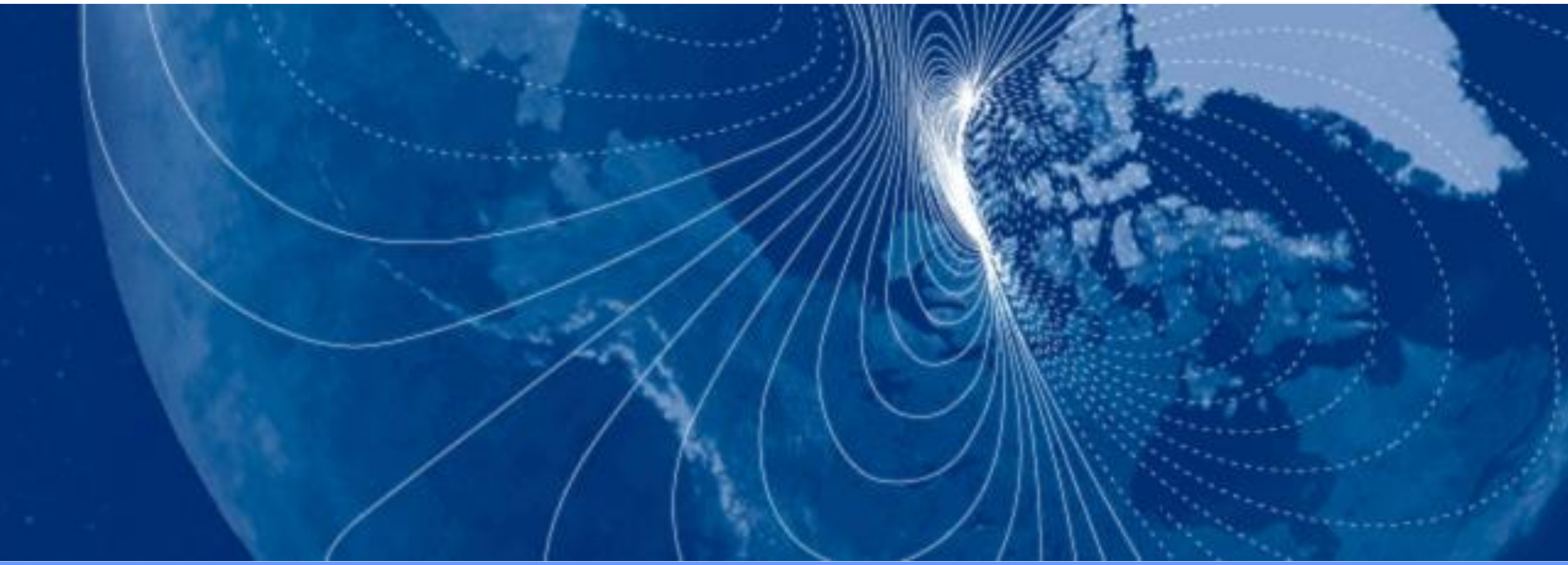


User Manual

TCM

High-Performance Tilt-Compensated Compass Module

PNI SENSOR CORPORATION

Table of Contents

List of Tables

List of Figures

1 Copyright & Warranty Information

Revised March 2014. For most recent version visit our website at www.pnicorp.com

PNI Sensor Corporation
2331 Circadian Way
Santa Rosa, CA 95407, USA
Tel: +1 (707) 566-2260
Fax: +1 (707) 566-2261

Warranty and Limitation of Liability. PNI Sensor Corporation ("PNI") manufactures its TCM 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 Sensor Corporation's TCM XB (pn 12810) or TCM MB (pn 13095) tilt-compensated 3-axis digital compass. The TCM is a high-performance, low-power consumption, tilt-compensated electronic compass module that incorporates PNI's advanced magnetic distortion compensation and calibration scoring algorithms to provide industry-leading heading accuracy. The TCM combines PNI's patented magneto-inductive sensors and measurement circuit technology with a 3-axis MEMS accelerometer for unparalleled cost effectiveness and performance.

PNI recognizes not all applications allow for significant tilt during calibration, so multiple calibration methods are available to ensure optimized performance can be obtained in the real world. These include Full-Range Calibration, when ≥45° of tilt is possible during calibration, 2D Calibration when constrained to calibration in a horizontal or near-horizontal plane, and Limited-Tilt Calibration when tilt is constrained to <45° but >5° of tilt is possible.

PNI also recognizes conditions may change over time, and to maintain superior heading accuracy it may be necessary to recalibrate the compass. So the TCM incorporates Hard-Iron-Only Calibration to easily account for gradual changes in the local magnetic distorting components. Plus, the accelerometer can be periodically recalibrated in the field to maintain maximum accuracy.

These advantages make PNI's TCM the choice for applications that require the highest accuracy and performance anywhere in the world under a wide range of conditions. Applications for the TCM include:

- Unmanned vehicles – underwater (UUV), ground (UGV), & aerial (UAV)
- Far target locaters and laser range finders
- Dead reckoning systems
- Systems in which the tilt angles used for calibration are physically constrained

With its many applications, the TCM incorporates a flexible and adaptable command set. Many parameters are user-programmable, including reporting units, a wide range of sampling configurations, output damping, and more.

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

3 Specifications

3.1 Characteristics & Requirements

Table 3-1: Performance Characteristics[1]

Parameter				Value
Heading	Accuracy	≤65° of pitch after Full-Range Calibration		<0.3° rms
		≤80° of pitch after Full-Range Calibration		<0.5° rms
		≤5° of pitch after 2D calibration		<2.0° rms
		≤2 times the calibration tilt angle when using limited-tilt calibration[2]		<2.0° rms
	Resolution			0.1°
	Repeatability			0.05° rms
Attitude	Range	Pitch		± 90°
		Roll		± 180°
	Accuracy	Pitch		0.2° rms
		Roll	≤65° of pitch	0.2° rms
			≤80° of pitch	0.4° rms
			≤86° of pitch	1.0° rms
	Resolution			0.01°
	Repeatability			0.05° rms
Maximum Operational Dip Angle[3]				85°
Magnetometers		Calibrated Field Range		± 125 µT
		Resolution		0.05 µT
		Repeatability		± 0.1 µT

Footnotes:

1. Specifications are subject to change. Assumes the TCM is motionless and the local magnetic field is clean relative to the user calibration.
2. For example, if the calibration was performed over ±10° of tilt, then the TCM would provide <2° rms accuracy over ±20° of tilt.
3. Performance at maximum operational dip angle will be somewhat degraded.

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 Operating Requirements

<table>
<tr><th colspan="3">Parameter</th><th>Value</th></tr>
<tr><td rowspan="2">Supply Voltage</td><td colspan="2">TCM XB</td><td>3.8 to 9 VDC</td></tr>
<tr><td colspan="2">TCM MB</td><td>3.3 to 9 VDC</td></tr>
<tr><td rowspan="8">Communication Lines</td><td rowspan="4">TCM XB</td><td>High Level Input</td><td>2.4 V minimum</td></tr>
<tr><td>Low Level Input</td><td>0.6 V maximum</td></tr>
<tr><td>Output Voltage Swing</td><td>±5.2 V typ., ±5.0 V min.</td></tr>
<tr><td>Tx Output Resistance</td><td>300 Ω</td></tr>
<tr><td rowspan="4">TCM MB</td><td>High Level Input</td><td>2.0 V minimum</td></tr>
<tr><td>Low Level Input</td><td>0.8 V maximum</td></tr>
<tr><td>Output Voltage Swing</td><td>0 – 3.3 V typical</td></tr>
<tr><td>Tx Output Resistance</td><td>330 Ω</td></tr>
<tr><td rowspan="4">Average Current Draw</td><td rowspan="2">TCM XB</td><td>@ max. sample rate</td><td>20 mA typical</td></tr>
<tr><td>@ 8 Hz sample rate</td><td>16 mA typical</td></tr>
<tr><td rowspan="2">TCM MB</td><td>@ max. sample rate</td><td>17 mA typical</td></tr>
<tr><td>@ 8 Hz sample rate</td><td>13 mA typical</td></tr>
<tr><td rowspan="2">Peak Current Draw</td><td colspan="2">During application of external power</td><td>120 mA pk, 60 mA avg over 2 ms</td></tr>
<tr><td colspan="2">During logical power up/down or Sync Trigger</td><td>135 mA pk, 60 mA avg over 4 ms</td></tr>
<tr><td rowspan="2">Sleep Mode Current Draw</td><td colspan="2">TCM XB</td><td>0.3 mA typical</td></tr>
<tr><td colspan="2">TCM MB</td><td>0.1 mA typical</td></tr>
</table>

Table 3-4: I/O Characteristics

Parameter		Value
Communication Interface	TCM XB	RS232 UART
	TCM MB	CMOS/TTL UART
Communication Protocol		PNI Binary
Communication Rate		300 to 115200 baud
Maximum Sample Rate[1]		~30 samples/sec
Time to Initial Good Data[2]	Initial power up	<210 ms
	Sleep Mode recovery	<80 ms

Footnotes:

1. The maximum sample rate is dependent on the strength of the magnetic field, and typically will be from 25 to 32 samples/sec.
2. FIR taps set to “0”.

