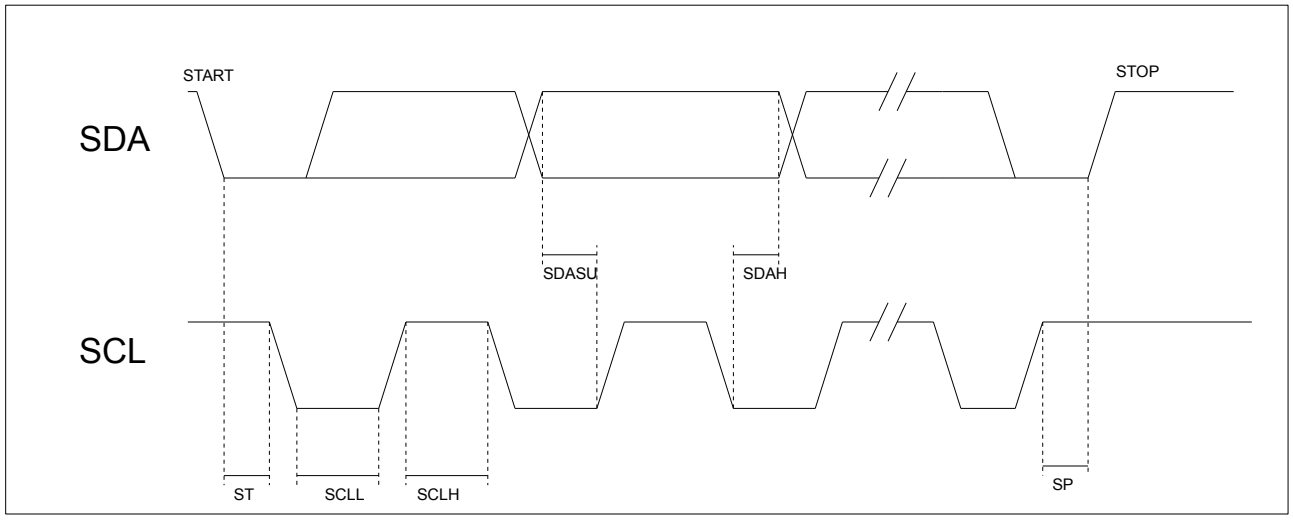
READ DATA:

MASTER	ST	ADD		I2C_CMD			MAK		NMAK	SP
SLAVE			SAK		SAK	DATA		DATA		

WRITE DATA:

MASTER	ST	ADD		I2C_CMD		DATA		DATA	NMAK	SP
SLAVE			SAK		SAK		SAK			

The diagram and parameter table below illustrates additional timing requirements and limits of the I2C interface:



Symbol	Parameter	Min.	Max.	Units
SCLf	SCL frequency	0	300	µs
SCLL	SCL low time	2.2		µs
SCLH	SCL high time	4.2		µs
SDASU	SDA setup time	2		µs
SDAH	SDA hold time	0	400	ns
ST	START condition setup time	1		µs
SP	STOP condition setup time	4		µs

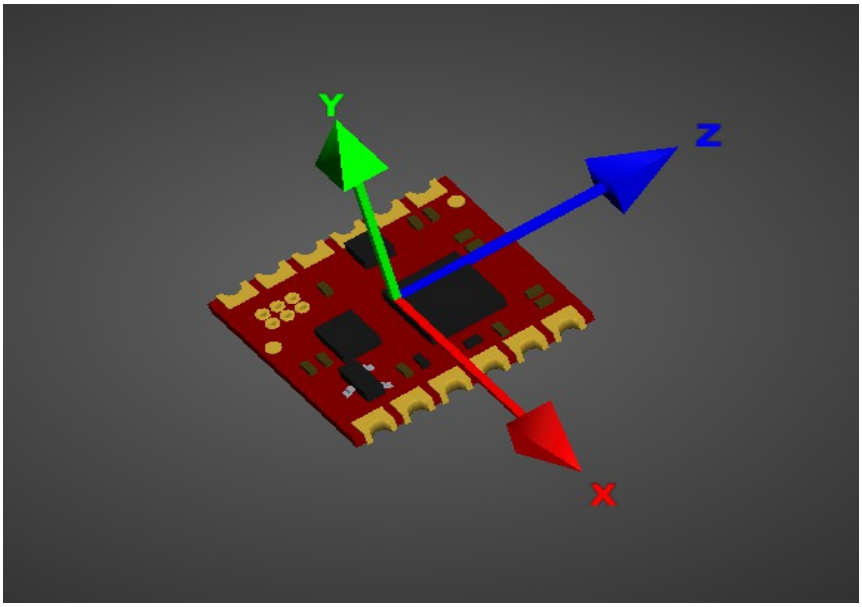
2.8 Axis Assignment

All Yost Labs 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 LX are as follows:

- The positive X-axis points out of the side of the sensor with pins 7 through 12.
- The positive Y-axis points out of the top of the sensor (the component side of the board).
- The positive Z-axis points out of the side of the sensor between pins 1 through 12.

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 Sensor Filter

The component sensor data and reference vectors are fed into a high-performance orientation 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 QGRAD2 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.

3.2 Pedestrian Tracking

The 3-Space Sensor LX includes an embedded pedestrian tracking algorithm that, once activated, uses the AHRS and the inertial sensor data to compute a step-by-step motion path of a body-worn or human carried sensor. For each step that is taken, the sensor logs multiple data items including the heading of travel, displacements in travel and other data relevant to each step in the sequence.

To use the pedestrian tracking feature, the following action are recommended:

Starting Tracking:

Turn on pedestrian tracking by using command 52(0x34), sub-command 0, with a data value of 1.0. This activates pedestrian tracking mode. Once activated, every step will be logged into the internal memory buffer to be read later.

Pausing Tracking:

To pause pedestrian tracking, use command 52(0x34), sub-command 0, with a data value of 0.0.

Resume Tracking:

To resume pedestrian tracking use command 52(0x34), sub-command 0, with a data value of 1.0.

Reading the Most Recent Step Data:

To read the most recent step data, use command 53(0x35), sub-command 22. This will return 48 bytes(12 floats) that represent the following data items:

- **Index** (float offset 0) – The step index number. Step indices start with 0 and increment up to 99.
- **Heading** (float offset 1) – The heading of travel in decimal as degrees in the range(0-360). 0 is North, 90 is East, 180 is South, 270 is West.
- **X_step_distance** (float offset 2) – Distance traveled along the east/west axis. Units of measure are determined by the units selected by the command 52(0x34), sub-command 12, the "Set Units"command.
- **Y_step_distance** (float offset 3) – Distance traveled along the north/south axis. Units of measure are determined by the units selected by the command 52(0x34), sub-command 12, the "Set Units"command.
- **Step_distance** (float offset 4) – Distance traveled during the current step. Units of measure are determined by the units selected by the command 52(0x34), sub-command 12, the "Set Units"command.
- **Step_time** (float offset 5) – The duration of the step, in seconds. This can be used to compute the speed of travel or the duration of travel.
- **Step_amplitude** (float offset 6) - The amplitude of the step. This can be used for additional post-processing and visualization.
- **Step_duty_cycle** (float offset 7) – The ratio of the step-cycle time duration in the positive region to the time in the negative region. This can be used for additional post-processing and visualization.
- **Step_ratio** (float offset 8) – The ratio of the integrated area beneath the positive region of the step to the integrated area above the negative region of the step.
- **Altitude (float offset 9)** – Not available on the LX sensors will always report 0, available on the Embedded sensor.
- **Confidence (float offset 10)** – This is a value 0.0 and 100.0 where 100.0 is the highest level of confidence in the step data. High levels of confidence indicate a high-probability of the detected distances and the captured motion path being accurate. Low confidence values indicate a higher-likelihood an inaccurate distance estimate and motion path.
- **Altitude_Offset ((float offset 11)** – Not available on the LX sensors will always report 0, available on the Embedded sensor.

Reading Step Data at an Index:

To read the step data at an index, first use command 52(0x34), sub-command 1 with a float passed that is the desired step index to read from. After the index has been set, use command 53(0x35), sub-command 23. This will return 48 bytes(12 floats) that represent the step data items as described above. Index values can range from 0 to 99. If more that 100 steps are needed for the application, the host machine can read out the step sequence and then issue a command to clear the step data.

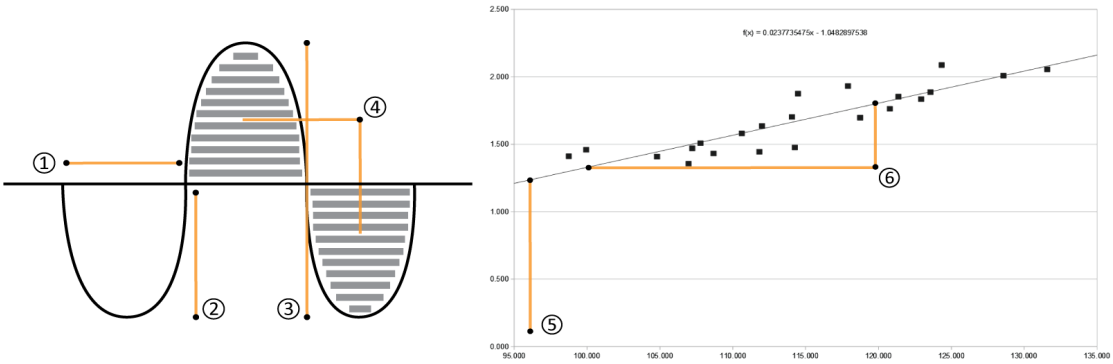
Clearing Step Data:

To clear the recorded step data and begin a new step sequence, use command 52(0x34), sub-command 15 with a single dummy float value of 0.0 passed to it. This discards the stored step sequence and allows a new step sequence to begin being logged.

Setting Pedestrian Tracking Parameters:

The pedestrian tracking algorithms make use of several parameters that can be used to control/tune performance and accuracy of the estimated motion path. The following figures can be used as a reference to understand the parameters.

