User Manual

Target Point TCM

Digital Compass and Attitude & Heading Reference System (AHRS)

PNI

Table of Contents

List of Figures

List of Tables

1 Copyright & Warranty Information

PNI Sensor
2331 Circadian Way
Santa Rosa, CA 95407, USA
Tel: (707) 566-2260
Fax: (707) 566-2261

Warranty and Limitation of Liability. PNI Sensor ("PNI") manufactures its TargetPoint TCM (TPTCM) products ("Products") from parts and components that are new or equivalent to new in performance. PNI warrants that each Product to be delivered hereunder, if properly used, will, for one year following the date of shipment unless a different warranty time period for such Product is specified: (i) in PNI's Price List in effect at time of order acceptance; or (ii) on PNI's web site (www.pnicorp.com) at time of order acceptance, be free from defects in material and workmanship and will operate in accordance with PNI's published specifications and documentation for the Product in effect at time of order. PNI will make no changes to the specifications or manufacturing processes that affect form, fit, or function of the Product without written notice to the OEM, however, PNI may at any time, without such notice, make minor changes to specifications or manufacturing processes that do not affect the form, fit, or function of the Product. This warranty will be void if the Products' serial number, or other identification marks have been defaced, damaged, or removed. This warranty does not cover wear and tear due to normal use, or damage to the Product as the result of improper usage, neglect of care, alteration, accident, or unauthorized repair.

THE ABOVE WARRANTY IS IN LIEU OF ANY OTHER WARRANTY, WHETHER EXPRESS, IMPLIED, OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, ANY WARRANTY OF MERCHANTABILITY, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION, OR SAMPLE. PNI NEITHER ASSUMES NOR AUTHORIZES ANY PERSON TO ASSUME FOR IT ANY OTHER LIABILITY.

If any Product furnished hereunder fails to conform to the above warranty, OEM's sole and exclusive remedy and PNI's sole and exclusive liability will be, at PNI's option, to repair, replace, or credit OEM's account with an amount equal to the price paid for any such Product which fails during the applicable warranty period provided that (i) OEM promptly notifies PNI in writing that such Product is defective and furnishes an explanation of the deficiency; (ii) such Product is returned to PNI's service facility at OEM's risk and expense; and (iii) PNI is satisfied that claimed deficiencies exist and were not caused by accident, misuse, neglect, alteration, repair, improper installation, or improper testing. If a Product is defective, transportation charges for the return of the Product to OEM within the United States and Canada will be paid by PNI. For all other locations, the warranty excludes all costs of shipping, customs clearance, and other related charges. PNI will have a reasonable time to make repairs or to replace the Product or to credit OEM's account. PNI warrants any such repaired or replacement Product to be free from defects in material and workmanship on the same terms as the Product originally purchased.

Except for the breach of warranty remedies set forth herein, or for personal injury, PNI shall have no liability for any indirect or speculative damages (including, but not limited to, consequential, incidental, punitive and special damages) relating to the use of or inability to use this Product, whether arising out of contract, negligence, tort, or under any warranty theory, or for infringement of any other party's intellectual property rights, irrespective of whether PNI had advance notice of the possibility of any such damages, including, but not limited to, loss of use, revenue or profit. In no event shall PNI's total liability for all claims regarding a Product exceed the price paid for the Product. PNI neither assumes nor authorizes any person to assume for it any other liabilities.

Some states and provinces do not allow limitations on how long an implied warranty lasts or the exclusion or limitation of incidental or consequential damages, so the above limitations or exclusions may not apply to you. This warranty gives you specific legal rights and you may have other rights that vary by state or province.

2 Introduction

Thank you for purchasing PNI's TargetPoint TCM (TPTCM) digital compass and AHRS. In digital compass mode, the TPTCM intelligently fuses PNI's patented Reference Magnetic Sensors with a 3-axis accelerometer. The result is an orientation device that provides accurate heading information.

TPTCM is also an attitude & heading reference system (AHRS). It employs a proprietary Kalman filtering algorithm that intelligently fuses PNI's patented Reference Magnetic Sensors with a 3-axis gyroscope and 3-axis accelerometer.

We're sure the TPTCM will help you to achieve the greatest performance from your system. Thank you for selecting the TPTCM.

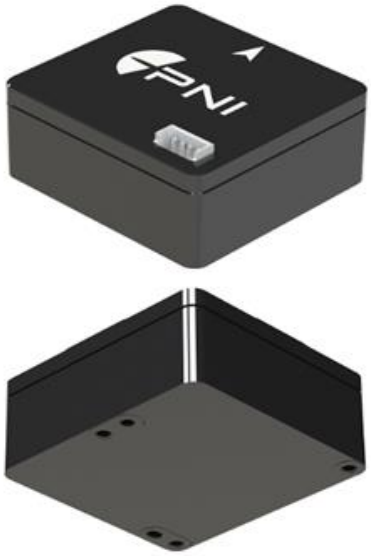


Figure 2-1: TPTCM Top and Bottom

3 Specifications

3.1 Characteristics & Requirements

Table 3-1: Performance Specifications[1]

Parameter				Value
Compass Mode	Heading	Static Accuracy[2]		0.25° rms
		Resolution		0.01°
		Repeatability		0.05° rms
	Attitude	Range	Pitch	± 90°
			Roll	± 180°
		Static Accuracy		0.2° rms
		Resolution		0.01°
		Repeatability		0.05° rms
AHRS Mode	Heading	Accuracy[3]		2.0° rms
		Resolution		0.1°
	Attitude	Range	Pitch	± 90°
			Roll	± 180°
		Accuracy		2.0° rms
		Resolution		0.01°
Magnetometer Calibrated Range				± 150 µT
Accelerometer Calibrated Range				± 1.0 g

Footnotes:

1. Specifications are typical unless otherwise noted, and subject to change.
2. Assumes TPTCM is motionless, the local magnetic field is clean relative to user calibration, ≤65° of pitch, and after a Full-Range calibration has been performed.
3. Assumes heading status is "1". See Section 6.6 or Section 7.5.3 for a discussion on heading status.

Table 3-2: Absolute Maximum Ratings

Parameter	Minimum	Maximum	Units
Supply Voltage	-0.3	+10	VDC
Storage Temperature	-40	+85	°C

CAUTION:

Stresses beyond those listed above may cause permanent damage to the device. These are stress ratings only. Operation of the device at these or other conditions beyond those indicated in the operational sections of the specifications is not implied.

Table 3-3: Electrical Requirements

Parameter		Value
Supply Voltage		3.7 to 9 VDC
Serial UART CMOS/TTL(1)	UART Rx (2) V_{IH}	2.3 V min 1.87V min Typical(1)
	UART Rx V_{IL}	1.1 V max
	UART Tx V_{OH} (8mA load)	2.9V min
	UART Tx V_{OL} (8ma load)	0.4V max
Current Draw	AHRS Mode@ max. sample rate	20 mA typical
	Compass Mode @ max. sample rate	17 mA typical
	During application of external power	See Figure 3-1
	Sleep Mode	0.5 mA typical

(1) CMOS 3.3V and TTL-compliant logic levels. UART RX V_{IH} is tested at 2.3V min. and is guarenteed by design to be 1.87V min

(2) Vin on UART Rx is rated at 4V MAX. Higher than 3.6V can cause excessive loading on source.

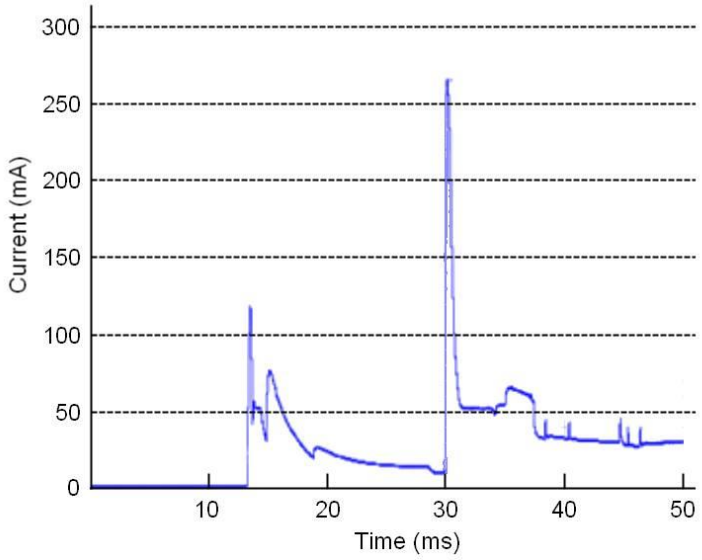


Figure 3-1: Typical Current Drawing During Application of External Power

Table 3-4: I/O Characteristics

Parameter	Value
Communication Interface	TTL serial UART
Communication Protocol	PNI Binary
Communication Rate[1]	2400 to 921,600 baud
Maximum Data Output Rate	≈30 samples/sec

Footnote:

1. The TPTCM can operate up to 921,600 baud, but native TTL is limited to 115,200 baud.

Table 3-5: Environmental Requirements

Parameter	Value
Operating Temperature[1]	-40C to +85C
Storage Temperature	-40C to +85C

Footnote:

1. To meet performance specifications across this range, recalibration will be necessary as the temperature varies.

Table 3-6: Mechanical Characteristics

Parameter	Value
Dimensions (l x w x h)	3.3 x 3.1 x 1.38 cm
Weight	15 gm
Connector	4-pin Molex, part number 53047-0410 Mating Molex connector part number 51021-0400

3.2 Mechanical Drawings

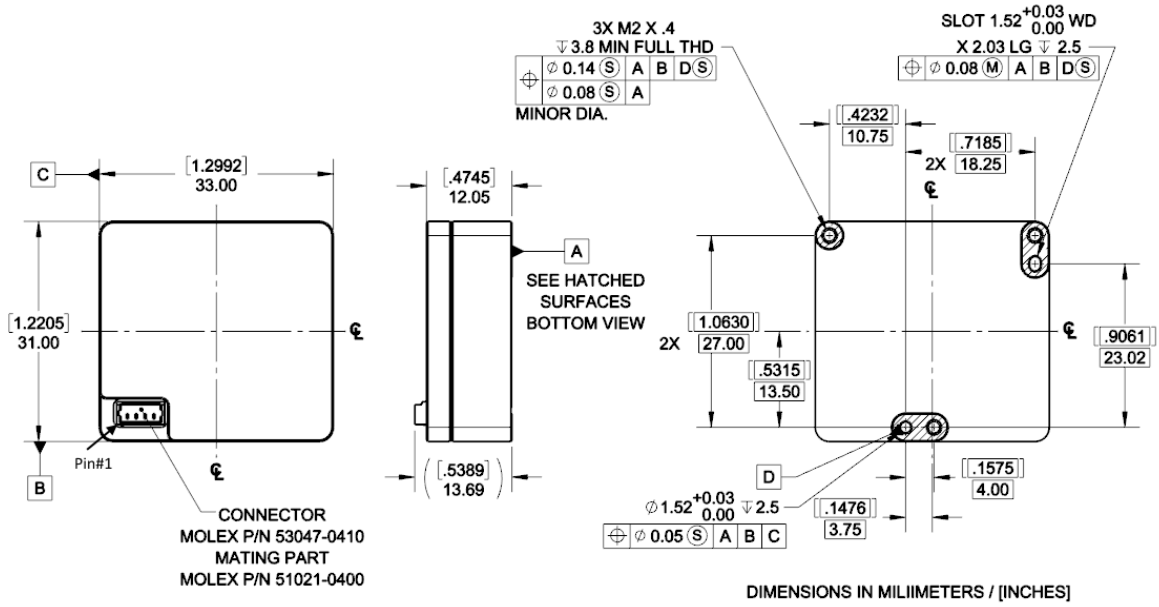


Figure 3-2: TPTCM Mechanical Drawing

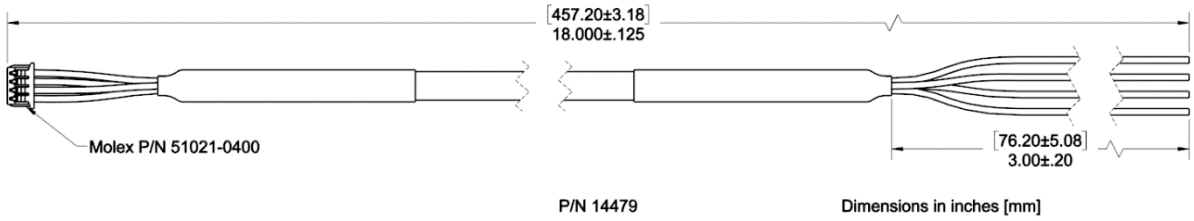


Figure 3-3: Molex-to-Pigtail Cable Drawing, pn 14479

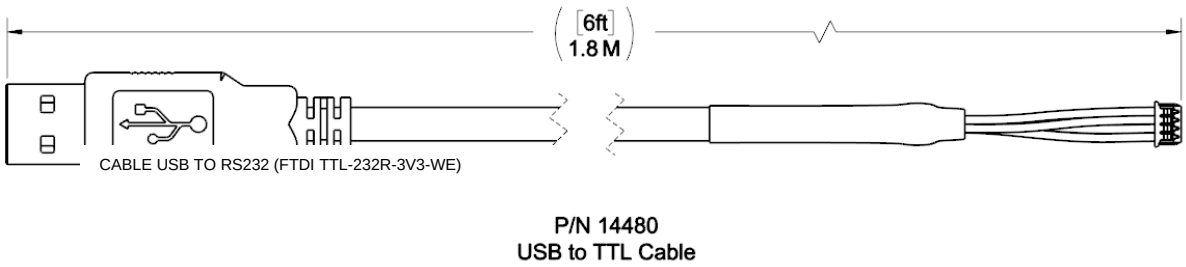


Figure 3-4: Molex-to-USB Cable Drawing, pn 14480

4 Set-Up and Evaluation

This section describes how to configure the TPTCM in your host system. To install the TPTCM into your system, follow these steps:

- Make electrical connections to the TPTCM.
- Evaluate the TPTCM using the TRAX Studio program, or a binary terminal emulation program, such as RealTerm or Tera Term, to ensure the TPTCM is working properly.
- Choose a mounting location.
- Mechanically mount the TPTCM in the host system.
- Perform a user calibration.

4.1 Electrical Connections

The TPTCM incorporates a 4 pin Molex connector, part number 53047-0410, which mates with Molex part 51021-0410 or equivalent. The pin-out for both is given below in Table 4-1.

Table 4-1: TPTCM Pin Descriptions

Pin #[1]	TPTCM		
	4 Pin Molex Connector	USB Demo Cable Wire Color	Pigtailed Cable
1	Ground	Black	Black
2	+3.3-8V Power	Red	Red
3	TTL UART Tx	Green	Green
4	TTL UART Rx	White	White

Footnote:

1. Pin #1 is the left most position of the connector as indicated on Figure 3-2

After making the electrical connections, it is a good idea to perform some simple tests to ensure the TPTCM is working as expected. See Section 6 for how to operate the TPTCM with TRAX Studio or Section 7 for how to operate the TPTCM using PNI's binary protocol.

4.2 Installation Location

The TPTCM's wide dynamic range and sophisticated algorithms allow it to operate in many environments. For optimal performance however, you should mount the TPTCM with the following considerations in mind:

4.2.1 Operate within the TPTCM's dynamic range

The TPTCM can be user calibrated to correct for static magnetic fields created by the host system. However, each axis of the TPTCM has a calibrated dynamic range of ±150 µT. If the total field exceeds this value for any axis, the TPTCM may not perform to specification. When mounting the TPTCM, consider the effect of any sources of magnetic fields in the host environment that, when added to Earth's field, may take the TPTCM out of its dynamic range. For example, large masses of ferrous metals such as transformers and vehicle chassis, large electric currents, permanent magnets such as electric motors, and so on.

4.2.2 Locate away from changing magnetic fields

Keep the TPTCM away from sources of local magnetic distortion that knowingly will change with time, such as electrical equipment that will be turned on and off, or ferrous bodies that will move.

4.2.3 Mount in a physically stable location

Choose a location that is isolated from persistent vibration or other dynamic motion. Accurate measurement of gravity vector is required for accurate heading but vibration and acceleration will induce errors in the gravity measurement by the accelerometer sensor on the TPTCM.

4.2.4 Choose Non-magnetic Mounting Pins

Use non-magnetic mounting pins. There are 3 mounting holes at the bottom of TPTCM, brass, copper or aluminum pins should be used to hold it in place to eliminate magnetic distortion in close proximity.

4.2.5 Keep Proper Distance

Keeping good distance from known magnetic interference such as battery, magnet and ferrous objects can reduce magnetic distortion significantly. Magnetic fields diminish mathematically as 1/distance ^{3}, for 2 cm, the effect is 1/8.

4.3 Mechanical Mounting

The TPTCM is factory calibrated with respect to its mounting holes. It must be aligned within the host system with respect to these mounting holes. Ensure any stand-offs or screws used to mount the TPTCM are non-magnetic. Refer to Section 3.2 for dimensions, hole locations, and the reference frame orientation.

4.3.1 Pitch and Roll Convention

The TPTCM utilizes Euler angles as the primary method for providing orientation data, although quaternions outputs also are available. The Euler angles are the common method used for aircraft orientation, where the outputs are heading, pitch and roll. When using Euler angles in aviation, roll is defined as the angle rotated around an axis through the center of the fuselage, while pitch is rotation around an axis through the center of the wings. These rotations are dependent on each other since the axes of rotation move with the plane.

As shown in Figure 4-1, for the TPTCM a positive pitch is when the front edge of the board is rotated upward, and a positive roll is when the right edge of the board is rotated downward. The order of rotation is given as heading, pitch, and then roll.