Table 3-5: Environmental Requirements

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

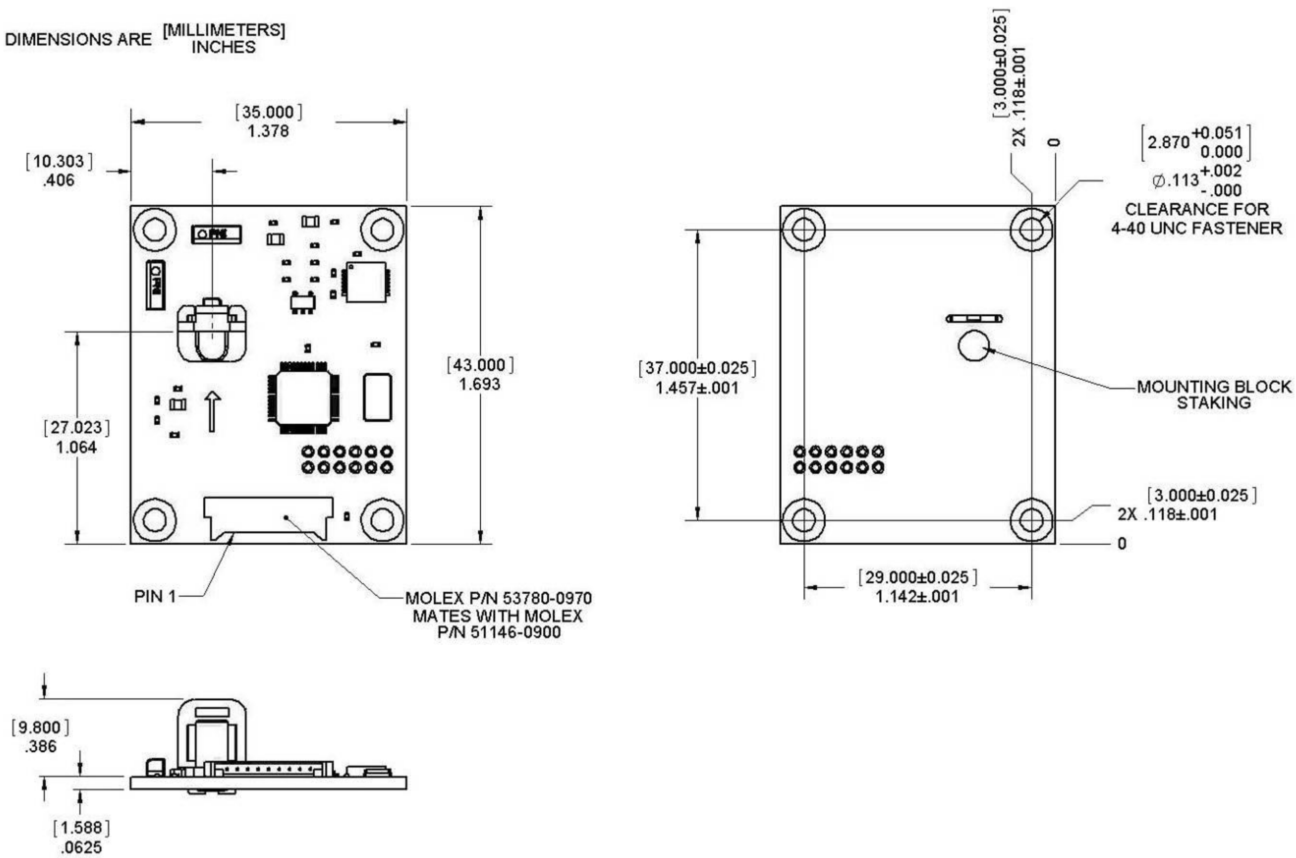
Footnote:

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*)	TCM XB	35 x 43 x 13 mm
	TCM MB	33 x 31 x 13 mm
Weight	TCM XB	6.8 gm
	TCM MB	5.3 gm
Connector	TCM XB	9-pin Molex, pn 53780-0970
	TCM MB	4-pin MIL-MAX, pn 850-10-004-10-001000

3.2 Mechanical Drawings



The default orientation is for the silk-screened arrow to point in the "forward" direction.

Figure 3-1: TCM XB Mechanical Drawing

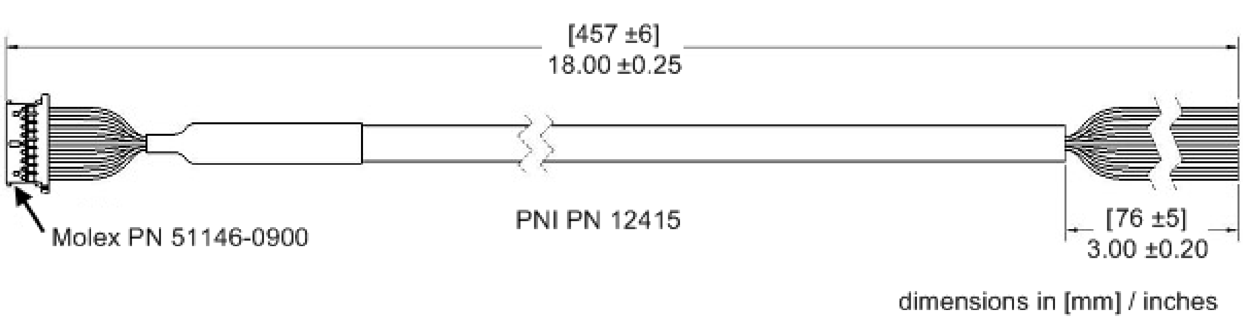
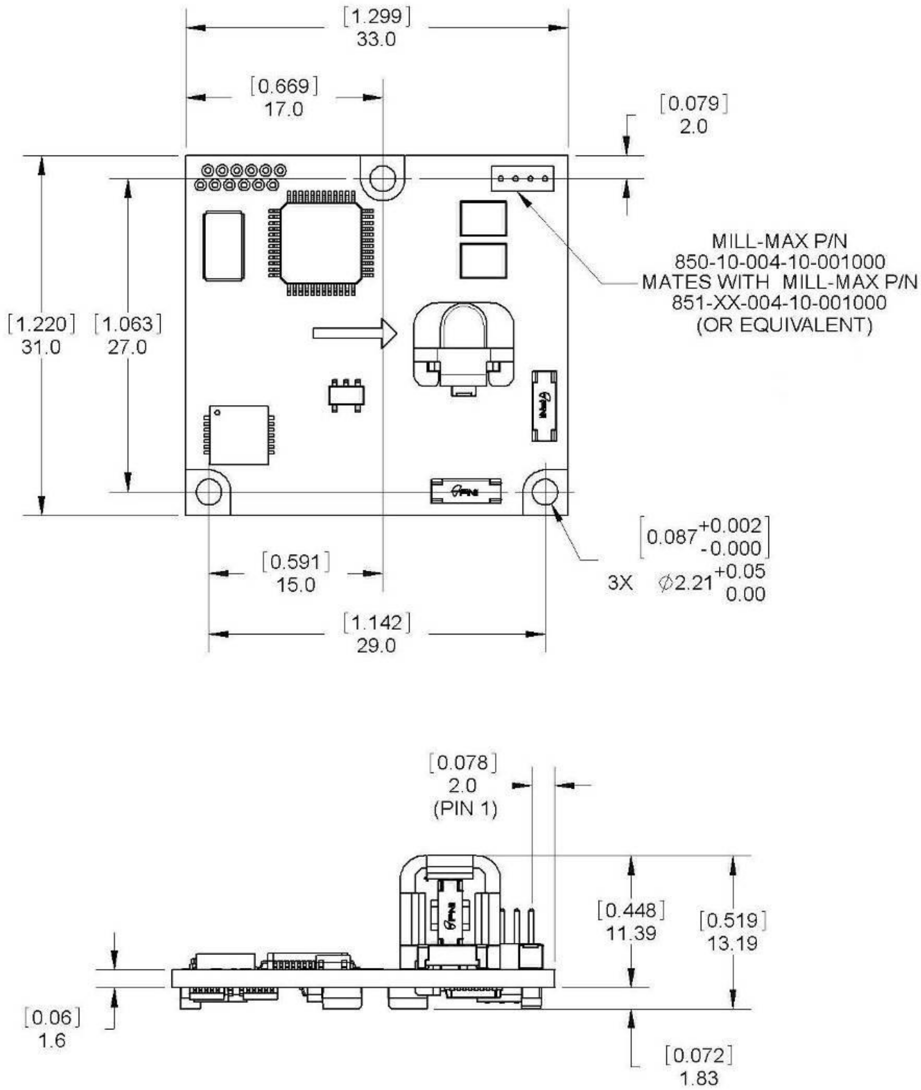


Figure 3-2: TCM XB Pigtailed Cable Drawing



The default orientation is for the silk-screened arrow to point in the “forward” direction.

Figure 3-3: TCM MB Mechanical Drawing

4 Set-Up

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

- Make electrical connections to the TCM.
- Evaluate the TCM using TCM Studio or a binary terminal emulation program, such as RealTerm or Tera Term, to ensure the compass generally works correctly.
- Choose a mounting location.
- Mechanically mount the TCM in the host system.
- Perform a user calibration.

4.1 Electrical Connections

The TCM XB incorporates a 9 pin Molex connector, part number 53780-0970, which mates with Molex part 51146-0900 or equivalent. The TCM MB incorporates a 4 pin Mil-Max connector, part number 850-10-004-10-001000, which mates with Mill-Max part 851-XX-004-10-001000 or equivalent. The pin-out is given below in Table 4-1.

Table 4-1: TCM Pin Descriptions

Pin Number[1]	TCM XB		TCM MB
	9 Pin Connector	Cable Wire Color	4 Pin Connector
1	GND	Black	GND
2	GND	Gray	Vin
3	GND	Green	UART Tx
4	NC	Orange	UART Rx
5	NC	Violet	
6	NC	Brown	
7	UART Tx	Yellow	
8	UART Rx	Blue	
9	Vin	Red	

Footnote:

1. For the TCM XB, pin #1 is indicated on Figure 3-1, while for the TCM MB, pin #1 is the pin closest to the corner.

After making the electrical connections, it is a good idea to perform some simple tests to ensure the TCM is working as expected. See Section 5 for how to operate the TCM with TCM Studio, or Section 7 for how to operate the TCM using the PNI binary protocol.

4.2 Installation Location

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

4.2.1 Operate within the TCM's dynamic range

The TCM can be user calibrated to correct for static magnetic fields created by the host system. However, each axis of the TCM has a calibrated dynamic range of ±125 µT. If the total field exceeds this value for any axis, the TCM may not perform to specification. When mounting the TCM, consider the effect of any sources of magnetic fields in the host environment that, when added to Earth's field, may take the TCM out of its dynamic regime. 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

It is not possible to calibrate for changing magnetic anomalies. Thus, for greatest accuracy, keep the TCM away from sources of local magnetic distortion that will change with time; such as electrical equipment that will be turned on and off, or ferrous bodies that will move. Make sure the TCM is not mounted close to cargo or payload areas that may be loaded with large sources of local magnetic fields.

4.2.3 Mount in a physically stable location

Choose a location that is isolated from excessive shock, oscillation, and vibration. The TCM works best when stationary. Any non-gravitational acceleration results in a distorted reading of Earth's gravitational vector, which affects the heading measurement.

4.2.4 Location-verification testing

Location-verification testing should be performed at an early stage of development to understand and accommodate the magnetic distortion contributors in a host system.

Determine the distance range of field distortion.

Place the compass in a fixed position, then move or energize suspect components while observing the output to determine when they are an influence.

Determine if the magnetic field is within the dynamic range of the compass.

With the compass mounted, rotate and tilt the system in as many positions as possible. While doing so, monitor the magnetometer outputs, observing if the maximum linear range is exceeded.

4.3 Mechanical Mounting

The TCM 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 module are non-magnetic. Refer to Section 3.2 for dimensions, hole locations, and the reference frame orientation.

Note: *Ensure that when attaching the TCM to the host system, the mounting method does not introduce stresses on the board, as this can affect the performance of the accelerometer, and therefore also negatively affect heading accuracy.*

4.3.1 Pitch and Roll Conventions

The TCM uses a MEMS accelerometer to measure the tilt angle of the compass. This data is output as pitch and roll data, and is also used in conjunction with the magnetometers to provide a tilt-compensated heading reading.

