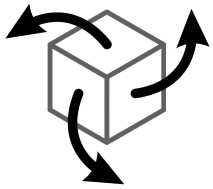
3-Space Sensor



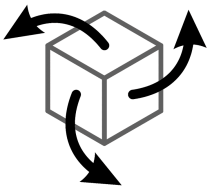
3-Space Sensor Embedded

Ultra-Miniature Attitude & Heading Reference System

User's Manual

YEI Technology
630 Second Street
Portsmouth, Ohio 45662

www.YeiTechnology.com
www.3SpaceSensor.com



3-Space Sensor Embedded

Ultra-Miniature Attitude & Heading Reference System

User's Manual

YEI Technology
630 Second Street
Portsmouth, Ohio 45662

www.YeiTechnology.com
www.3SpaceSensor.com

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

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 YEI 3-Space Embedded 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 Embedded module. Placing magnetic components, ferrous metal containing components, high-current conductors, and high-frequency digital signal lines should be avoided during PCB layout.

1.2 Technical Support and Repairs

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

2. Overview of the YEI 3-Space Sensor

2.1 Introduction

The YEI 3-Space Sensor™ Embedded is an ultra-miniature, high-precision, high-reliability, low-cost 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.

Orientation can be returned in absolute terms or relative to a designated reference orientation. The proprietary multi-reference vector mode increases accuracy and greatly reduces and compensates for sensor error. The YEI 3-Space Sensor Embedded system also utilizes a dynamic sensor confidence algorithm that ensures optimal accuracy and precision across a wide range of operating conditions.

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

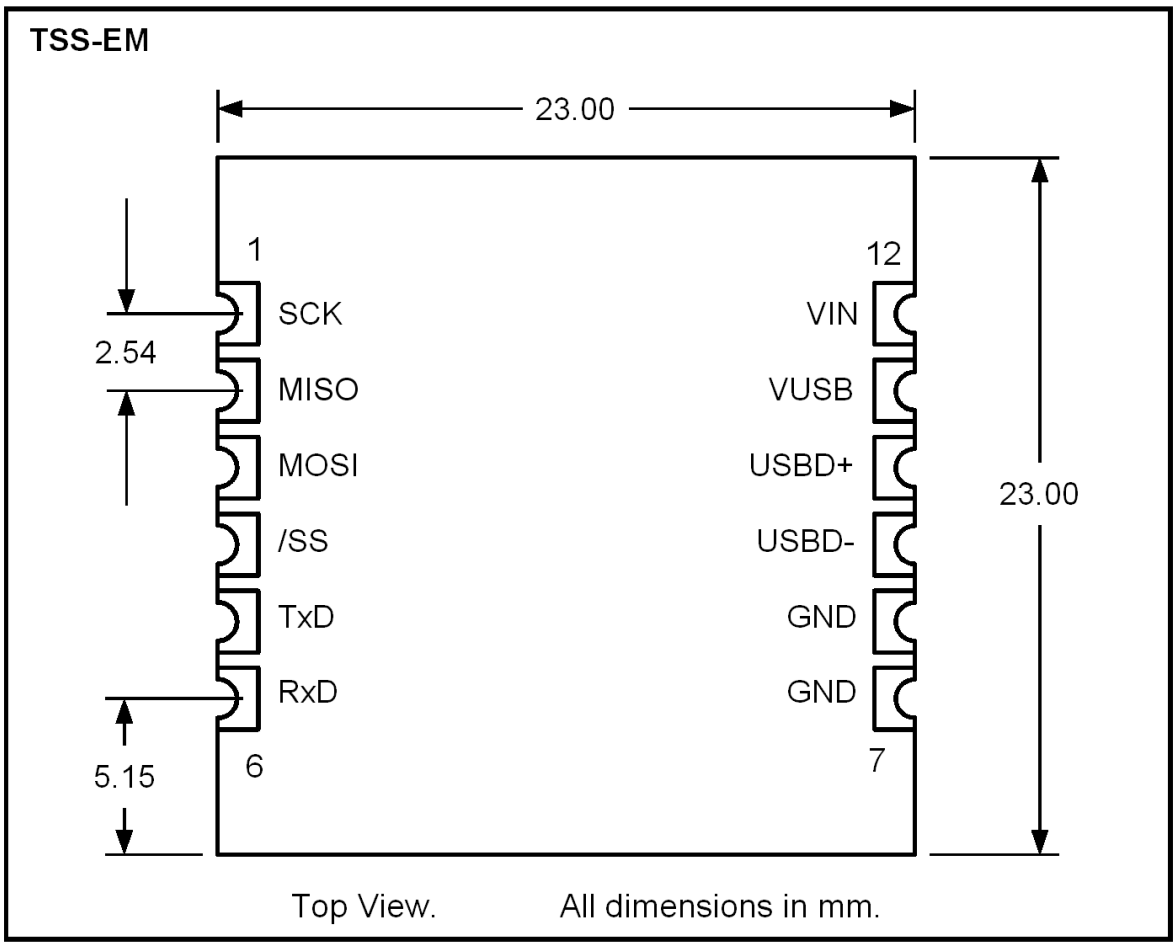
When used as a USB device, the Embedded 3-Space Sensor™ provides mouse emulation and joystick emulation modes that ease integration with existing applications.

2.2 Applications

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

2.3 Hardware Overview

The YEI 3-Space Embedded is packaged as a 23mmx23mmx2.2mm castellated edge SMT module. Alternatively, the module can be through-hole mounted by adding standard 0.1" header strips to the castellated edge pads.