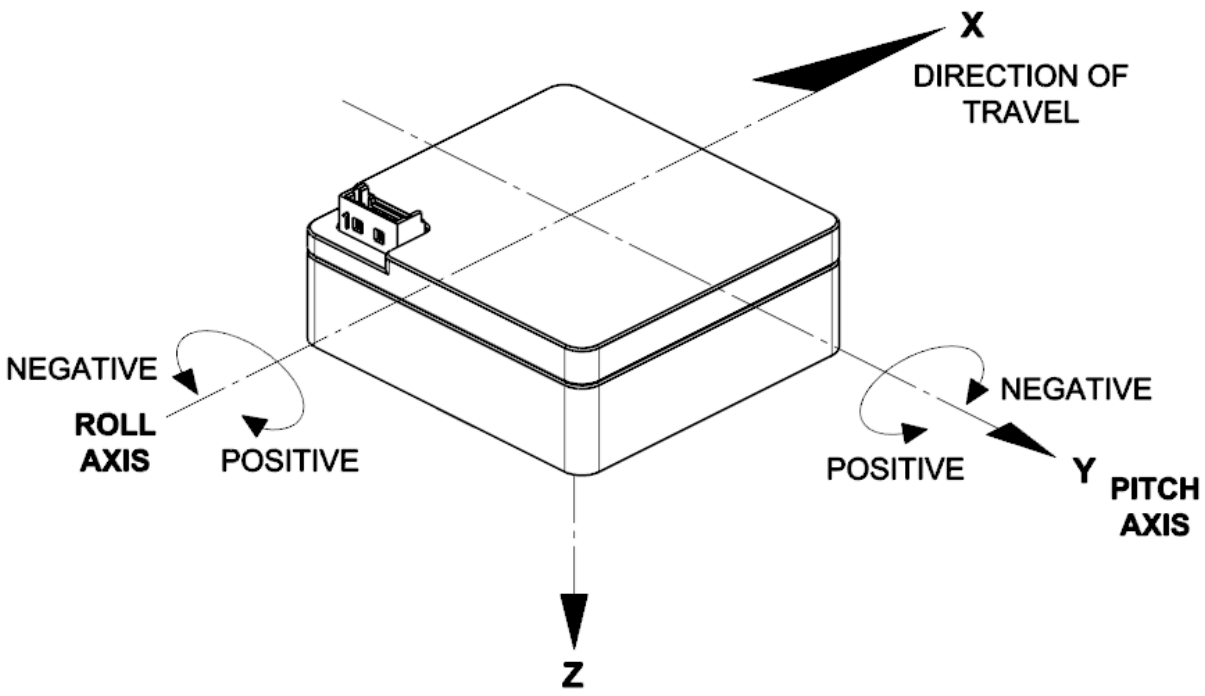


Figure 4-1: Positive & Negative Roll and Pitch Definition

4.3.2 Coordinate System

The TPTCM utilizes North East Down (NED) coordinate system (frame) to define X, Y and Z axes as shown in Figure 4-1.

4.3.3 Mounting Orientation

The TPTCM can be mounted in 16 different orientations, as shown in Figure 4-2. All reference points are based on the default orientation shown in Figure 4-1. The orientation should be programmed in the TPTCM using the Configuration Tab in TRAX Studio or using the kSetConfig command and the kMountingRef setting in the PNI Protocol, as described in Section 7.4.2. The default orientation is "STD 0°".

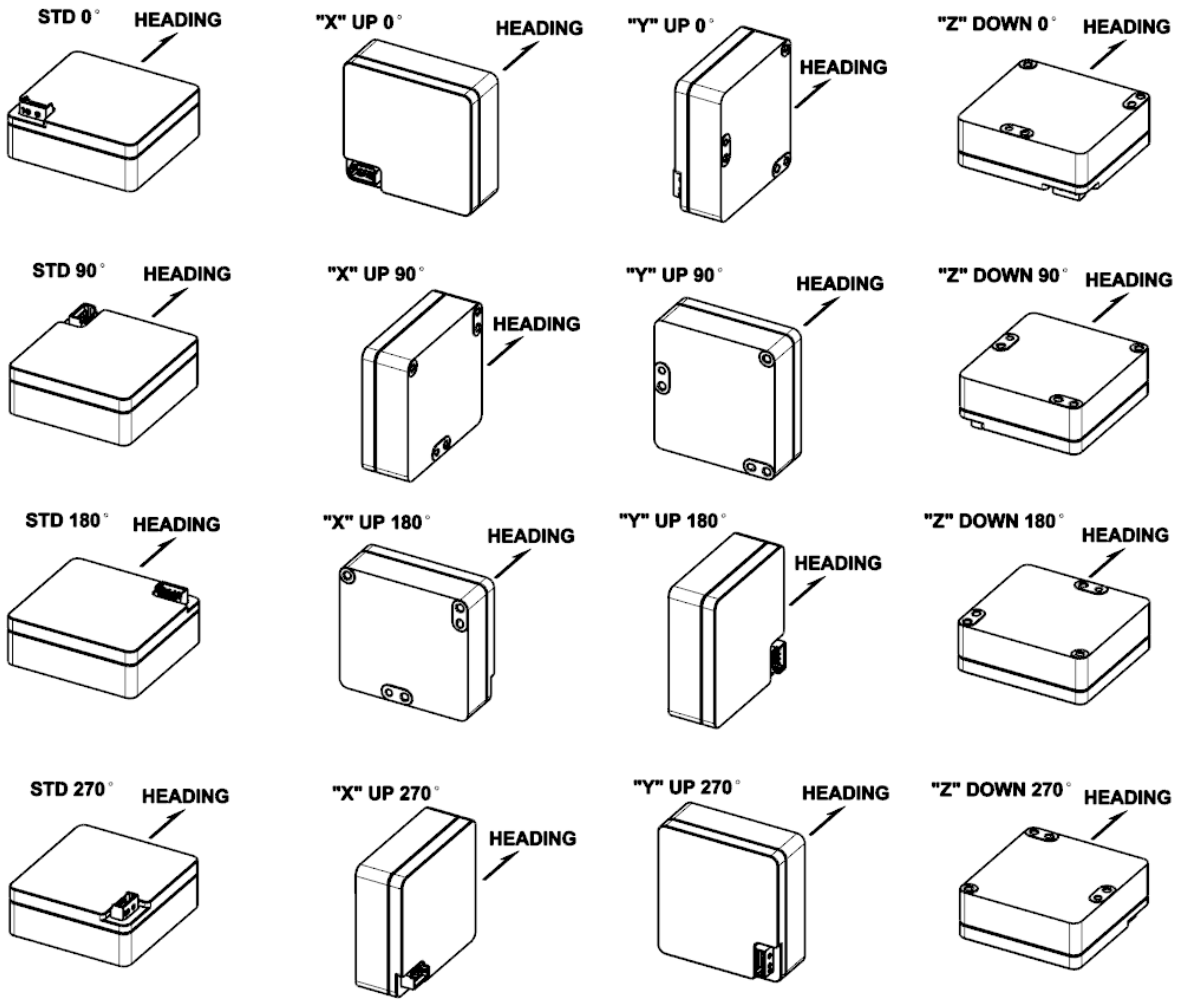


Figure 4-2: TPTCM Enclosed Mounting Orientations

5 User Calibration

The magnetic sensor in the TPTCM is calibrated at PNI's factory in a magnetically controlled environment. However, sources of magnetic distortion positioned near the TPTCM in the user's system will distort Earth's magnetic field and should be compensated for in the host system with a user calibration. Examples of such sources include ferrous metals and alloys (ex. iron, nickel, steel, etc.), batteries, audio speakers, current-carrying wires, and electric motors. Compensation is accomplished by mounting the TPTCM in the host system and performing a user calibration. It is expected the sources of magnetic distortion remain fixed relative to the TPTCM's position within the host system. By performing a calibration, the TPTCM identifies the local sources of magnetic distortion and negates their effects from the overall reading to provide an accurate heading.

Key Points:

- Magnetic calibration:
  - Requires incorporating the TPTCM into the user's host system such that the magnetic components of the user's system can be compensated for.
  - Allows for 4 different methods of calibration. Full-Range Calibration provides the highest heading accuracy but requires ≥30° of pitch. 2D and Limited-Tilt Calibration allow for good calibration when the range of allowable motion is limited. Hard-Iron-Only Calibration updates the hard-iron compensation coefficients with a relatively easy procedure.
- If the TPTCM will experience different states during operation, such as operating with a nearby shutter sometimes closed and sometimes open, or operating over a broad temperature range, then different sets of calibration coefficients can be saved for the various states. Up to 8 magnetic calibration coefficient sets can be saved.

As with the magnetic sensor, the accelerometer in the TPTCM is calibrated at PNI's factory. In the unlikely event the accelerometer drifts the user can perform a user accelerometer calibration or return the unit to PNI for recalibration. Please refer to the Accelerometer Field Calibration Application Note for further information.

5.1 Magnetic Calibration

Two fundamental types of magnetic distortion exist: hard-iron and soft-iron. These are discussed in the following paragraphs, plus a discussion on how temperature also affects magnetic fields and other considerations. For more information on magnetic distortion and

calibration, see PNI's white paper "Local Magnetic Distortion Effects on 3-Axis Compassing" at PNI's website (http://www.pnicorp.com/technology/papers).

Hard-Iron Effects

Hard-iron distortions are caused by permanent magnets and magnetized objects in close proximity to the sensors. These distortions add or subtract a fixed component to each axis of the magnetic field reading. Hard-iron distortions usually are unchanging and in a constant location relative to the sensors, for all heading orientations.

Soft-Iron Effects

Magnetically "soft" materials effectively bend the magnetic field near them. These materials have a high magnetic permeability, meaning they easily serve as a path for magnetic field lines. Unlike hard-iron effects, soft-iron effects do not increase or decrease the total field in the area. However, the effect of the soft-iron distortion changes as the host system's orientation changes. Because of this, it is more difficult to compensate for soft-iron materials.

Temperature Effects

While the hard-iron and soft-iron distortion of a system may remain quite stable over time, normally the distortion signature will change over temperature. As a general rule, the hard-iron component will change 1% per 10°C temperature change. Exactly how this affects heading depends on several factors, most notably the hard-iron component of the system and the inclination, or dip angle.

Consider the example of a host system with a 100 µT hard-iron component. This is a fairly large hard-iron component, but not completely uncommon. A 10°C temperature change will alter the magnetic field by ~1 µT in the direction of the hard-iron component. Around San Francisco, with an inclination of ~60°, this results in up to a couple of degrees of heading change over 10°C.

Consequently, no matter how stable a compass is over temperature, it is wise to recalibrate over temperature since the magnetic signature of the host system will change over temperature. The TPTCM helps accommodate this issue by allowing the user to save up to 8 sets of magnetic calibration coefficient sets, so different calibration coefficients can be generated and loaded at different temperatures.

Other Considerations

TPTCM can store up to 8 different sets of magnetic calibration coefficients, hence if the system magnetic signature will change in different known unique states, perform a user calibration in these unique states and save each of the calibration coefficients. The unique calibration coefficients can be recalled when the system is operated in that specific state again.

The main objective of a magnetic user calibration is to compensate for hard-iron and soft-iron distortions to the magnetic field caused by components within the user's host system. To that end, the TPTCM needs to be mounted within the host system and the entire host system needs to be moved as a single unit during a user calibration. The TPTCM allows the user to perform a calibration only in a 2D plane or with limited tilt but provides the greatest accuracy if the user can rotate through 360° of heading and at least ±30°of tilt.

The following subsections provide instructions for performing a magnetic calibration of a TPTCM system. Several calibration mode options exist, as summarized in Table 5-1. To meet the accuracy specification, the number of samples should be the "Minimum Recommended" value, or greater. Calibration may be performed using TRAX Studio or using the PNI binary protocol, and up to 8 sets of magnetic calibration coefficients may be saved. The recommended calibration patterns described in the following sub-sections provide a good distribution of sample points.

Table 5-1: Magnetic Calibration Mode Summary

Calibration Mode	Static Accuracy in Compass Mode	Pitch Range during Cal	Minimum Recommended # of Samples	Allowable Range of # of Samples[1]
Full-Range	0.25° rms	>±30°	12	10 – 32
2D Calibration	<2°	<±5°	12	10 – 32
Limited-Tilt	<2° over 2x tilt range	±5° to ±30°	12	10 – 32
Hard-Iron-Only	Restores prior accuracy	>±30°	6	4 - 32

Footnote:

1. Maximum number of sample point is 32 in TPTCM. When TRAX studio is used to evaluate, the studio has limited sample point up to 18.

Before proceeding with a calibration, ensure the TPTCM is properly installed in the host system. The device should be installed as discussed in Section 4, and the software should be properly configured with respect to the mounting orientation, Endianness, north reference, etc.

Section 6.5 outlines how to perform a calibration using TPTCM Studio, while Section 7.6.2 provides a step-by-step example of how to perform a calibration using the PNI protocol.

5.1.1 Full-Range Calibration

A Full-Range Calibration is appropriate when the system with the TPTCM installed can be tilted ±30° or more. This method compensates for hard and soft-iron effects in three dimensions, and allows for the highest accuracy readings. The recommended 12-point calibration pattern is a series of 2 circles of evenly spaced points, as illustrated in **Error! Reference source not found.** as the top view, Figure 5-1 as the realistic view and listed in Table 5-2. The pitch used in the two circles of the calibration should at least match the maximum and minimum pitch the device is expected to encounter in use.

Table 5-2: 12 Point Full-Range Calibration Pattern

Sample #[4]	Heading[1]	Pitch[2]	Roll[3]
First Circle			
1	0°	≥ +30°	0°
2	60°	≥ +30°	0°
3	120°	≥ +30°	0°
4	180°	≥ +30°	0°
5	240°	≥ +30°	0°
6	300°	≥ +30°	0°
Second Circle			
7	0°	≤ -30°	0°
8	60°	≤ -30°	0°
9	120°	≤ -30°	0°
10	180°	≤ -30°	0°
11	240°	≤ -30°	0°
12	240°	≤ -30°	0°

Footnote:

2. Heading listings are not absolute heading directions, but rather relative heading referenced to the first sample.

3. Pitch 30° or more is recommended; acceptable range is between ±10° to ±60°.

4. Roll is not required.

5. The sequence order of sample points is not required. The 12 points can be sampled in any order.

Figure 5-1: 12 Point Full-Range Calibration (Realistic View)

Notes

- *The location of the device changing in realist view is for illustration purpose. If possible, it is best to rotate around the center of the device as a pivot point.*
- *It is not necessary for the unit to be pointing North to start; however, it may provide better results, especially in high latitudes.*

5.1.2 2D Calibration

A 2D Calibration is intended for applications with very low tilt operation (<5°) and where calibrating the TPTCM with greater tilt is not practical.

This procedure calibrates for hard and soft-iron effects in only two dimensions, and in general is effective for operation and calibration in the tilt range of -5° to +5°. The recommended 12-point calibration pattern is a circle of evenly spaced points, as given in Table 5-3.

Table 5-3: 12 Point 2D Calibration Pattern

Sample #	Heading	Pitch[1,2]	Roll[2]
1	0°	0°	0°
2	30°	max. negative	0°
3	60°	0°	0°
4	90°	max. positive	0°
5	120°	0°	0°
6	150°	max. negative	0°
7	180°	0°	0°
8	210°	max. positive	0°
9	240°	0°	0°
10	270°	max. negative	0°
11	300°	0°	0°
12	330°	max. positive	0°

Footnote:

1. For best results, the pitch experienced during calibration should match that experienced in service. For example, if the TPTCM is restrained to a level plane in service, then calibration should be in a plane, where “max. positive” and “max. negative” are 0°.

2. If the device frame has a fixed tilt, the 2D calibration still works, when the fixed tilt is constant through the 2D rotation. The device frame fixed tilt can be in the range of -30° to +30°.

5.1.3 Limited-Tilt Calibration

A Limited-Tilt Calibration is recommended when 30° of pitch isn’t feasible, but >5° of pitch is possible. It provides both hard-iron and soft-iron distortion correction. The recommended 12-point calibration pattern given below is a series of 3 circles of evenly spaced points, with as much tilt variation as expected during use.

Table 5-4: 12 Point Limited-Tilt Calibration Pattern

Sample #	Yaw	Pitch	Roll
First Circle			
1	0°	0°	0°
2	90°	0°	0°
3	180°	0°	0°
4	270°	0°	0°
Second Circle			
5	45°	> +5°	0°
6	135°	> +5°	0°
7	225°	> +5°	0°
8	315°	> +5°	0°
Third Circle			
9	45°	< -5°	0°
10	135°	< -5°	0°
11	225°	< -5°	0°
12	315°	< -5°	0°

Alternatively, a similar and acceptable pattern would be to follow the recommended 12 point Full-Range Calibration pattern but substituting the >±30° of pitch with whatever pitch can be achieved. See Section 5.1.1 for more information.

Table 5-5: 12 Point Limited-Tilt Calibration Alternative Pattern

Sample #	Heading	Pitch	Roll
First Circle			
1	0°	> +5°	0°
2	60°	> +5°	0°
3	120°	> +5°	0°
4	180°	> +5°	0°
5	240°	> +5°	0°
6	300°	> +5°	0°
Second Circle			
7	0°	< -5°	0°
8	60°	< -5°	0°
9	120°	< -5°	0°
10	180°	< -5°	0°
11	240°	< -5°	0°
12	300	< -5°	0°

5.1.4 Hard-Iron-Only Calibration

It is not uncommon for the hard-iron magnetic distortions around the TPTCM to change. Some reasons for this include significant temperature change or temperature shock to a system, as well as gradual aging of components. A Hard-Iron-Only Calibration allows for quick recalibration of the TPTCM for hard-iron effects, and generally is effective for operation. Calibration in the pitch range of ≥30° is preferred, as low as 3° is possible but will result in loss of accuracy. The recommended 6-point calibration pattern given below is a circle of alternately tilted, evenly spaced points, with as much tilt as expected during use.

Table 5-6: 6 Point Hard-Iron-Only Calibration Pattern

Sample #	Heading	Pitch[1]	Roll[1]
1	0°	max. negative	0°
2	60°	max. positive	0°
3	120°	max. negative	0°
4	180°	max. positive	0°
5	240°	max. negative	0°
6	300°	max. positive	0°

Footnote:

1. For best results, the tilt experienced during calibration should match that experienced in service. For example, if the TPTCM will be subject to ±30° of pitch when in service, then "max negative" should be -30° and "max positive" should be +30°.

5.2 Accelerometer Calibration

The TPTCM uses a MEMS accelerometer to measure attitude. This data is output as pitch and roll data. The accelerometer data is critical for establishing an accurate heading reading when the TPTCM is tilted, as discussed in the PNI white paper "Tilt-Induced Heading Error in a 2-Axis Compass", which can be found on PNI's web site: (http://www.pnicorp.com/technology/papers).

The TPTCM algorithms assume the accelerometer only measures the gravitational field. If the TPTCM is accelerating during a measurement, this will result in the TPTCM calculating an inaccurate gravitational vector, which in turn will result in an inaccurate heading reading. For this reason, the TPTCM should be stationary when taking a measurement.