Step Algorithm Parameter Diagram



① Step Duration ② Amin / Amax ③ Dmin / Dmax ④ Rmin / Rmax ⑤ Stride Offset ⑥ Step Slope

The pedestrian tracking parameters are as follows:

- **Rmin** - The minimum allowable value of the ratio of the positive step area to the negative step area for a valid step (4).
- **Rmax** - The maximum allowable value of the ratio of the positive step area to the negative step area for a valid step (4).
- **Dmin** - The minimum allowable duty-cycle for a detected step wave-form (3).
- **Dmax** - The maximum allowable duty-cycle for a detected step wave-form (3).
- **Amin** - The minimum allowable amplitude for a detected step wave-form (2).
- **Amax** - The maximum allowable amplitude for a detected step wave-form (2).
- **Step_Duration_Min** - The minimum allowable duration for a detected step wave-form (1).
- **Step_Duration_Max** - The maximum allowable duration for a detected step wave-form (1).
- **Step_Stride_Slope** - The slope of the stride-to-cadence relation used to convert step time to distance (6).
- **Step_Stride_Offset** – The offset of the stride-to-cadence relation used to convert step time to distance (5).
- **Units** - The units of measurement output by the pedestrian tracking commands. 0=meters, 1=feet.
- **Altitude_Offset** – Not available on the LX sensors will always report 0, available on the Embedded sensor.

Pedestrian Tracking Performance:

To achieve good results with the pedestrian tracking algorithms within the sensor it is important to observe certain operating conditions.

Here is a list of considerations that will lead to good performance of the pedestrian tracking feature:

Consistency

When using the position tracking algorithm, the user should try to orient the device's forward direction with their own direction to achieve an accurate and consistent heading. Please note that the device's orientation should remain consistent throughout the duration of tracking for the best distance and step tracking results. The device should not transition from upright to upside down during position tracking. Note that walking up steps or across steep grades may reduce the accuracy of the distance calculated. Note that confidence is based on distance vs. steps taken.

Interference

The LX 3-Space Sensor will be affected by interference from sources of magnetic perturbation such as active stepper motors, servo motors, and other sources of magnetic interference. To ensure minimal interference on sensing components all potentially interfering components should be ideally placed a reasonable distance from the sensor and all motors or other sources of magnetic fields should be powered down during tracking.

Calibration

Best results are achieved when the sensor is calibrated within the device being tracked. Currently the best method of calibrating the sensor embedded within a device is with standard IMU calibration methods available within the 3-Space Sensor suite application. Please refer to sphere or gradient decent calibration methods found in the calibration menu. Every sensor's barometer may potentially require calibration to achieve accurate altitude outputs. The barometer can be calibrated using command 52(0x34), sub-command 13, which sets the altitude offset parameter. Alternatively, command 52(0x34), sub-command 14 can be used to calibrate the barometer by providing a known altitude.

Pedestrian Tracking Example Commands:

Like all other 3-Space commands, the pedestrian tracking commands can be sent in either ASCII format, or in Binary format. For simple testing or debugging, ASCII commands can be more convenient. For production code that requires efficiency, the binary command mode is generally preferred.

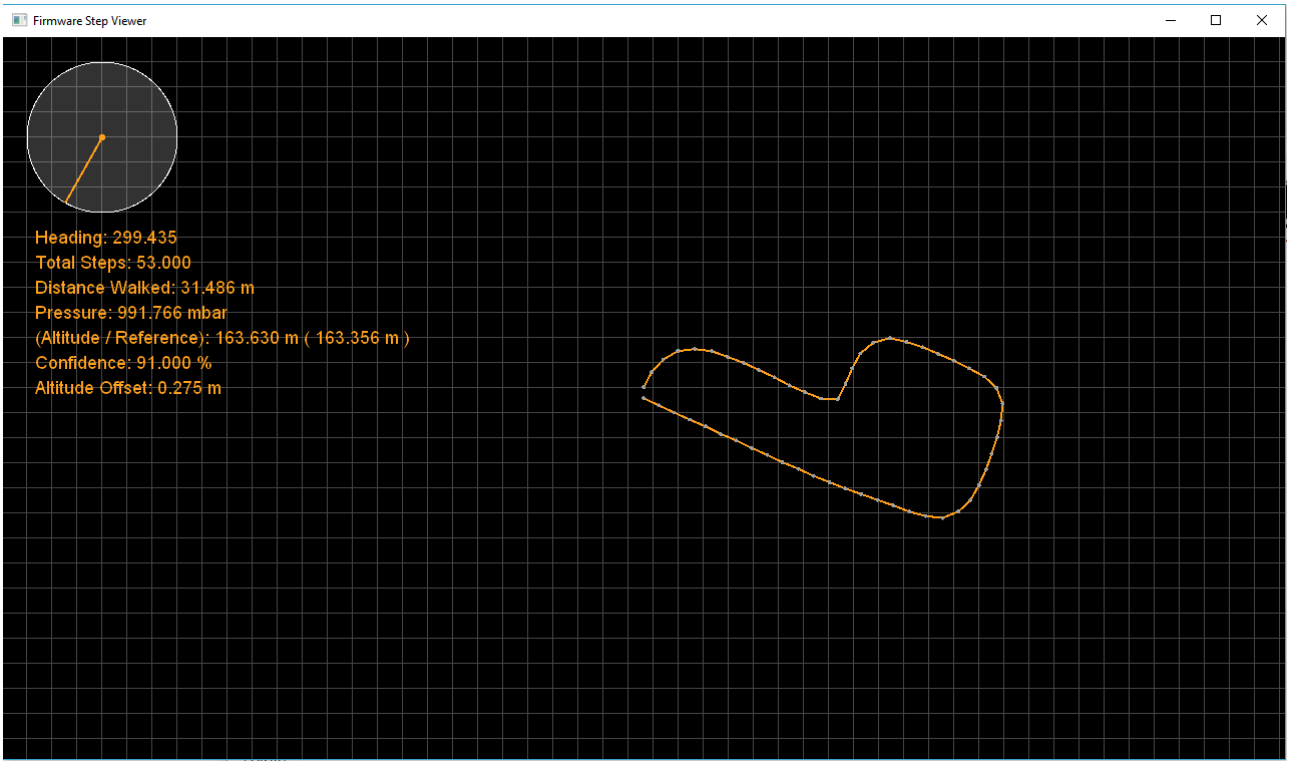
Below are some examples of pedestrian tracking commands as issued in ACSII mode.

```
> 52,15
// This will reset the step count and erase the recorded step data.

> 52,0,1
// This will start the recording of steps.

> 53,22
// This will get and return the latest step data.

> 52,0,0
// This will stop/pause the recording of steps.

> 52,1,4
// This will set the selected step index to the 5th step in the buffer.

> 53,23
```

```
// This will get and return the step at the selected index.

> 53,21
// This will return the confidence in distance recorded from the last
recording session.
```

Pedestrian Tracking Example Output:

The pedestrian tracking data can be visualized using the Yost Labs Visual Step Debugger application. A visualization of a captured step sequence representing a walk through a building and back to the starting position is shown below.



3.3 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 either a USB connection, an asynchronous serial UART connection, an SPI connection, or an I2C connection. 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 communicate through USB or serial port. To learn how to use this, read the 3-Space Suite manual.

3.4 Sensor Settings

3.4.1 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.

3.4.2 Natural Axes

All Yost Labs 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 LX are as follows:

- The positive X-axis points out of the side of the sensor with pins 7 through 12.
- The positive Y-axis points out of the top of the sensor (the component side of the board).
- The positive Z-axis points out of the back of the sensor (towards pins 1 and 12).

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.

Upon restoration of factory settings, the axis are returned to the default configuration.

The natural axes are illustrated in section 2.8.

3.4.3 Settings and Defaults

Setting Name	Purpose	Default Value
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)
Reference Mode	Determines how the accelerometer and compass reference vectors are determined	Single Auto
UART Baud Rate	Determines the speed of the Serial UART communication	115200
CPU Speed	Determines how fast the CPU will run	80MHz

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. Only binary commands are available when using SPI and I2C.

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 checksum 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

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 easiest way to find out which COM port belongs to a certain sensor is to take note of what COM port appears when that sensor is plugged in(provided the drivers have been installed on that computer already. Otherwise, find out what COM port appears once driver installation has finished.) For more information on how to install the sensor software on a computer and begin using it, see the Quick Start guide.

4.1.3 Electronic Interfacing Overview

The 3-Space Sensor LX module offers four interfacing /communications options: USB 2.0, Asynchronous Serial, Serial Peripheral Interface (SPI), and Inter-Integrated Circuit (I2C). One or more of the interfaces may be connected and used together. When using multiple interfaces, care should be taken to avoid the sending overlapping concurrent commands from multiple interfaces. Overlapping concurrent commands from multiple interfaces could result in a command being dropped. Thus, in situations where multiple overlapping concurrent commands cannot be avoided, a simple command verification, timeout, and retry paradigm should be used. The sections below describe the necessary pin connections and typical circuits used for using each of the respective interface options.