The TCM utilizes Euler angles as the method for determining accurate orientation. This method is the same used in aircraft orientation where the outputs are heading (also called yaw or azimuth), pitch and roll. When using Euler angles, 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 two rotations are independent of each other since the rotation axes rotate with the plane body.

As shown in Figure 4-1, for the TCM 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 down.

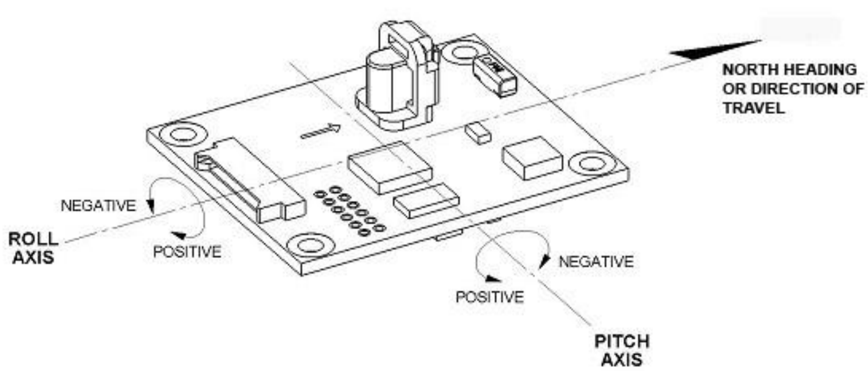
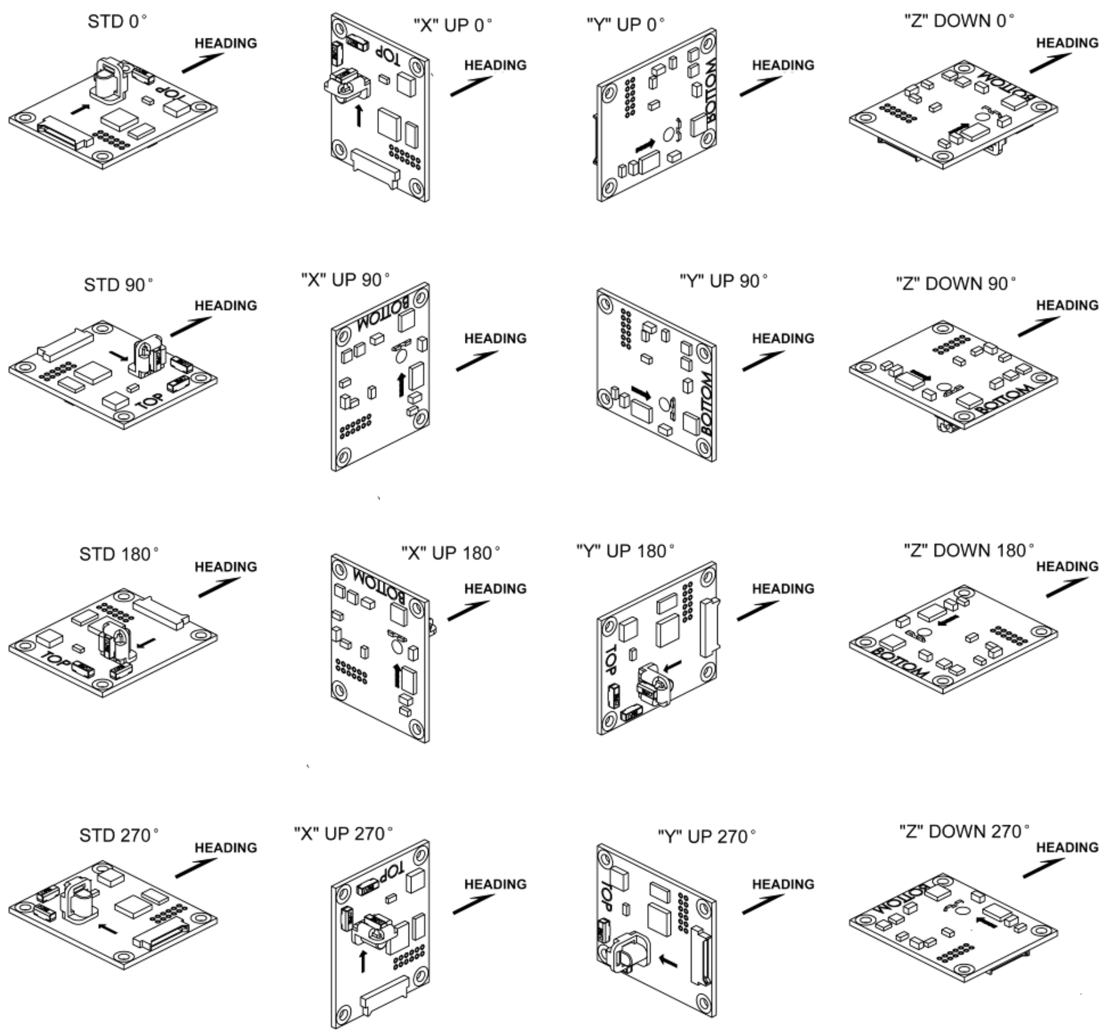


Figure 4-1: Positive & Negative Roll and Pitch Definition

4.3.2 Mounting Orientation

The TCM can be mounted in various orientations, as shown for the TCM XB in Figure 4-2. All reference points are based on the white silk-screened arrow on the top side of the board. The orientation should be programmed in the TCM using TCM Studio or the kSetConfig command. The default orientation is “STD 0°”.



Note: *TCM XB is shown. The Z axis sensor and the connector are on the module’s top surface, regardless of model.*

Figure 4-2: Mounting Orientations

5 User Calibration

The magnetic sensors in the TCM are calibrated at PNI's factory in a magnetically controlled environment. However sources of magnetic distortion positioned near the TCM 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 TCM in the host system and performing a user calibration. It is expected the sources of magnetic distortion remain fixed relative to the TCM's position within the host system. By performing a calibration, the TCM identifies the local sources of magnetic distortion and negates their effects from the overall reading to provide an accurate heading.

As with the magnetic sensor, the accelerometer in the TCM is calibrated at PNI's factory. But the accelerometer will gradually change over time, and the user either will need to periodically perform a user accelerometer calibration or return the unit to PNI for recalibration. As a general rule-of-thumb, the accelerometer should be recalibrated every 6 to 12 months. Unlike a magnetic calibration, the accelerometer may be calibrated outside the host system. Accelerometer calibration is more sensitive to noise or hand jitter than magnetic calibration, especially for subsequent use at high tilt angles. Because of this, ideally a stabilized fixture would be used for accelerometer calibration, although resting the unit against a stable surface often is sufficient.

Key Points:

- Magnetic calibration:
    - Requires incorporating the TCM into the host system to compensate for magnetic sourcing and distorting components with the user's system.
    - Allows for 4 different methods of calibration. Full-Range Calibration provides the highest heading accuracy, while 2D and Limited-Tilt Calibration support a limited range of motion during calibration. Hard-Iron-Only Calibration updates just the hard-iron coefficients with a relatively easy procedure.
- Accelerometer calibration requires rotating the TCM through a full sphere of coverage, but the TCM does not need to be incorporated into the user's system during calibration.
- If the TCM 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 and 3 accelerometer calibration coefficient sets can be saved.

5.1 Magnetic Calibration

Two fundamental types of magnetic distortion exist, hard-iron distortion and soft-iron distortion. A given component can exhibit both hard-iron and soft-iron distortions. These distortions are reviewed in the ensuing paragraphs, and are followed by discussions on temperature effects 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 TCM 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

The TCM measures the total magnetic field within its vicinity, which is a combination of Earth's magnetic field and local magnetic sources and distortions. While the TCM's calibration algorithms can compensate for local static magnetic sources, it is not possible to compensate for dynamic changes in the magnetic field. Consequently, it is recommended to keep the TCM away from dynamic magnetic fields. If this is not possible, then take measurements only when the state of the magnetic field is known. For example, if an electric motor is nearby take measurements only when the motor is off. Alternatively, different sets of magnetic calibration coefficients can be generated in advance for various states and then called when appropriate. Using the prior example, generate and use one set of coefficients for when the motor is off and another set for when the motor is on.

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 TCM 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 TCM 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 ±45°of tilt.

The following subsections provide instructions for performing a magnetic calibration of a TCM 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 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. Also, PNI recommends the location of the TCM remain fairly constant while only the orientation is changed.

Table 5-1: Magnetic Calibration Mode Summary

Calibration Mode	Accuracy	Tilt Range during Cal	Number of Samples	
			Minimum Recommend	Allowable Range
Full Range	0.3° rms	>±45°	12	10 – 18
2D Calibration	<2°	<±5°	12	10 – 18
Limited Tilt Range	<2° over 2x tilt range	±5° to ±45°	12	10 – 18
Hard Iron Only	Restores prior accuracy	>±3°	6	4 - 18

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

Section 6.4 outlines how to perform a calibration using Studio, while Section 7.3.10 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 TCM can be tilted ±45° 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 3 circles of evenly spaced points, as illustrated in Figure 5-1 and listed in Table 5-2. The pitch used in the second and third circles of the calibration should at least match the maximum and minimum pitch the device is expected to encounter in use.