PNI calibrates the accelerometer in its factory prior to shipment. In the unfortunate event that the bias and offset of the accelerometer drifts, refer to the "Accelerometer Field Calibration application note" for instructions on how to calibrate in the field or return it to PNI for recalibration.

6 Operation with TRAX Studio

The TRAX Studio program puts an easy-to-use, graphical-user interface (GUI) onto the binary command language used by the TPTCM. TRAX Studio is intended for evaluating, demonstrating, and calibrating both the TRAX and TPTCM. Among other features, the program can log and save the outputs from the TPTCM to a file for off-line evaluation. Anything that can be done using TRAX Studio also can be done using PNI's binary protocol, as discussed in Section 7. Please note that the TRAX Studio has an option to switch between Compass and AHRS mode.

NOTE: TRAX Studio works with the TPTCM module configured to Big Endian. This is the default TPTCM setting, but if the module was set to Little Endian, reset to Big Endian per section 7.4.2

6.1 Installation

TRAX Studio is provided as an executable program which can be downloaded from PNI's website. It will work with Windows 7, and Windows 10 operating systems. Check the PNI web page at www.pnicorp.com for the latest version.

Copy the "TRAXStudio.msi" file onto your computer. Double-click on the icon and step through the Setup Wizard. The program will be installed into the following directory unless you direct it otherwise: Program Files\PNI Sensor Corporation\TRAX Studio\. A "TRAX Studio" shortcut icon will be placed on your computer's desktop.

Now plug the TPTCM into the USB or Serial port of your computer. Since the ***TPTCM interface is TTL, in order to use the TRAX Studio a USB to TTL cable as shown in Figure 3-4 is required to interface with the TRAX Studio.*** **If using it in AHRS mode, ensure the TPTCM is completely still after it has been plugged into the USB port as the gyros initialize during the first five seconds after being plugged in.**

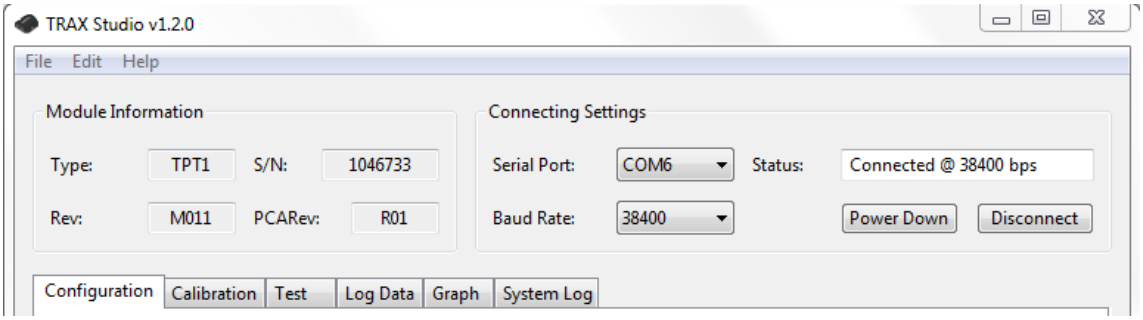
If you are using Windows 7 or later version the computer automatically searches for and installs the driver. If there is a problem with this, download the driver from FTDI's website at http://www.ftdichip.com/Drivers/VCP.htm.

You have now completed the installation of TRAX Studio.

6.2 TRAX Studio Header and Connecting to TRAX Studio

If the TPTCM is not already plugged into the computer, then do so.

Double-click on the TRAX Studio icon to launch TRAX Studio. Below is a picture of the TRAX Studio header. The header includes "Module Information" and "Connection Settings", and under the header are the various TRAX Studio tabs, which are discussed in the subsequent sections. The header is the same regardless of which tab is selected.



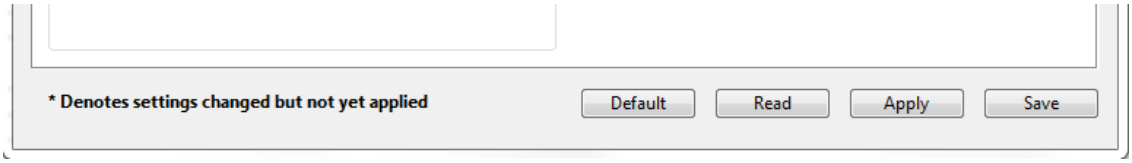
To connect, set the "Serial" field to the appropriate COM port. To determine this, in Windows right click on "My Computer", select "Manage", then select "Device Manager". Expand "Ports" and note the port for "USB Serial Port". This should be the assigned COM port for the TPTCM.

Set the baud rate. The default is 38400, and this should be the setting the first time the TPTCM is used. Once a connection has been established, the desired baud rate can be changed on the Configuration Tab, as discussed in Section 6.4.1.

Click the <Connect> button. When the connection is established, the light to the left of "Status" will turn green, "Status" will change to "Connected @ [the baud rate]", the <Connect> button will now say <Disconnect>, the <Power Down> button will be live, and the "Device Information" section of the header will populate.

6.3 TRAX Studio Footer and Saving/Applying Settings

Below is a picture of the TRAX Studio footer. The footer includes 4 buttons which are discussed below. The footer is the same regardless of which tab is selected. Note that if a field is changed on any of the TRAX Studio tabs, then either <Apply> or <Save> must be selected for the change to take effect.



Save

Clicking the <Save> button will save any changes made in TRAX Studio to the TPTCM's onboard non-volatile and volatile memory.

Apply

Selecting the <Apply> button will apply any changes to the TPTCM's volatile memory, but not to the non-volatile memory. If the TPTCM is powered off/on, the settings will revert to whatever was last saved in the non-volatile memory.

Read

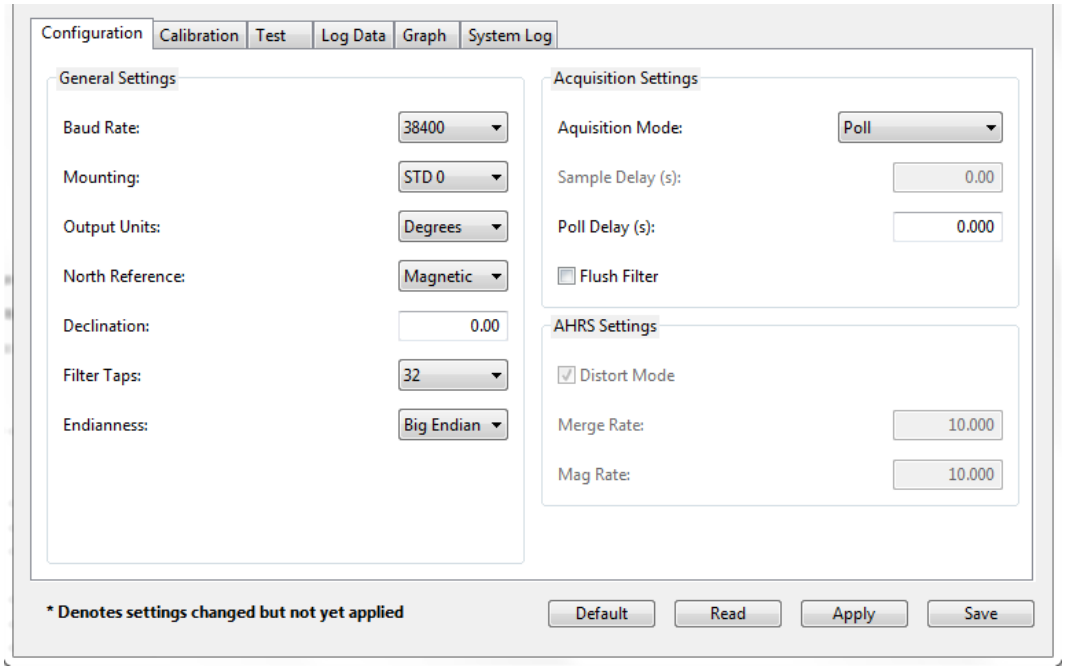
Clicking the <Read> button will read the current settings in the TPTCM's volatile memory and display them. This will change any item in bold, which indicates a setting was changed in TRAX Studio but not applied to the TPTCM's volatile memory, back to regular type and the setting change will not be applied to the TPTCM's memory.

Default

Selecting the <Default> button will display the default settings, which are stored in TRAX Studio. Selecting this button will not automatically apply these values, and either <Apply> or <Save> must be selected for the read values to be applied.

6.4 Configuration Tab

The Configuration Tab is shown below and its contents discussed in the following subsections. For any changes to take effect, the <Save> button must be selected. Clicking on the <Default> button will load the factory default values, although these must be saved to take effect. Clicking the <Retrieve> button will show the current settings of the device.



6.4.1 General Settings

Baud Rate

The baud rate can be altered by selecting the desired baud rate from the pull down menu, clicking on "Save", and then powering the device off and back on. The new baud rate will not take effect until the device has been powered off/on. The default baud rate is 38400.

Mounting

TRAX Studio supports 16 mounting orientations, as previously illustrated in Figure 4-2. The default is "Standard".

Output Units

The TPTCM can output heading, pitch, and roll in either degrees or mils. The default is "Degrees". (There are 6400 mils in a circle, such that 1 degree = 17.7778 mils and 1 mil = 0.05625 degree.)

North Reference

When "Magnetic" is selected, the heading output will be relative to magnetic north. When "True" is selected, heading will be relative to true north, and the declination value should be entered in the next field. The default is "Magnetic".

Declination

The declination represents the heading difference between magnetic north and true north, and needs to be entered if "True" is selected as the "North Reference". Declination varies primarily with location, although it also gradually changes over time (years) for a given location. To find the declination for where the TPTCM will be used, go to http://www.ngdc.noaa.gov/geomagmodels/Declination.jsp.

Filter Taps

When operating the TPTCM in Compass Mode, the TPTCM will use a finite impulse response (FIR) filter to effectively provide a stable heading reading. When operating in AHRS Mode a completely different filtering method is used and the number of FIR Taps is not relevant.

The number of taps (or samples) represents the amount of filtering to be performed. Either 0, 4, 8, 16, or 32 taps may be selected, with zero taps representing no filtering. Note that selecting a larger number of taps can significantly slow the time for the initial sample reading and, if "Flush Filters" is selected, the rate at which data is output. The default setting is 32.

The FIR filtering operates on a first-in, first-out (FIFO) basis. For example, with FIR Taps set to 32, once the initial 32 samples are taken the next sample (#33) is included in the filtering and the initial sample (#1) is dropped from the filtering.

Flush Filter

The Flush Filter setting is only relevant when in Compass Mode, as it affects how the FIR filters are implemented. Flushing the FIR filter clears all the filter values so it is necessary to fully repopulate the filter before a good reading can be given. For example, if 32 FIR taps is set, then 32 new samples must be taken to provide a good reading. It is particularly prudent to flush the filter if the Sampling Delay is set to a non-zero value as this will purge old data. Note that flushing the filters increases the delay until data is output, with the length of the delay being directly correlated to the number of FIR taps. The default is not to Flush Filters.

6.4.2 Acquisition Settings

Acquisition Mode

When operating in Continuous Acquisition Mode, the TPTCM continuously outputs data to the host system. The rate is set by the Sample Delay. When operating in Polled Mode, TRAX Studio simulates a host system and polls the TPTCM for a single measurement; but TRAX Studio makes this request at a fixed rate which is set by the Polling Delay. In both cases data is continuously output, but in Continuous Acquisition Mode the TPTCM controls the data rate while in Polled Mode the TRAX Studio program controls the data rate. Polled Mode is the default.

Sample Delay

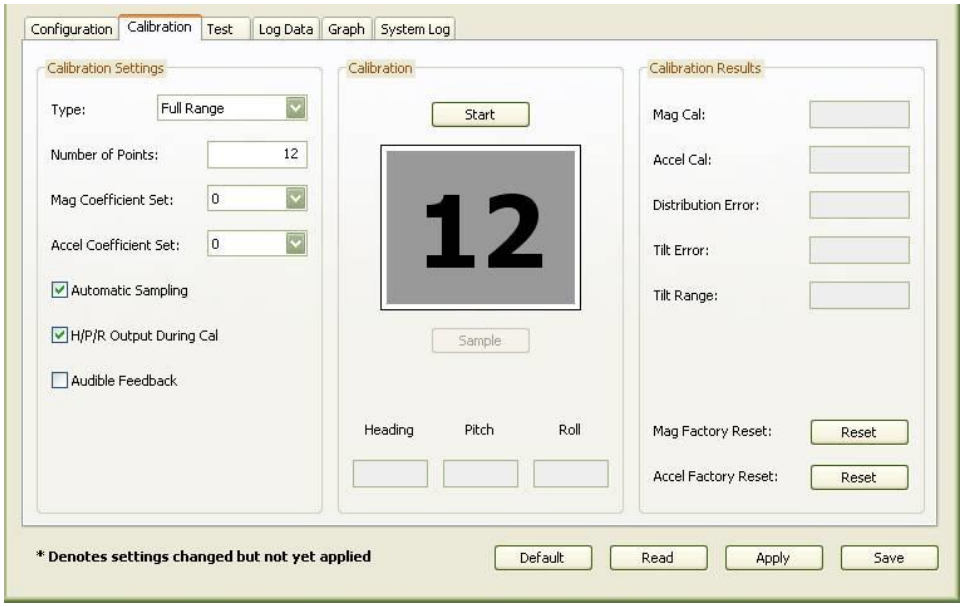
The Sample Delay is relevant when Continuous Mode is selected. It is the time delay, in seconds, between completion of TPTCM sending one set of data and the start of sending the next sample set. If the delay is set to 0, then TPTCM will begin sending new data as soon as the previous data set has been sent. Note that the inverse of the Sample Delay is greater than the sample rate, since the Sample Delay does not include the actual measurement acquisition time. The default is 0.

Polling Delay

The Polling Delay is relevant when Polled Mode is selected. It represents the time delay, in seconds, between the completion of TRAX Studio receiving one set of sampled data and requesting the next sample set. If the delay is set to 0, then TRAX Studio requests new data as soon as the previous request is fulfilled. Note that the inverse of the Polling Delay is greater than the sample rate, since the Polling Delay does not include the actual measurement acquisition time. The default is 0.

6.5 Calibration and the Calibration Tab

As discussed in Section 5, a user calibration should be performed to optimize TPTCM performance. The Calibration Tab is shown below, and following this are discussions of how to set the calibration settings, how to perform a calibration, and how to interpret the results.



6.5.1 Calibration Settings

Calibration Type

The "Calibration Type" pull down menu establishes the type of calibration to be performed. The options are Full, 2D, Hard-Iron-Only, Limited-Tilt, and Accelerometer. These are briefly discussed below and in more detail in Section 5.

Full Range – recommended calibration method when ≥30° of tilt is possible.

2D – recommended when the available tilt range is limited to ≤5°.

Hard Iron Only – serves as a hard-iron recalibration to a prior calibration. If the hard-iron distortion around the device changes, this procedure can bring the device back into specification more quickly than other methods.

Limited Tilt Range – recommended calibration method when >5° of tilt calibration is available, but tilt is restricted to <30°. (i.e. full range calibration is not possible.)

Accelerometer – The user should select this when accelerometer calibration will be performed. Accelerometer calibration is recommended every 6 to 12 months to

compensate for bias drift in the accelerometer. The TPTCM can also be returned to PNI for accelerometer calibration.

Number of Points

This establishes how many samples will be taken during the calibration. The minimum and recommended number of samples depends on the calibration method and is summarized in The following subsections provide instructions for performing a magnetic calibration of a TPTCM system. Several calibration mode options exist, as summarized in Table 5-1. To meet the accuracy specification, the number of samples should be the "Minimum Recommended" value, or greater. Calibration may be performed using TRAX Studio or using the PNI binary protocol, and up to 8 sets of magnetic calibration coefficients may be saved. The recommended calibration patterns described in the following sub-sections provide a good distribution of sample points.

Table 5-1. The maximum number of samples is 32. The evaluation program TRAX Studio limits number of sample points to 18.

Mag Coefficient Set & Accel Coefficient Set

At any one time, the TPTCM will use one set of magnetic calibration coefficients and one set of accelerometer calibration coefficients. The magnetic coefficients compensate for measured magnetic distortions in the host system as determined during a magnetic calibration. The accelerometer coefficients compensate for bias and offset of the accelerometers, as determined during an accelerometer calibration.

However, the TPTCM can store eight (8) sets of magnetic calibration coefficients and eight (8) sets of accelerometer calibration coefficients. This feature is useful if the compass will be placed in multiple locations that have different local magnetic field properties. The default is index number 0 and initially this is populated at PNI with factory-generated coefficients for the device itself. The other sets initially are unpopulated.

To store a coefficient set, first select the index number (0 to 7), then perform a calibration. The coefficient values will be stored in the defined index number, assuming the <Save> is selected after the calibration. To recall and use a different set of coefficients, change the "Mag Coefficient Set" and/or "Accel Coefficient Set" number, then click the <Save> button.

Automatic Sampling

If selected, the TPTCM will take a sample point once predefined minimum change and stability requirements have been satisfied. If the minimum change and stability criteria are not met in 5 seconds as a timeout period, TPTCM will take a sample at the end of

the timeout. If the user wants to have more control over when the point will be taken then Auto Sampling should be deselected.

H/P/R Output During Cal

When selected, the heading, pitch, and roll of the device will be output below the sample number during a calibration. Using this feature, the user can monitor the device's orientation to easily follow the appropriate recommended calibration pattern. Since a calibration is being performed, these values are relative and should not be considered accurate.

Audible Feedback

If selected, TRAX Studio will give an audible signal when a calibration sample is taken.

6.5.2 Performing a Calibration

Before proceeding, ensure you are familiar with the recommended calibration pattern corresponding to the "Type" selected. These are discussed in Section 5.

To perform a calibration, follow the following steps:

- Click the <Start> button to begin the calibration process.
- If "Automatic Sampling" is not checked the first sample will be taken automatically assuming the TPTCM is relatively stationary. After this, it is necessary to click the <Take Sample> button to take a calibration sample point. This should be repeated until the total number of samples is taken, changing the orientation of the device between samples as discussed in Section 0.
- If "Automatic Sampling" is checked the TPTCM needs to be held steady for a short time and then a sample automatically will be taken. Once the window indicates the next number, the device's orientation should be changed and held steady for the next sample. Once the pre-set number of samples has been taken (as set on the Configuration tab) the calibration is complete.