4.1.3.1 USB Interfacing

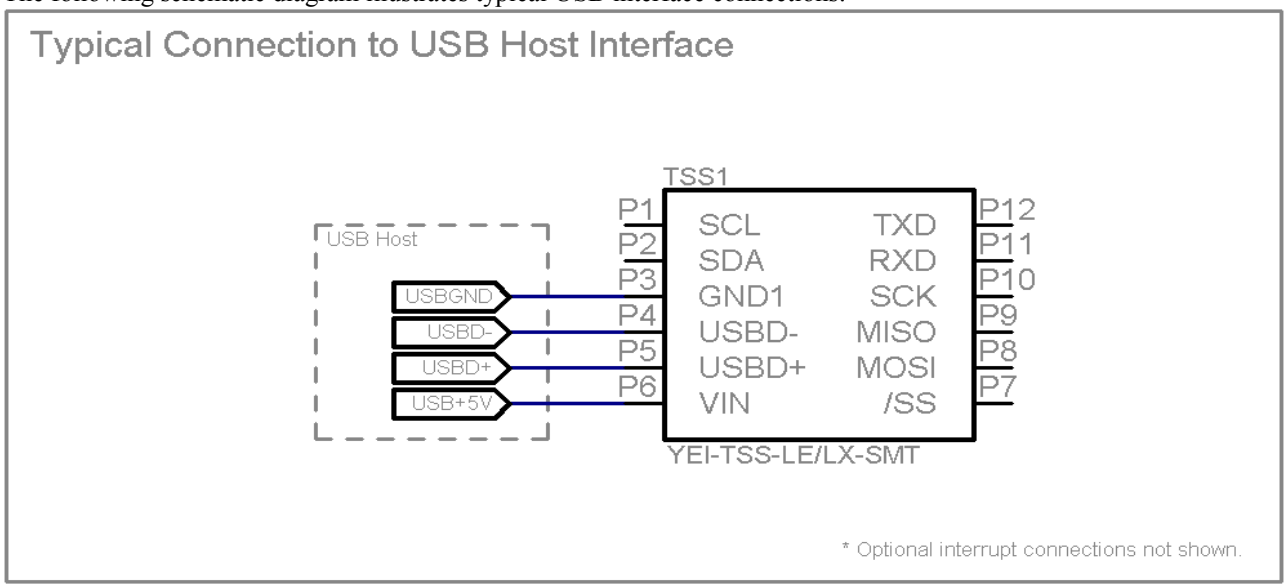
The USB 2.0 interface of the 3-Space Sensor LX requires the connection of signals as follows:

Pin	Signal	Description
3	GND1	USB Ground.
4	USBD-	USB Data Minus.
5	USBD+	USB Data Plus.
6	VIN	+5v USB Power Supply Input .

Additionally, one of the following optional interrupt pins may be configured for use during USB mode:

Pin	Signal	Description
2	SDA / INT	Configurable as filter update interrupt when I2C interface is unused.
9	MISO / INT	Configurable as filter update interrupt when SPI interface is unused.

The following schematic diagram illustrates typical USB interface connections:



4.1.3.2 Asynchronous Serial Interfacing

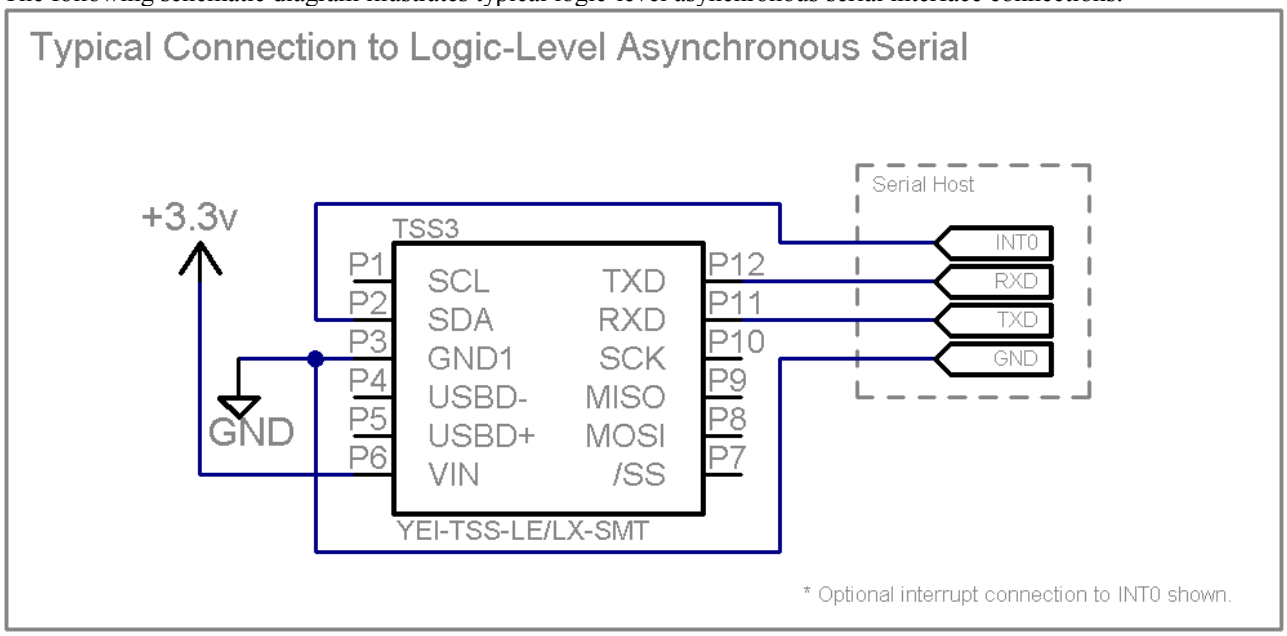
The asynchronous serial interface of the 3-Space Sensor LX requires the connection of signals as follows:

Pin	Signal	Description
3	GND1	Ground.
6	VIN	Voltage Input +3.3v ~ +6.0v.
11	RXD	UART Asynchronous Receive Data. Input to Module.
12	TXD	UART Asynchronous Transmit Data. Output from Module.

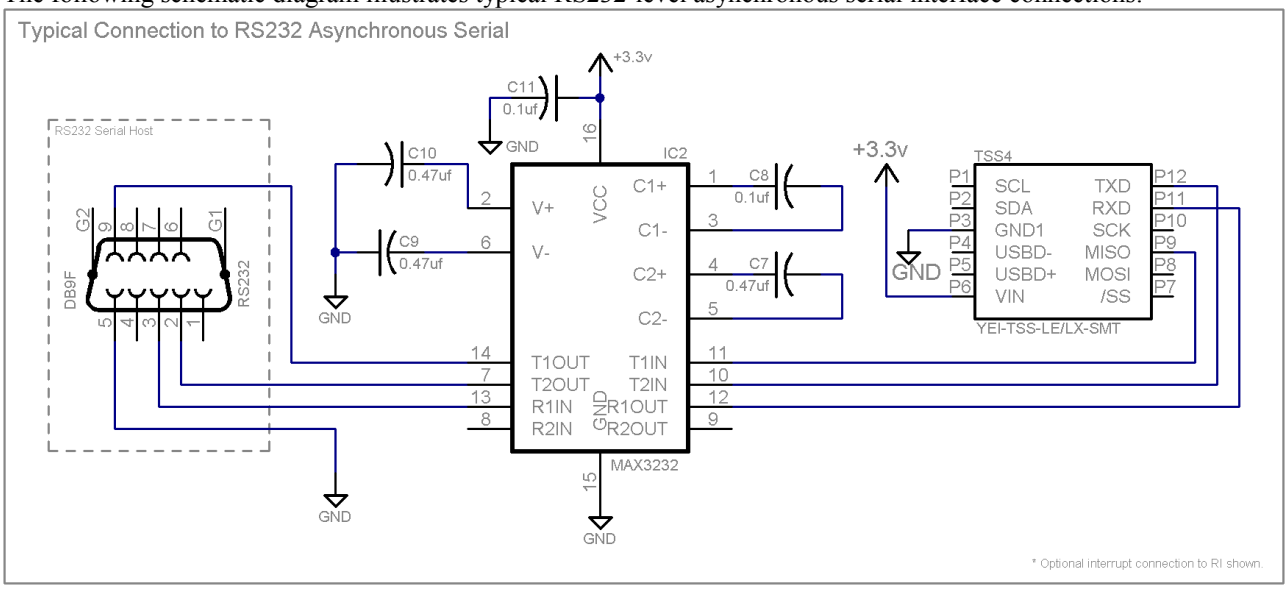
Additionally, one of the following optional interrupt pins may be configured for use during asynchronous serial mode:

Pin	Signal	Description
2	SDA / INT	Configurable as filter update interrupt when I2C interface is unused.
9	MISO / INT	Configurable as filter update interrupt when SPI interface is unused.

The following schematic diagram illustrates typical logic-level asynchronous serial interface connections:



The following schematic diagram illustrates typical RS232-level asynchronous serial interface connections:



4.1.3.2.1 Chain Configuration

Multiple 3-Space LX sensors can be chained together in order to communicate to a single serial host. This configuration requires the TXD lines of each sensor to be connected to the RXD of the serial host and the RXD lines of each sensor to be connected to the TXD of the serial host. Each sensor will require a logical ID in order to properly receive and send messages to and from the host device. The logical ID of a sensor can be set using command 209 (0xD1).