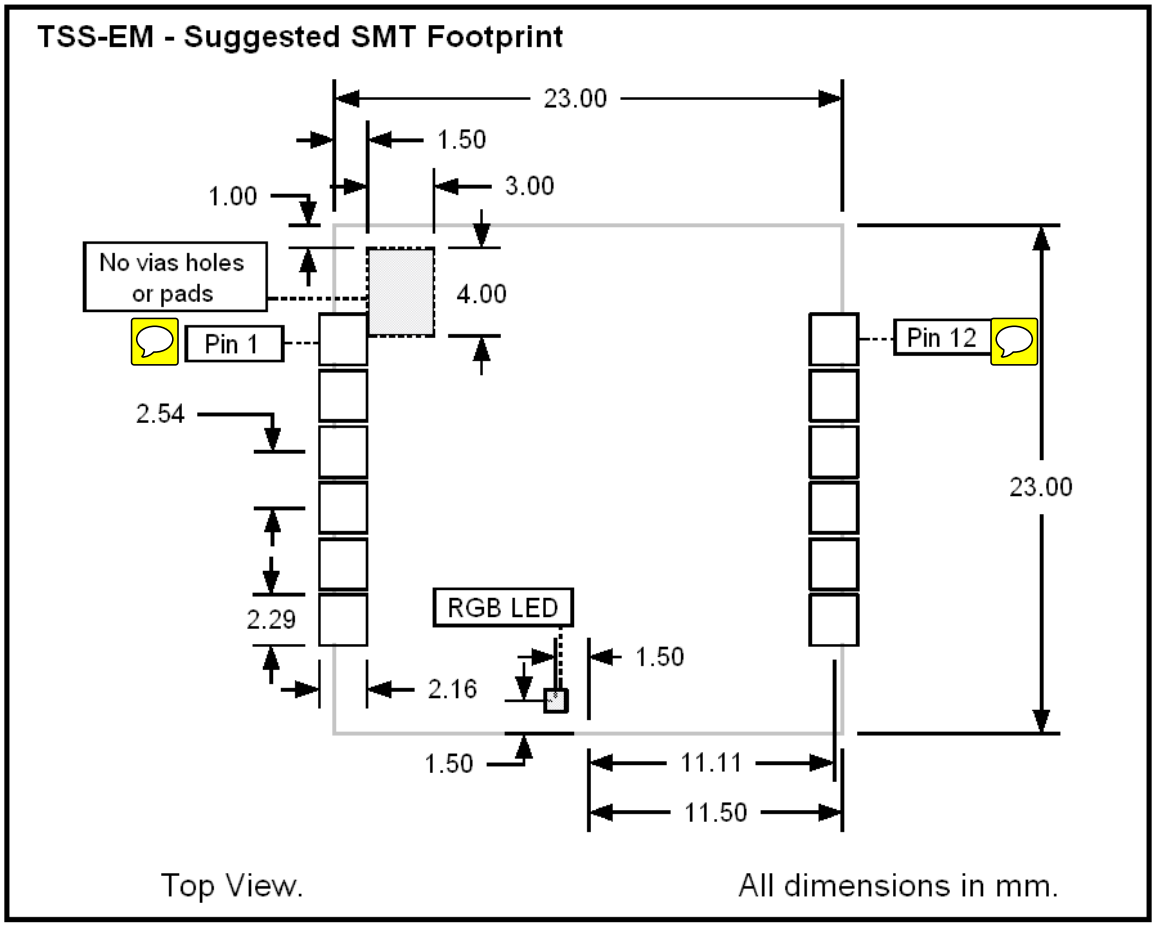


2.3.1 Pin Functions

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

2.3.2 PCB Layout

PCB layout should follow follow the suggested SMT footprint below.



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

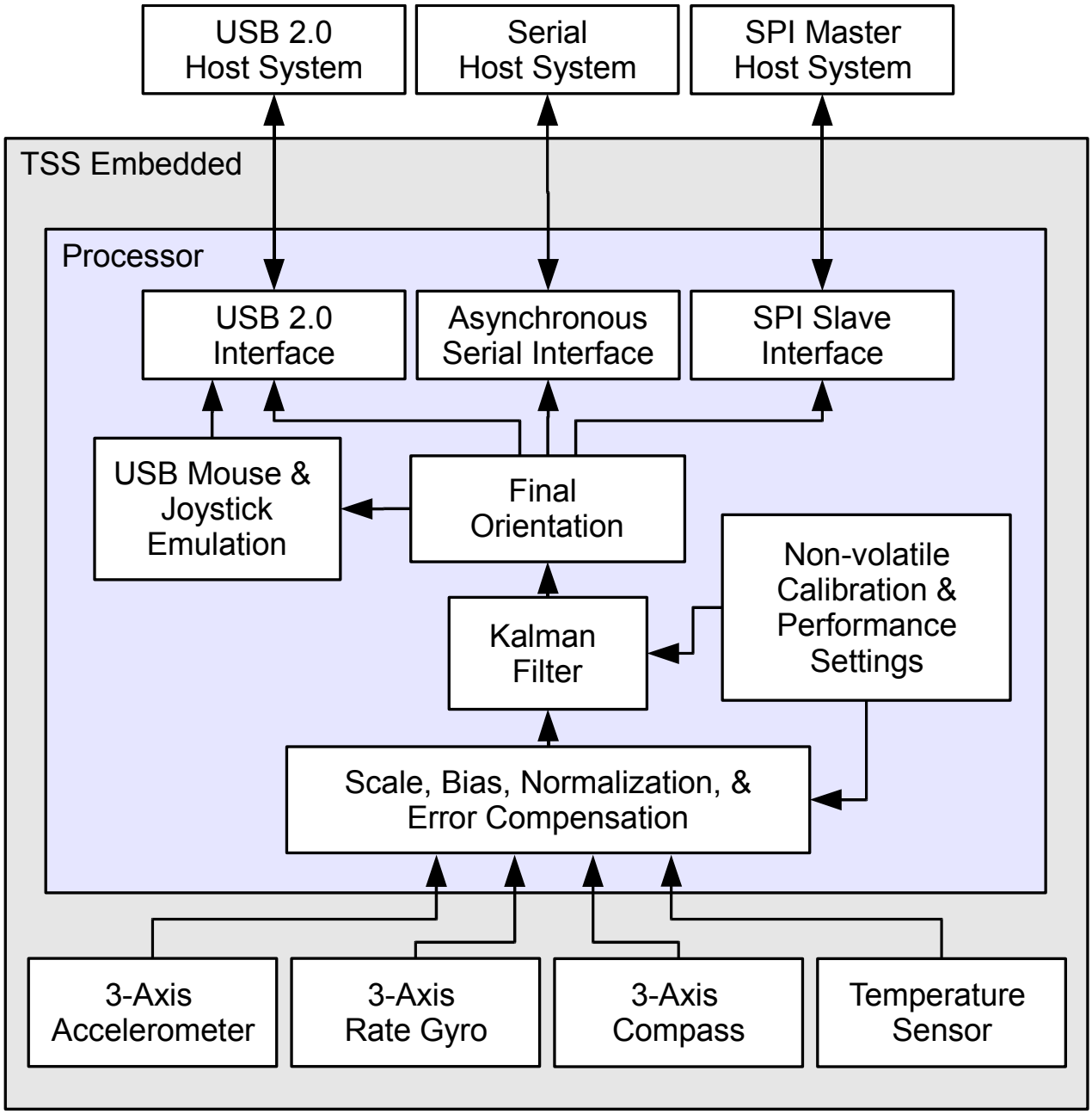
- Do not place untented pads, vias, or holes beneath the restricted area in the diagram.
- Do not place magnetic components such as speakers and motors in close proximity to the module since the magnetic fields generated can adversely affect the performance of the compass module.
- Do not place components 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-EM module may result in unforeseen performance consequences due to layout.