Regardless of whether "Automatic Sampling" is selected, two criteria must be met for a calibration sample to be taken. First, the TPTCM must be held steady enough to meet PNI-defined stability criteria. Second, the TPTCM's orientation must have changed enough to meet PNI-defined orientation change criteria.

If "H/P/R Output During Cal" is selected, then the "Heading", "Pitch", and "Roll" fields will be populated in real-time once the <Start> button is selected. Note that the readings in these fields are relative, since calibration is in process.

6.5.3 Calibration Results

Once the calibration is complete the "Calibration Results" section will indicate the quality of the calibration. This may take a few seconds to populate. The primary score of concern is the MagCalScore. The other parameters provide information that may assist in improving the MagCalScore should it be unacceptably high. If a calibration is acceptable, then click the <Save> button to save the calibration coefficients to the coefficient set defined on the Configuration Tab.

Note: If a calibration is aborted, all the scores will read "179.80", and the calibration coefficients will not be changed. (Clicking the <Save> button will not change the calibration coefficients.)

Mag Cal

Represents the over-riding indicator of the quality of a magnetic calibration. Acceptable scores are <1 for Full-Range Calibration and <2 for other methods. Note that it is possible to obtain acceptable Distribution Error and Tilt Error scores and still have a rather high Mag Cal value. The most likely reason for this is the TPTCM is close to a source of magnetic distortion that is not fixed with respect to the device.

Distribution Error

Indicates if the distribution of sample points is good, with an emphasis on the heading distribution. The score should be 0. Significant clumping or a lack of sample points in a particular section can result in a poor score.

Tilt Error

Indicates if the TPTCM experienced sufficient tilt during the calibration, taking into account the calibration method. The score should be 0.

Tilt Range

This reports the half of the full pitch range of all sample points. For example, if the TPTCM is pitched +25º to -15º, the Tilt Range value would be 20º, as derived from [+25º - {-15º}]/2. For Full-Range Calibration and Hard-Iron-Only Calibration, this should be ≥30°. For 2D Calibration, this ideally would be ~2°. For Limited-Tilt Calibration the value should be as large as possible given the user's constraints.

Mag Factory Reset & Accel Factory Reset

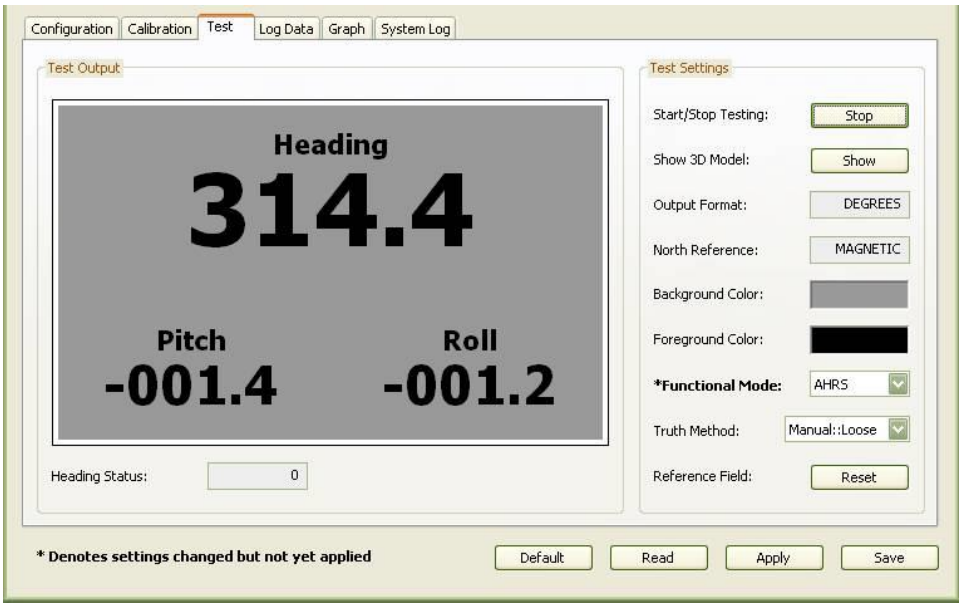
Clicking the Mag Factory Reset and/or Accel Factory Reset will reset the calibration coefficients currently indexed to those established at PNI. For example, if "Mag Coefficient Set 1" is selected as the current set, "Mag Factory Reset" will reset it to factory coefficients. Note that the mag factory coefficients will not take into account magnetic influences inherent in the host system, which could result in very large

heading errors. And the accel factory coefficients will not correct for any possible accelerometer drift. If either of these Reset buttons are selected, it is necessary to click <Save> or <Apply> to have the factory coefficients take effect.

6.6 Test Tab

The Test Tab is used to demonstrate and evaluate the performance of the TPTCM. The Heading, Pitch, and Roll are output during testing. TPTCM is set default as Compass mode.

In AHRS mode, the **Heading Status** provides an indication of the accuracy of the heading. A green box represents a heading accuracy of <2°, a yellow box means the heading uncertainty is approximately 2° to 10°, and a red box means the uncertainty is >10°. These heading uncertainties assume the TPTCM was properly calibrated in a clean magnetic environment.



Start/Stop Testing

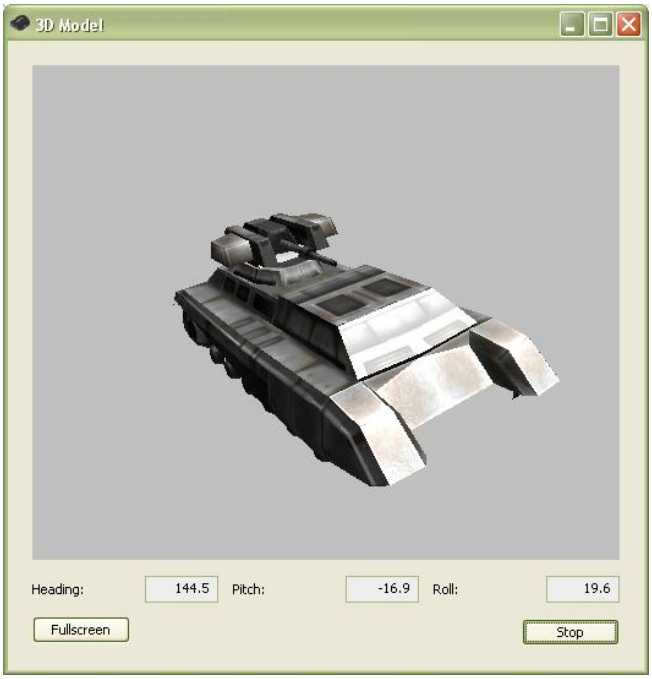
Clicking the <Start> button results in the Heading, Pitch, and Roll values being updated on the Test Tab screen, as well as the Heading Status.

If the Acquisition Mode is set to "Continuous", then the data will be continuously updated. The button will change to read <Stop>, such that clicking it again will stop the screen from updating.

If the Acquisition Mode is set to "Single", then only one measurement will be displayed on the screen. The button will briefly gray out while the data is being output to the screen, then return to reading <Start>.

Show 3D Model

Selecting the <Show> button launches the 3D Model window, as shown below.



Clicking the <Start> button begins continuous updating of the orientation of the rendered model, and of the heading, pitch, and roll output fields. The <Start> button on the Test Tab screen and on the 3D Model screen are linked such that selecting either of them will stop or start both screens.

Clicking on <Fullscreen> will expand the window to the full computer screen, and the button will now read <Windowed>. Clicking <Windowed> will shrink the window back down.

Output Format & North Reference

The "Output Format" and "North Reference" fields mimic the settings on the Configuration Tab. To change these, return to the Configuration Tab, make the change, then <Apply> or <Save> the change.

Background & Foreground Color

The foreground and background colors of the screen can be changed by the user. Simply click on the color square and select the new desired color. The change in color automatically is saved. The default is black lettering on a grey background.

Functional Mode

The TPTCM is designed to operate either in Compass Mode or AHRS Mode. And Compass Mode is the default operation mode. While the TPTCM is intended to be

used as a Compass, there are times when it is necessary or beneficial to place it in AHRS Mode. Before using it in either mode, it is necessary to calibrate the TPTCM in Compass Mode. Depending on the application, one mode may be better than another. Also, it is beneficial to operate in Compass Mode to conserve battery life, since in Compass Mode the TPTCM turns off the gyros, generally uses less CPU power, and can be placed in Sleep Mode to reduce current consumption.

Compass Mode

Compass Mode uses the magnetic sensor and accelerometer readings to determine heading, pitch and roll. When used in static condition such as target acquisition or surveying, Compass Mode will provide more accurate heading readings than AHRS Mode in a magnetic distortion free environment.

AHRS Mode

Two types of behavior can be configured in AHRS mode, *Magnetic Distortion Rejection AHRS* or *Gyro Stabilized Compass*. The Magnetic Distortion Rejection AHRS configuration is recommended.

Magnetic Distortion Rejection AHRS Configuration

There is a fine balance between magnetic distortion rejection and absolute heading accuracy. Magnetic Distortion Rejection Configuration requires user to define the "Truth Method" using the commands described in section 7.4.2. It requires users to be aware of the magnetic environment and have the knowledge of when to use the "Reference Field Reset" commands to define a clean magnetic reference field. If executed properly for the user's application, the TPTCM in this configuration can then provide protection against transient magnetic distortions.

Truth Method

When Magnetic Distortion Rejection is enabled, this field allows the user to have control over the criteria used to establish if the local magnetic field conforms to the reference magnetic field criteria. Three options are available, Standard (Loose), Tight and AutoMerge. Standard is recommended. Please see section 7.4.2 for details.

If Gyro Stabilized Compass Configuration is enabled, Settings related to Truth Method is ignored.

Reference Field Reset

When Magnetic Distortion Rejection Configuration is enabled, clicking the Reference Field <Reset> button on TRAX studio or sending the kSetResetRef command re-establishes the criteria for a clean magnetic field. In the TRAX

studio, after clicking this button the Heading Status box will go green. Over time, if the host system magnetic field changes significantly for a long duration from the clean magnetic field, the heading status displayed in the TRAX studio will eventually turn red, since it would have relied on only the gyroscope for its heading without proper correction from the magnetic truth reference. To correct for gyroscope drift, the distortion should be removed from the host system or if the user is confident the local magnetic field is free from distortion, Reference Field Reset command should be sent.

If Gyro Stabilized Compass Configuration is enabled, the Reference Field Reset command will immediately set the heading, pitch, and roll based on the mag and accel references.

Gyro Stabilized Compass Configuration

This setting utilizes the three-axis gyroscope's quick responsiveness in a less restricted magnetic clean environment. The heading is always using the magnetic sensor's measurement, merging to the magnetic sensor's heading continuously at a controlled rate that can be set using the kSetMergeRate command as described in section 7.4.2.

AHRS Configuration Examples

Drone

- Gyro Stabilized Compass Configuration
- KMergRate = 10 (default value)
- KMagRate = 10 (default value)

Robot

- Magnetic Distortion Rejection AHRS Configuration
- kSetMagTruthMethod = Standard ("1")
- KMergRate = 1
- KMagRate = 1
- Occasional Reference Field Reset may be required

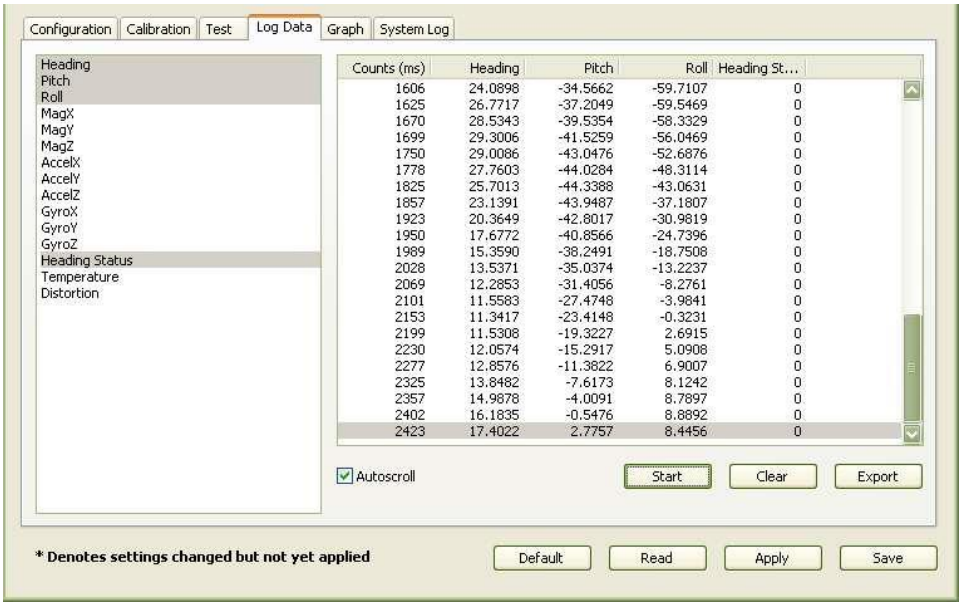
Platform

- Magnetic Distortion Rejection AHRS Configuration
- kSetMagTruthMethod = Tight ("2")
- KMergRate = 30

- KMagRate = 30
- Depending on local field Reference Field Reset may be required often

6.7 Log Data Tab

TRAX Studio can capture measurement data and then export it to a text file.



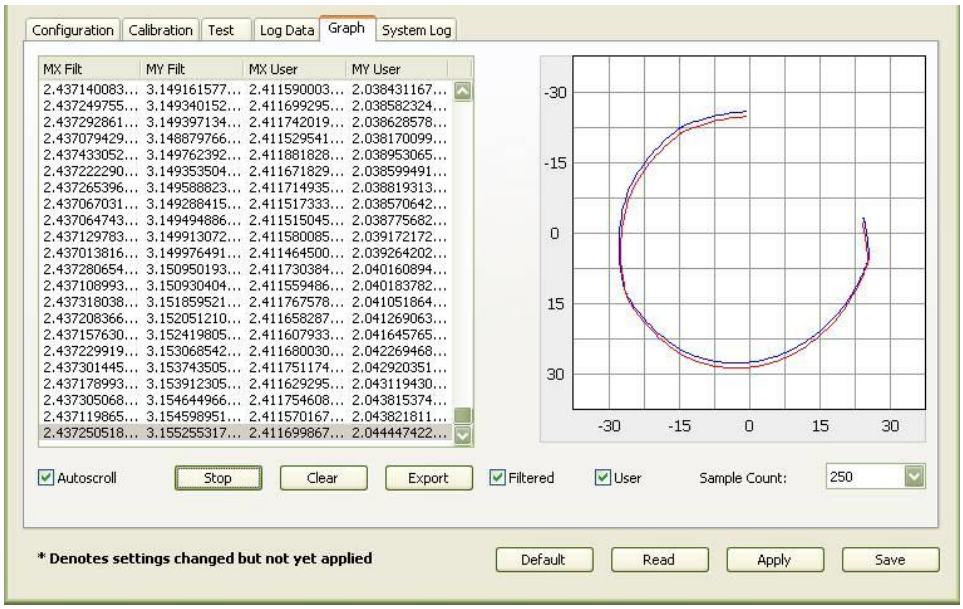
To acquire data and export it, follow the procedure below:

- Select the parameters you wish to log in the window on the left. Use Shift -Click and Ctrl-Click to select multiple items. In the screen shot above, “Heading”, “Pitch”, “Roll”, and “Heading Status” were selected. Note that Heading Status can be 1, 2, or 3, corresponding to “green”, “yellow”, or “red”.
- Click the <Start> button to start logging. The <Start> button changes to a <Stop> button after data logging begins.
- Click the <Stop> button to stop logging data.
- Click the <Export> button to save the data to a file.
- Click the <Clear> button to clear the data from the window.

Note that the “Distortion” log indicates if the magnetic field is >|±125 µT| for any of the magnetic sensors. It is only applicable in Compass Mode and will always read “FALSE” in AHRS Mode.

6.8 Graph Tab

The Graph Tab provides a plot of the measured field strength on the x-axis and y-axis magnetic sensors.

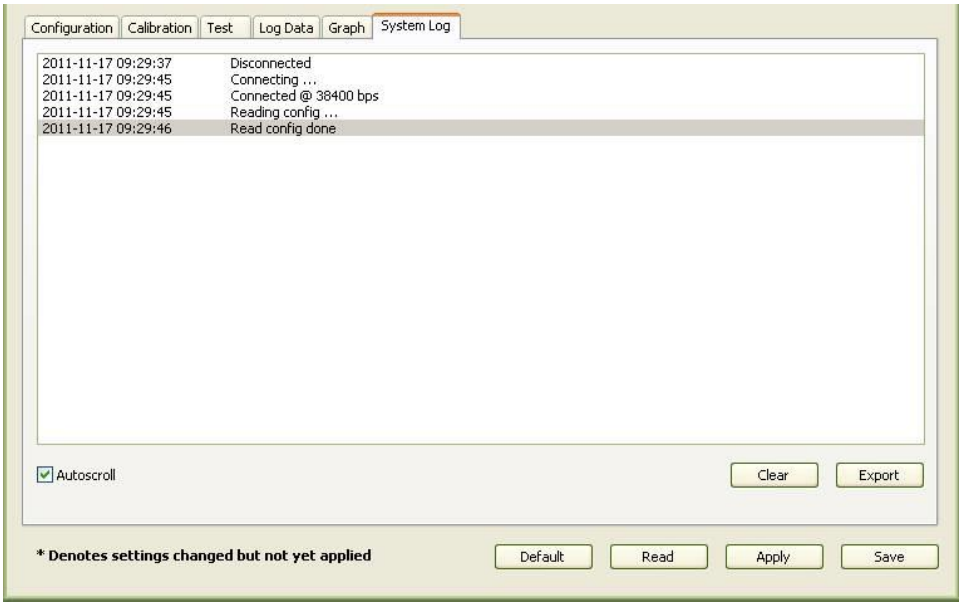


The "Filt" data and plot (blue) provides magnetic field strength measurements after the FIR filter taps are applied, but prior to applying the user calibration coefficients. The "User" data and plot (red) provides data after applying the user calibration coefficients. The graph can be used to visually see hard and soft-iron effects within the environment measured by the TPTCM, as well as corrected output after a user calibration has been performed.

The data can be saved to a .txt log file by clicking <Export>. To clear the data, select <Clear>. To clear both the data and the plot, select <Apply>.