4.1.3.3 SPI Interfacing

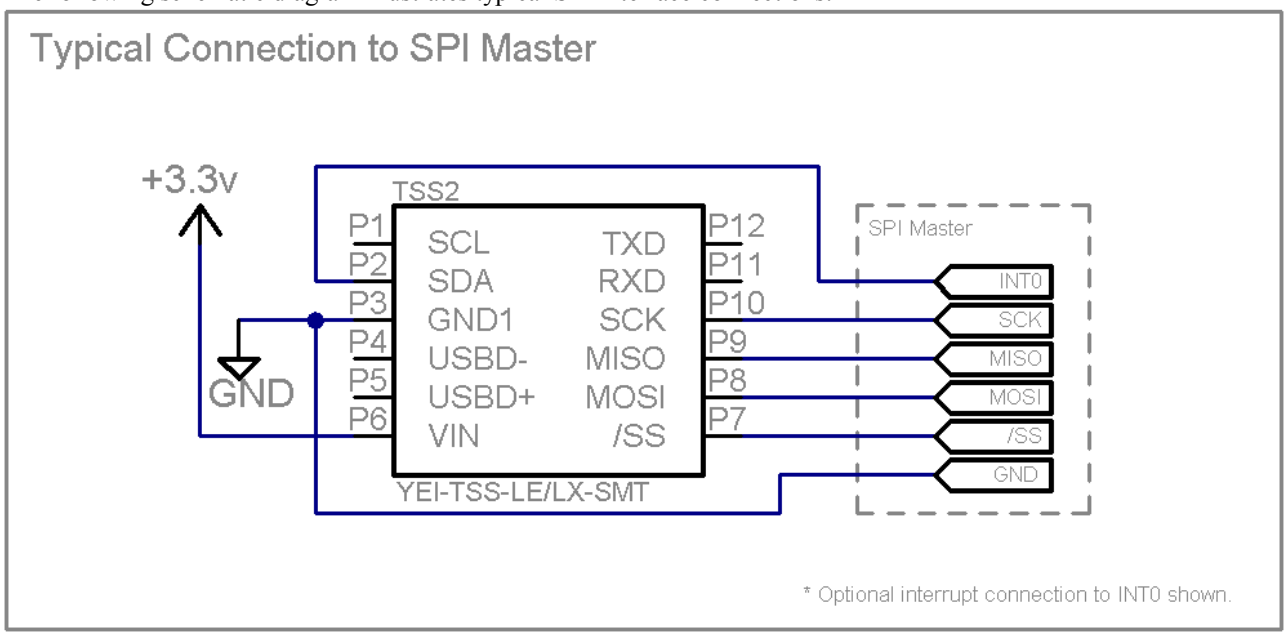
The Serial Peripheral Interface (SPI) of the 3-Space Sensor LX requires the connection of signals as follows:

Pin	Signal	Description
7	/SS	SPI Slave Select. Active Low Input to Module.
8	MOSI	SPI Master Out Slave In. Input to Module.
9	MISO	SPI Master In Slave Out. Output from Module.
10	SCK	SPI Serial Clock. Input to Module.

Additionally, one of the following optional interrupt pins may be configured for use during SPI mode:

Pin	Signal	Description
2	SDA / INT	Configurable as filter update interrupt when I2C interface is unused.

The following schematic diagram illustrates typical SPI interface connections:



4.1.3.4 I2C Interfacing

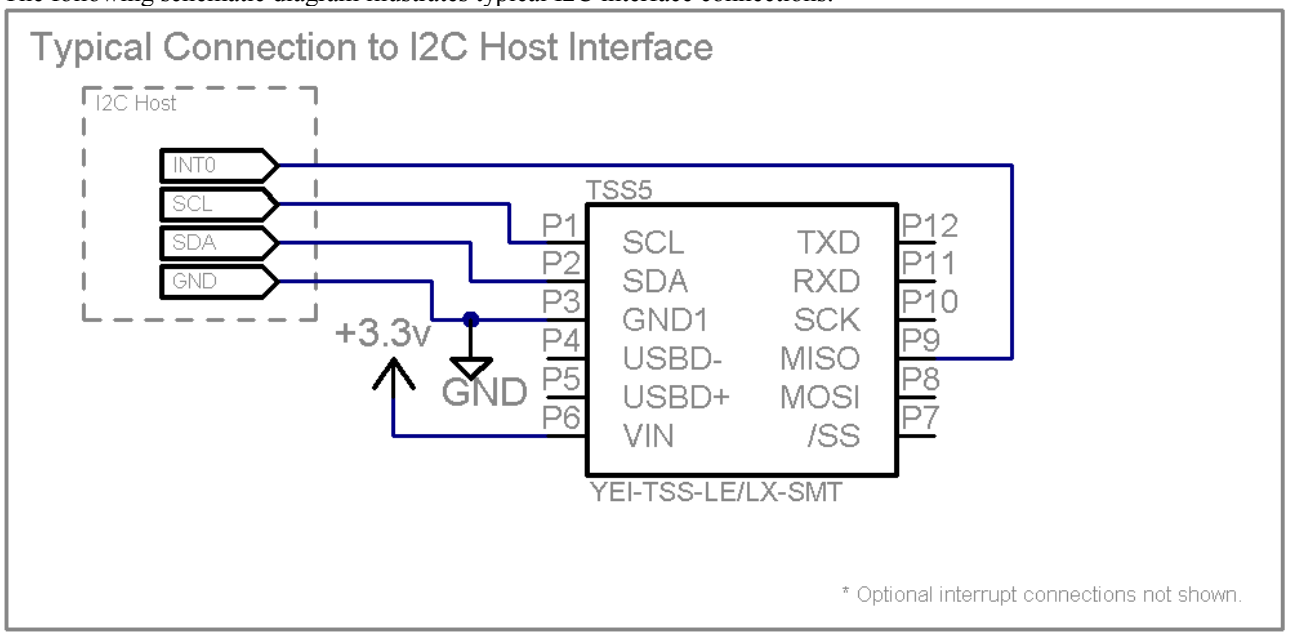
The Serial Peripheral Interface (SPI) of the 3-Space Sensor LX requires the connection of signals as follows:

Pin	Signal	Description
1	SCL	I2C Serial Clock. Input to Module.
2	SDA	I2C Data Line. Input to / Output from Module.
3	GND1	Ground.
6	VIN	Voltage Input +3.3v ~ +6.0v.

Additionally, one of the following optional interrupt pins may be configured for use during I2C mode:

Pin	Signal	Description
9	MISO / INT	Configurable as filter update interrupt when SPI interface is unused.

The following schematic diagram illustrates typical I2C interface connections:



4.1.3.5 Interrupt Generation

The LX 3-Space Sensor is capable of generating a signal on certain pins which can be used to trigger an interrupt when new orientation data becomes available. The pin will be high by default. The signal can be set to act in pulse mode, where the pin is set low for 5 microseconds and then pulled back to high, or it can be set to level mode, where the pin is set low until the interrupt status is read (see command 18). By default, no pin is set to act as the interrupt generation pin. Either the SPI MISO pin or the I2C SDA pin may be set to act as the interrupt pin, meaning that while interrupt generation is active, either the I2C or SPI will be unusable. For more information on setting the interrupt pin and mode, see command 16.

Pin	Signal	Description
2	SDA	Configurable as filter update interrupt when I2C interface is unused.
9	MISO	Configurable as filter update interrupt when SPI interface is unused.

4.2 Protocol Packet Format (USB and Serial)

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

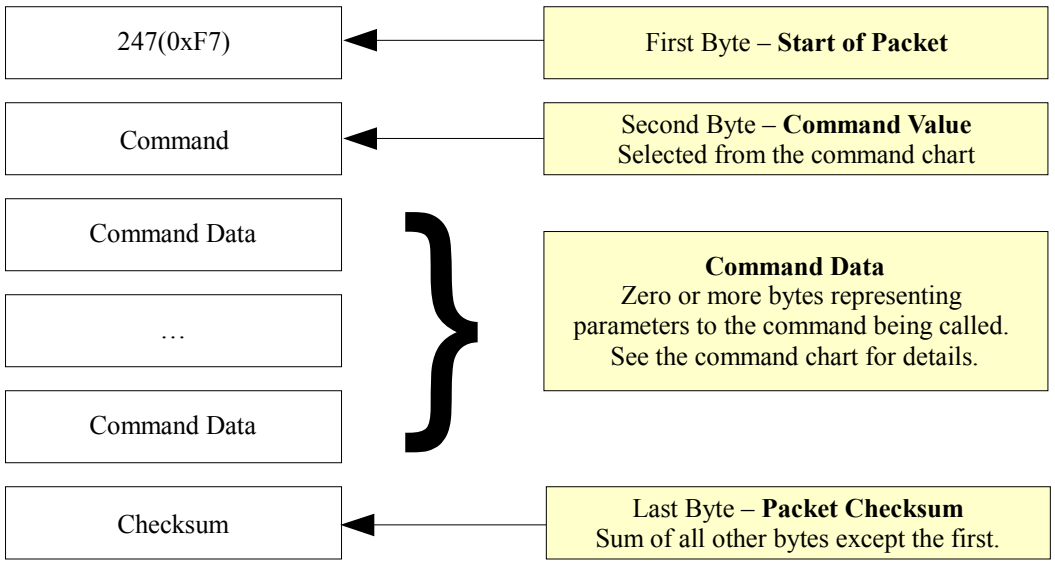


Figure 1 - Typical Binary Command Packet Format

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

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.

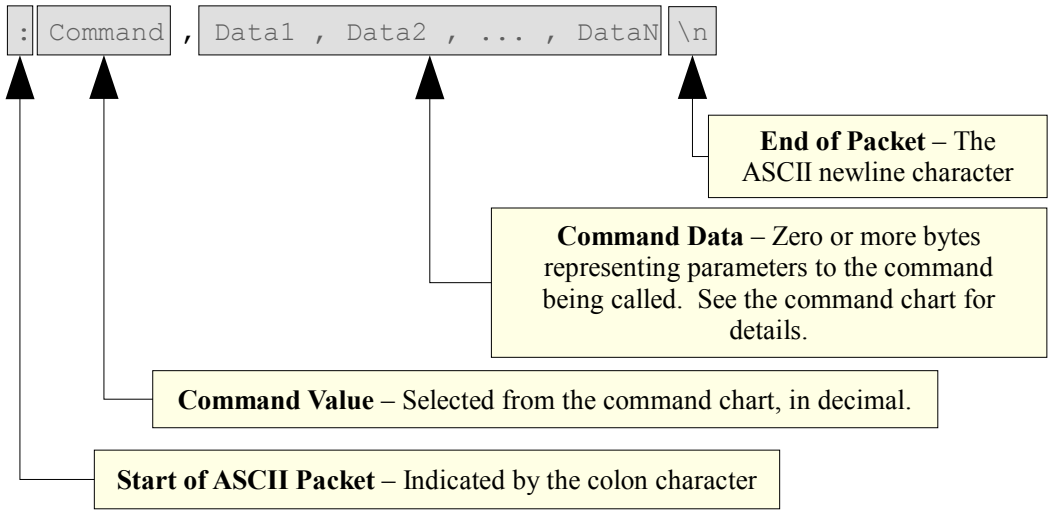


Figure 2 - Typical ASCII Command Packet Format

Thus the ASCII packet consists of the 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 Protocol Packet Format (Chain Configuration)

4.3.1 Binary Packet Format

The chain binary packet format is very similar to the wired format. Each packet consists of an initial **"address"** 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 chain binary packet is at least 4 bytes in length and is formatted as shown in figure 3.