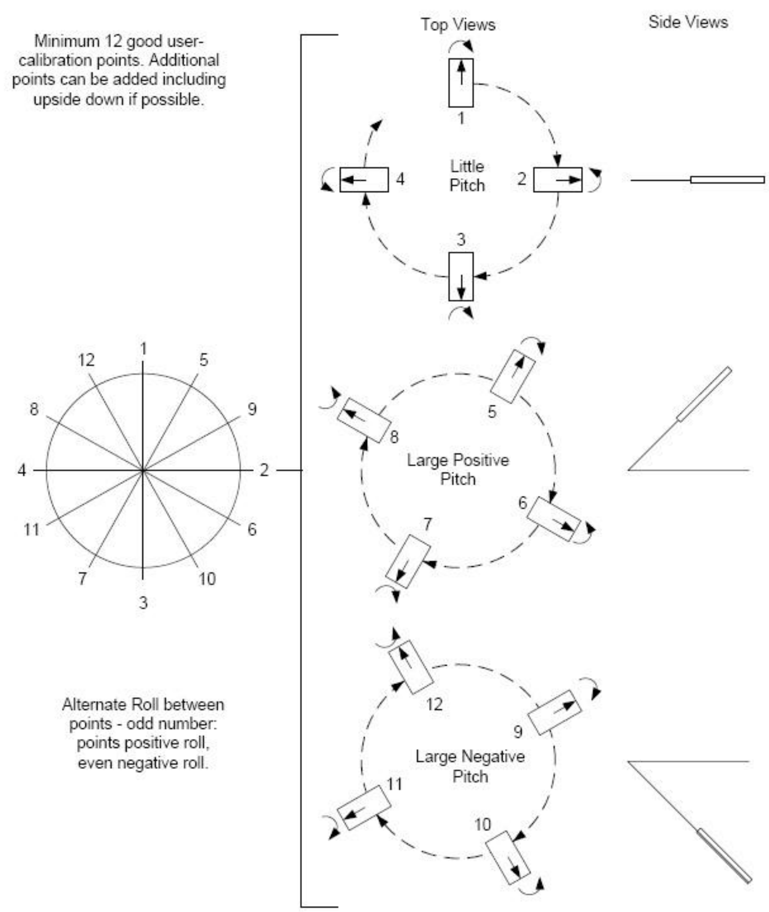


Figure 5-1: 12 Point Full-Range Calibration

Note: *While Figure 5-1 shows the location of the device changing, this is for illustration purposes and it is best for the location of the device to remain constant while only the orientation is changed.*

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

	Sample #	Yaw[1]	Pitch	Roll
First Circle				
	1	0°	±5°	30° to 40°
	2	90°	±5°	-30° to -40°
	3	180°	±5°	30° to 40°
	4	270°	±5°	-30° to -40°
Second Circle				
	5	30°	> +45°	30° to 40°
	6	120°	> +45°	-30° to -40°
	7	210°	> +45°	30° to 40°
	8	300°	> +45°	-30° to -40°
Third Circle				
	9	60°	< -45°	30° to 40°
	10	150°	< -45°	-30° to -40°
	11	240°	< -45°	30° to 40°
	12	330°	< -45°	-30° to -40°

Footnote:

1. Yaw listings are not absolute heading directions but rather relative heading referenced to the first sample.

5.1.2 2D Calibration

A 2D Calibration is intended for very low tilt operation (<5°) where calibrating the TCM 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 #	Yaw	Pitch[1]	Roll[1]
1	0°	0°	0°
2	30°	max. negative	max. negative
3	60°	0°	0°
4	90°	max. positive	max. positive
5	120°	0°	0°
6	150°	max. negative	max. negative
7	180°	0°	0°
8	210°	max. positive	max. positive
9	240°	0°	0°
10	270°	max. negative	max. negative
11	300°	0°	0°
12	330°	max. positive	max. positive

Footnote:

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

5.1.3 Limited Tilt Range Calibration

A Limited Tilt Range Calibration is recommended when 45° of tilt isn't feasible, but >5° of tilt is possible. It provides both hard-iron and softiron 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°
6	270°	0°	0°
Second Circle			
7	45°	> +5°	> +5°
8	135°	> +5°	> +5°
11	225°	> +5°	> +5°
12	315°	> +5°	> +5°
Third Circle			
13	45°	< -5°	< -5°
14	135°	< -5°	< -5°
17	225°	< -5°	< -5°
18	315°	< -5°	< -5°

Note that a similar and acceptable alternative pattern would be to follow the recommended 12 point Full-Range Calibration pattern, but substituting the >±45° of pitch with whatever pitch can be achieved and the ±10° to ±20° or roll with whatever roll can be achieved up to these limits.

5.1.4 Hard-Iron-Only Calibration

It is not uncommon for the hard-iron magnetic distortions around the TCM 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 TCM for hard-iron effects, and generally is effective for operation and calibration in the tilt range of 3° or more (≥45° is preferred). 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 5-5: 6 Point Hard-Iron-Only Calibration Pattern

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

Footnote:

1. For best results, the tilt experienced during calibration should match that experienced in service. For example, if the TCM will be subject to ±45° of pitch and roll when in service, then "max negative" should be -45° and "max positive" should be +45°.

5.2 Accelerometer Calibration

The TCM uses a MEMS accelerometer to measure the attitude of the compass. This data is output as pitch and roll data. Additionally, the accelerometer data is critical for establishing an accurate heading reading when the TCM 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 TCM algorithms assume the accelerometer only measures the gravitational field. If the TCM is accelerating, this will result in the TCM calculating an inaccurate gravitational vector, which in turn will result in an inaccurate heading reading. For this reason, the TCM should be stationary when taking a measurement.

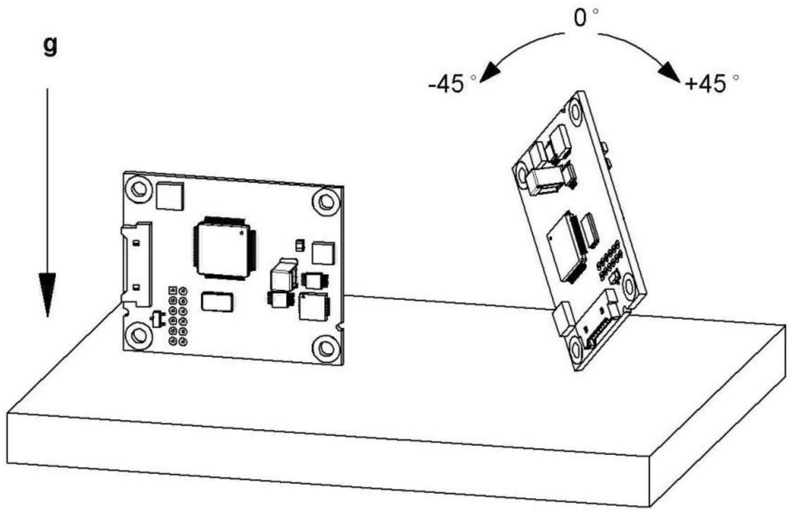
As previously mentioned, PNI calibrates the accelerometer in its factory prior to shipment. But over time the bias and offset of the accelerometer will drift. For this reason PNI recommends the accelerometer be recalibrated every 6 to 12 months. The user may return the TCM to PNI for accelerometer calibration, or the user may perform a user accelerometer calibration. The remainder of this section covers the user accelerometer calibration.

5.2.1 Accelerometer-Only Calibration

The requirements for a good user accelerometer calibration differ significantly from the requirements for a good magnetic calibration. Specifically, a good accelerometer calibration involves the TCM experiencing a wide range of pitch and roll values, preferably seeing both ±180° of pitch and ±180° of roll. Also, it is necessary for the TCM to be very still during an accelerometer calibration. If possible, PNI recommends using a fixture to hold the device during calibration, although resting the TCM on a hard surface normally is sufficient.

The accelerometer either can be calibrated while mounted in the host system or it may be removed and calibrated outside the system. The advantage of performing the calibration while mounted in the host system is the user does not need to remove the TCM from the system, which can be burdensome, and a simultaneous Mag-and-Accel Calibration may be appropriate. The advantage of performing the calibration outside of the system is it may be much simpler to obtain the desired range of pitch and roll.

Figure 5-2 shows the two basic starting positions for the recommended 18-point calibration pattern. Starting with the TCM as shown on the left in Figure 5-2, rotate the device about its z axis such that it sits on each of its 4 edges, taking one calibration sample on each edge. Then place the TCM flat on the surface and take a calibration sample, then flip it over (roll it 180°) and take another sample. Next, starting with the TCM as shown on the right, take a calibration point with it being vertical (0°). Now tilt the TCM back 45° and take another calibration point (+45°), then tilt the device forward 45° and take another calibration point (-45°). Repeat this 3-point calibration process for the TCM with it resting on each of its 4 corners. Note that it is possible to perform an Accelerometer Calibration with as few as 12 sample points, although it generally is more difficult to obtain a good calibration with just 12 sample points. Also, the maximum number of calibration points is 18.



Note: While the TCM is shown removed from the host system, the Accelerometer Calibration may be performed with the TCM mounted in the host system.

Figure 5-2: Accelerometer Calibration Starting Orientations

5.2.2 Mag-and-Accel Calibration

The TCM allows for a simultaneous magnetometer and accelerometer calibration. This requires a full-coverage calibration pattern, physically stable measurements, and installation in the user's system so the host system's magnetic signature is present. PNI recommends 18 to 32 calibration points for a Mag-and-Accel Calibration. The Accelerometer-Only Calibration pattern discussed in Section 5.2 will work for a Mag-and-Accel Calibration. Optimal performance is obtained when all rotations of the TCM are performed towards magnetic north to achieve the widest possible magnetic field distribution.

Note that combining calibrations only makes sense if all the host system's magnetic distortions (steel structures or batteries, for instance) are present and fixed relative to the module when calibrating. If an Accelerometer-Only Calibration is performed, the user's system distortions are not relevant, which allows the TCM to be removed from the host system in order to perform the Accelerometer-Only Calibration.

6 Operation with TCM Studio

TCM Studio puts an easy-to-use, graphical-user interface (GUI) onto the binary command language used by the TCM. TCM Studio is intended for evaluating, demonstrating, and calibrating the TCM module. The program includes the ability to log and save the outputs from the TCM to a file for off-line evaluation. Check the PNI website for the latest TCM Studio updates at www.pnicorp.com.

Note: *TCM Studio v3.X and higher is compatible with the TCM XB, TCM MB and legacy TCM 6, but not other legacy TCM models. The TCM XB also will work with TCM Studio v3 and higher, while the TCM MB will work with TCM Studio v4 and higher. The version of Studio is identified in the upper left corner of the GUI.*

The TCM Studio evaluation software communicates with the TCM through the RS232 serial port of a computer. The TCM MB requires a user-supplied level shifter to make it compatible with the computer's RS232 interface.

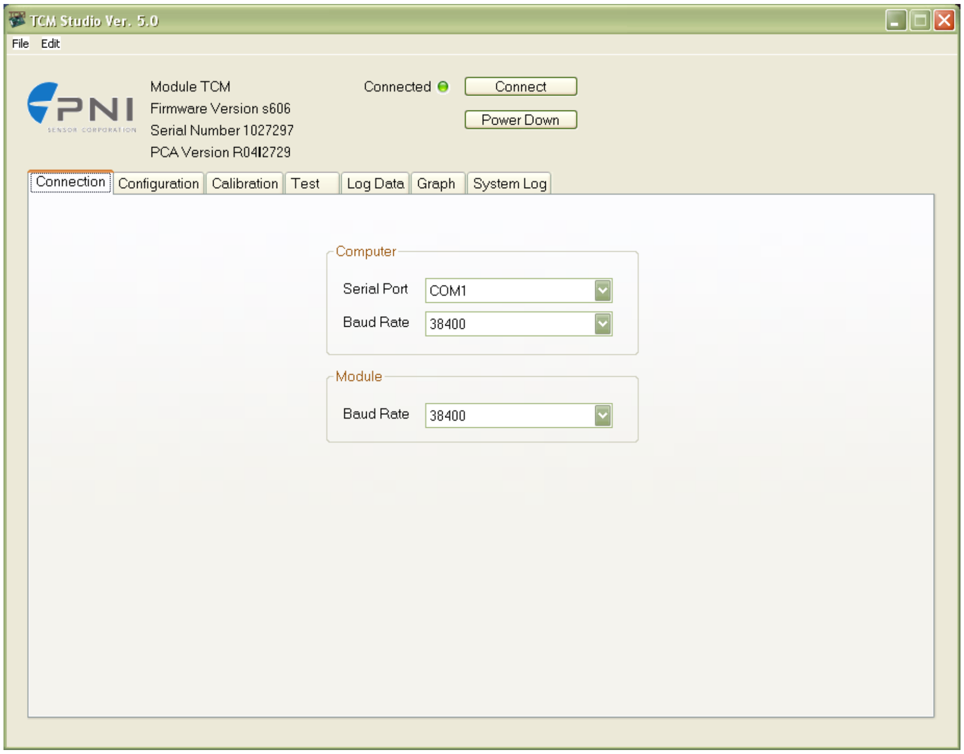
6.1 Installation

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

For Windows computers, copy the TCMStudio.msi file onto your computer. Then, open the file and step through the Setup Wizard.