6.9 System Log Tab

The System Log tab shows all communication between TRAX Studio and TPTCM. Closing TRAX Studio will erase the system log. Select the <Export> button, at the bottom right of the screen, to save the system log to a text file.



7 Operation with PNI Binary Protocol

The TPTCM utilizes a binary communication protocol, where the communication parameters should be configured as follows:

Table 7-1: Port Configuration

Parameter	Value
Number of Data Bits	8
Start Bits	1
Stop Bits	1
Parity	none

7.1 Datagram Structure

The data structure is shown below:

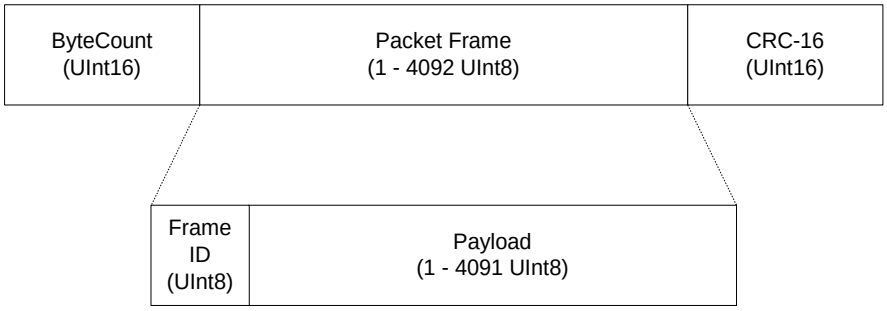


Figure 7-1: Datagram Structure

The ByteCount is the total number of bytes in the packet including the CRC-16 (checksum). The CRC-16 is calculated starting from the ByteCount to the last byte of the Packet Frame. The ByteCount and CRC-16 are always transmitted in big Endian. Two examples follow.

Example: The complete packet for the kGetModInfo command, which has no payload is:

00 05	01	EF D4
ByteCount	Frame ID	Checksum

Example: Below is a complete sample packet to start a 2D Calibration (kStartCal):

00 09	0A	00 00	00 14	5C F9
ByteCount	Frame ID	CalOption (MSBs)	CalOption (2D Calibration)	Checksum

7.2 Parameter Formats

7.2.1 Endianness

TPTCM can treat 32-bit and 16-bit parameters as having big or little Endian formatting. For 32-bit parameters, the big Endian byte order is ABCD EFGH, while the little Endian byte order is DCBA HGFE. For 16-bit parameters the big Endian byte order is ABCD, while the little Endian byte order is DCBA. The Endianness is selectable by the user per Section 7.4.2. The default is big Endian, and this is generally assumed in the manual.

7.2.2 Floating Point

Floating-point parameters conform to ANSI/IEEE Std 754-1985. A hex-to-floating-point converter can be found at http://babbage.cs.qc.cuny.edu/IEEE-754.old/32bit.html. Note that for this converter, little Endian values must be manually converted to big Endian.

Please refer to the Standard for more information. PNI also recommends referencing the user's compiler instructions to understand how the compiler implements floating-point.

64-Bit Floating Point (Float64)

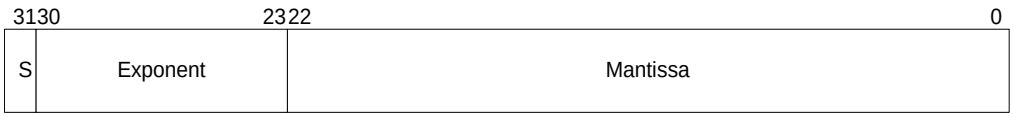
Below is the 64-bit float format in big Endian. In little Endian the bytes are in reverse order in 4-byte groups (e.g. DCBA HGFE).

The value (v) is determined as:

v = (-1)S * 2(Exponent-1023) * 1.Mantissa, if and only if 0 < Exponent < 2047

32-Bit Floating Point (Float32)

Shown below is the 32-bit float format in big Endian. In little Endian, the 4 bytes are in reverse order (e.g. DCBA).



The value (v) is determined as:

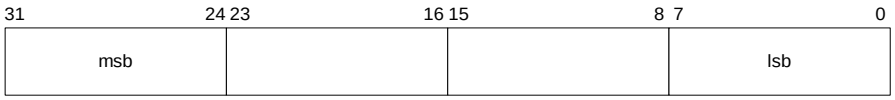
v = (-1)S * 2(Exponent-127) * 1.Mantissa, if and only if 0 < Exponent < 255

7.2.3 Signed Integer

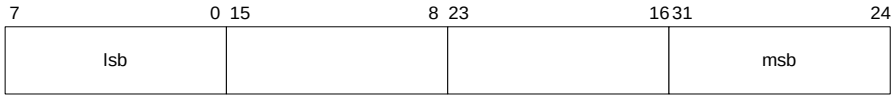
For signed integers, the most significant bit (msb) represents the sign of the value, where 0=positive and 1=negative. Signed integers are represents in 2's compliment.

Signed 32-Bit Integer (SInt32)

SInt32-based parameters are signed 32-bit numbers (2's compliment). Bit 31 represents the sign of the value, where 0=positive and 1=negative.



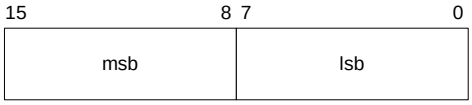
Big Endian



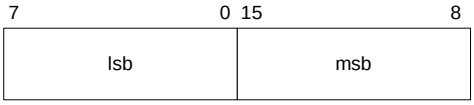
Little Endian

Signed 16-Bit Integer (SInt16)

SInt16-based parameters are signed 16-bit numbers (2's compliment). Bit 15 represents the sign of the value, where 0=positive and 1=negative.



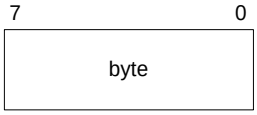
Big Endian



Little Endian

Signed 8-Bit Integer (SInt8)

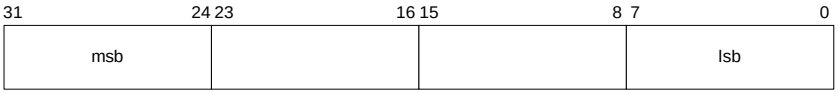
UInt8-based parameters are unsigned 8-bit numbers. Bit 7 represents the sign of the value, where 0=positive and 1=negative.



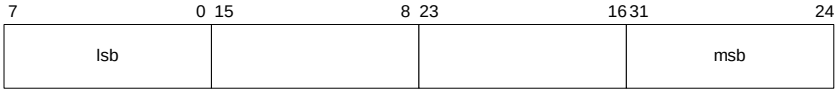
7.2.4 Unsigned Integer

Unsigned 32-Bit Integer (UInt32)

UInt32-based parameters are unsigned 32-bit numbers.



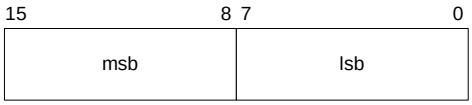
Big Endian



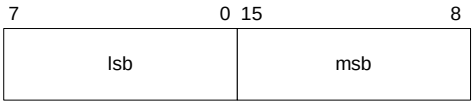
Little Endian

Unsigned 16-Bit Integer (UInt16)

UInt16-based parameters are unsigned 16-bit numbers.



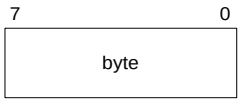
Big Endian



Little Endian

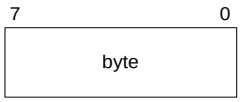
Unsigned 8-Bit Integer (UInt8)

UInt8-based parameters are unsigned 8-bit numbers.



7.2.5 Boolean

Boolean is a 1-byte parameter that MUST have the value 0=FALSE or 1=TRUE.



7.3 Commands Overview

Table 7-2 provides a summary of the basic commands available with the TPTCM.

Table 7-2: TPTCM Command Set

Frame ID Dec	Frame ID Hex	Command	Description
1	0x01	kGetModInfo	Queries the device's type and firmware revision.
2	0x02	kGetModInfoResp	Response to kGetModInfo
3	0x03	kSetDataComponents	Sets the data components to be output.
4	0x04	kGetData	Queries the TPTCM for data
5	0x05	kGetDataResp	Response to kGetData
6	0x06	kSetConfig	Sets internal configurations in TPTCM
7	0x07	kGetConfig	Queries TPTCM for the current internal configuration
8	0x08	kGetConfigResp	Response to kGetConfig
9	0x09	kSave	Saves the current internal configuration and any new user calibration coefficients to non-volatile memory.
10	0x0A	kStartCal	Commands the TPTCM to start user calibration
11	0x0B	kStopCal	Commands the TPTCM to stop user calibration
12	0x0C	kSetFIRFilters	Sets the FIR filter settings for the magnetometer & accelerometer sensors.
13	0x0D	kGetFIRFilters	Queries for the FIR filter settings for the magnetometer & accelerometer sensors.
14	0x0E	kGetFIRFiltersResp	Contains the FIR filter settings for the magnetometer & accelerometer sensors.
15	0x0F	kPowerDown	Powers down the module
16	0x10	kSaveDone	Response to kSave
17	0x11	kUserCalSampleCount	Sent from the TPTCM after taking a calibration sample point
18	0x12	kUserCalScore	Contains the calibration score
19	0x13	kSetConfigDone	Response to kSetConfig
20	0x14	kSetFIRFiltersDone	Response to kSetFIRFilters
21	0x15	kStartContinuousMode	Commands the TPTCM to output data at a fixed interval
22	0x16	kStopContinuousMode	Stops data output when in Continuous Mode

23	0x17	kPowerUpDone	Confirms the TPTCM has received a signal to power up
24	0x18	kSetAcqParams	Sets the sensor acquisition parameters
25	0x19	kGetAcqParams	Queries for the sensor acquisition parameters
26	0x1A	kSetAcqParamsDone	Response to kSetAcqParams
27	0x1B	kGetAcqParamsResp	Response to kGetAcqParams
28	0x1C	kPowerDownDone	Response to kPowerDown
29	0x1D	kFactoryMagCoeff	Resets magnetometer calibration coefficients to original factory-established values
30	0x1E	kFactoryMagCoeffDone	Response to kFactoryMagCoeff
31	0x1F	kTakeUserCalSample	Commands the TPTCM to take a sample during user calibration
36	0x24	kFactoryIAccelCoeff	Resets accelerometer calibration coefficients to original factory-established values
37	0x25	kFactoryAccelCoeffDone	Respond to kFactoryAccelCoeff
43	0x2B	kCopyCoeffSet	Copy one set of calibration coefficient to another set
44	0x2C	kCopyCoeffSetDone	Respond to kCopyCoeffSet
52	0x34	kSerialNumber	Request Serial Number of TPTCM unit
53	0x35	kSerialNumberResp	Respond to kSerialNumber
79	0x4F	kSetFunctionalMode	Puts TPTCM in Compass Mode or AHRS Mode
80	0x50	kGetFunctionalMode	Queries for Compass Mode or AHRS Mode
81	0x51	kGetFunctionalModeResp	Response to kGetFunctionalMode
107	0x6B	kSetDistortMode	Set Distortion Mode as On (1) or Off (0), default Off
108	0x6C	kGetDistortMode	Get Distortion Mode
109	0x6D	kGetDistortModeResp	Response to kGetDistortMode, return 1 as On, 0 Off
110	0x6E	kSetResetRef	Establishes criteria for the reference magnetic field.
119	0x77	kSetMagTruthMethod	Sets if dip angle & radius establish mag truth (standard) or additional criteria (tight).
120	0x78	kGetMagTruthMethod	Queries for standard or tight truth method.
121	0x79	kGetMagTruthMethodResp	Response to kGetMagTruthMethod
128	0x80	kSetMergeRate	Sets merge rate for process, mag, accel or gyro

129	0x81	kGetMergeRate	Queries for merge rate for process, mag, accel or gyro.
130	0x82	kGetMergeRateResp	Response to kGetMergeRate

7.4 Set-Up Commands

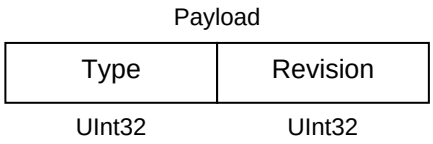
7.4.1 Module Information

kGetModInfo (frame ID 1_d, 0X01)

This frame queries the device's type and firmware revision number. The frame has no payload.

kGetModInfoResp (frame ID 2_d, 0X02)

The response to kGetModInfo is given below. The payload contains the device type identifier followed by the firmware revision number.



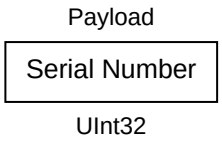
Note that the Type and Revision can be decoded from the binary format to character format using the ASCII standard. For example, the hex string "00 0D 02 54 50 54 31 31 32 30 38 C7 87" can be decoded to read "TPT1 1208". A Hex-to-ASCII converter can be found at: http://www.dolcevie.com/js/converter.html

kSerialNumber (frame ID 52_d, 0X34)

This frame queries the device's serial number. The frame has no payload.

kSerialNumberResp (frame ID 53_d, 0X35)

The response to kSerialNumber is given below. The payload contains the device serial number.

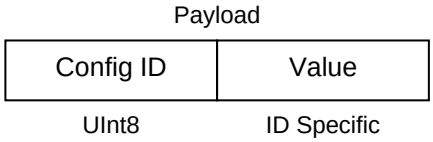


For example, the hex string "00 09 35 00 0F BE 43 0E CF" can be decoded with payload "00 0F BE 43" to read "1031747" as the serial number.

7.4.2 Module Configuration

kSetConfig (frame ID 6_d, 0X06)

This frame sets internal configurations in the TPTCM. The first byte of the payload is the configuration ID followed by a format-specific value. These configurations can only be set one at time. To save these in non-volatile memory, the kSave command must be issued.



Example: To configure the declination, the payload would look like:

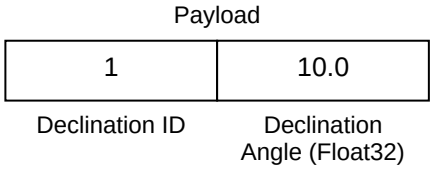


Table 7-3: Configuration Identifiers

Settings	Config. ID_d	Format	Values / Range	Default
kDeclination	1	Float32	-180˚ to +180˚	0
kTrueNorth	2	Boolean	True or False	False
kBigEndian	6	Boolean	True or False	True
kMountingRef	10	UInt8	1 = STD 0° 2 = X UP 0° 3 = Y UP 0° 4 = STD 90° 5 = STD 180° 6 = STD 270° 7 = Z DOWN 0° 8 = X UP 90° 9 = X UP 180° 10 = X UP 270° 11 = Y UP 90° 12 = Y UP 180° 13 = Y UP 270° 14 = Z DOWN 90° 15 = Z DOWN 180° 16 = Z DOWN 270°	1
kUserCalNumPoints	12	UInt32	4 – 18	12
kUserCalAutoSampling	13	Boolean	True or False	True

kBaudRate	14	UInt8	4 – 2400 5 – 3600 6 – 4800 7 – 7200 8 – 9600 9 – 14400 10 – 19200 11 – 28800 12 – 38400 13 – 57600 14 - 115200	12
kMilOut	15	Boolean	True or False	False
kHPRDuringCal	16	Boolean	True or False	True
kMagCoeffSet	18	UInt32	0 - 7	0
kAccelCoeffSet	19	UInt32	0 - 7	0

Configuration parameters and settings for kSetConfig:

kDeclination (Config. ID 1d)

This sets the declination angle to determine True North heading. Positive declination is easterly declination and negative is westerly declination. This is not applied unless kTrueNorth is set to TRUE.

kTrueNorth (Config. ID 2_d)

Flag to set compass heading output to true north heading by adding the declination angle to the magnetic north heading.

kBigEndian (Config. ID 6_d)

Sets the Endianness of packets. TRUE is Big Endian. FALSE is Little Endian.

Note: TRAX Studio requires Big Endian. Return the module to Big Endian if Little Endian has been set if it will be used with TRAX Studio.

kMountingRef (Config. ID 10_d)

This sets the reference orientation for the TPTCM. Please refer to Figure 4-2 for additional information.

kUserCalNumPoints (Config. ID 12_d)

The user must select the number of points to take during a calibration. Table 7-4 provides the "Minimum Recommended" number of sample points, as well as the full "Allowable Range". The "Minimum Recommended" number of samples normally is sufficient to meet the TPTCM's heading accuracy specification, while less than this may make it difficult to meet specification. See Section 5 for additional information.

Table 7-4: Sample Points

Calibration Mode	Number of Samples	
	Allowable Range	Minimum Recommended
Full-Range	10 to 32	12
Limited-Tilt	10 to 32	12
2D Calibration	10 to 32	12
Hard-Iron-Only	4 to 32	6
Accelerometer-Only	12 to 18	18
Mag-and-Accel	12 to 18	18

kUserCalAutoSampling (Config. ID 13_d)

This flag is used during user calibration. If set to TRUE, the TPTCM automatically takes calibration sample points once the minimum change and stability requirements are met. If set to FALSE, the device waits for kTakeUserCalSample to take a sample with the condition that a magnetic field vector component delta is greater than 5 µT from the last sample point. If the user wants to have maximum control over when the calibration sample points are taken then this flag should be set to FALSE.

kBaudRate (Config. ID 14_d)

Baud rate index value. A power-down, power-up cycle is required when changing the baud rate.

kMilOut (Config. ID 15_d)

Sets the output units as mils (TRUE) or degrees (FALSE). The default is FALSE.

kHPRDuringCal (Config. ID 16_d)

This flag sets whether or not heading, pitch, and roll data are output simultaneously while the TPTCM is being calibrated. The default is TRUE, such that heading, pitch, and roll are output during calibration. FALSE disables simultaneous output.

kMagCoeffSet (Config. ID 18_d)

This command provides the flexibility to store up to eight (8) sets of magnetometer calibration coefficients in the TPTCM. The default is set number 0. To store a set of coefficients, first establish the set number (number 0 to 7) using kMagCoeffSet, then perform the magnetometer calibration. The coefficient values will be stored in the defined set number. This feature is useful if the compass will be placed in multiple locations that have different local magnetic field properties.

kAccelCoeffSet (Config. ID 19$_d$)

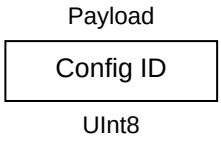
This command provides the flexibility to store up to eight (8) sets of accelerometer calibration coefficients in the TPTCM. The default is set number 0. To store a set of coefficients, first establish the set number (number 0 to 7) using kAccelCoeffSet, then perform the accelerometer calibration. The coefficient values will be stored in the defined set number.

kSetConfigDone (frame ID 19$_d$, 0X13)

This frame is the response to kSetConfig frame. The frame has no payload.