2.4 Features

The YEI 3-Space Sensor Embedded 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 23mm x 23mm x 2mm and only 1.3 grams
- Fast sensor update and filter rate allow use in real-time applications, including stabilization, virtual reality, real-time immersive simulation, and robotics
- Highly customizable orientation sensing with options such as tunable filtering, oversampling, and orientation error correction
- Advanced integrated Kalman filtering allows sensor to automatically reduce the effects of sensor noise and sensor error
- Robust open protocol allows commands to be sent in human readable form, or more quickly in machine readable form
- Orientation output format available in absolute or relative terms in multiple formats (quaternion, rotation matrix, axis angle, two-vector)
- Absolute or custom reference axes
- Access to raw sensor data
- Flexible communication options: SPI, USB 2.0, or asynchronous serial
- USB communication through a virtual COM port
- When used as a USB device, USB joystick/mouse emulation modes ease integration with existing applications
- Castellated SMT edge pads provide secure SMT mounting and allow optional through-hole mounting
- Upgradeable firmware
- RGB status LED
- Programmable interrupt capability
- Development kit available
- RoHS Compliant
- +5v tolerant I/O signals

2.5 Block Diagram of Sensor Operation



2.6 Specifications

General	
Part number	TSS-EM
Dimensions	23mm x 23mm x 2.2mm (0.9 x 0.9 x 0.086 in.)
Weight	1.3 grams (0.0458 oz)
Supply voltage	+3.3v ~ +6.0v
Power consumption	45mA @ 5v
Communication interfaces	USB 2.0, SPI, Asynchronous Serial
Filter update rate	Up to 200Hz with full functionality
Orientation output	absolute & relative quaternion, Euler angles, axis angle, rotation matrix, two vector
Other output	raw sensor data, normalized sensor data, temperature
SPI clock rate	6 MHz 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 @ 60MHz
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 selectable
Accelerometer resolution	14 bit
Accelerometer noise density	99µg/√ Hz
Accelerometer sensitivity	0.00024g/digit for ±2g range 0.00048g/digit for ±4g range 0.00096g/digit for ±8g range
Accelerometer temperature sensitivity	±0.008%/°C
Gyro scale	±250/±500/±2000 º/sec selectable
Gyro resolution	16 bit
Gyro noise density	0.03º/sec/√ Hz
Gyro bias stability @ 25°C	11º/hr average for all axes
Gyro sensitivity	0.00875º/sec/digit for ±250º/sec 0.01750º/sec/digit for ±500º/sec 0.070º/sec/digit for ±2000º/sec
Gyro non-linearity	0.2% full-scale
Gyro temperature sensitivity	±0.016%/°C
Compass scale	±1.3 Ga default. Up to ±8.1 Ga available
Compass resolution	12 bit
Compass sensitivity	5 mGa/digit
Compass non-linearity	0.1% full-scale

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_{ACT}	Active Current Consumption		45	60	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.

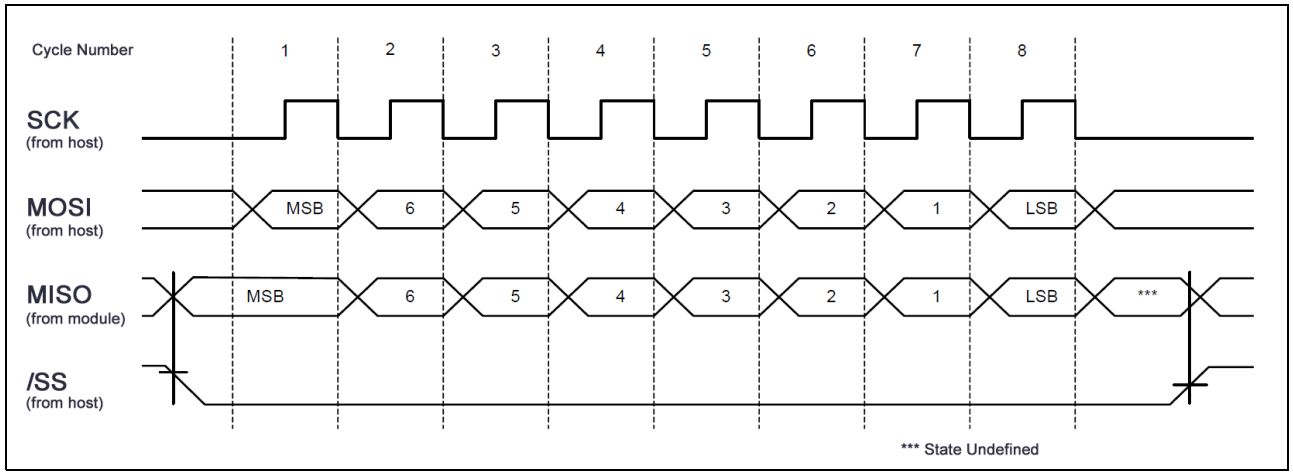
The Asynchronous Serial uses 8N1 (8 data bits, no parity, 1 stop bit) format and supports the following standard baud rates: 1200, 2400, 4800, 9600, 19200, 28800, 38400, 57600, 115200, 230400, 460800, 921600.

The factory default baud rate is 115200.

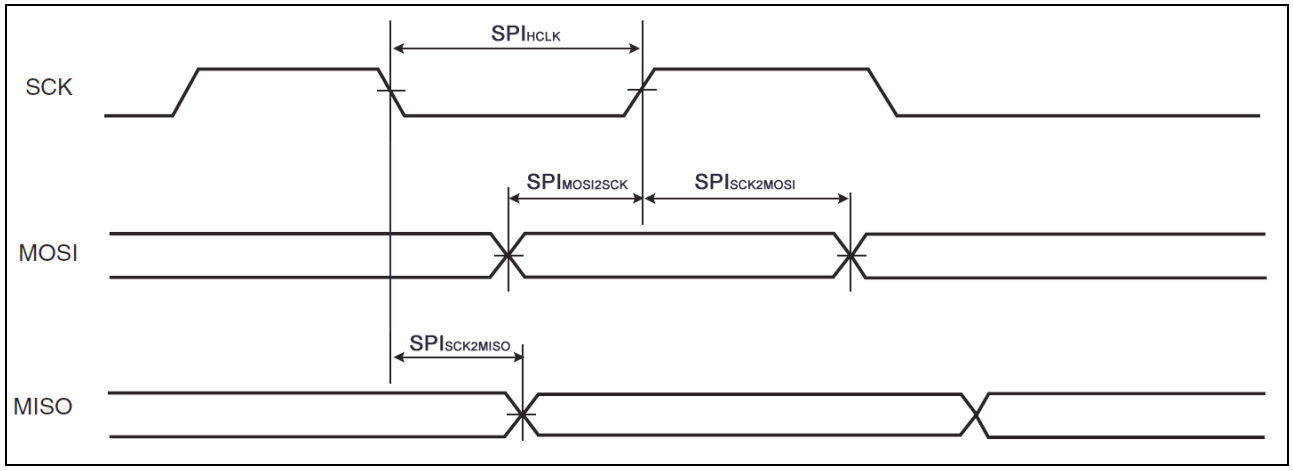
2.7.5 SPI Characteristics

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 6MHz. 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	80		ns
$SPI_{SCK2MISO}$	SPI SCK falling to MISO Delay		26.5	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