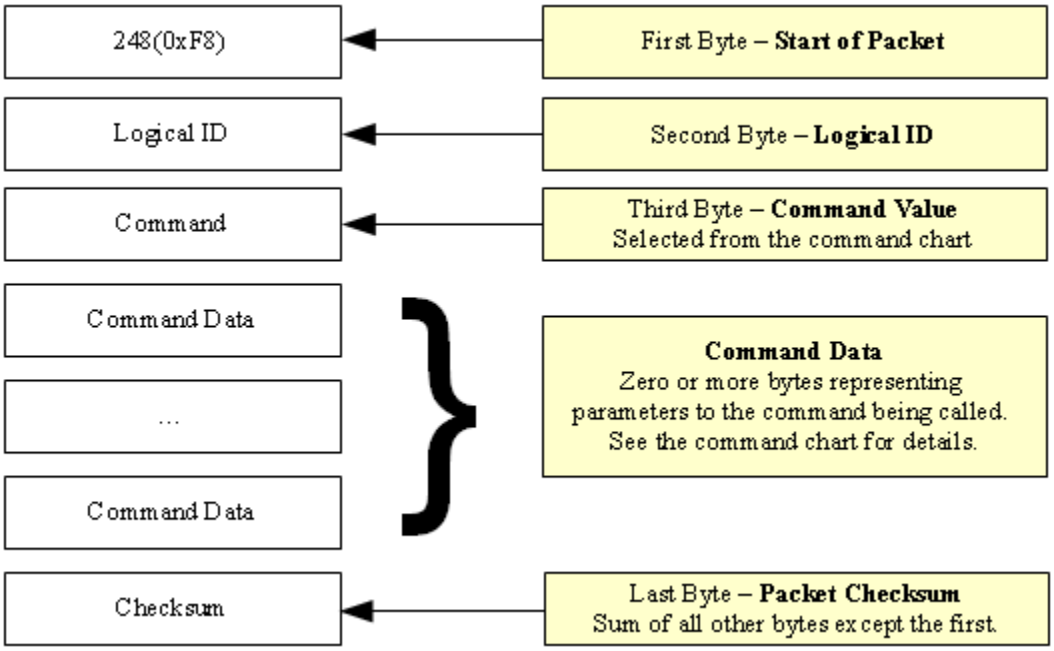


Figure 3 - Chain Binary Command Packet Format

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

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.

4.3.2 ASCII Text Packet Format

chain ASCII packets are very similar to regular ASCII packets. Each chain ASCII packet is formatted as shown in figure 4.

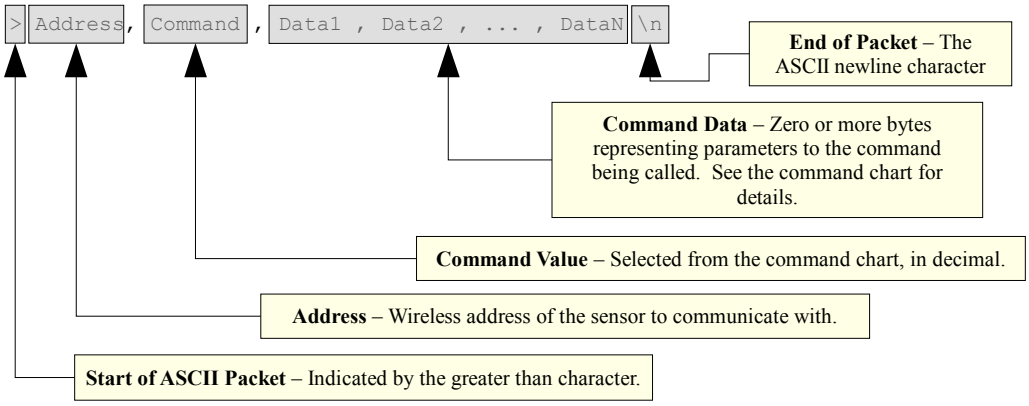


Figure 4 - Chain ASCII Command Packet Format

Thus the ASCII packet consists of the the following characters:

- **>** – the ASCII greater than 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:

`>1,0\n`	Read orientation as a quaternion from senor with ID 1
`>0,106,2\n`	Set oversample rate to 2 on sensor with ID 0

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.

4.4 Protocol Packet Format (SPI and I2C)

4.4.1 SPI Command Packet Format

In order to initiate an SPI data transfer, one of three command bytes must be sent in order to enter the proper protocol mode. The SPI commands supported by the TSS LX are as followed:

0xE9 (SPI Write Command) This command prepares the sensor to receive a TSS command and any additional arguments.

0x69 (SPI Read Data) This command prepares the sensor to send out data to the master device.

0x81 (SPI Read Status) This command prepares the sensor to send the status value of the slave device.

If the byte 0xE9 is sent then it should be followed by a TSS command byte and any additional arguments that may be required. If the command byte has a return value, such as command 0x0 or 0x6, then the return data will be stored into an internal buffer that can be read from by sending the byte 0x69 and then reading out all of the command data. The data buffer will constantly be updated based off of the command sent until a new command byte is received or if the sensors settings are reset using command 0xE0. If the sensor receives the byte 0x81, then the data buffer will be loaded with the 8 bit status value of the SPI device. This status will contain:

(Bit 0) – New data flag: Flag is set when there is new data to be read by the master device.

(Bit 1) – Update disabled flag: Flag is set when reading from SPI device to prevent the data buffer from updating while being read.

(Bit 2-7) – Command size: The return data size of the currently set command.

4.4.2 I2C Command Packet Format

I2C packet format is very similar to SPI. In order to initiate an I2C communication, the I2C address byte 0xEE must be sent first before sending one of three I2C commands:

0x42 (I2C Write Command) This command prepares the sensor to receive a TSS command and any additional arguments.

0x43 (I2C Read Data) This command prepares the sensor to send out data to the master device.

0x41 (I2C Read Status) This command prepares the sensor to send the status value of the slave device.

If the byte 0x42 is sent then it should be followed by a TSS command byte and any additional arguments that may be required. If the command byte has a return value, such as command 0x0 or 0x6, then the return data will be stored into an internal buffer that can be read from by sending the byte 0x43 and then reading out all of the command data. The data buffer will constantly be updated based off of the command sent until a new command byte is received or if the sensors settings are reset using command 0xE0. If the sensor receives the byte 0x41, then the data buffer will be loaded with the 8 bit status value of the I2C device.

4.5 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 for USB and serial packets. Thus, the total message size for USB and serial can be calculated by adding three bytes to the “Data Len” listed in the table. The total message size for SPI is Data Len plus the one Command byte.
- 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.

4.5.1 Commands for Reading Filtered Sensor Data

These commands return sensor data which has been filtered using the QGRAD2 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)
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)	Get untared orientation (two vector)	24	North Vector (float x3), Gravity Vector (float x3)
11(0x0B)	Get tared two vector in sensor frame	24	Forward Vector (float x3), Down Vector (float x3)
12(0x0C)	Get untared two vector in sensor frame	24	North Vector (float x3), Gravity Vector (float x3)

4.5.2 Commands for Interfacing with Electronic Systems

These commands are used to configure the LX to signal a data ready interrupt on either the SDA or MISO pin.

Command	Description	Return Data Len	Return Data Details	Data Len	Data Details
29(0x1D)	Set interrupt type	0		3	Mode(byte, 0 for off, 1 for pulse, 2 for level), pin(byte, 0 for SDA, 1 for MISO), polarity(byte, 0 for LOW, 1 for HIGH)
30(0x1E)	Read interrupt type	3	Mode(byte, 0 for off, 1 for pulse, 2 for level), pin(byte, 0 for SDA, 1 for MISO), polarity (byte, 0 for LOW, 1 for HIGH)	0	
31(0x1F)	Read interrupt status	1	Interrupt status(byte, 1 for new data since last orient read, 0 for no new data)	0	

4.5.3 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)
43(0x2B)	Read temperature (Celsius)	4	float
44(0x2C)	Read temperature (Fahrenheit)	4	float

4.5.4 Commands for Reading Corrected Sensor Data

These commands return sensor data which has been converted from a raw form to a form that represents a real world quantity. None of these commands take any parameters, they only return data.

Command	Description	Return Len	Return Data Details
37(0x25)	Read all (gyro, accelerometer, compass)	36	Vector(float x3), Vector(float x3), Vector(float x3)
38(0x26)	Read gyros	12	Vector(float x3)
39(0x27)	Read accelerometer	12	Vector(float x3)
40(0x28)	Read compass	12	Vector(float x3)
41(0x29)	Read linear acceleration	12	Vector(float x3)
48(0x30)	Correct raw gyro data	12	Gyro Rate in units of radians/sec (float x3)
49(0x31)	Correct raw accel data	12	Acceleration Vector in units of G (float x3)
50(0x32)	Correct raw compass data	12	Compass Vector in units of gauss (float x3)

4.5.5 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
64(0x40)	Read all raw (gyro, accelerometer, compass)	36	Vector(float x3), Vector(float x3), Vector(float x3)
65(0x41)	Read gyro raw	12	Vector(float x3)
66(0x42)	Read accelerometer raw	12	Vector(float x3)
67(0x43)	Read compass raw	12	Vector(float x3)

4.5.6 Commands for Streaming Data

These commands allow for the configuration of parameters associated with the streaming of data asynchronously from the sensor.