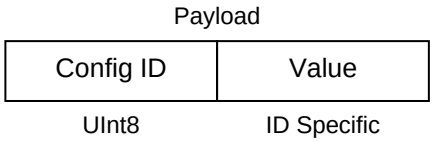
kGetConfig (frame ID 7$_d$, 0X07)

This frame queries the TPTCM for the current internal configuration value. The payload contains the configuration ID requested.

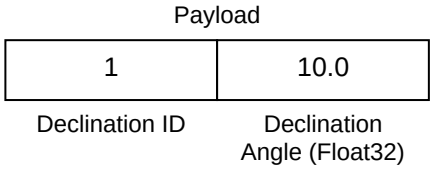


kGetConfigResp (frame ID 8$_d$, 0X08)

The response to kGetConfig is given below. The payload contains the configuration ID and value.



Example: If a request to get the set declination angle, the payload would look like:

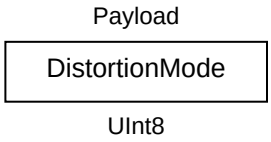


kSetDistortionMode (frame ID 107$_d$, 0X6B)

This frame allows the user to set Magnetic Distortion Rejection AHRS Configuration. Distortion mode switches TPTCM between two different AHRS mode configurations; Gyro Stabilized Compass or Magnetic Distortion Rejection AHRS Configurations.

When Distortion Mode is turned on, user needs to establish a good heading reference in a magnetic clean environment with KSetResetRef and KSetMagTruthMethod commands. Once the good heading reference is established, TPTCM will be able to

reject magnetic transient distortion and maintain desired heading accuracy. When distortion mode is off, MagTruthMethod is not relevant and TPTCM will operate in Gyro Stablized Compass Configuration where its output will slowly merge to stationary local magnetic field. While distortion mode is off, kSetResetRef will act to immediately align heading, pitch, and roll to the current mag/accel frame; under most use cases this alignment will be negligible due to the constant merging of the system. The "Distortion Mode" is Off by default. The payload is defined below.



DistortionMode Payload Value:
"0": Off
"1": On

kGetDistortionMode (frame ID 108$_d$, 0X6C)

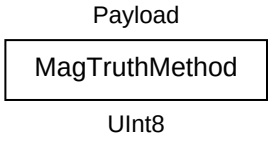
This frame queries the setting of the Distortion Mode. It has no payload.

kGetDistortionModeResp (frame ID 109$_d$, 0X6D)

This frame is the response of kGetDistortionMode and the payload is the same as for kSetDistortionMode.

kSetMagTruthMethod (frame ID 119$_d$, 0X77)

This frame allows the user to have control over the breadth of criteria used to establish if the local magnetic field conforms to the reference magnetic field criteria. The "AutoMerge" criteria is the default. The payload is defined below. Tight setting is not recommended unless the host system is in a fixed location and the user understands the local magnetic field well enough to send the Reset Reference Command as needed.



MagTruthMethod Payload Value:
"1": Standard (Loose)
"2": Tight
"3": AutoMerge

kGetMagTruthMethod (frame ID 120$_d$, 0X78)

This frame queries the setting of the Mag Truth Method. It has no payload.

kGetMagTruthMethodResp (frame ID 121$_d$, 0X79)

This frame is the response of kGetMagTruthMethod and the payload is the same as for kSetMagTruthMethod.

kSetMergeRate (frame ID 128_d, 0X80)

Merge rates are Time Constants (TC), they control how quickly the TPTCM algorithm converges to a specific sensor's reading. The smaller the value, the more reliant the algorithm is on the sensor. The following description is a good approximation of the time constant merge rate. It is not meant to be an exact mathematical representation:

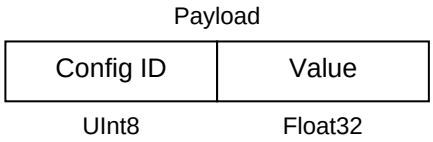
The time constant is square proportional to the time it takes the system to settle, the settle time is proportional to the TC value of the power of 2.

$$settle\ time \propto (TC)^2$$

A TC value of 0.1 merges almost instantaneously, value of 1 within 3 to 4 seconds, value of 10 within 300s. Following equation is an estimation between settle time and TC value.

$$settle\ time = 3(TC)^2$$

This frame sets merge rates for process and magnetic sensor, which will have the same value. The first byte of the payload is the rate ID followed by a 4-byte value. These rates can only be set one at time. To save these in non-volatile memory, the kSave command must be issued.



Example: To set the magnetic merge rate, the payload would look like:

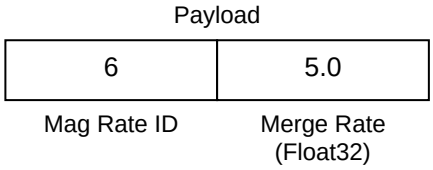


Table 7-5: Merge Rate Identifiers

Settings	Rate ID_d	Format	Values / Range	Default
kMergeRate	5	Float32	0.1 – 20.0	10.0
kMagRate	6	Float32	0.1 – 20.0	10.0

Rate parameters and settings for kSetMergeRate:

kMergeRate (Rate ID 5d)

This sets the process merge rate to control the combined magnetic and accelerometer merge times. This is the overall merging rate; it is recommended to set this and kMagRate to the same value.

kMagRate (Rate ID 6d)

This sets the mag merge rate to control how quickly the TPTCM algorithm converges to the mag reading, which primarily affects the heading value. The smaller the value, the more reliant the algorithm is on the mag. It is recommended to set this to the same value as kMergeRate.

kGetMergeRate (frame ID 129_d, 0X81)

This frame queries the setting of the Mag Truth Method. It has no payload.

kGetMergeRateResp (frame ID 130_d, 0X82)

This frame is the response of kGetMergeRate and the payload is the same as for kSetMergeRate.

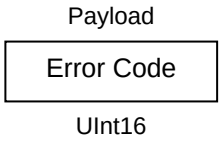
7.4.3 Saving Settings

kSave (frame ID 9_d, 0X09)

This frame commands the TPTCM to save internal configurations and user calibration to non-volatile memory. Internal configurations and user calibration are restored on power up. The frame has no payload. This is the ONLY command that causes the device to save information to non-volatile memory.

kSaveDone (frame ID 16_d, 0X10)

This frame is the response to kSave frame. The payload contains a UInt16 error code: 0 indicates no error; 1 indicates an error when attempting to save data to memory.



7.5 Measurement Commands

7.5.1 Setting the Reference Magnetic Field Criteria

Prior to operating the TPTCM in AHRS mode it is necessary to establish the criteria for a known distortion-free local field. This should be done with the TPTCM installed in the host system, as the distortion presented by the host system is constant and will be compensated for by the TPTCM algorithms.

kSetResetRef (frame ID 110_d)

This frame re-aligns the TPTCM 9-axis heading to the 6-axis (mag and accel) heading and establishes the criteria for the reference magnetic field. The frame should be sent when the user is confident the local magnetic field is not distorted. It has no payload.

7.5.2 Data Acquisition Parameters

kSetAcqParams (frame ID 24_d)

This frame sets the sensor acquisition parameters in the TPTCM. The payload should contain the following:

Payload

AcquisitionMode	FlushFilter	PNIReserved	SampleDelay
UInt8	UInt8	Float32	Float32

AcquisitionMode:

This flag sets whether output will be presented in Continuous or Polled Acquisition Mode. Poll Mode is TRUE and should be selected when the host system will poll the TPTCM for each data set. Continuous Mode is FALSE and should be selected if the user will have the TPTCM output data to the host system at a relatively fixed rate. Poll Mode is the default.

FlushFilter:

This is only relevant in Compass Mode. Setting this flag to TRUE results in the FIR filter being flushed (cleared) after every measurement. The default is FALSE.

Flushing the filter clears all tap values, thus purging old data. This can be useful if a significant change in heading has occurred since the last reading, as the old heading data would be in the filter. Once the taps are cleared, it is necessary to fully repopulate the filter before data is output. For example, if 32 FIR taps is set, 32 new samples must be taken before a reading will be output. The length of the delay before outputting data is directly correlated to the number of FIR taps.

PNIReserved:

These 4 bytes serve no function. PNI recommends populating the bytes with 0.

SampleDelay:

The SampleDelay is relevant when the Continuous Acquisition Mode is selected. It is the time delay, in seconds, between completion of TPTCM sending one set of data and the start of sending the next data set. The default is 0 seconds, which means TPTCM will send new data as soon as the previous data set has been sent. Note that the inverse of the SampleDelay is somewhat greater than the actual sample rate, since the SampleDelay does not include actual acquisition time.

kSetAcqParamsDone (frame ID 26 d)

This frame is the response to kSetAcqParams frame. The frame has no payload.

kGetAcqParams (frame ID 25 d)

This frame queries the unit for acquisition parameters. The frame has no payload.

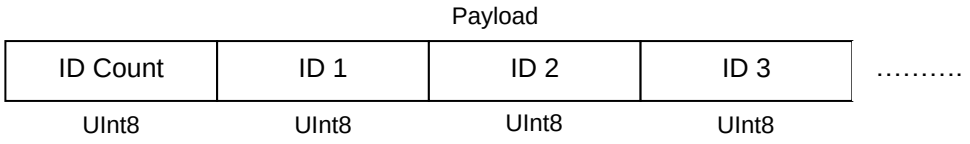
kGetAcqParamsResp (frame ID 27 d)

This frame is the response to kGetAcqParams frame. The payload should contain the same payload as the kSetAcqParams frame.

7.5.3 Data Components

kSetDataComponents (frame ID 3d)

This frame defines what data is output when kGetData is sent. Table 7-6 summarizes the various data components and more detail follows this table. Note that this is not a query for the device's model type and software revision (see kGetModInfo). The first byte of the payload indicates the number of data components followed by the data component IDs. Note that the sequence of the data components defined by kSetDataComponents will match the output sequence of kGetDataResp.



Example: To query for heading and heading status, the payload should contain:

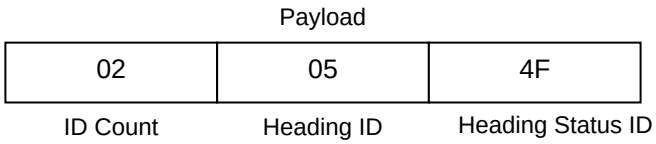


Table 7-6: Component Identifiers

Component	Component ID		Format	Units
	Decimal	Hex		
kHeading	5	0x05	Float32	degrees
kPitch	24	0x18	Float32	degrees
kRoll	25	0x19	Float32	degrees
kHeadingStatus	79	0x4F	UInt8	value
kQuaternion	77	0x4D	4x Float32	quaternion
kTemperature	7	0x07	Float32	˚ Celsius
kDistortion	8	0x08	Boolean	True or False (Default)
kCalStatus	9	0x09	Boolean	True or False (Default)
kAccelX	21	0x15	Float32	G
kAccelY	22	0x16	Float32	G
kAccelZ	23	0x17	Float32	G
kMagX	27	0x1B	Float32	µT
kMagY	28	0x1C	Float32	µT
kMagZ	29	0x1D	Float32	µT
kGyroX	74	0x4A	Float32	radians/sec
kGyroY	75	0x4B	Float32	radians/sec
kGyroZ	76	0x4C	Float32	radians/sec

Component types are listed below. All are read-only values.

kHeading, kPitch, kRoll (Component IDs 5_d, 24_d, 25_d)

Provides compass heading, pitch and roll outputs. The heading range is 0.0˚ to +359.9˚, the pitch range is -90.0˚ to +90.0˚, and the roll range is to -180.0˚ to +180.0˚.

kTemperature (Component ID 7_d)

This value is provided by the device's internal temperature sensor. Its value is in degrees Celsius and has an accuracy of 0.25° C.

kDistortion (Component ID 8_d)

This flag indicates at least one magnetometer axis reading is beyond ±125 µT. It is only applicable in Compass Mode, and will always read "FALSE" in AHRS Mode.

kCalStatus (Component ID 9_d)

This flag indicates the user calibration status. False means it is not user calibrated and this is the default value.

kAccelX, kAccelY & kAccelZ (Component IDs 21_d, 22_d, 23_d)

These values represent the accelerometer sensor data for the x, y, and z axis, respectively. The values are normalized to g (Earth's gravitational force).

kMagX, kMagY & kMagZ (Component IDs 27_d, 28_d, 29_d)

These values represent the magnetic sensor data for the x, y, and z axis, respectively. The values are given in µT.

kGyroX, kGyroY, kGyroZ (Component IDs 74_d, 75_d, 76_d)

These values represent the gyroscope sensor data for rotation around the x, y, and z axis, respectively. The values are in radians per second.

7.5.4 Making a Measurement

kGetData (frame ID 4_d)

If the TPTCM is configured to operate in Polled Acquisition Mode (see kSetAcqParams), then this frame requests a single measurement data set. The frame has no payload.

kStartContinuousMode (frame ID 21_d)

If the TPTCM is configured to operate in Continuous Acquisition Mode (see kSetAcqParams), then this frame initiates the outputting of data at a relatively fixed data rate, where the data rate is established by the SampleDelay parameter. The frame has no payload.

kStopContinuousMode (frame ID 22_d)

This frame commands the TPTCM to stop data output when in Continuous Acquisition Mode. The frame has no payload.

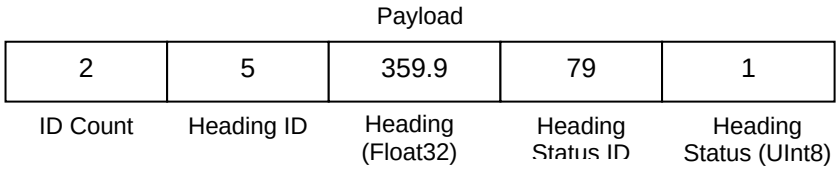
kGetDataResp (frame ID 5_d)

The response to kGetData and kStartContinuousMode is kGetDataResp. The specific data fields that will be output (ID 1, Value ID 1, etc.) should have been previously established by the kSetDataComponents command frame.

Payload

ID Count	ID 1	Value ID 1	ID 2	Value ID 2	ID 3	Value ID 3
UInt8	UInt8	ID Specific	UInt8	ID Specific	UInt8	ID Specific

Example: If heading and heading status are set to be output per the kSetDataComponents command, the payload would look like:



7.5.5 Continuous Data Output After Power Cycle

TPTCM is configurable to continuously send out data at next power on. Following procedure shows the steps on how to enable this feature, and how to disable it.

To Enable the Continuous Data Output

kSetAcqParams (frame ID 24 d)

Configurate TPTCM data acquisition as Continuous Mode (see AcquisitionMode) and set SampleDelay in expected interval in seconds, for example, interval 0.5s will result in 2Hz output rate.

kSetDataComponents (frame ID 3d)

This frame defines what data is output by selecting the various data components by their IDs (See Component ID).

kStartContinuousMode (frame ID 21 d)

This frame initiates the outputting of data at the expected fixed data rate, where the data rate is established by the SampleDelay parameter. The frame has no payload.

kSave (frame ID 9 d)

This frame commands the TPTCM to save internal configurations and user calibration to non-volatile memory. The frame has no payload. This is the ONLY command that causes the device to save information to non-volatile memory.

Power Cycle TPTCM

Power off and then power on, TPTCM will start to output the same data as the set of data before the power off.

To Disable the Continuous Data Output

Send the following two commands, TPTCM will stop outputting data after next power cycle.

kStopContinuousMode (frame ID 22 $_d$)

This frame commands the TPTCM to stop data output when in Continuous Acquisition Mode. The frame has no payload.

kSave (frame ID 9 $_d$)

Power Cycle TPTCM

Power off and then power on, TPTCM will not automatically output data.

7.6 Calibration Commands

7.6.1 User Calibration Commands

First, note that in order to perform a user calibration, it is necessary to place the TPTCM in Compass Mode, as discussed in Section 7.7. Note that TPTCM allows for a maximum of 18 calibration points.

kStartCal (frame ID 10$_d$)

This frame commands the TPTCM to start user calibration with the current sensor acquisition parameters, internal configurations and FIR filter settings.

Payload

CalOption

UInt32

CalOption Payload Value:
"10$_d$" or "0A$_h$": Full-Range Calibration (default)
"20$_d$" or "14$_h$": 2D Calibration
"30$_d$" or "1E$_h$": Hard-Iron-Only Calibration
"40$_d$" or "28$_h$": Limited-Tilt Calibration
"100$_d$" or "64$_h$": Accel-Only Calibration
"110$_d$" or "6E$_h$": Mag-and-Accel Calibration

Note: The payload needs to be 32 bit (4 byte). If no payload is entered or if less than 4 bytes are entered, the unit will default to the previous calibration method.

The CalOption values are given below, along with basic descriptions of the options.

Full-Range Calibration

Recommended calibration method when >30° of pitch is possible. Can be used for between 20° and 30° of pitch, but accuracy will not be as good

2D Calibration

Recommended when the available tilt range is limited to ≤5°. Can be used for 5° to 10° of tilt, but accuracy will not be as good.

Hard-Iron-Only Calibration

Recalibrates the hard-iron offset for a prior calibration. If the local field hard-iron distortion has changed, this calibration can bring the TPTCM back into specification.

Limited-Tilt Calibration

Recommended calibration method when >5° of tilt calibration is available, but tilt is restricted to <30°. (i.e. full range calibration is not possible.)

Accel-Only Calibration

Select this when an accelerometer calibration will be performed.

Accelerometer and Magnetic Calibration ($110_d = 6E_h$)

Selected when magnetic and accelerometer calibration will be done simultaneously.

Below is a complete sample packet to start a 2D Calibration (kStartCal):

00 09	0A	00 00	00 14	5C F9
ByteCount	Frame ID	CalOption (MSBs)	CalOption (2D Calibration)	Checksum

Heading, pitch and roll information is output via the kGetDataResp frame during the calibration process. This feature provides guidance during the calibration regarding calibration sample point coverage. During calibration, in the kGetDataResp frame, the number of data components is set to be 3 and then followed by the data component ID-value pairs. The sequence of the component IDs are kHeading, kPitch and kRoll.

kTakeUserCalSample (frame ID 31_d)

This frame commands the TPTCM to take a sample during user calibration. The frame has no payload.

kUserCalSampleCount (frame ID 17_d)

This frame is sent from the TPTCM after taking a calibration sample point. The payload contains the sample count with the range of 0 to 32. Payload 0 is sent from TPTCM after kStartCal is received by TPTCM, it indicates user calibration start, and TPTCM is ready to take samples. Payload 1 to 32 indicates each point sampled successfully.