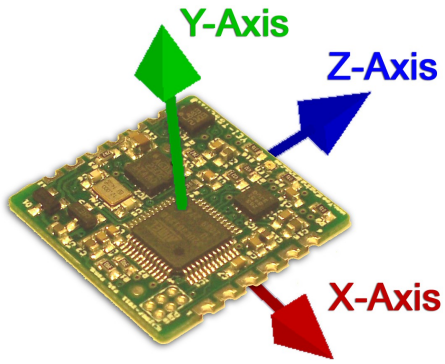
2.8 Axis Assignment

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

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

- The positive X-axis points out of the side of the sensor with pins 1 through 6.
- 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 (the side with the LED, towards pins 6 and 7).

The natural axes are illustrated in the diagram below:



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

3. Description of the 3-Space Sensor

3.1 Orientation Estimation

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

3.1.1 Component Sensors

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

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

3.1.2 Scale, Bias, and Cross-Axis Effect

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

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

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

3.1.3 Reference Vectors

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

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

3.1.4 Kalman Filter

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

3.1.5 Reference Orientation/Taring

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

3.1.6 Other Estimation Parameters

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

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

3.2 Communication

Obtaining data about orientation from the sensor or giving values for any of its settings is done through the sensor's communication protocol. The protocol can be used through either a USB connection, an asynchronous serial UART connection, or an SPI 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.

3.3 Input Device Emulation

3.3.1 Axes and Buttons

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

3.3.2 Joystick

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

3.3.3 Mouse

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

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

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

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

- The positive X-axis points out of the side of the sensor with pins 1 through 6.
- 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 (the side with the LED, towards pins 6 and 7).

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 Rho Value	Determine how trusted the accelerometer is	Confidence Mode, 5 to 100
Compass Rho Value	Determine how trusted the compass is	Confidence Mode, 5 to 100
Accelerometer Coefficients	Determines the scale, bias, and cross-axis parameters for the accelerometer	Factory calibrated
Compass Coefficients	Determines the scale, bias, and cross-axis parameters for the compass	Factory calibrated
Accelerometer Enabled	Determines whether the compass is enabled or not	TRUE
Compass Enabled	Determines whether the accelerometer is enabled or not	TRUE
Gyroscope Enabled	Determines whether the gyroscope is enabled or not	TRUE
Axis Directions	Determines what natural axis direction each data axis faces	+X, +Y, +Z
Sample Rate	Determines how many samples the sensor takes per cycle	1 from each component sensor
Running Average Percentage	Determines how heavy of a running average to run on the final orientation	0(no running average)
Desired Update Rate	Determines how long each cycle should take(ideally)	0 microseconds
Reference Mode	Determines how the accelerometer and compass reference vectors are determined	Single Auto
UART Baud Rate	Determines the speed of the Serial UART communication	115200
CPU Speed	Determines how fast the CPU will run	60 MHz
LED Color	Determines the RGB color of the LED	0,0,1(Blue)
Joystick Enabled	Determines whether the joystick is enabled or not	TRUE
Mouse Enabled	Determines whether the mouse is enabled or not	FALSE
Button Gyro Disable Length	Determines how many cycles the gyro is ignored after a button is pressed	5
Multi Reference Weight Power	Determines what power each multi reference vector weight is raised to	10
Multi Reference Cell Divisions	Determines how many cells the multi reference lookup table is divided into per axis	4
Multi Reference Nearby Vectors	Determines how many nearby vectors each multi reference lookup table cell stores	8
Interrupt Generation Mode	Determines how interrupts are generated	Off, pin TXD

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.

The 3-Space Sensor buffers the incoming command stream and will only take an action once the entire packet has been received and the checksum has been verified as correct(ASCII mode commands do not use checksums for convenience). Incomplete packets and packets with incorrect checksums will be ignored. This allows the controlling system to send command data at leisure without loss of functionality. The command buffer will, however, be cleared whenever the 3-Space Sensor is either reset or powered off/on.

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

4.1.2 Computer Interfacing Overview

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 Embedded module offers three interfacing /communications options: USB 2.0, Asynchronous Serial, and Serial Peripheral Interface (SPI). 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

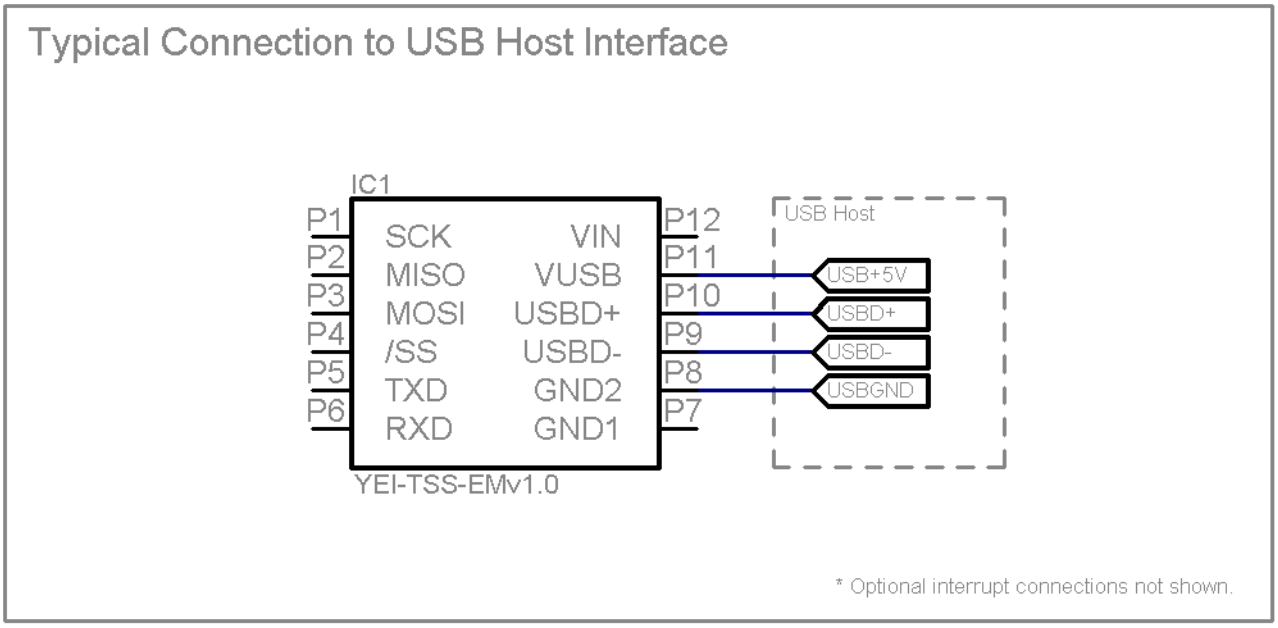
The USB 2.0 interface of the 3-Space Sensor Embedded requires the connection of signals as follows:

Pin	Signal	Description
8	GND	USB Ground. Required connection during USB mode use.
9	USBD-	USB Data Minus. Required connection during USB mode use.
10	USBD+	USB Data Plus. Required connection during USB mode use.
11	VUSB	+5v USB Power Supply Input . Required connection during USB mode use.