For Mac computers, copy the TCMStudio.zip file onto your computer. This automatically places the application in the working directory of your computer. The Quesa plug-in, also in the .zip file, needs to be moved to /Library/CFMSupport, if it is not already there.

6.2 Connection Tab



6.2.1 Initial Connection

If using the PNI dual-connectorized cable, ensure the batteries are well-charged.

- Select the serial port the module is plugged into, which is generally COM 1.
- Select 38400 as the baud rate.
- Click the <Connect> button if the connection is not automatic.

Once a connection is made the “Connected” light will turn green and the module’s firmware version, serial number, and PCA version will be displayed in the header section.

6.2.2 Changing Baud Rate

To change the baud rate:

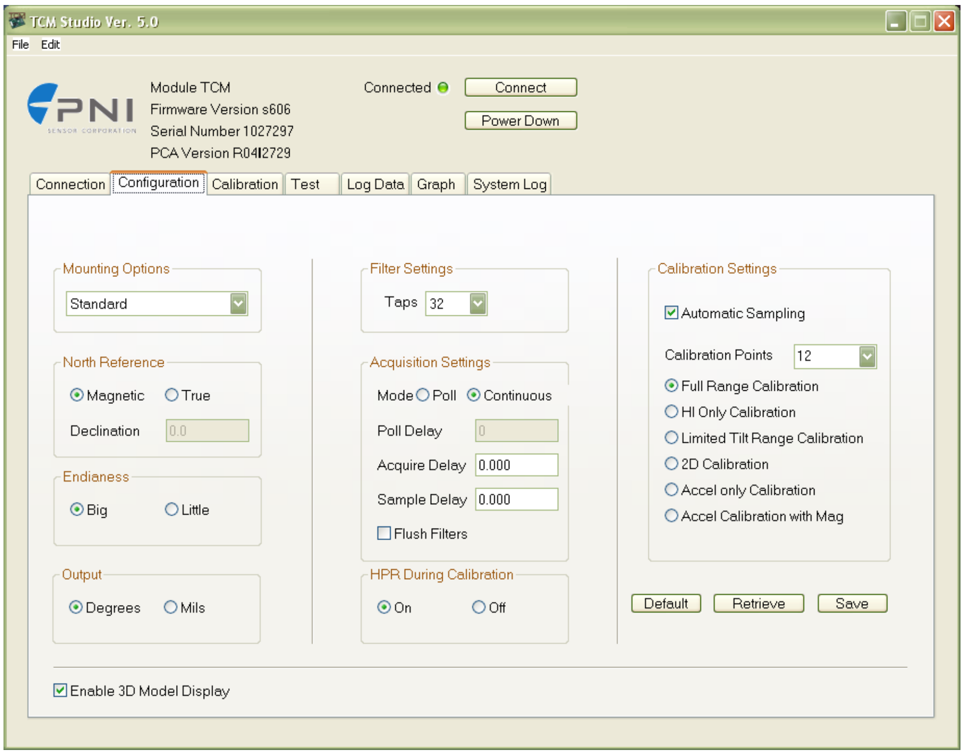
- In the Module window, select the new baud rate for the module.
- Click the <Power Down> button. The button will change to read <Power Up>.
- In the Computer window, select same baud rate for the computer.
- Click the <Power Up> button. The button will revert back to <Power Down>.

Note: *While the TCM can operate at a baud rate of 230400, a PC serial port normally will not operate this fast.*

6.2.3 Changing Modules

Once a connection has been made, TCM Studio will recall the last settings. If a different module is used, click the <Connect> button once the new module is attached. This will reestablish a connection, assuming the module baud rate is unchanged.

6.3 Configuration Tab



Note: *No settings will be changed in the module until the <SAVE> button has been selected.*

6.3.1 Mounting Options

TCM Studio supports 16 mounting orientations, as illustrated previously in Figure 4-2. The descriptions in TCM Studio are slightly different from those shown in Figure 4-2, and the relationship between the two sets of descriptions is given below.

Table 6-1: Mounting Orientations

TCM Studio Description	Figure 4-2 Description	TCM Studio Description	Figure 4-2 Description
Standard	STD 0°	Y Sensor Up	"Y" Up 0°
Standard 90 Degrees	STD 90°	Y Sensor Up Plus 90 Degrees	"Y" Up 90°
Standard 180 Degrees	STD 180°	Y Sensor Up Plus 180 Degrees	"Y" Up 180°
Standard 270 Degrees	STD 270°	Y Sensor Up Plus 270 Degrees	"Y" Up 270°
X Sensor Up	"X" Up 0°	Z Sensor Down	"Z" Down 0°
X Sensor Up Plus 90 Degrees	"X" Up 90°	Z Sensor Down Plus 90 Degrees	"Z" Down 90°
X Sensor Up Plus 180 Degrees	"X" Up 180°	Z Sensor Down Plus 180 Degrees	"Z" Down 180°
X Sensor Up Plus 270 Degrees	"X" Up 270°	Z Sensor Up Plus 270 Degrees	"Z" Down 270°

6.3.2 North Reference

Declination, also called magnetic variation, is the difference between true and magnetic north. It is measured in degrees east or west of true north. Correcting for declination is accomplished by storing the correct declination angle, and then changing the heading reference from magnetic north to true north. Declination angles vary throughout the world, and change very slowly over time. For the greatest possible accuracy, go to the National Geophysical Data Center web page below to get the declination angle based on your latitude and longitude:

http://www.ngdc.noaa.gov/geomagmodels/Declination.jsp

Magnetic

When the <Magnetic> button is selected, heading will be relative to magnetic north.

True

When the <True> button is selected, heading will be relative to true north. In this case, the declination needs to be set in the "Declination" window.

6.3.3 Endianess

Select either the <Big> or <Little> Endian button. The default setting is <Big>. See Sections 7.2 and 7.3 for additional information.

6.3.4 Output

The TCM module can output heading, pitch, and roll in either degrees or mils. Click either the <Degrees> or <Mils> button. The default is <Degrees>. (There are 6400 mils in a circle, such that 1 degree = 17.7778 mils and 1 mil = 0.05625 degree.)

6.3.5 Enable 3D Model

TCM Studio's Test tab includes a live-action 3-D rendering of a helicopter. Some computer systems may not have the graphics capability to render the 3D Model, for this reason it may be necessary to turn off this feature.

6.3.6 Filter Setting (Taps)

The TCM incorporates a finite impulse response (FIR) filter to effectively provide a more stable heading reading. The number of taps (or samples) represents the amount of filtering to be performed. The user should select either 0, 4, 8, 16, or 32 taps, 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.

6.3.7 Acquisition Settings

Mode

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

Poll Delay

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

Acquire Delay

The Acquire Delay sets the time between samples taken by the module, in seconds. This is an internal setting that is NOT tied to the time with which the module transmits data to TCM Studio or the host system. Generally speaking, the Acquire Delay is either set to 0, in which case the TCM is constantly sampling or set to equal either the Poll Delay or Sample Delay values. The advantage of running with an Acquire Delay of 0 is that the FIR filter can run with a relatively high Tap value to provide stable and timely data. The advantage of using a greater Acquire Delay is that power consumption can be reduced, assuming the Sample or Poll Delay are no less than the Acquire Delay.

Sample Delay

The Sample Delay is relevant when Continuous Mode is selected. It is the time delay, in seconds, between completion of the TCM sending one set of data and the start of sending the next sample set. If the delay is set to 0, then the TCM 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.

Flush Filters

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 Sample 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.

6.3.8 HPR During Calibration

When the <On> button is selected, heading, pitch, and roll will be output on the Calibration tab during a calibration.

6.3.9 Calibration Settings

Automatic Sampling

When selected, the module will take a sample point once the minimum change and stability requirements have been satisfied. If the user wants to have more control over when the point will be taken, then Auto Sampling should be deselected. Once deselected, the <Take Sample> button on the Calibration tab will be active. Selecting the <Take Sample> button will indicate to the module to take a sample once the minimum change and stability requirements are met.

Calibration Points

Select the number of points to take during a calibration. The minimum recommended number of points for an initial magnetic calibration is 12, although a Hard-Iron-Only (re)Calibration can be performed with only 6 recommended samples. The TCM will need to be rotated through at least 180° in the horizontal plane with a minimum of at least 1 positive and 1 negative Pitch and at least 1 positive and 1 negative Roll as part of the 12 points.

Calibration Method Buttons

Full Range Calibration - recommended calibration method when >45° of tilt is possible. The minimum recommended number of calibration points is 12.

HI Only Calibration - serves as a hard iron recalibration to a prior calibration. If the hard iron distortion around the module has changed, this calibration can bring the module back into specification. The minimum recommended number of calibration points is 6.

Limited Tilt Range Calibration - recommended calibration method when >5° of tilt calibration is available, but tilt is restricted to <45°. (i.e. Full-Range Calibration is not possible.) The minimum recommended number of calibration points is 12.

2D Calibration - Recommended when the available tilt range is limited to ≤5°. The minimum recommended number of calibration points is 12.

Accel Only Calibration – Select this when only an accelerometer calibration will be performed. The minimum recommended number of calibration points is 18.

Accel Calibration with Mag – The user should select this when magnetometer and accelerometer calibration will be performed simultaneously. The minimum recommended number of calibration points is 18.