Payload
SampleCount#
UInt32

kStopCal (frame ID 11_d)

This command aborts the calibration process. The prior calibration results are retained.

7.6.2 Performing a User calibration

The steps below provide an example of the steps to perform a user calibration.

- Using the kSetParam command, set the number of tap filters to 0.
- Using the kSetConfig command, set kUserCalAutoSampling. "False" is generally recommended, but "True" may be more convenient.
- Using the kSetConfig command, set kMagCoeffSet (magnetometer calibration) and/or kAccelCoeffSet (accelerometer calibration). These fields allow the user to save multiple sets of calibration coefficients. "0" is the default.
- Using the kSetConfig command again, set kUserCalNumPoints to the appropriate number of calibration points.
- Initiate a calibration using the kStartCal command. Note that this command requires indentifying the type of calibration procedure (i.e. Full-Range, 2D, etc.).
- Follow the appropriate calibration procedure, as discussed in Section 5. If kUserCalAutoSampling was set to "False", then send a kTakeUserCalSample command when ready to take a calibration point. If kUserCalAutoSampling was set to "True", then look for kUserCalSampleCount to confirm when a calibration point has been taken. During the calibration process, heading, pitch, and roll information will be output from the TPTCM, and this can be monitored using kDataResp.
- When the final calibration point is taken, the device will present the calibration score using kUserCalScore.
- If the calibration is acceptable (see Section 7.6.2), save the calibration coefficients using kSave.

7.6.3 Calibration Score

kUserCalScore (frame ID 18_d)

The calibration score is automatically sent upon taking the final calibration point. The payload is defined below, and the various payload components are discussed after this.

Payload

MagCalScore	Bytes 5-8	AccelCalScore	DistributionError	TiltError	TiltRange
Float32	Float32	Float32	Float32	Float32	Float32

MagCalScore:

Represents the over-riding indicator of the quality of the magnetometer calibration. Acceptable scores will be ≤1 for full range calibration, ≤2 for other methods. Note that it is possible to get acceptable scores for DistributionError and TiltError and still have a rather high MagCalScore value. The most likely reason for this is the TPTCM is close to a source of local magnetic distortion that is not fixed with respect to the device.

Bytes 5-8:

Reserved for PNI use.

AccelCalScore:

Represents the over-riding indicator of the quality of the accelerometer calibration. An acceptable score is ≤1.

DistributionError:

Indicates if the distribution of sample points is good, with an emphasis on the heading distribution. The score should be 0. Significant clumping or a lack of sample points in a particular section can result in a poor score.

TiltError:

Indicates if the TPTCM experienced sufficient tilt during the calibration, taking into account the calibration method. The score should be 0**.**

TiltRange:

This reports half the full pitch range of sample points. For example, if the device is pitched +25º to -15º, the TiltRange value would be 20º (as derived from [+25º - {-15º}]/2). For Full-Range Calibration and Hard-Iron-Only Calibration, this should be ≥30°. For 2D Calibration, ideally this should be ≈2°. For Limited-Tilt Calibration the value should be as large a possible given the user's constraints.

7.6.4 Factory Calibration

kFactoryMagCoeff (frame ID 29 d)

This frame clears the magnetometer calibration coefficients and loads the original factory-generated coefficients. The frame has no payload. This frame must be followed by the kSave frame to save the change in non-volatile memory.

kFactoryMagCoeffDone (frame ID 30 d)

This frame is the response to kFactoryMagCoeff frame. The frame has no payload.

kFactoryAccelCoeff (frame ID 36 d)

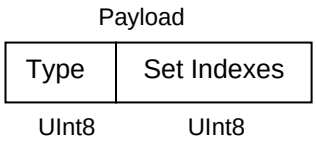
This frame clears the accelerometer calibration coefficients and loads the original factory-generated coefficients. The frame has no payload. This frame must be followed by the kSave frame to save the change in non-volatile memory.

kFactoryAccelCoeffDone (frame ID 37 d)

This frame is the response to kFactoryAccelCoeff frame. The frame has no payload.

kCopyCoeffSet (frame ID 43 d)

This frame copies one set of calibration coefficients to another. Trax2 supports 8 sets of magnetic calibration coefficients, and 8 sets of accel calibration coefficients. The set index is from 0 to 7. This frame must be followed by the kSave frame to save the change in non-volatile memory.



Type:

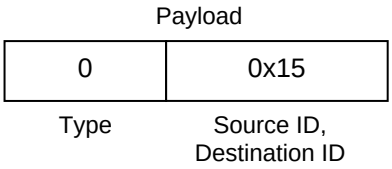
Value 0 to copy magnetic calibration coefficient set (default), 1 to copy accel coefficient set

Set Indexes:

bit 7 - 4: source coefficient set index from 0 to 7, default 0

bit 0 - 3: destination coefficient set index from 0 to 7, default 0

Example: Copy magnetic calibration coefficient set 1 to set 5 command, the payload would look like:



kCopyCoeffSetDone (frame ID 44 d)

This frame is the response to kCopyCoeffSet frame. The frame has no payload.

7.7 Performance Commands

The following commands will switch TPTCM's functional modes, change its performance to filter noise and save power.

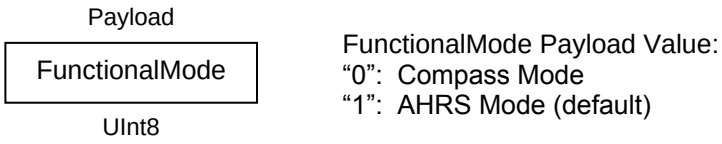
The TPTCM normally operates in Compass Mode as default, wherein it only uses the inputs from the accelerometer and magnetometer to establish heading, pitch, and roll to provide higher heading accuracy of 0.25º rms, for a static measurement in a known clean local magnetic field. In Compass Mode it can subsequently be placed into Sleep Mode which will significantly reduce power consumption when not taking measurements. User calibration is built into Compass Mode too.

However, the user can place the TPTCM in AHRS mode, which uses a proprietary Kalman algorithm to fuse the inputs of its 3-axis gyroscope, 3-axis accelerometer, and 3-axis magnetometer.

7.7.1 Switching Functional Mode

kSetFunctionalMode (frame ID 79 d)

This frame establishes whether the TPTCM operates in Compass Mode or AHRS Mode. The frame has a one byte payload. The payload byte is "0" to operate in Compass Mode and "1" to operate in AHRS Mode. The default is "0".



Sending the kSave command after changing the mode will save the new setting in non-volatile memory.

kGetFunctionalMode (frame ID 80 d)

This frame queries the TPTCM functional mode. The frame has no payload.

kGetFunctionalModeResp (frame ID 81 d)

This is the response of kGetFunctionalMode, and it has the same payload definition as kSetFunctionalMode.

7.7.2 FIR Filters

When operating in Compass Mode, the TPTCM incorporates a finite impulse response (FIR) filter to provide a more stable heading reading. The number of taps (or samples)

represents the amount of filtering to be performed. The number of taps directly affects the time for the initial sample reading, as all the taps must be populated before data is output. The FIR filter settings have no affect when operating in AHRS Mode.

The TPTCM can be configured to clear, or flush, the filters after each measurement, as discussed in Section 7.5.2. Flushing the filter clears all tap values, thus purging old data. This can be useful if a significant change in heading has occurred since the last reading, as the old heading data would be in the filter. Once the taps are cleared, it is necessary to fully repopulate the filter before data is output. For example, if 32 FIR taps is set, 32 new samples must be taken before a reading will be output. The length of the delay before outputting data is directly correlated to the number of FIR taps.

kSetFIRFilters (frame ID 12_d)

The payload for kSetFIRFilters is given below.

Payload

Byte 1	Byte 2	Count N	Value 1	Value 2	Value 3	Value N
UInt8	UInt8	UInt8	ID Specific	ID Specific	ID Specific	ID Specific

Byte 1 should be set to 3 and Byte 2 should be set to 1. The third payload byte indicates the number of FIR taps to use, which can be 0 (no filtering), 4, 8, 16, or 32. This is followed by the tap values (0 to 32 total Values can be in the payload), with each Value being a Float64, and suggested values given in Table 7-7.

Table 7-7: Recommended FIR Filter Tap Values

Count	4-Tap Filter	8-Tap Filter	16-Tap Filter	32-Tap Filter
1	04.6708657655334e-2	01.9875512449729e-2	07.9724971069144e-3	01.4823725958818e-3
2	04.5329134234467e-1	06.4500864832660e-2	01.2710056429342e-2	02.0737124095482e-3
3	04.5329134234467e-1	01.6637325898141e-1	02.5971390034516e-2	03.2757326624196e-3
4	04.6708657655334e-2	02.4925036373620e-1	04.6451949792704e-2	05.3097803863757e-3
5		02.4925036373620e-1	07.1024151197772e-2	08.3414139286254e-3
6		01.6637325898141e-1	09.5354386848804e-2	01.2456836057785e-2
7		06.4500864832660e-2	01.1484431942626e-1	01.7646051430536e-2
8		01.9875512449729e-2	01.2567124916369e-1	02.3794805168613e-2
9			01.2567124916369e-1	03.0686505921968e-2
10			01.1484431942626e-1	03.8014333463472e-2
11			09.5354386848804e-2	04.5402682509802e-2
12			07.1024151197772e-2	05.2436112653103e-2

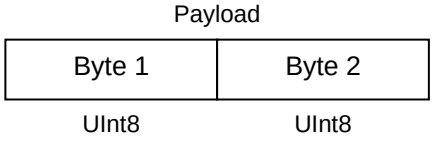
13			04.6451949792704e-2	05.8693165018301e-2
14			02.5971390034516e-2	06.3781858267530e-2
15			01.2710056429342e-2	06.7373451424187e-2
16			07.9724971069144e-3	06.9231186101853e-2
17				06.9231186101853e-2
18				06.7373451424187e-2
19				06.3781858267530e-2
20				05.8693165018301e-2
21				05.2436112653103e-2
22				04.5402682509802e-2
23				03.8014333463472e-2
24				03.0686505921968e-2
25				02.3794805168613e-2
26				01.7646051430536e-2
27				01.2456836057785e-2
28				08.3414139286254e-3
29				05.3097803863757e-3
30				03.2757326624196e-3
31				02.0737124095482e-3
32				01.4823725958818e-3

kSetFIRFiltersDone (frame ID 20 d)

This frame is the response to kSetFIRFilters. The frame has no payload.

kGetFIRFilters (frame ID 13d)

This frame queries the FIR filter settings for the sensors. Byte 1 should be set to 3 and Byte 2 should be set to 1.



kGetFIRFiltersResp (frame ID 14d)

This is the response to kGetFIRFilters and it has the same payload definition as kSetFIRFilters.

7.7.3 Power Down/Up

TPTCM may be powered down to save energy when heading data is not required.

kPowerDown (frame ID 15_d)

This frame is used to power-down the module. The frame has no payload. The command will power down all peripherals including the sensors, microprocessor, and RS-232 driver. However, the driver chip has a feature to keep the Rx line enabled. The TPTCM will power up when it receives any signal on the native UART Rx line.

kPowerDownDone (frame ID 28_d)

This frame confirms the TPTCM received a command to power down. The frame has no payload.

kPowerUpDone (frame ID 23_d)

This frame confirms the TPTCM received a command to power up. (The TPTCM will power up when it receives any signal on the native UART Rx line.) The frame has no payload.

7.8 Using Multiple Coefficient Sets

The ability to store and access multiple calibration coefficients sets the TPTCM apart from our Prime or legacy TCM. This section will detail the command list and provide two examples for utilizing this functionality.

Table 7-8: Multiple Coefficient Command List

Magnetometer Calibration					
kSetConfig (frame ID)	**kCoeffCopySet (config ID)**	**Value (UInt32)**	**Examples**	**Command Bytes**	**TPTCM Response**
0x06	0x12	0-7	Set kCoeffCopySet to be copy 0	0x00 0x0A 0x06 0x12 0x00 0x00 0x00 0x00 0x3E 0x76	0x00 0x05 0x13 0xDD 0xA7
			Set kCoeffCopySet to be copy 1	0x00 0x0A 0x06 0x12 0x00 0x00 0x00 0x01 0x2E 0x57	0x00 0x05 0x13 0xDD 0xA8
			Set kCoeffCopySet to be copy 4	0x00 0x0A 0x06 0x12 0x00 0x00 0x00 0x04 0x7E 0xF2	0x00 0x05 0x13 0xDD 0xA9
kGetConfig (frame ID)	**kCoeffCopySet (config ID)**	**Value (UInt32)**	**Examples**	**Command Bytes**	**TPTCM Response**
0x07	0x12		get kCoeffCopySet value which is currently used in TPTCM	0x00 0x06 0x07 0x12 0x19 0x44	0x00 0x0A 0x08 0x12 0x00 0x00 0x00 0x?? CRC1 CRC2
Accelerometer Calibration					
kSetConfig (frame ID)	**AccelCoeffCopySet (config ID)**	**Value (UInt32)**	**Examples**	**Command Bytes**	**TPTCM Response**
0x06	0x13	0 - 2	Set kAccelCoeffCopySet to be copy 0	0x00 0x0A 0x06 0x13 0x00 0x00 0x00 0x00 0x94 0x27	0x00 0x05 0x13 0xDD 0xA7
			Set kAccelCoeffCopySet to be copy 1	0x00 0x0A 0x06 0x13 0x00 0x00 0x00 0x01 0x84 0x06	0x00 0x05 0x13 0xDD 0xA8
			Set kAccelCoeffCopySet to be copy 2	0x00 0x0A 0x06 0x13 0x00 0x00 0x00 0x02 0xB4 0x65	0x00 0x05 0x13 0xDD 0xA9
kGetConfig (frame ID)	**AccelCoeffCopySet (config ID)**	**Value (UInt32)**	**Examples**	**Command Bytes**	**TPTCM Response**
0x07	0x13		get kAccelCoeffCopySet value which is currently used in TPTCM	0x00 0x06 0x07 0x13 0x09 0x65	0x00 0x0A 0x08 0x13 0x00 0x00 0x00 0x?? CRC1 CRC2

Examples

Example 1: Save Magnetic Calibration result to Coeff Copy Set 4.

Set the kCoeffCopySet to copy 4 by sending the following command.

0x00 0x0A 0x06 0x12 0x00 0x00 0x00 0x04 0x7E 0xF2

Get the kCoeffCopySet to verify by sending the following command. (Optional)

0x00 0x06 0x07 0x12 0x19 0x44

Send kSave command to save the kCoeffCopySet to flash so that it will be still available after power cycle. The kSave command is as following.

0x00 0x05 0x09 0x6E 0xDC

Start a user calibration, when completes, save calibration coeffs to TCM. The coeffs have been saved into coeff set copy 4.

Example 2: Use Magnetic Coeff Copy Set 1 in TCM. (The assumption is user has saved calibration coeffs to set 1 before)

Set the kCoeffCopySet to copy 1 by sending the following command.

0x00 0x0A 0x06 0x12 0x00 0x00 0x00 0x01 0x2E 0x57

Get the kCoeffCopySet to verify by sending the following command. (Optional)

0x00 0x06 0x07 0x12 0x19 0x44

Send kSave command to save the kCoeffCopySet to flash so that it will be still available after power cycle. The kSave command is as following.

0x00 0x05 0x09 0x6E 0xDC

7.9 Communication Protocol Example

Below is a procedure to generate heading, pitch, and roll outputs in Polled Mode.

1. Write kGetModInfo (frame ID = 0x01)

   ```
   00 05 01 ef d4
   ```

2. Read response from Trax kGetModInfoResp (frame ID = 0x2). In ASCII it will contain "TRAXP733" version number.

   ```
   00 0d 02 54 52 41 58 50 37 33 33 5b 76
   ```

3. Write kSetDataComponents (frame ID 0x03). There are 4 components: kHeading, kPitch, kRoll, kHeadingStatus (component IDs = 0x5, 0x18, 0x19, 0x4f)

   ```
   00 0a 03 04 05 18 19 4f e2 ef
   ```

4. Write kGetData (frame ID=0x04)

   ```
   00 05 04 bf 71
   ```

5. Read kGetDataResp (frame ID=0x05). There are 4 components: kHeading, kPitch, kRoll, kHeadingStatus

   ```
   00 17 05 04 05 43 b3 df 5e 18 be 88 ed bd 19 3d b5 15 53 4f 03 91
   34
   ```

6. Repeat steps 4 and 5 whenever data is desired.

Appendix – Sample Code

The following example files, CommProtocol.h, CommProtocol.cpp, TPTCM.h and TPTCM.cpp would be used together for proper communication with a TPTCM module.