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

Pin	Signal	Description
2	MISO / INT	Configurable as filter update interrupt when SPI interface is unused.
5	TxD / INT	Configurable as filter update interrupt when asynchronous serial interface is unused.

The following schematic diagram illustrates typical USB interface connections:



4.1.3.2 Asynchronous Serial Interfacing

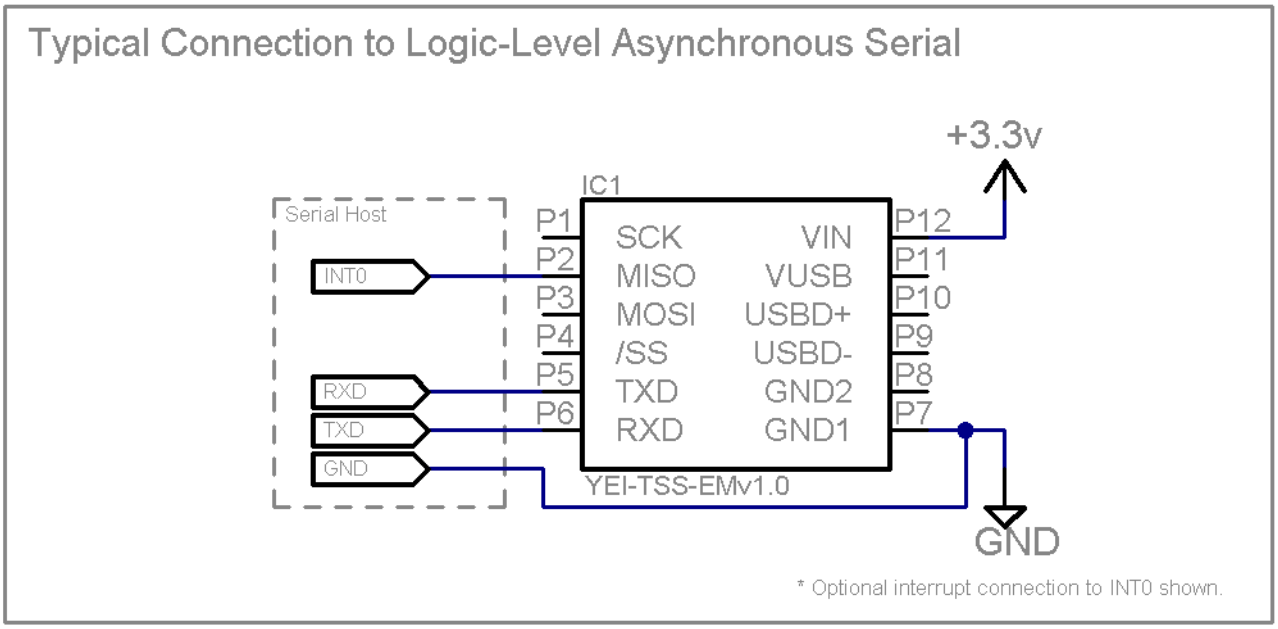
The asynchronous serial interface of the 3-Space Sensor Embedded requires the connection of signals as follows:

Pin	Signal	Description
5	TxD	UART Asynchronous Transmit Data. Output from Module.
6	RxD	UART Asynchronous Receive Data. Input to Module.
7,8	GND	Ground. Only one ground pad must be connected.
12	VIN	Voltage Input +3.3v ~ +6.0v. Only required when USB power is not being used.

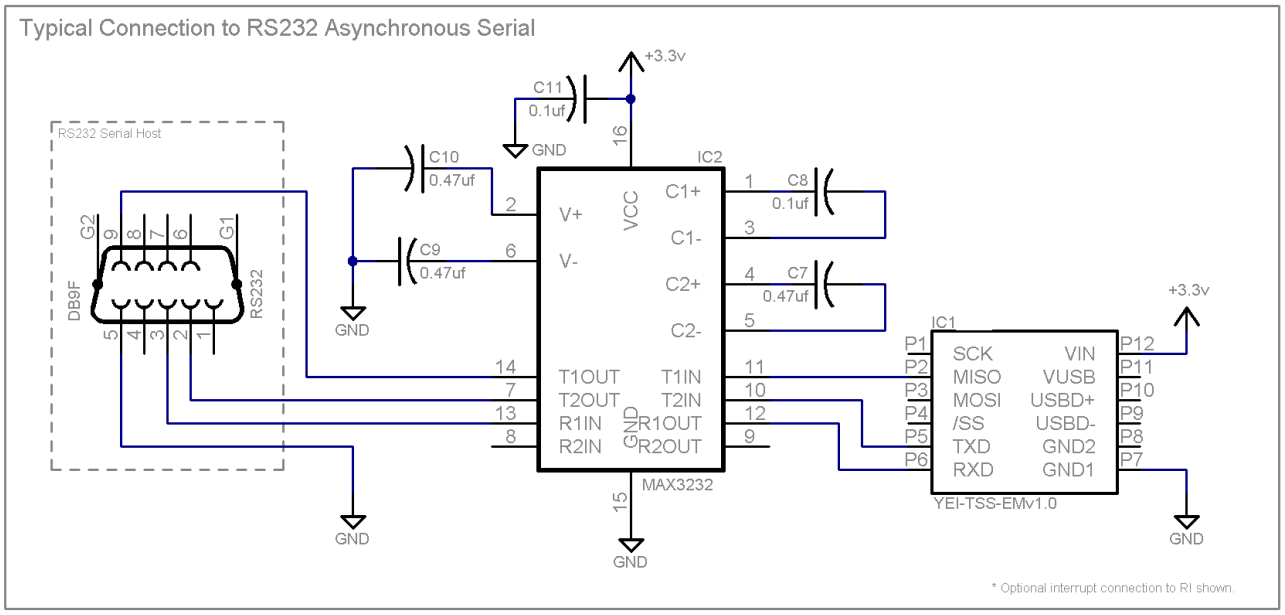
Additionally, the following optional interrupt pin may be configured for use during asynchronous serial mode:

Pin	Signal	Description
2	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:



4.1.3.3 SPI Interfacing

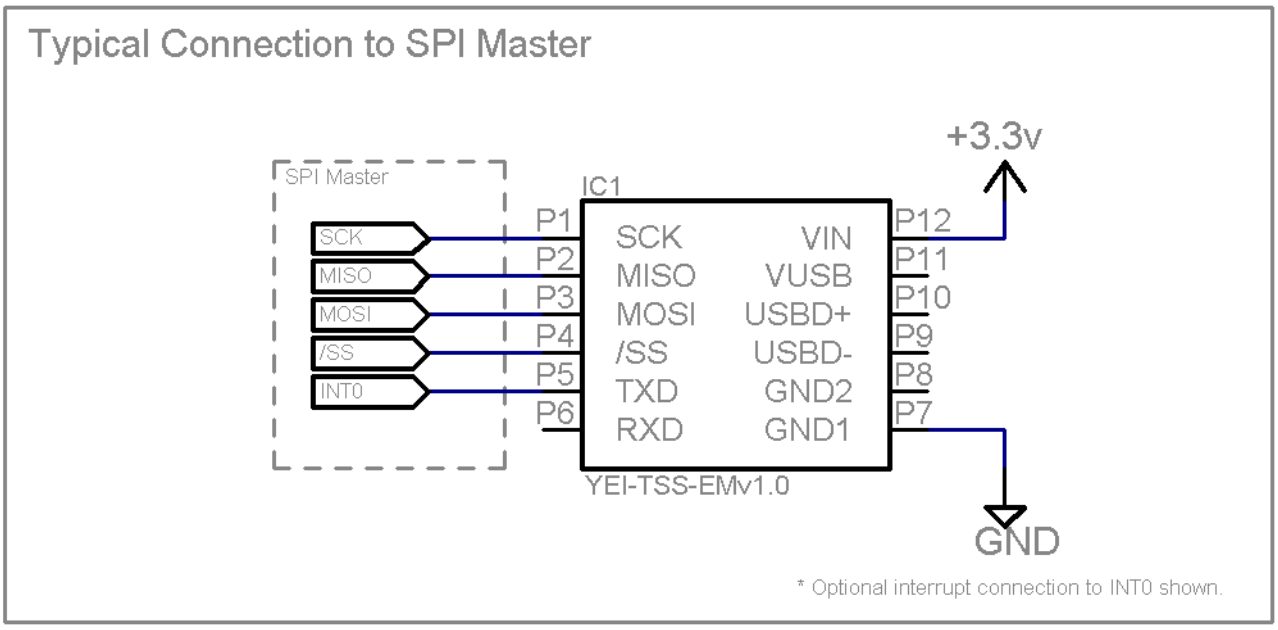
The Serial Peripheral Interface (SPI) of the 3-Space Sensor Embedded requires the connection of signals as follows:

Pin	Signal	Description
1	SCK	SPI Serial Clock. Input to Module.
2	MISO	SPI Master In Slave Out. Output from Module.
3	MOSI	SPI Master Out Slave In. Input to Module.
4	/SS	SPI Slave Select. Active Low Input to Module.

Additionally, the following optional interrupt pin may be configured for use during SPI mode:

Pin	Signal	Description
5	TxD / INT	Configurable as filter update interrupt when asynchronous serial interface is unused. Additionally, can be configured as command complete interrupt when asynchronous serial nterface is unused.

The following schematic diagram illustrates typical SPI interface connections:



4.1.3.4 Interrupt Generation

The Embedded 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. This pin will be high by default. The signal can be set to act in filter pulse mode, where the pin is set low for 5 microseconds and then pulled back to high, or it can be set to filter level mode, where the pin is set low until the interrupt status is read(see command 18). Additionally, the signal may be set to act in SPI command pulse mode, where the pin pulses once any given command has been completed. By default, no pin is set to act as the interrupt generation pin. Either the SPI MISO pin or the UART TXD pin may be set to act as the interrupt pin, meaning that while interrupt generation is active, either the UART or SPI will be unusable. In SPI command pulse mode, the UART TXD pin will be used automatically to signal the completion of a command, regardless of which pin is specified with command 16. For more information on setting the interrupt pin and mode, see command 16.

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

5	TxD / INT	Configurable as filter update interrupt when asynchronous serial interface is unused. Additionally, can be configured as command complete interrupt when SPI interface is used.

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

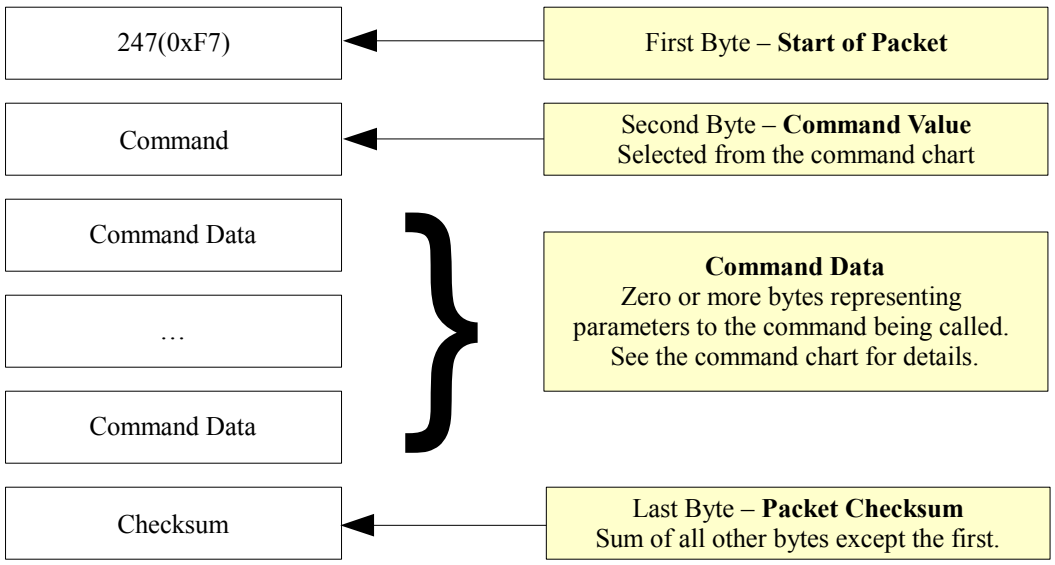


Figure 1 - Typical Binary Command Packet Format

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.