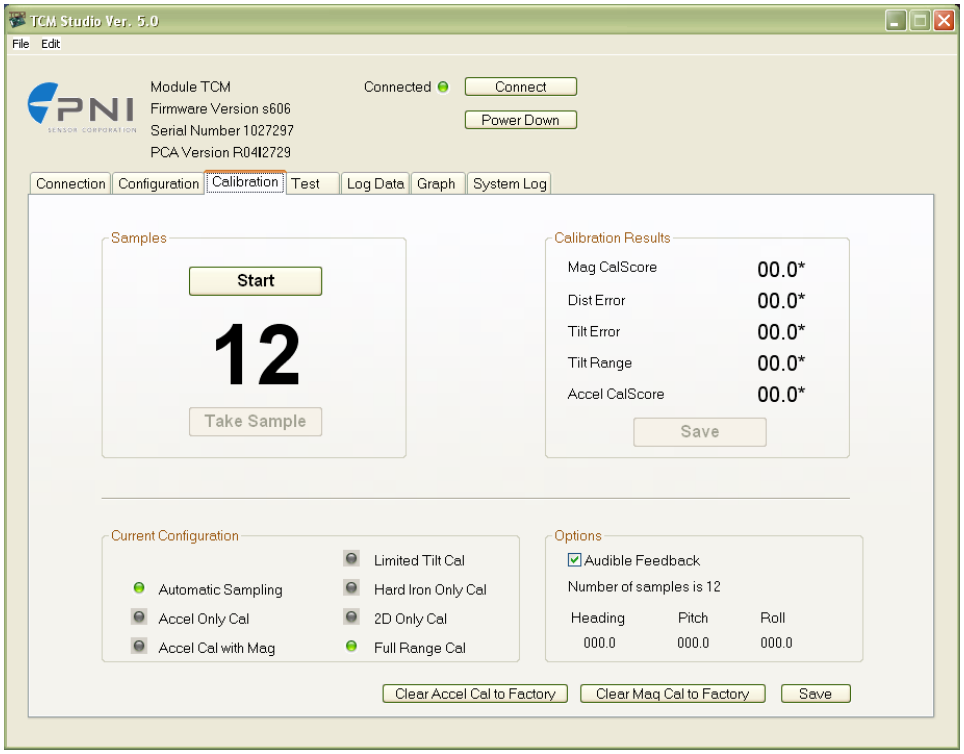
6.3.10 Default

Clicking this button restores the TCM Studio program to the factory default settings.

6.3.11 Retrieve

Clicking on this button causes TCM Studio to read the settings from the module and display them on the screen.

6.4 Calibration Tab



Note: *The default settings are recommended for the highest accuracy and quality of calibration.*

6.4.1 Samples

Before proceeding, refer to Section 5 for the recommended calibration procedure corresponding to the calibration method selected on the Configuration tab.

Clicking the <Start> button begins the calibration process.

If "Automatic Sampling" is not checked on the Configuration tab, 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, as set on the Configuration tab, is taken while changing the orientation of the module between samples as discussed in Section 5.

If "Automatic Sampling" is checked, the module will need to be held steady for a short time and then a sample automatically will be taken. Once the window indicates the next number, the module'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.

6.4.2 Calibration Results

Once a calibration is complete, the "Calibration Results" window will indicate the quality of the calibration. This may take a minute or more to populate. The primary purpose of these scores is to confirm the calibration was successful, as indicated by a low Mag and/or Accel CalScore. The other scores provide information that may assist in improving the CalScore, should it be unacceptably high. If either CalScore is too high, click the <Start> button to begin a new calibration. If the calibration is acceptable, click the <Save> button to save the calibration to the module's flash. If the <Save> button is not selected then the module will need to be recalibrated after a power cycle.

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

Mag CalScore

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 Dist Error and Tilt Error and still have a rather high Mag CalScore value. The most likely reason for this is the TCM is close to a source of local magnetic distortion that is not fixed with respect to the module.

Dist Error

Indicates the quality of the sample point distribution, primarily looking for an even yaw distribution. Significant clumping or a lack of sample points in a particular section can result in a poor score. The score should be <1 and close to 0.

Tilt Error

Indicates the contribution to the Mag CalScore caused by tilt or lack thereof, and takes into account the calibration method. The score should be <1 and close to 0.

Tilt Range

This reports the larger of either half the full pitch range or half the full roll range of sample points. For example, if the module is pitched +10° to -20º, and rolled +25º to -15º, the Tilt Range value would be 20º, as derived half the full roll range. For Full-Range Calibration and Hard-Iron-Only Calibration, this should be ≥45°. For 2D Calibration, this ideally should be ≈2°. For Limited Tilt Range Calibration the value should be as large a possible given the user's constraints.

Accel CalScore

Represents the over-riding indicator of the quality of the accelerometer calibration. Acceptable scores will be <1.

6.4.3 Current Configuration

These indicators mimic the pertinent selections made on the Configuration tab.

6.4.4 Options

This window indicates how many samples are to be taken and provides real time heading, pitch, and roll information if "HPR During Calibration" is set to <On>, both as defined on the Configuration tab.

Audible Feedback

If selected TCM Studio will give an audible signal once a calibration point has been taken. Note that an audible signal also will occur when the <Start> button is clicked, but no data will be taken.

6.4.5 Clear

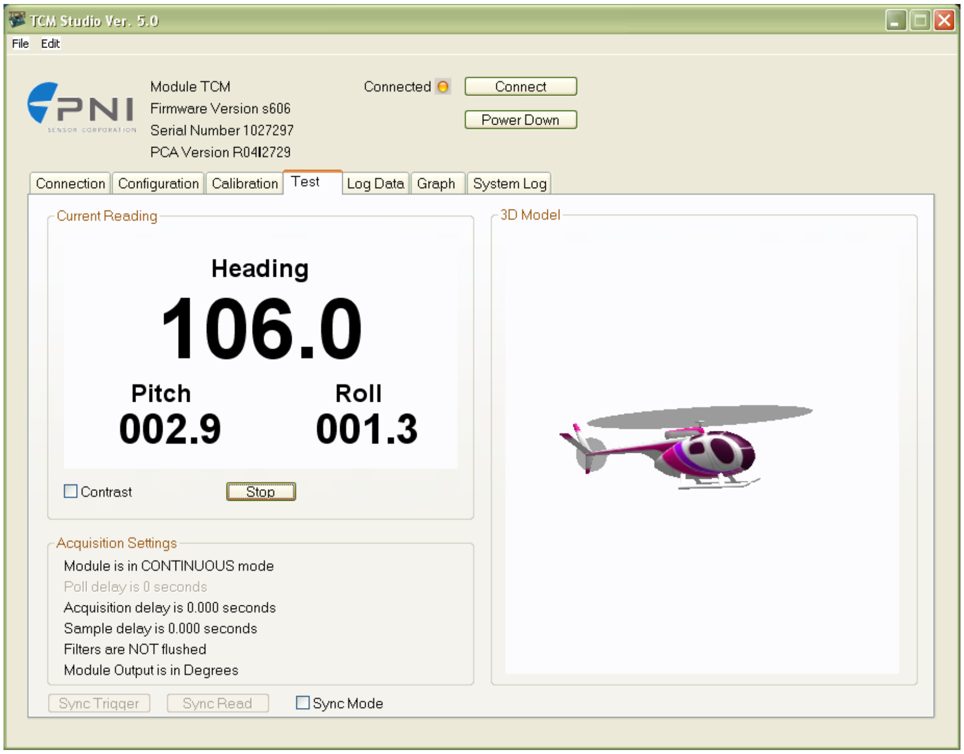
Clear Mag Cal to Factory

This button clears the user's calibration of the magnetometers. Once selected, the module reverts to its factory magnetometer calibration. To save this action in nonvolatile memory, click the <Save> button. It is not necessary to clear the current calibration in order to perform a new calibration.

Clear Accel Cal to Factory

This button clears the user's calibration of the accelerometer. Once selected, the module reverts back to its factory accelerometer calibration. To save this action in non-volatile memory, click the <Save> button. It is not necessary to clear the current calibration in order to perform a new calibration.

6.5 Test Tab



6.5.1 Current Reading

Once the <Go> button is selected the module will begin outputting heading, pitch and roll information. Selecting the <Stop> button or changing tabs will halt the output of the module.

Contrast

Selecting this box sets the "Current Readings" window to have yellow lettering on a black background, rather than black lettering on a white background.

6.5.2 3D Model

The helicopter will follow the movement of the TCM and give a visual representation of the module's orientation, assuming the "Enable 3D Model Display" box is selected on the Configuration tab.

6.5.3 Acquisition Settings

These indicators mimic the pertinent selections made on the Configuration tab.

6.5.4 Sync Mode

Sync Mode enables the module to stay in Sleep Mode until the user's system sends a trigger to report data. When so triggered, the TCM will wake up, report data once, then return to Sleep Mode. One application of this is to lower power consumption. Another use of the Sync Mode is to trigger a reading during an interval when local magnetic sources are well understood. For instance, if a system has considerable magnetic noise due to nearby motors, the Synch Mode can be used to take measurements when the motors are turned off.

Enter Sync Mode

On the Test tab, above the tabs and 3D model, click the "Sync Mode" check box to enter Sync Mode.

Sync Mode Output

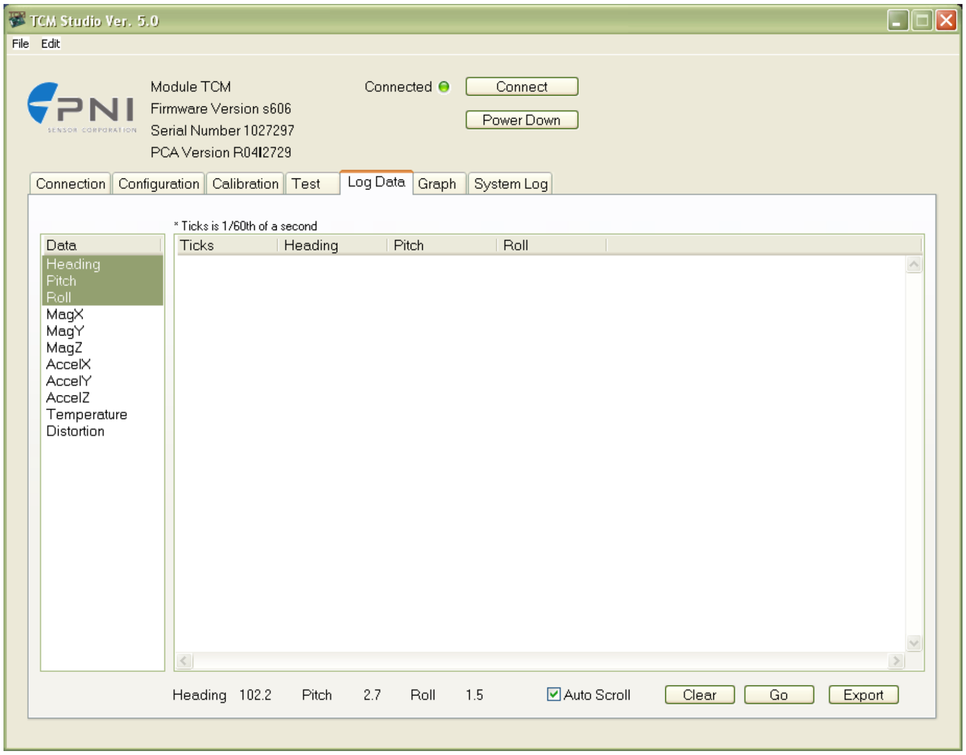
To retrieve the first reading, click the <Sync Read> button. Heading, pitch and roll information will be displayed on Current Reading window. If the "Enable 3D Model Display" box is selected on the Configuration tab, then the helicopter will follow the movement as well. The module will enter Sleep Mode after outputting the heading, pitch, and roll information. To obtain subsequent readings, the user should first click on the <Sync Trigger> button to wake up the module and then click on the <Sync Read> button to get the readings, after which the module will return to sleep.