Command	Description	Return Len	Return Data Details	Data Len	Data Details
80(0x50)	Set streaming slots	0		8	Commands (Bytes x 8)
81(0x51)	Get streaming slots	8	Commands (Bytes x 8)	0	
82(0x52)	Set streaming timing	0		12	Interval, duration, delay (Unsigned int x 3)
83(0x53)	Get streaming timing	12	Interval, duration, delay (Unsigned int x 3)	0	
84(0x54)	Get streaming batch	Varies		0	
85(0x55)	Start streaming	0		0	
86(0x56)	Stop streaming	0		0	
95(0x5f)	Update current timestamp	0		4	Timestamp (Unsigned int)

4.5.7 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
16(0x10)	Set euler angle decomposition order	1	Euler angle decomposition order (byte)
19(0x13)	Offset with current Orientation	0	
20(0x14)	Reset base offset	0	
21(0x15)	Offset with quaternion	16	Quaternion (float x4)
22(0x16)	Set base offset with current orientation	0	
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)
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)	Set filter parameters	12	Kp, Max Smooth Factor, Min Smooth Factor (Float x 3)
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)
121(0x79)	Set accelerometer range	1	Accel range(byte, 0 for 2G, 1 for 4G, 2 for 8G, 3 for 16G)
125(0x7d)	Set gyroscope range	1	Gyro range mode(byte, 0 for 125 dps, 1 for 245 dps, 2 for 500 dps, 3 for 1000 dps, 4 for 2000 dps)
126(0x7e)	Set compass range	1	Compass range mode(byte, see command details for options)

4.5.8 Commands for Reading Filter Parameters

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

Command	Description	Return Len	Return Data Details
128(0x80)	Read tare orientation(Quaternion)	16	Quaternion(float x4)
129(0x81)	Read tare orientation(Rotation Matrix)	36	Rotation Matrix(float x9)
132(0x84)	Read current update rate	4	Update rate in microseconds(int)
133(0x85)	Read compass reference vector	12	Vector(float x3)
134(0x86)	Read accelerometer reference vector	12	Vector(float x3)
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)
140(0x8c)	Read gyro enabled state	1	Mode(byte, 0 for disabled, 1 for enabled)
141(0x8d)	Read accelerometer enabled state	1	Mode(byte, 0 for disabled, 1 for enabled)
142(0x8e)	Read compass enabled state	1	Mode(byte, 0 for disabled, 1 for enabled)
143(0x8f)	Read axis directions	1	Axis direction byte
144(0x90)	Read oversample rate	1	Rate(1 for off, 2+ for number of samples per frame)
145(0x91)	Read running average percent	4	Percent(float)
148(0x94)	Read accelerometer range	1	Accel range(byte, 0 for 2G, 0x10 for 4G, 0x30 for 8G)
154(0x9a)	Read gyroscope range	1	Gyro range mode(byte, 0 for 250 dps, 1 for 500 dps, 2 for 2000 dps)
155(0x9b)	Read compass range	1	Gyro range mode(byte, see command details for options)
156(0x9c)	Get euler angle decomposition order	1	Euler angle decomposition order (byte)
159(0x9f)	Get offset orientation as Quaternion	16	Quaternion (float x4)

4.5.9 Commands for Calibration

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

Command	Description	Return Data 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)
171(0xab)	Set auto calibration mode	0		1	Enable mode (Byte)
172(0xac)	Get auto calibration mode	1	Enable mode (Byte)	0	
173(0xad)	Set auto calibration factors	0		14	Too Close Angle (float), Bias Movement Percentage (float), Coplanarity Tolerance (float), Max Averages (byte), Max Bad Deviation (byte)
174(0xae)	Get auto calibration factors	14	Too Close Angle (float), Bias Movement Percentage (float), Coplanarity Tolerance (float), Max Averages (byte), Max Bad Deviation (byte)	0	
175(0xaf)	Get auto calibration counts	1	Counter (Byte)	0	

4.5.10 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 Len	Return Data Details	Data Len	Data Details
221(0xdd)	Set response header	0		4	Response header configuration bitfield (unsigned int)
222(0xde)	Get response header	4	Response header configuration bitfield (unsigned int)	0	
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)	Set sleep mode	0		1	Sleep mode (byte)
228(0xe4)	Get sleep mode	1	Sleep mode (byte)	0	
229(0xe5)	Enter firmware update mode	0		0	
230(0xe6)	Get version	12	Version number(string)	0	
231(0xe7)	Set UART baud rate	0		4	Baud rate (int, one of:1200,2400,4800, 9600,19200,28800,38400,57600, 115200, 230400, 460800, 921600)
232(0xe8)	Get UART baud rate	4	Baud rate (int, one of:1200,2400,4800, 9600,19200,28800,38400,57600, 115200, 230400, 460800, 921600)	0	
237(0xed)	Get serial number	4	Serial number (int)	0	
244(0xf4)	Set protocol timeout	0		4	Protocol Timeout (float)
245(0xf5)	Get protocol timeout	4	Protocol Timeout (float)	0	

4.5.11 Commands for Chain Communication

These commands are for the configuration of a senor placed in a chain of TSS-LX sensors.

Command	Description	Return Len	Return Data Details	Data Len	Data Details
208(0xd0)	Get logical ID	1	Logical ID (byte)	0	
209(0xd1)	Set logical ID	0		5	Logical ID (byte), serial number (unsigned int)
210(0xd2)	Set chain streaming settings	0		3	Streaming command (byte), number of packets (byte), sensors in chain (byte)
211(0xd3)	Set chain streaming delay	0		4	Delay (unsigned int)
212(0xd4)	Start chain streaming	0		1	Mode (byte)
213(0xd5)	Get chain streaming parameters	7	Streaming command (byte), number of packets (byte), sensors in chain (byte), delay (unsigned int)	0	

4.5.12 Commands for Pedestrian Tracking

These commands are for accessing the pedestrian tracking features of the sensor. The basic commands allow the setting and getting of data and configuration parameters related to pedestrian tracking. Individual pedestrian tracking related commands and parameters are implemented as sub-commands. See the pedestrian tracking command and sub-command details section for complete documentation.

Command	Description	Return Len	Return Data Details	Data Len	Data Details
52 (0x34)	Set command for pedestrian tracking	0		5	Sub-command(byte), Parameter(float)
53 (0x35)	Get command for pedestrian tracking	varies	Sub-command 0-12 return 1 flloat	1	Sub-command(byte)

4.5.12.1 Pedestrian Tracking Sub Commands

Command	Sub-command	Description	Return Len	Return Data Details	Data Len	Data Details
52 (0x34)	0 (0x00)	Set Pedestrian Tracking On/Off	0		4	On_off(float): 0=off, 1=on
52 (0x34)	1 (0x01)	Set Selected Step Index	0		4	Index(float)
52 (0x34)	2 (0x02)	Set Rmin Parameter	0		4	Rmin(float)
52 (0x34)	3 (0x03)	Set Rmax Parameter	0		4	Rmax(float)
52 (0x34)	4 (0x04)	Set DMin Parameter	0		4	Dmin(float)
52 (0x34)	5 (0x05)	Set DMax Parameter	0		4	Dmax(float)
52 (0x34)	6 (0x06)	Set AMin Parameter	0		4	Amin(float)
52 (0x34)	7 (0x07)	Set AMax Parameter	0		4	Amax(float)
52 (0x34)	8 (0x08)	Set Step_Duration_Min Parameter	0		4	StepDurMin(float)
52 (0x34)	9 (0x09)	Set Step_Duration_Max Parameter	0		4	StepDurMax(float)
52 (0x34)	10 (0x0a)	Set Step_Stride_Slope Parameter	0		4	StepSlope(float)
52 (0x34)	11 (0x0b)	Set Step_Stride_Offset Parameter	0		4	StepOffset(float)
52 (0x34)	12 (0x0c)	Set Units	0		4	Units(float): 0=meters, 1=feet
52 (0x34)	15 (0x0f)	Reset Step Count And Clear Steps	0		4	Dummy_byte(float)
53 (0x35)	0 (0x00)	Get Pedestrian Tracking On/Off	4	On_off(float): 0=off, 1=on		
53 (0x35)	1 (0x01)	Get Selected Step Index	4	Index(float)		
53 (0x35)	2 (0x02)	Get Rmin Parameter	4	Rmin(float)		
53 (0x35)	3 (0x03)	Get Rmax Parameter	4	Rmax(float)		
53 (0x35)	4 (0x04)	Get DMin Parameter	4	Dmin(float)		
53 (0x35)	5 (0x05)	Get DMax Parameter	4	Dmax(float)		
53 (0x35)	6 (0x06)	Get AMin Parameter	4	Amin(float)		
53 (0x35)	7 (0x07)	Get AMax Parameter	4	Amax(float)		
53 (0x35)	8 (0x08)	Get Step_Duration_Min Parameter	4	StepDurMin(float)		

Command	Sub-command	Description	Return Len	Return Data Details	Data Len	Data Details
53 (0x35)	9 (0x09)	Get Step_Duration_Max Parameter	4	StepDurMax(float)		
53 (0x35)	10 (0x0a)	Get Step_Stride_Slope Parameter	4	StepSlope(float)		
53 (0x35)	11 (0x0b)	Get Step_Stride_Offset Parameter	4	StepOffset(float)		
53 (0x35)	12 (0x0c)	Get Units	4	Units(float): 0=meters, 1=feet		
53 (0x36)	18 (0x12)	Get Heading in Degrees	4	Heading_degrees(float)		
53 (0x35)	19 (0x13)	Get Step Buffer Length	4	NumStepsRecorded(float)		
53 (0x35)	20 (0x14)	Get Total Distance Traveled	4	TotalDistanceTraveled(float)		
53 (0x35)	21 (0x15)	Get Confidence Value	4	ConfidenceValue(float)		
53 (0x35)	22 (0x16)	Get Latest Step	48	StepDataVector(floatx12): Index, Heading, X_step_distance, Y_step_distance, Step_distance, Step_time, Step_amplitude, Step_duty_cycle, Step_ratio, Altitude, Confidence, Altitude_Offset		
53 (0x35)	23 (0x17)	Get Step From Current Index	48	StepDataVector(floatx12): Index, Heading, X_step_distance, Y_step_distance, Step_distance, Step_time, Step_amplitude, Step_duty_cycle, Step_ratio, Altitude, Confidence, Altitude_Offset		

4.6 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.