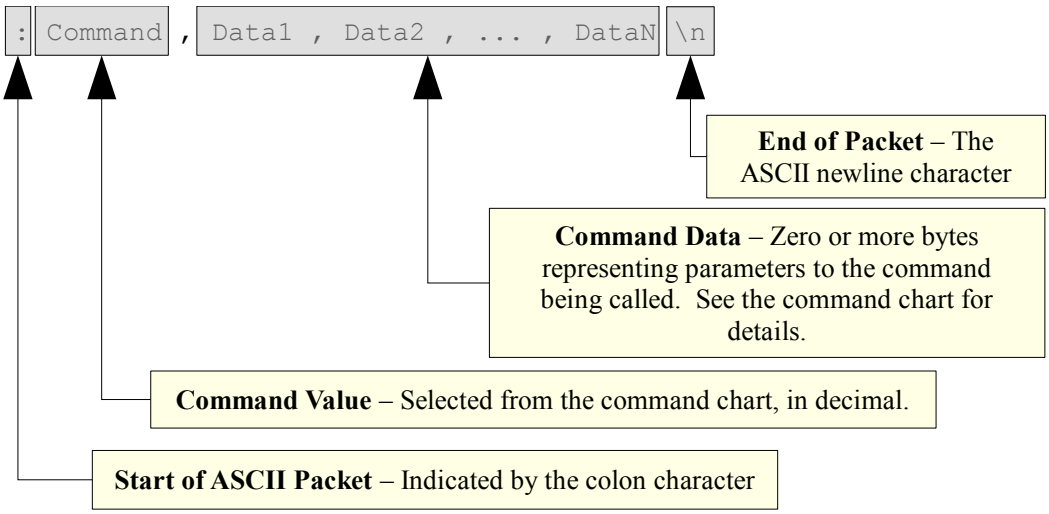


Figure 2 - Typical ASCII Command Packet Format

Thus the ASCII packet consists of the 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(SPI)

4.3.1 Command Packet Format

In order to initiate an SPI data transfer, the byte 0xF6 must be sent to signal the start of an incoming command packet. Afterwards, the command byte should be sent as well as any required command parameter bytes. After the command has been processed, the byte 0xFF must be sent repeatedly to read any bytes returned from the sensor. While the sensor is not currently processing a command, any byte sent to it other than 0xF6 and 0xFF will cause the internal data buffer to reset, thus clearing any response data prepared by the sensor. Once the sensor has responded with a 1 (indicating the command has finished), the user must send repeated bytes of 0xFF until all command data is read. In other words, if a command returns 12 bytes, 12 bytes of 0xFF must be sent after the 1 has been received. Additionally, there are several internal states that the sensor maintains while processing SPI commands:

0x0 (IDLE) The sensor is waiting on a command. Any bytes sent to the sensor besides 0xF6 will have no effect.

0x1 (READY) The sensor has processed a command and data is available to read. Any byte sent to the sensor other than 0xFF will reset the internal data buffer.

0x2 (BUSY) The sensor is currently processing a command.

0x4 (ACCUMULATING) The sensor is accumulating command bytes, but has not received enough to run the command. Anything sent to the sensor in this state will be interpreted as command data.

The following diagram illustrates the process for sending command data and reading response data. Command 230(0xE6) is the id command, and returns 32 total bytes, where the first three bytes are “TSS”. First, 0xF6 is sent to the sensor over SPI, which responds with a 0x0.The 0xE6 byte is sent to the sensor over SPI, which will receive a response of 0x4. The byte 0xFF is sent to the sensor until it responds with a 1. Once it does, 32 bytes of 0xFF are sent to the sensor until all data is retrieved. Only 3 of the data byte communications are illustrated here for brevity.

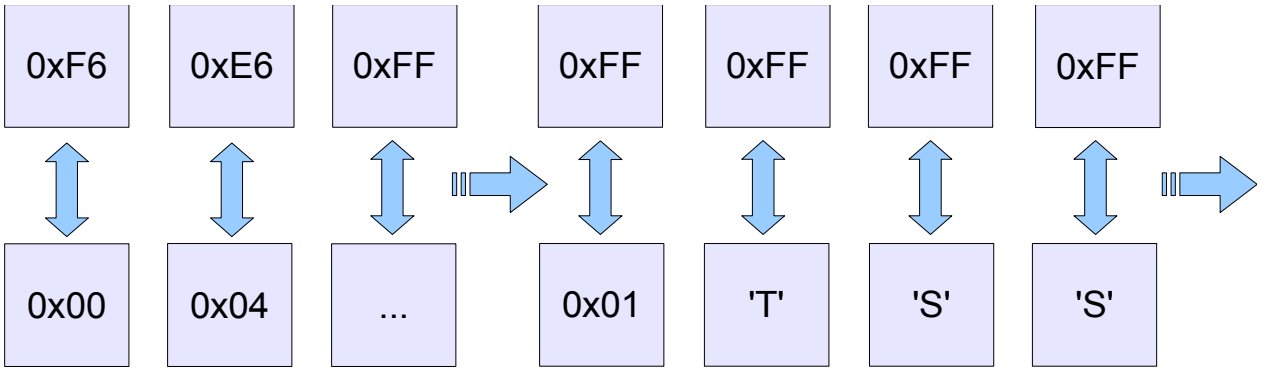


Figure 3 – Sample SPI Communication

4.4 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.

4.3.1 Commands for Reading Filtered Sensor Data

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

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

4.3.2 Commands for Interfacing with Electronic Systems

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

4.3.3 Commands for Reading Normalized Sensor Data

These commands return sensor data which has been converted from a raw form. In the case of the normalized accelerometer and compass, the data returned are unit vectors, and as such, have no magnitude data associated with them. For data that represents real world quantities, both the unnormalized accelerometer command and unnormalized compass command return data in units of g and gauss respectively. Normalized gyro data is returned in radians/sec.

Command	Description	Return Len	Return Data Details
32(0x20)	Read all(gyro, accelerometer, compass)	36	Vector(float x3), Vector(float x3), Vector(float x3)
33(0x21)	Read gyros	12	Vector(float x3)
34(0x22)	Read accelerometer	12	Vector(float x3)
35(0x23)	Read compass	12	Vector(float x3)
36(0x24)	Read temperature C	4	float
37(0x25)	Read temperature F	4	float
38(0x26)	Read confidence factor	4	float
39(0x27)	Read accelerometer unnormalized	12	Vector(float x3)
40(0x28)	Read compass unnormalized	12	Vector(float x3)

4.3.4 Commands for Reading Raw Sensor Data

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

Command	Description	Return Len	Return Data Details	Data Len	Data Details
64(0x40)	Read all raw	36	Vector(float x3), Vector(float x3), Vector(float x3)	0	
65(0x41)	Read gyro raw	12	Vector(float x3)	0	
66(0x42)	Read accelerometer raw	12	Vector(float x3)	0	
67(0x43)	Read compass raw	12	Vector(float x3)	0	

4.3.5 Commands for Setting Filter Parameters

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

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

4.3.6 Commands for Reading Filter Parameters

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

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

4.3.7 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)