Exit Sync Mode

Click on the <Sync Trigger> button and then uncheck the "Sync Mode" check box to exit Sync Mode.

Note that <Sync Trigger> sends a 0xFF signal as an external interrupt to wake up the module. This is not done for the first reading as the module is already awake.

6.6 Log Data Tab

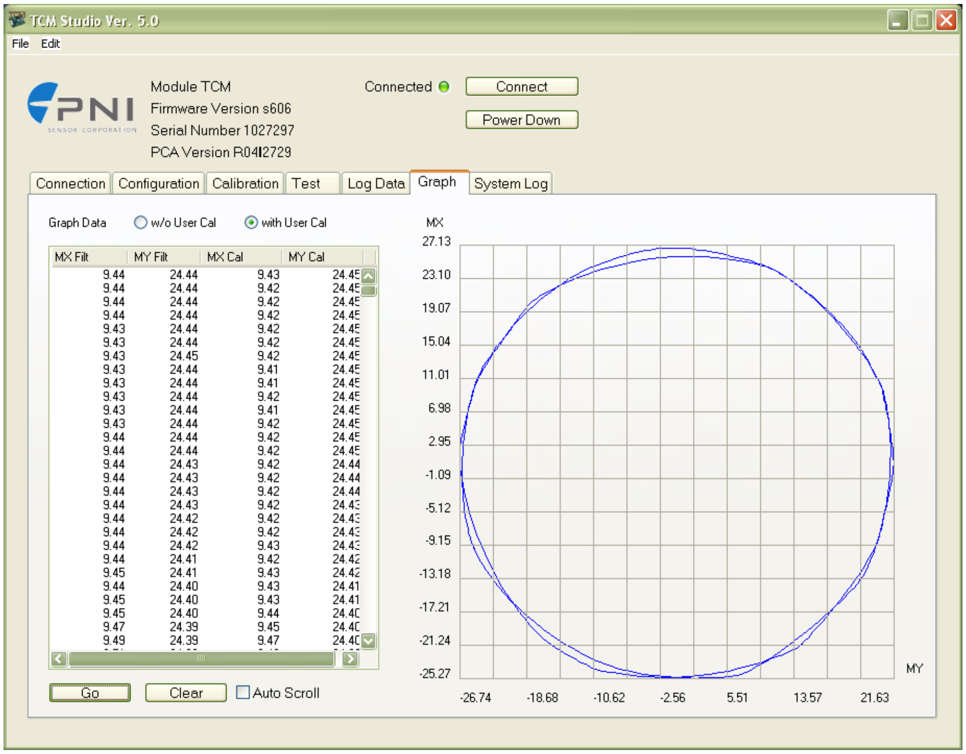


TCM Studio can capture measurement data and then export it to a text file. To acquire data and export it, follow the procedure below:

- Select the parameters you wish to log in the "Data" window. Use Shift -Click and Ctrl-Click to select multiple items. In the screen shot above, "Heading", "Pitch", and "Roll" were selected.
- Click the <Go> button to start logging. The <Go> 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.

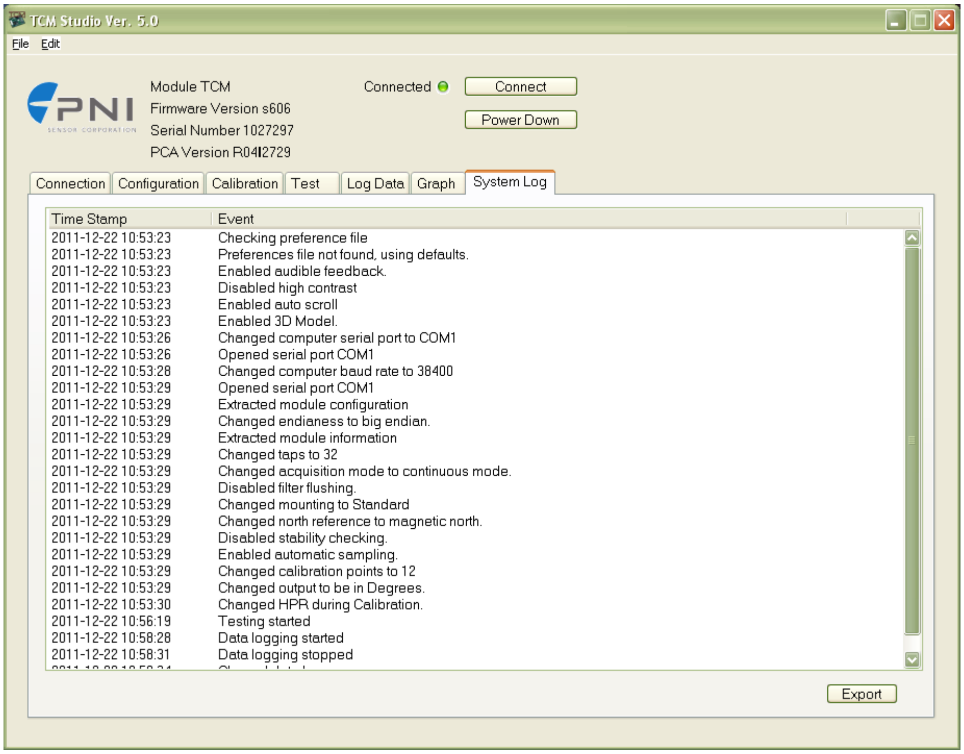
Note: *The data logger use ticks for time reference. A tick is 1/60 second.*

6.7 Graph Tab



The graph provides a 2-axis (X,Y) plot of the measured field strength. If "w/o User Cal" graph data is selected, the plot and data provide magnetic field strength measurements after the FIR taps are applied, but prior to applying the user calibration coefficients. If "with User Cal" graph data is selected, the plot and data provide data after applying the FIR filter and the user calibration coefficients. The sample plot shows a 360° rotation in the horizontal plane, with both "w/o User Cal" and "with User Cal" selected. The offset between these two plots represents the effect of the calibration coefficients. The graph can be used to visually see hard and soft iron effects within the environment measured by the TCM, as well as corrected output after a user calibration has been performed.

6.8 System Log Tab



The System Log tab shows all communication between TCM Studio and the TCM module since launching TCM Studio. Closing TCM 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 TCM utilizes a binary communication protocol, where the communication parameters should be configured as follows:

Table 7-1: UART 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:

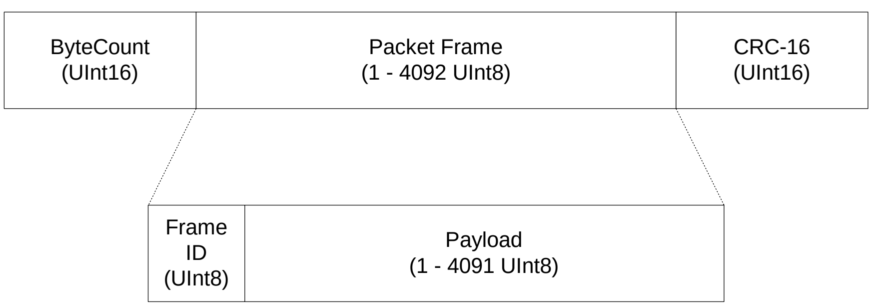


Figure 7-1: Datagram Structure

The ByteCount is the total number of bytes in the packet including the CRC-16 checksum. 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	CalOption (2D Calibration)	Checksum

7.2 Parameter Formats

Note: *Floating-point based parameters conform to ANSIring/IEEE Std 754-1985. Please refer to the Standard for more information. PNI also recommends refer to the user's compiler instructions to understand how the compiler implements floating-point format.*

64-Bit Floating Point (Float64)

The 64-bit float format is given below in big Endian. In little Endian, the bytes are in reverse order in 4 byte groups. (eg. big Endian: ABCD EFGH; little Endian: DCBA HGFE).

63	62 52	51 0
S	Exponent	Mantissa

The value (v) is determined as: "if and only if" $0 < Exponent < 2047$, then

$$v = (-1)*S*2(Exponent-1023)*1.Mantissa$$

32-Bit Floating Point (Float32)

Shown below is the 32-bit float format in big Endian. In little Endian format, the 4 bytes are in reverse order, with LSB first.

31	30 23	22 0
S	Exponent	Mantissa

The value (v) is determined as: "if and only if" $0 < Exponent < 255$, then

$$v = (-1)*S*2(Exponent-127)*1.Mantissa$$

Signed 32-Bit Integer (SInt32)

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

31 24	23 16	15 8	7 0
msb			lsb

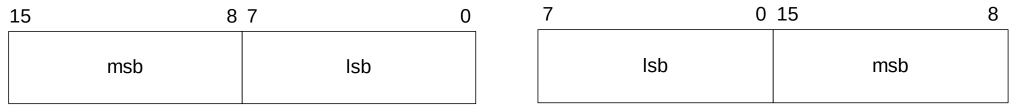
Big Endian

7 0	15 8	23 16	31 24
lsb			msb

Little Endian

Signed 16-Bit Integer (SInt16)

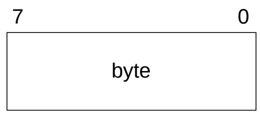
SInt16-based parameters are signed 16-bit numbers, in 2's compliment. Bit 15 represents the sign of the value, where 0=positive and 1=negative.



Big Endian   Little Endian

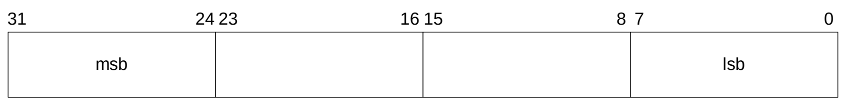
Signed 8-Bit Integer (SInt8)

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

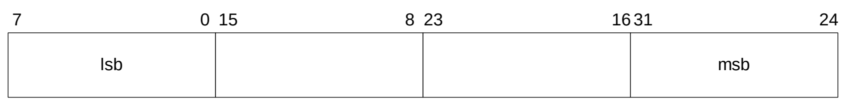


Unsigned 32-Bit Integer (UInt32)

UInt32-based parameters are unsigned 32-bit numbers.