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

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

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

Function:	Read filtered, tared orientation(Euler Angles)
Command Value:	3 (0x03)
Return Data Bytes:	16
Return Data Format:	Axis(float x3), Angle(float)
Description:	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.

Function:	Read filtered, tared orientation(Two Vector(Forward, Down))
Command Value:	4 (0x04)
Return Data Bytes:	24
Return Data Format:	Vector(float x3), Vector(float x3)
Description:	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.

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

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

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

Function:	Read filtered, untared orientation(Euler Angles)
Command Value:	9 (0x09)
Return Data Bytes:	16
Return Data Format:	Axis(float x3), Angle(float)
Description:	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.

Function:	Get untared orientation as two vector
Command Value:	10 (0x0A)
Return Data Bytes:	24
Return Data Format:	North Vector (float x3), Gravity Vector (float x3)
Description:	Returns the filtered, untared orientation estimate in two vector form, where the first vector refers to north and the second refers to gravity.

Function:	Get tared two vector in sensor frame
Command Value:	11 (0x0B)
Return Data Bytes:	24
Return Data Format:	Forward Vector (float x3), Down Vector (float x3)
Description:	Returns the filtered, tared orientation estimate in two vector form, where the first vector refers to forward and the second refers to down. These vectors are given in the sensor reference frame and not the global reference frame.

Function:	Get untared two vector in sensor frame
Command Value:	12 (0x0C)
Return Data Bytes:	24
Return Data Format:	North Vector (float x3), Gravity Vector (float x3)
Description:	Returns the filtered, untared orientation estimate in two vector form, where the first vector refers to north and the second refers to gravity. These vectors are given in the sensor reference frame and not the global reference frame.

Function:	Set euler angle decomposition order
Command Value:	16 (0x10)
Data Bytes:	1
Data Format:	Euler angle decomposition order (byte)
Description:	Sets the current euler angle decomposition order, which determines how the angles returned from command 0x1 are decomposed from the full quaternion orientation. Possible values are 0x0 for XYZ, 0x1 for YZX, 0x2 for ZXY, 0x3 for ZYX, 0x4 for XZY or 0x5 for YXZ (default).

Function:	Offset with current Orientation
Command Value:	19 (0x13)
Description:	Sets the offset orientation to be the same as the current filtered orientation.

Function:	Reset base offset
Command Value:	20 (0x14)
Description:	Sets the base offset to an identity quaternion.

Function:	Offset with quaternion
Command Value:	21 (0x15)
Return Data Bytes:	16
Return Data Format:	Quaternion (float x4)
Description:	Sets the offset orientation to be the same as the supplied orientation, which should be passed as a Quaternion.

Function:	Set base offset with current orientation
Command Value:	22 (0x16)
Description:	Sets the base offset orientation to be the same as the current filtered orientation.

Function:	Set interrupt type
Command Value:	29 (0x1D)
Data Bytes:	3
Data Format:	Mode(byte, 0 for off, 1 for pulse, 2 for level), pin(byte, 0 for SDA, 1 for MISO), polarity(byte, 0 for LOW, 1 for HIGH)
Description:	Sets up interrupt generation. If mode is 0, no generation will occur. If mode is 1, the interrupt generator will set the specified pin low for 5 microseconds when new data is available. If mode is 2, the interrupt generator will set the pin low until the interrupt status is read with command 18. The pin byte specifies which pin the interrupt will be generated on. The method of communication each pin belongs to cannot be used while that pin is being used for interrupt generation. Note that the pin cannot be changed to SDA over I2C, and cannot be changed to MISO over SPI. If polarity is 0, then interrupts are generated by the interrupt pin being pulled low, if 1, then the pin will be pulled high.

Function:	Read interrupt type
Command Value:	30 (0x1E)
Return Data Bytes:	3
Return Data Format:	Mode(byte, 0 for off, 1 for pulse, 2 for level), pin(byte, 0 for SDA, 1 for MISO), polarity(byte, 0 for LOW, 1 for HIGH)
Description:	Read the current interrupt mode, pin, and polarity.

Function:	Read interrupt status
Command Value:	31 (0x1F)
Return Data Bytes:	1
Return Data Format:	Interrupt status(byte, 1 for new data since last orient read, 0 for no new data)
Description:	Read the current interrupt state. If there is new data since the last orientation read(commands 0-4 and 6-10), the value will be a 1, otherwise it will be 0. Calling this command while interrupts are in level mode will cause the interrupt pin to return to high.

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

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

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

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

Function:	Read all corrected data (gyro, accelerometer, compass)
Command Value:	37 (0x25)
Return Data Bytes:	36
Return Data Format:	Vector(float x3), Vector(float x3), Vector(float x3)
Description:	Returns the corrected gyro, accelerometer, and compass values.

Function:	Read gyros corrected
Command Value:	38 (0x26)
Return Data Bytes:	12
Return Data Format:	Vector(float x3)
Description:	Returns the corrected gyro rates.

Function:	Read accelerometers corrected
Command Value:	39 (0x27)
Return Data Bytes:	12
Return Data Format:	Vector(float x3)
Description:	Returns the corrected gravity vector.

Function:	Read compass corrected
Command Value:	40 (0x28)
Return Data Bytes:	12
Return Data Format:	Vector(float x3)
Description:	Returns the corrected north vector.

Function:	Read linear acceleration
Command Value:	41 (0x29)
Return Data Bytes:	12
Return Data Format:	Vector(float x3)
Description:	Returns the linear acceleration of the device. Uses the untared orientation.

Function:	Read temperature C
Command Value:	43 (0x2B)
Return Data Bytes:	4
Return Data Format:	float
Description:	Returns the temperature of the sensor in Celsius

Function:	Read temperature F
Command Value:	44 (0x2C)
Return Data Bytes:	4
Return Data Format:	float
Description:	Returns the temperature of the sensor in Fahrenheit

Function:	Correct raw gyro data
Command Value:	48 (0x30)
Return Data Bytes:	12
Return Data Format:	Gyro Rate in units of radians/sec (float x3)
Description:	Converts the supplied raw data gyroscope vector to its corrected data representation.

Function:	Correct raw accel data
Command Value:	49 (0x31)
Return Data Bytes:	12
Return Data Format:	Acceleration Vector in units of G (float x3)
Description:	Converts the supplied raw data accelerometer vector to its corrected data representation.

Function:	Correct raw compass data
Command Value:	50 (0x32)
Return Data Bytes:	12
Return Data Format:	Compass Vector in units of gauss (float x3)
Description:	Converts the supplied raw data compass vector to its corrected data representation.

Function:	Set command for pedestrian tracking
Command Value:	52 (0x34)
Data Bytes:	5
Data Format:	Sub-command(byte), Value(float)
Description:	This command allows the setting of parameters related to pedestrian tracking. Sub-command is a single-byte that is used to indicate the parameter that is to be set, Value is a float indicating the value of the parameter to be set. See the pedestrian tracking and pedestrian tracking commands section for command details.

Function:	Get command for pedestrian tracking
Command Value:	53 (0x35)
Data Bytes	1
Data Format	Sub-command(byte)
Return Data Bytes:	varies
Return Data Format:	Sub-commands 0-21 return a single float for the requested data value, Sub-commands 22-23return 12 floats representing the individual data elements of the step data.
Description:	This command allows the getting of parameters related to pedestrian tracking. The sub-command parameter is a byte indicating the value that is to be read. Sub-commands 22 and 23 return a block of 12 floats that represents the step data for a single step. See the pedestrian tracking and pedestrian tracking commands section for command details.

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

Function:	Read gyro raw
Command Value:	65 (0x41)
Return Data Bytes:	12
Return Data Format:	Vector(float x3)
Description:	Returns the raw gyro values in the specified measurement range.

Function:	Read accelerometer raw
Command Value:	66 (0x42)
Return Data Bytes:	12
Return Data Format:	Vector(float x3)
Description:	Returns the raw accelerometer values in the specified measurement range.

Function:	Read compass raw
Command Value:	67 (0x43)
Return Data Bytes:	12
Return Data Format:	Vector(float x3)
Description:	Returns the raw compass values in the specified measurement range.

Function:	Set streaming slots
Command Value:	80 (0x50)
Data Bytes:	8
Data Format:	Commands (Bytes x 8)
Description:	Configures data output slots for streaming mode.

Function:	Get streaming slots
Command Value:	81 (0x51)
Return Data Bytes:	8
Return Data Format:	Commands (Bytes x 8)
Description:	Reads data output slots for streaming mode.

Function:	Set streaming timing
Command Value:	82 (0x52)
Data Bytes:	12
Data Format:	Interval (Unsigned int), Duration (Unsigned int), Delay (Unsigned int)
Description:	Configures timing information for a streaming session. All parameters are specified in microseconds.

Function:	Get streaming timing
Command Value:	83 (0x53)
Return Data Bytes:	12
Return Data Format:	Interval (Unsigned int), Duration (Unsigned int), Delay (Unsigned int)
Description:	Returns the current streaming timing information.

Function:	Get streaming batch
Command Value:	84 (0x54)
Data:	Varies
Description:	Return a single packet of streaming data using the current slot configuration.

Function:	Start streaming
Command Value:	85 (0x55)
Description:	Start a streaming session using the current slot and Timing configuration.

Function:	Stop streaming
Command Value:	86 (0x56)
Description:	Stop the current streaming session.

Function:	Update current timestamp
Command Value:	95 (0x5F)
Data Bytes:	4
Data Format:	Timestamp (Unsigned int)
Description:	Set the current internal timestamp to the specified value.

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

Function:	Tare with quaternion
Command Value:	97 (0x61)
Data Bytes:	16
Data Format:	Quaternion (float x4)
Description:	Sets the tare orientation to be the same as the supplied orientation, which should be passed as a Quaternion.