4.3.8 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
223(0xdf)	Read software version	12	Software version string	0	
224(0xe0)	Restore factory settings	0		0	
225(0xe1)	Commit Settings	0		0	
226(0xe2)	Software reset	0		0	
227(0xe3)	Enable watchdog timer	0		4	Timeout rate in microseconds(int)
228(0xe4)	Disable watchdog timer	0		0	
229(0xe5)	Enter firmware update mode	0		0	
230(0xe6)	Read hardware version	32	Hardware version 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	
233(0xe9)	Set USB mode	0		1	Mode(byte, 1 for FTDI, 0 for CDC)
234(0xea)	Get USB mode	1	Mode(byte, 1 for FTDI, 0 for CDC)	0	
235(0xeb)	Set clock speed	0		4	Clock speed(int, Hz)
236(0xec)	Get clock speed	4	Clock speed(int, Hz)	0	
237(0xed)	Get serial number	4	Serial number(int)	0	
238(0xee)	Set LED color	0		12	Red(float), Green(float), Blue(float)
239(0xef)	Get LED color	12	Red(float), Green(float), Blue(float)	0	
240(0xf0)	Enable/Disable joystick	0		1	Mode(byte, 0 for disabled, 1 for enabled)
241(0xf1)	Enable/Disable mouse	0		1	Mode(byte, 0 for disabled, 1 for enabled)
242(0xf2)	Read joystick enabled state	1	Mode(byte, 0 for disabled, 1 for enabled)	0	
243(0xf3)	Read mouse enabled state	1	Mode(byte, 0 for disabled, 1 for enabled)	0	
244(0xf4)	Set control mode	0		3	Control class(byte), control index(byte), handler index(byte)
245(0xf5)	Set control data	0		7	Control class(byte), control index(byte), data point index(byte), data point(float)
246(0xf6)	Read control mode	1	Handler index(byte)	2	Control class(byte), control index(byte)
247(0xf7)	Read control data	4	Data point(float)	3	Control class(byte), control index(byte), data point index(byte)
251(0xfb)	Set mouse absolute/relative	0		1	Mode(0 for absolute, 1 for relative)
252(0xfc)	Read mouse absolute/relative	1	Mode(0 for absolute, 1 for relative)	0	
253(0xfd)	Set joystick and mouse present/removed	0		2	Mode(byte for each, 0 for removed, 1 for present)
254(0xfe)	Read joystick and mouse present/removed	2	Mode(byte for each, 0 for removed, 1 for present)	0	

4.4 Command Details

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

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

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:	Read filtered, untared orientation(Two Vector(Forward, Down))
Command Value:	10 (0x0A)
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 reference vectors. This version returns the data as the forward and down vectors of the coordinate system defined by the orientation.

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

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

Function:	Set interrupt type
Command Value:	29 (0x1D)
Data Bytes:	2
Data Format:	Mode(byte, 0 for off, 1 for pulse, 2 for level, 3 for SPI pulse mode), pin(byte, 0 for TXD, 1 for MISO)
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. If mode is 3, the interrupt generator will pulse the pin each time a command is ran while communicating over SPI. Note that regardless of which pin is specified for this mode, TXD will be used. 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 TXD over the UART, and cannot be changed to MISO over SPI.

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

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

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

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

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

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

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

Function:	Read accelerometer unnormalized
Command Value:	39 (0x27)
Return Data Bytes:	12
Return Data Format:	Vector(float x3)
Description:	Returns the unnormalized gravity vector. This reading is given in terms of g.

Function:	Read compass unnormalized
Command Value:	40 (0x28)
Return Data Bytes:	12
Return Data Format:	Vector(float x3)
Description:	Returns the unnormalized north vector. This reading is given in terms of gauss.

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:	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 the orientation specified by the given 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 the orientation specified by the given rotation matrix.

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

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

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

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

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

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

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

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

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

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

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

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

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

Function:	Set accelerometer ramge
Command Value:	121 (0x79)
Data Bytes:	1
Data Format:	Accel range(byte, 0 for ±2g, 1 for ±4g, 2 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 multi reference weight power
Command Value:	122 (0x7a)
Data Bytes:	4
Data Format:	Weight power(float)
Description:	Set weighting power for multi reference vector weights. Multi reference vector weights are all raised to the weight power before they are summed and used in the calculation for the final reference vector. Setting this value nearer to 0 will cause the reference vectors to overlap more, and setting it nearer to infinity will cause the reference vectors to influence a smaller set of orientations.

Function:	Enable/disable filter
Command Value:	123 (0x7b)
Data Bytes:	1
Data Format:	Mode (byte, 0 for disabled, 1 for full kalman)
Description:	Used to disable the orientation filter or set the orientation filter mode. Changing this parameter can be useful for tuning filter-performance versus orientation-update rates. When using the sensor as an IMU, it will improve performance to disable the orientation filter.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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:	Set compass calibration parameters
Command Value:	160 (0xA0)
Data Bytes:	48
Data Format:	Matrix(float x9), Bias(float x3)
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:	Matrix(float x9), Bias(float x3)
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:	Read software version string
Command Value:	223 (0xDF)
Return Data Bytes:	12
Return Data Format:	Version(string)
Description:	Reads the software 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:	Enable watchdog timer
Command Value:	227 (0xE3)
Data Bytes:	4
Data Format:	Timeout rate in microseconds(int)
Description:	Enables the watchdog timer with the given timeout rate. If a frame takes more than this amount of time, the sensor will automatically reset. This is useful for dealing with sensor hangs or crashes, as the sensor would reset and continue normal operation.

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

Function:	Enter firmware update 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:	Read hardware version string
Command Value:	230 (0xE6)
Return Data Bytes:	32
Return Data Format:	Version(string)
Description:	Reads the version identifier of the sensor firmware.

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

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

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

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

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

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

Function:	Enable/Disable joystick
Command Value:	240 (0xf0)
Data Bytes:	1
Data Format:	Mode(byte, 0 for disabled, 1 for enabled)
Description:	Turns the data feed to the joystick on and off. If disabled, the sensor will still enumerate as a joystick, but the joystick will not function. This allows normal communication to occur at a faster rate.

Function:	Enable/Disable mouse
Command Value:	241 (0xf1)
Data Bytes:	1
Data Format:	Mode(byte, 0 for disabled, 1 for enabled)
Description:	Turns the data feed to the mouse on and off. If disabled, the sensor will still enumerate as a mouse, but the mouse will not function. This allows normal communication to occur at a faster rate.

Function:	Read joystick enabled state
Command Value:	242 (0xf2)
Return Data Bytes:	1
Return Data Format:	Mode(byte, 0 for disabled, 1 for enabled)
Description:	Read whether or not the joystick is enabled.

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

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

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

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

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

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

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

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

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

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: ______________________________

YEI Technology

630 Second Street
Portsmouth, Ohio 45662

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

www.YeiTechnology.com
www.3SpaceSensor.com