Note: *The following files are not included in the sample codes and need to be created by the user: Processes.h & TickGenerator.h. The comments in the code explain what is needed to be sent or received from these functions so the user can write this section for the user's platform. For example, with the TickGenerator.h, the user needs to write a routing that generates 10 msec ticks.*

Header File & CRC-16 Function

```
//Header File & CRC-16 Function
// type declarations
typedef struct
{
    UInt8 AcquisitionMode, FlushFilter;
    Float32 AcquireDelay, SampleDelay;
} __attribute__ ((packed)) AcqParams;

typedef struct
{
    Float32 MagCalScore;
    Float32 reserve1;
    Float32 AccelCalScore;
    Float32 DistError;
    Float32 TiltError;
    Float32 TiltRange;
} __attribute__ ((packed)) MagCalScore;

enum
{
    // Frame IDs (Commands)
    kGetModInfo = 1,         // 1
    kGetModInfoResp,         // 2
    kSetDataComponents, // 3
    kGetData,           // 4
    kGetDataResp,       // 5
    kSetConfig,         // 6
    kGetConfig,         // 7
    kGetConfigResp,     // 8
    kSave,          // 9
    kStartCal,          // 10
    kStopCal,           // 11
    kSetFilters,        // 12
    kGetFilters,        // 13
    kGetFiltersResp,        // 14
    kPowerDown,         // 15
    kSaveDone,          // 16
    kUserCalSampCount,  // 17
    kCalScore,          // 18
    kSetConfigDone,     // 19
    kSetFiltersDone,        // 20
    kStartContinuousMode,   // 21
    kStopContinuousMode,    // 22
    kPowerUp,           // 23
    kSetAcqParams,      // 24
    kGetAcqParams,      // 25
    kAcqParamsDone,     // 26
    kGetAcqParamsResp,  // 27
    kPowerDoneDown,     // 28
    kFactoryUserCal,        // 29
    kFactoryUserCalDone,    // 30
    kTakeUserCalSample, // 31
```

```
    kFactoryInclCal = 36,   // 36
    kFactoryInclCalDone,    // 37
    kSetSyncMode = 46,  // 46
    kSetSyncModeDone,       // 47
    kSyncRead = 49,     // 49

    // Cal Option IDs
    kFullRangeCal = 10, // 10 - type Float32
    k2DCal = 20,        // 20 - type Float32
    kHIOnlyCal = 30,        // 30 - type Float32
    kLimitedTiltCal = 40,   // 40 - type Float32
    kAccelCalOnly = 100,    // 100 - type Float32
    kAccelCalwithMag =110,  // 110 - type Float32

    // Param IDs
    kSetDataComponents =3,  // 3-AxisID(UInt8) + Count(UInt8) +
                    // Value (Float64) +...

    // Data Component IDs
    kHeading = 5,       // 5 - type Float32
    kTemperature = 7,       // 7 - type Float32
    kDistortion,        // 8 - type boolean
    kAccelX = 21,       // 21 - type Float32
    kAccelY,            // 22 - type Float32
    kAccelZ,            // 23 - type Float32
    kPitch,         // 24 - type Float32
    kRoll,          // 25 - type Float32
    kMagX = 27,         // 27 - type Float32
    kMagY,          // 28 - type Float32
    kMagZ,          // 29 - type Float32

    // Configuration Parameter IDs
    kDeclination = 1,       // 1 - type Float32
    kTrueNorth,         // 2 - type boolean
    kMountingRef = 10,  // 10 - type UInt8
    kUserCalStableCheck,    // 11 - type boolean
    kUserCalNumPoints,  // 12 - type UInt32
    kUserCalAutoSampling,   // 13 - type boolean
    kBaudRate,          // 14 - UInt8
    kMilOutPut,         // 15 - type Boolean
    kDataCal            // 16 - type Boolean
    kMagCoeffSet = 18,  // 18 - type UInt32
    kAccelCoeffSet, // 19 - type UInt32

    // Mounting Reference IDs
    kMountedStandard = 1,   // 1
    kMountedXUp,        // 2
    kMountedYUp,        // 3
    kMountedStdPlus90,  // 4
    kMountedStdPlus180, // 5
    kMountedStdPlus270, // 6
    kMountedZDown       // 7
    kMountedXUpPlus90       // 8
    kMountedXUpPlus180  // 9
    kMountedXUpPlus270  // 10
    kMountedYUpPlus90       // 11
    kMountedYUpPlus180  // 12
    kMountedYUpPlus270  // 13
    kMountedZDownPlus90 // 14
    kMountedZDownPlus180    // 15
    kMountedZDownPlus270    // 16


    // Result IDs
    kErrNone = 0,       // 0
    kErrSave,           // 1
};


// function to calculate CRC-16
UInt16 CRC(void * data, UInt32 len)
```

```
{
    UInt8 * dataPtr = (UInt8 *)data;
    UInt32 index = 0;
    // Update the CRC for transmitted and received data using
    // the CCITT 16bit algorithm (X^16 + X^12 + X^5 + 1).
    UInt16 crc = 0;
    while(len--)
    {
        crc = (unsigned char)(crc >> 8) | (crc << 8);
        crc ^= dataPtr[index++];
        crc ^= (unsigned char)(crc & 0xff) >> 4;
        crc ^= (crc << 8) << 4;
        crc ^= ((crc & 0xff) << 4) << 1;
    }
    return crc;
}
```

CommProtocol.h File

```
//CommProtocol.h File

#pragma once

#include "SystemSerPort.h"
#include "Processes.h"


//
//CommHandler is a base class that provides a callback for
//incoming messages.
//
class CommHandler
{
public:
    // Call back to be implemented in derived class.
    virtual void HandleComm(UInt8 frameType, void * dataPtr =
        NULL, UInt16 dataLen = 0) {}
};

//
//  CommProtocol handles the actual serial communication with the
//  module.
//  Process is a base class that provides CommProtocol with
//  cooperative parallel processing.  The Control method will be
//  called by a process manager on a continuous basis.
//
class CommProtocol : public Process
{
public:
    enum
    {
        // Frame IDs (Commands)
        kGetModInfo         // 1
        kGetModInfoResp,    // 2
        kSetDataComponents, // 3
        kGetData,           // 4
        kGetDataResp,       // 5

        // Data Component IDs
        kHeading = 5,       // 5 - type Float32
        kTemperature = 7,   // 7 - type Float32
        kAccelX = 21,       // 21 - type Float32
        kAccelY,            // 22 - type Float32
        kAccelZ,            // 23 - type Float32
        kPitch,             // 24 - type Float32
        kRoll,              // 25 - type Float32
    };

    enum
    {
        kBufferSize = 512,  // max size of input buffer
        kPacketMinSize = 5  // min size of serial packet
    };

    // SerPort is a serial communication object abstracting
    // the hardware implementation
    CommProtocol(CommHandler * handler = NULL, SerPort * serPort = NULL);

    void Init(UInt32 baud = 38400);

    void SendData(UInt8 frame, void * dataPtr = NULL, UInt32 len = 0);
    void SetBaud(UInt32 baud);

protected:
    CommHandler * mHandler;
    SerPort * mSerialPort;

    UInt8 mOutData[kBufferSize], mInData[kBufferSize];
```

```
    UInt16 mExpectedLen;
    UInt32 mOutLen, mOldInLen, mTime, mStep;

    UInt16 CRC(void * data, UInt32 len);
    void Control();
};
```

CommProtocol.cpp File

```
//CommProtocol.cpp File

#include "CommProtocol.h"

// import an object that will provide a 10mSec tick count through
// a function called Ticks()
#include "TickGenerator.h"

// SerPort is an object that controls the physical serial
// interface.  It handles sending out
// the characters, and buffers the characters read in until
// we are ready for them.
//
CommProtocol::CommProtocol(CommHandler * handler, SerPort * serPort)
    : Process("CommProtocol")
{
    mHandler = handler;
    // store the object that will parse the data when it is fully
    // received
    mSerialPort = serPort;
    Init();
}

//  Initialize the serial port and variables that will control
// this process
void CommProtocol::Init(UInt32 baud)
{
    SetBaud(baud);
    mOldInLen = 0;
    // no data previously received
    mStep = 1;
    // goto the first step of our process
}

//
// Put together the frame to send to the module
//
void CommProtocol::SendData(UInt8 frameType, void * dataPtr, UInt32 len)
{
    UInt8 * data = (UInt8 *)dataPtr;        // the data to send
    UInt32 index = 0;
    // our location in the frame we are putting together
    UInt16 crc;
    // the CRC to add to the end of the packet
    UInt16 count;
    // the total length the packet will be

    count = (UInt16)len + kPacketMinSize;

    // exit without sending if there is too much data to fit
    // inside our packet
    if (len > kBufferSize - kPacketMinSize) return;

    // Store the total len of the packet including the len bytes
    // (2), the frame ID (1),
    // the data (len), and the crc (2).  If no data is sent, the
    // min len is 5
    mOutData[index++] = count >> 8;
    mOutData[index++] = count & 0xFF;

    // store the frame ID
    mOutData[index++] = frameType;

    // copy the data to be sent
    while (len--) mOutData[index++] = *data++;

    // compute and add the crc
    crc = CRC(mOutData, index);
    mOutData[index++] = crc >> 8;
```

```
    mOutData[index++] = crc & 0xFF;

    // Write block will copy and send the data out the serial port
    mSerialPort->WriteBlock(mOutData, index);
}

//
// Call the functions in serial port necessary to change the
// baud rate
//
void CommProtocol::SetBaud(UInt32 baud)
{
    mSerialPort->SetBaudRate(baud);
    mSerialPort->InClear();
    // clear any data that was already waiting in the buffer
}

//
// Update the CRC for transmitted and received data using the
// CCITT 16bit algorithm (X^16 + X^12 + X^5 + 1).
//

UInt16 CommProtocol::CRC(void * data, UInt32 len)
{
    UInt8 * dataPtr = (UInt8 *)data;
    UInt32 index = 0;

    UInt16 crc = 0;
    while (len--)
    {
        crc = (unsigned char)(crc >> 8) | (crc << 8);
        crc ^= dataPtr[index++];
        crc ^= (unsigned char)(crc & 0xff) >> 4;
        crc ^= (crc << 8) << 4;
        crc ^= ((crc & 0xff) << 4) << 1;
    }
    return crc;
}

//
//  This is called each time this process gets a turn to execute.
//
void CommProtocol::Control()
{
    // InLen returns the number of bytes in the input buffer of
    //the serial object that are available for us to read.
    UInt32 inLen = mSerialPort->InLen();

    switch (mStep)
    {
    case 1:
    {
        // wait for length bytes to be received by the serial object
        if (inLen >= 2)
        {
            // Read block will return the number of requested (or available)
            // bytes that are in the serial objects input buffer.
            // read the byte count
            mSerialPort->ReadBlock(mInData, 2);

            // byte count is ALWAYS transmitted in big endian, copy byte
            // count to mExpectedLen to native endianess
            mExpectedLen = (mInData[0] << 8) | mInData[1];

            // Ticks is a timer function.  1 tick = 10msec.
            // wait up to 1/2s for the complete frame (mExpectedLen) to be
            // received
            mTime = Ticks() + 50;
            mStep++;
            // goto the next step in the process
        }
```

```
            break;
        }

        case 2:
        {
            // wait for msg complete or timeout
            if (inLen >= mExpectedLen - 2)
            {
                UInt16 crc, crcReceived;
                // calculated and received crcs.

                // Read block will return the number of
                // requested (or available) bytes that are in the
                // serial objects input buffer.
                mSerialPort->ReadBlock(&mInData[2], mExpectedLen - 2);
                // in CRC verification, don't include the CRC in the recalculation (-2)
                crc = CRC(mInData, mExpectedLen - 2);
                // CRC is also ALWAYS transmitted in big endian
                crcReceived = (mInData[mExpectedLen - 2] << 8) | mInData[mExpectedLen - 1];

                if (crc == crcReceived)
                {
                    // the crc is correct, so pass the frame up for processing.
                    if (mHandler) mHandler->HandleComm(mInData[2], &mInData[3], mExpectedLen -
kPacketMinSize);
                }
                else
                {
                    // crc's don't match so clear everything that is currently in the
                    // input buffer since the data is not reliable.
                    mSerialPort->InClear();
                }

                // go back to looking for the length bytes.
                mStep = 1;
            }
            else
            {
                // Ticks is a timer function.  1 tick = 10msec.
                if (Ticks() > mTime)
                {
                    // Corrupted message.  We did not get the length we were
                    // expecting within 1/2sec of receiving the length bytes.  Clear
                    // everything in the input buffer since the data is unreliable
                    mSerialPort->InClear();
                    mStep = 1;
                    // Look for the next length bytes
                }
            }
            break;
        }

        default:
            break;
        }
}
```

TPTCM.h File

```
//TPTCM.h File

#pragma once

#include "Processes.h"
#include "CommProtocol.h"


//
//  This file contains the object providing communication to TRAX.
//  It will set up the module and parse packets received.
//  Process is a base class that provides TRAX with cooperative
//  parallel processing.  The Control method will be
//  called by a process manager on a continuous basis.
//
class TPTCM : public Process, public CommHandler
{
    public:
        TPTCM(SerPort * serPort);
        ~TPTCM();

    protected:
        CommProtocol * mComm;

        UInt32 mStep, mTime, mResponseTime;

        void HandleComm(UInt8 frameType, void * dataPtr = NULL, UInt16 dataLen = 0);
        void SendComm(UInt8 frameType, void * dataPtr = NULL, UInt16 dataLen = 0);

        void Control();
};
```

TPTCM.cpp File

```
//TPTCM.cpp File

#include "TPTCM.h"
#include "TickGenerator.h"

const UInt8 kDataCount = 4;
// We will be requesting 4 components (heading, pitch, roll, and
// temperature)
//
//  This object polls the compass module once a second for
//  heading, pitch, roll and temperature.
//

TPTCM::TPTCM(SerPort * serPort)
    : Process("TPTCM")
{
    // Let the CommProtocol know this object will handle any
    // serial data returned by the module
    mComm = new CommProtocol(this, serPort);

    mTime = 0;
    mStep = 1;
}

TPTCM::~TPTCM ()
{
}

//
// Called by the CommProtocol object when a frame is completely // received
//
void TPTCM::HandleComm(UInt8 frameType, void * dataPtr, UInt16 dataLen)
{
    UInt8 * data = (UInt8 *)dataPtr;

    switch (frameType)
    {
    case CommProtocol::kGetDataResp:
    {
        // Parse the data response
        UInt8 count = data[0];
        // The number of data elements returned
        UInt32 pntr = 1;
        // Used to retrieve the returned elements

            // The data elements we requested
        Float32 heading, pitch, roll, temperature;

        if (count != kDataCount)
        {
            // Message is a function that displays a C formatted string
            // (similar to printf)
            Message("Received %u data elements instead of the %u requested\r\n", (UInt16)count,
                (UInt16)kDataCount);
            return;
        }

        // loop through and collect the elements
        while (count)
        {
            // The elements are received as {type (ie. kHeading), data}
            switch (data[pntr++])
                // read the type and go to the first byte of the data
            {
                // Only handling the 4 elements we are looking for
            case CommProtocol::kHeading:
            {
                // Move(source, destination, size (bytes)).  Move copies the
                // specified number of bytes from the source pointer to the
```

```
            // destination pointer.  Store the heading.
            Move(&(data[pntr]), &heading, sizeof(heading));

            // increase the pointer to point to the next data element type
            pntr += sizeof(heading);
            break;
        }

        case CommProtocol::kPitch:
        {
            // Move(source, destination, size (bytes)).  Move copies the
            // specified number of bytes from the source pointer to the
            // destination pointer.  Store the pitch.
            Move(&(data[pntr]), &pitch, sizeof(pitch));

            // increase the pointer to point to the next data element type
            pntr += sizeof(pitch);
            break;
        }

        case CommProtocol::kRoll:
        {
            // Move(source, destination, size (bytes)).  Move copies the
            // specified number of bytes from the source pointer to the
            // destination pointer.  Store the roll.
            Move(&(data[pntr]), &roll, sizeof(roll));

            // increase the pointer to point to the next data element type
            pntr += sizeof(roll);
            break;
        }

        case CommProtocol::kTemperature:
        {
            // Move(source, destination, size (bytes)).  Move copies the
            // specified number of bytes from the source pointer to the
            // destination pointer.  Store the heading.
            Move(&(data[pntr]), &temperature, sizeof(temperature));

            // increase the pointer to point to the next data element type
            pntr += sizeof(temperature);
            break;
        }

        default:
            // Message is a function that displays a formatted string
        // (similar to printf)
            Message("Unknown type: %02X\r\n", data[pntr - 1]);
            // unknown data type, so size is unknown, so skip everything
            return;
            break;
        }

        count--;
        // One less element to read in
    }

    // Message is a function that displays a formatted string
    // (similar to printf)
    Message("Heading: %f, Pitch: %f, Roll: %f, Temperature: %f\r\n", heading, pitch, roll,
        temperature);
    mStep--;
    // send next data request
    break;
}

default:
{
    // Message is a function that displays a formatted string
// (similar to printf)
    Message("Unknown frame %02X received\r\n", (UInt16)frameType);
```

```
        break;
    }
    }
}

//
// Have the CommProtocol build and send the frame to the module.
//
void TPTCM::SendComm(UInt8 frameType, void * dataPtr, UInt16 dataLen)
{
    if (mComm) mComm->SendData(frameType, dataPtr, dataLen);
    // Ticks is a timer function.  1 tick = 10msec.
    mResponseTime = Ticks() + 300;      // Expect a response within 3 seconds
}
//
//  This is called each time this process gets a turn to execute.
//
void TPTCM::Control()
{
    switch (mStep)
    {
    case 1:
    {
        UInt8 pkt[kDataCount + 1];
        // the compents we are requesting, preceded by the number of
        // components being requested

        pkt[0] = kDataCount;
        pkt[1] = CommProtocol::kHeading;
        pkt[2] = CommProtocol::kPitch;
        pkt[3] = CommProtocol::kRoll;
        pkt[4] = CommProtocol::kTemperature;

        SendComm(CommProtocol::kSetDataComponents, pkt, kDataCount + 1);

        // Ticks is a timer function.  1 tick = 10msec.
        mTime = Ticks() + 100;
        // Taking a sample in 1s.
        mStep++;
        // go to next step of process
        break;
    }

    case 2:
    {
        // Ticks is a timer function.  1 tick = 10msec.
        if (Ticks() > mTime)
        {
            // tell the module to take a sample
            SendComm(CommProtocol::kGetData);
            mTime = Ticks() + 100;  // take a sample every second
            mStep++;
        }
        break;
    }

    case 3:
    {
        // Ticks is a timer function.  1 tick = 10msec.
        if (Ticks() > mResponseTime)
        {
            Message("No response from the module.  Check connection and try again\r\n");
            mStep = 0;
        }
        break;
    }

    default:
        break;
    }
}
```

Revision Control Block

Revision	Description of Change	Effective Date	Approval
Draft	Initial version	July 1, 2019	B. Zhang
Preliminary	For review and feedback	July 8, 2019	B. Zhang
Preliminary	Add AHRS mode back	Sep 10, 2019	B. Zhang
Ver1.0	Update Calibration Chapters	Mar 20, 2020	B. Zhang
Ver1.1	Update AHRS	Nov 10, 2020	B. Zhang
Ver1.2	Value correction on heading status	Dec 7, 2020	B. Zhang
Ver1.3	Table 3-1 add calibration ranges for mag and Accel	Dec 27, 2020	B. Zhang
Ver1.4	Content update, and link error correction	Feb 11, 2021	RS, BZ
Ver1.5	Table 3-1 Update on performance	Jan 10, 2022	B. Zhang