Function:	Tare with rotation matrix
Command Value:	98 (0x62)
Data Bytes:	36
Data Format:	Rotation Matrix (float x9)
Description:	Sets the tare orientation to be the same as the supplied orientation, which should be passed as a rotation matrix.

Function:	Set reference vector mode
Command Value:	105 (0x69)
Data Bytes:	1
Data Format:	Mode(byte, 0 for single static, 1 for single auto, 2 for single auto continuous, 3 for multi)
Description:	Sets reference vector mode. -Single static mode uses a certain reference vector for the compass and another certain reference vector for the accelerometer at all times. -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. -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. -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.

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

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

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

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

Function:	Get filter parameters
Command Value:	111 (0x6f)
Data Bytes:	12
Data Format:	Kp, Max Smooth Factor, Min Smooth Factor (Float x 3)
Description:	Read out adjustable filter parameters

Function:	Set axis directions
Command Value:	116 (0x74)
Data Bytes:	1
Data Format:	Axis direction byte
Description:	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 *after* 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)

Function:	Set running average percent
Command Value:	117 (0x75)
Data Bytes:	4
Data Format:	Percent(float)
Description:	Sets what percentage of running average to use on the sensor's orientation. This is computed as follows: total_orient = total_orient * percent total_orient = total_orient + current_orient * (1 – percent) current_orient = total_orient If the percentage is 0, the running average will be shut off completely.

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

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

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

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

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

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

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

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

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

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

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

Function:	Get gyroscope enabled state
Command Value:	140 (0x8c)
Return Data Bytes:	1
Return Data Format:	Mode(byte, 0 for disabled, 1 for enabled)
Description:	Reads the enabled state of the gyros.

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

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

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

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

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

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

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

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

Function:	Get euler angle decomposition order
Command Value:	156 (0x9c)
Return Data Bytes:	1
Return Data Format:	Euler angle decomposition order (byte)
Description:	Reads the current euler angle decomposition order.

Function:	Get offset orientation as Quaternion
Command Value:	159 (0x9f)
Return Data Bytes:	16
Return Data Format:	Quaternion (float x4)
Description:	Returns the current offset orientation as a Quaternion.

Function:	Set compass calibration parameters
Command Value:	160 (0xA0)
Data Bytes:	48
Data Format:	Bias(float x3), Matrix(float x9)
Description:	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.

Function:	Set accelerometer calibration parameters
Command Value:	161 (0xA1)
Data Bytes:	48
Data Format:	Bias(float x3), Matrix(float x9)
Description:	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.

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

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

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

Function:	Begin gyro auto-calibration
Command Value:	165 (0xA5)
Description:	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.

Function:	Set accelerometer calibration parameters
Command Value:	166 (0xA6)
Data Bytes:	24
Data Format:	Bias(float x3), High range bias(float x3)
Description:	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.

Function:	Set auto calibration mode
Command Value:	171 (0xAB)
Data Bytes:	1
Data Format:	Enable mode (Byte)
Description:	Sets the operating mode for auto calibration. Mode 0 will Disable auto calibration, 1 enables Single Mode that does the calibration once at sensor start up and when the mode is set to single, 2 enables Auto Mode that does the calibration as needed and will not perform calibration if it evaluates the current calibration as OK, 3 enables Auto continuous that does the calibration steps non-stop, it will perform calibration regardless of the quality of the current calibration.

Function:	Get auto calibration mode
Command Value:	172 (0xAC)
Data Bytes:	1
Data Format:	Enable mode (Byte)
Description:	Read out operating mode for auto calibration.

Function:	Set auto calibration factors
Command Value:	173 (0xAD)
Data Bytes:	12
Data Format:	Too Close Angle (float), Bias Movement Percentage (float), Coplanarity Tolerance (float), Max Averages (byte), Max Bad Deviation (byte)
Description:	Sets auto calibration factors: -Too Close Angle: Compass samples are not taken if the angle between two samples is too close. Samples that are too close produce a bad calibration. -Bias Movement Percentage: Determines the allowable variance between new calibration biases and the current running average bias. Helps prevent bad calibrations. -Coplanarity Tolerance: Determines the threshold for when a set of samples are considered coplanar. Coplanar samples produce a bad calibration. -Max Averages: The max allowable number of calibrations for the running average of the bias. Higher = Slower Calibration But more accuracy. -Max Bad Deviations: The max allowable number of calibrations that are too variable before the running average bias is reset.

Function:	Get auto calibration factors
Command Value:	174 (0xAE)
Data Bytes:	14
Data Format:	Too Close Angle (float), Bias Movement Percentage (float), Coplanarity Tolerance (float), Max Averages (byte), Max Bad Deviation (byte)
Description:	Read out adjustable auto calibration factors.

Function:	Get auto calibration counts
Command Value:	175 (0xAF)
Data Bytes:	1
Data Format:	Counter (Byte)
Description:	Get number of calibrations gathered.

Function:	Start chain streaming
Command Value:	208 (0xD0)
Return Data Bytes:	1
Return Data Format:	Logical ID (byte)
Description:	Returns the local logical ID of the sensor.

Function:	Set logical ID
Command Value:	209 (0xD1)
Return Data Bytes:	5
Return Data Format:	Logical ID (byte), Serial Number (unsigned int)
Description:	Sets the local logical ID of the sensor to be used in a daisy chain configuration. Should be used to designate the sensors position in the chain.

Function:	Set chain streaming settings
Command Value:	210 (0xD2)
Return Data Bytes:	3
Return Data Format:	Stream Command (byte), Number of packets (byte), Sensors in chain (byte)
Description:	Sets the command to be streamed for the sensor, the number of packets to be output by the sensor, and the total number of sensors in the chain. If the nuber of packets is set to 0, then the sensor will stream until stopped by command 0x56.

Function:	Set chain streaming delay
Command Value:	211 (0xD3)
Return Data Bytes:	4
Return Data Format:	Delay (unsigned long int)
Description:	Sets the delay between output packets form each sensor.

Function:	Start chain streaming
Command Value:	212 (0xD4)
Return Data Bytes:	1
Return Data Format:	Mode (byte)
Description:	Starts streaming data from all sensors in the chain. Data is produced at intervals specified by command 0xD3. This command requires a Mode byte that acts either as a start command or a synchronization command: 1: Loads all streaming parameters and begins streaming 0: Continues to operate with most recent settings and Synchronizes the timers of all streaming devices

Function:	Get chain streaming parameters
Command Value:	213 (0xD5)
Return Data Bytes:	7
Return Data Format:	Stream Command (byte), Number of packets(byte), Sensors in chain (byte), Delay (unsigned long int)
Description:	Returns the set streaming command, the number of output packets, total sensor count in the chain, and the delay between each output packet.

Function:	Set wired response header
Command Value:	221(0xDD)
Data Bytes:	4
Data Format:	Response header configuration bitfield (unsigned int)
Description:	Configures the response header for data returned over a wired connection. The only parameter is a four-byte bitfield that determines which data is prepended to all data responses. The following bits are used: 0x1: (1 byte) Success/Failure, with non-zero values representing failure. 0x2: (4 bytes) Timestamp, in microseconds. 0x4: (1 byte) Command echo—outputs the called command. Returns 0xFF for streamed data. 0x8: (1 byte) Additive checksum over returned data, but not including response header. 0x10: (1 byte) Logical ID, returns 0xFE for wired sensors. Meant to be used with 3-Space Dongle response header (For more info, see command 0xDB). 0x20: (4 bytes) Serial number 0x40: (1 byte) Data length, returns the length of the requested data, not including response header. This setting can be committed to non-volatile flash memory by calling the Commit Settings command.

Function:	Get wired response header
Command Value:	222 (0xDE)
Return Data Bytes:	4
Return Data Format:	Response header configuration bitfield (unsigned int)
Description:	Return the current wired response header bitfield.

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

Function:	Restore factory settings
Command Value:	224 (0xE0)
Description:	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.

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

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

Function:	Set sleep mode
Command Value:	227 (0xE4)
Data Bytes:	1
Data Format:	Sleep mode (byte)
Description:	Sets the current sleep mode of the sensor. Supported sleep modes are 0 for NONE and 1 for IDLE. IDLE mode merely skips all filtering steps. NONE is the default state.

Function:	Get sleep mode
Command Value:	228 (0xE5)
Return Data Bytes:	1
Return Data Format:	Sleep mode (byte)
Description:	Reads the current sleep mode of the sensor, which can be 0 for NONE or 1 for IDLE.

Function:	Enter bootloader mode
Command Value:	229 (0xE5)
Description:	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.

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

Function:	Set UART baud rate
Command Value:	231 (0xE7)
Data Bytes:	4
Data Format:	Baud rate(int)
Description:	Sets the baud rate of the physical UART. This setting does not need to be committed, but does not take effect until the sensor is reset. The baud rate will be set to the valid value nearest the requested value. Valid baud rates are: 1200, 2400, 4800, 9600, 19200, 28800, 38400, 57600, 115200, 230400, 460800, 921600. The factory default baud rate is 115200.

Function:	Get UART baud rate
Command Value:	232 (0xE8)
Return Data Bytes:	4
Return Data Format:	Baud rate(int)
Description:	Reads the baud rate of the physical UART. Possible baud rates are: 1200, 2400, 4800, 9600, 19200, 28800, 38400, 57600, 115200, 230400, 460800, 921600. The factory default baud rate is 115200.

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

Function:	Set protocol timeout
Command Value:	244 (0xF4)
Data Bytes:	4
Data Format:	Protocol Timeout (float)
Description:	Sets a timeout value for the command processing protocol in microseconds.

Function:	Get protocol timeout
Command Value:	245 (0xF5)
Return Data Bytes:	4
Return Data Format:	Protocol Timeout (float)
Description:	Gets the timeout value for the command processing protocol in microseconds.

Appendix

Hex / Decimal Conversion Chart

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

Notes:

Serial Number: _____

Yost Labs
630 Second Street
Portsmouth, Ohio 45662

Phone: 740-876-4936

www.yostlabs.com