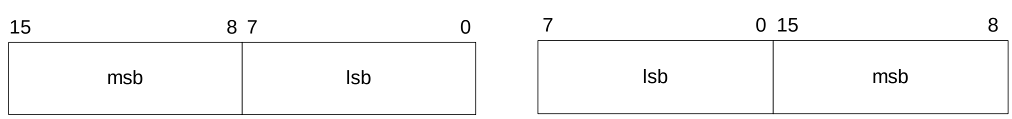
Big Endian



Little Endian

Unsigned 16-Bit Integer (UInt16)

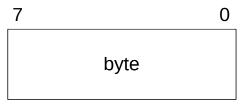
UInt16-based parameters are unsigned 16-bit numbers.



Big Endian   Little Endian

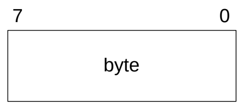
Unsigned 8-Bit Integer (UInt8)

UInt8-based parameters are unsigned 8-bit numbers.



Boolean

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



7.3 Commands & Communication Frames

Table 7-2, below, provides the TCM's command set.

Table 7-2: TCM Command Set

Frame ID_d	Command	Description
1	kGetModInfo	Queries the device's type and firmware revision.
2	kGetModInfoResp	Response to kGetModInfo
3	kSetDataComponents	Sets the data components to be output.
4	kGetData	Queries the TCM for data
5	kGetDataResp	Response to kGetData
6	kSetConfig	Sets internal configurations in TCM
7	kGetConfig	Queries TCM for the current internal configuration
8	kGetConfigResp	Response to kGetConfig
9	kSave	Saves the current internal configuration and any new user calibration coefficients to non-volatile memory.
10	kStartCal	Commands the TCM to start user calibration
11	kStopCal	Commands the TCM to stop user calibration
12	kSetFIRFilters	Sets the FIR filter settings for the magnetometer & accelerometer sensors.
13	kGetFIRFilters	Queries for the FIR filter settings for the magnetometer & accelerometer sensors.
14	kGetFIRFiltersResp	Contains the FIR filter settings for the magnetometer & accelerometer sensors.
15	kPowerDown	Powers down the module
16	kSaveDone	Response to kSave

17	kUserCalSampleCount	Sent from the TCM after taking a calibration sample point
18	kCalScore	Contains the calibration score
19	kSetConfigDone	Response to kSetConfig
20	kSetFIRFiltersDone	Response to kSetFIRFilters
21	kStartContinuousMode	Commands the TCM to output data at a fixed interval
22	kStopContinuousMode	Stops data output when in Continuous Mode
23	kPowerUpDone	Confirms the TCM has received a signal to power up
24	kSetAcqParams	Sets the sensor acquisition parameters
25	kGetAcqParams	Queries for the sensor acquisition parameters
26	kSetAcqParamsDone	Response to kSetAcqParams
27	kGetAcqParamsResp	Response to kGetAcqParams
28	kPowerDownDone	Response to kPowerDown
29	kFactoryMagCoeff	Resets magnetometer calibration coefficients to original factory-established values
30	kFactoryMagCoeffDone	Response to kFactoryMagCoeff
31	kTakeUserCalSample	Commands the TCM to take a sample during user calibration
36	kFactoryIAccelCoeff	Resets accelerometer calibration coefficients to original factory-established values
37	kFactoryAccelCoeffDone	Respond to kFactoryAccelCoeff
46	kSetSyncMode	Sets whether the TCM is in normal or Sync Mode
47	kSetSyncModeResp	Response to kSetSyncMode
49	kSyncRead	Queries the module for data in Sync Mode

7.3.1 kGetModInfo (frame ID 1_d)

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

7.3.2 kGetModInfoResp (frame ID 2_d)

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

Payload

Type	Revision
UInt32	UInt32

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 43 4D 35 31

32 30 38 C7 87” can be decoded to read “TCM5 1208”. Also, the TCM XB is referenced as Type “TCM6” since the number of Type characters is limited to 4.

7.3.3 kSetDataComponents (frame ID 3_d)

This frame defines what data is output when kGetData is sent. Table 7-3 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.

Payload

ID Count	ID 1	ID 2	ID 3	……….
UInt8	UInt8	UInt8	UInt8	

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

Payload

2	5	24
ID Count	Heading ID	Pitch ID

When querying for data (kGetData frame), the sequence of the data component output follows the sequence of the data component IDs as set in this frame.

Table 7-3: Component Identifiers

Component	Component ID_d	Format	Units
kHeading	5	Float32	degrees
kPitch	24	Float32	degrees
kRoll	25	Float32	degrees
kTemperature	7	Float32	˚ Celsius
kDistortion	8	Boolean	True or False (Default)
kCalStatus	9	Boolean	True or False (Default)
kAccelX	21	Float32	G
kAccelY	22	Float32	G
kAccelZ	23	Float32	G
kMagX	27	Float32	µT
kMagY	28	Float32	µT
kMagZ	29	Float32	µT

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 in degrees Celsius, and has an accuracy of ±3° C.

kDistortion (Component ID 8_d)

This flag indicates at least one magnetometer axis reading is beyond ±125 µT.

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.

7.3.4 kGetData (frame ID 4_d)

If the TCM is configured to operate in Poll Acquisition Mode, as defined by kSetAcqParams, then this frame requests a single measurement data set. The frame has no payload. The response is kGetDataResp.

7.3.5 kGetDataResp (frame ID 5_d)

The response to kGetData, kStartContinuousMode, and kSyncRead 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 pitch are set to be output per the kSetDataComponents command, the payload would look like:

Payload

2	5	359.9	24	10.5
ID Count	Heading ID	Heading (Float32)	Pitch ID	Pitch Output (Float32)

7.3.6 kSetConfig (frame ID 6_d)

This frame sets internal configurations in the TCM. 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.

Payload

Config ID	Value
UInt8	ID Specific

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

Payload

1	10.0
Declination ID	Declination Angle (Float32)

Table 7-4: 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[1]	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 – 32	12
kUserCalAutoSampling	13	Boolean	True or False	True
kBaudRate	14	UInt8	0 – 300 1 – 600 2 – 1200 3 – 1800 4 – 2400 5 – 3600 6 – 4800 7 – 7200 8 – 9600 9 – 14400 10 – 19200 11 – 28800 12 – 38400 13 – 57600 14 - 115200	12
kMilOutput	15	Boolean	True or False	False
kHPRDuringCal	16	Boolean	True or False	True
kMagCoeffSet	18	UInt32	0 - 7	0
kAccelCoeffSet	19	UInt32	0 - 2	0

Note:

1. Refer to Figure 4-2 for additional information on mounting orientations.

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.

kMountingRef (Config. ID 10_d)

This sets the reference orientation for the module. Please refer to and 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 7-5 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 TCM's heading accuracy specification, while less than this may make it difficult to meet specification. See Section 5 for additional information.

Table 7-5: Sample Points

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

kUserCalAutoSampling (Config. ID 13_d)

This flag is used during user calibration. If set to TRUE, the module automatically takes calibration sample points once the minimum change requirement is met. If set to FALSE, the module 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 point 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.

kMilOutput (Config. ID 15_d)

This flag sets the heading, pitch and roll output to mils. By default, kMilOutput is set to FALSE and the heading, pitch and roll output are in degrees. Note that 360 degrees = 6400 mils, such that 1 degree = 17.778 mils or 1 mil = 0.05625 degree.

kDataCal (Config. ID 16_d)

This flag sets whether or not heading, pitch, and roll data are output simultaneously while the TCM 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 setting provides the flexibility to store up to eight (8) sets of magnetometer calibration coefficients in the module. These different coefficient sets can be used for storing coefficients for varying conditions, such as when a door is open or closed near the sensor, or when the temperature varies, since the magnetic signature of the host system may change over temperature. Also, if the existing coefficients are acceptable but not great and you want to recalibrate, you should recalibrate to a different set number so you can retrieve the old set if necessary. If you don't do this then you will need to reboot the TCM to retrieve the old set.

The initial default is set 0. To store a new set of coefficients, first establish the set number (0 to 7) using kMagCoeffSet, then perform the magnetometer calibration. The new coefficient values and coefficient set number will be stored in volatile memory and will be applied immediately. Save the coefficient set to non-volatile memory by sending kSave. When the TCM is powered down and back up again, it will load the last saved coefficient set and apply its coefficient values.

For example, assume:

- the kSetConfig frame is sent with kMagCoeffSet = 2
- a calibration is performed
- the kSave frame is sent
- the kSetConfig frame is sent again, but with kMagCoeffSet = 3, and
- a calibration is performed.

After this second calibration, the coefficients values from the second calibration are immediately applied, even thought kSave has not been sent. If the TCM is now powered down and powered back up again, kMagCoeffSet = 2 would be recalled and its coefficient values would be applied, since kMagCoeffSet = 3 was not saved and kMagCoeffSet = 2 was the last saved calibration set.

kAccelCoeffSet (Config. ID 19_d)

This setting provides flexibility to store up to three (3) sets of accelerometer calibration coefficients in the module. As with kMagCoeffSet, this can be useful for storing coefficients under a variety of conditions, such as different temperature settings, or if you want to fine-tune the coefficient values but not lose the current set. The initial default is set 0. To store a new set of coefficients, first establish the set number (0 to 2) using kAccelCoeffSet, then perform an accelerometer calibration. The new coefficient values will be stored in volatile memory in the defined set number and will be implemented immediately. Save the coefficient set to non-volatile memory by sending kSave. When the TCM is powered down and back up again, it will load the last saved coefficient set.

7.3.7 kGetConfig (frame ID 7_d)

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

Payload

Config ID
UInt8

7.3.8 kGetConfigResp (frame ID 8_d)

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

Payload

Config ID	Value
UInt8	ID Specific

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

Payload

1	10.0
Declination ID	Declination Angle (Float32)

7.3.9 kSave (frame ID 9_d)

This frame commands the TCM to save internal configurations and user calibration coefficients to non-volatile memory. Internal configurations and user calibration coefficients 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.

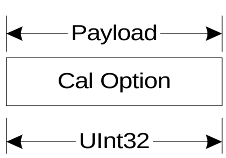
7.3.10 kStartCal (frame ID 10_d)

Before proceeding with this section, ensure you are familiar with Section 5. Also, note the following:

- Multiple sets of calibration coefficients can be saved using kMagCoeffSet and kAccelCoeffSet. These different coefficient sets can be used for storing coefficients for varying conditions, such as when a door is open or closed, or when the temperature varies, since the magnetic signature of the host system may change over temperature.
- Immediately after performing a successful calibration the new calibration coefficients will be will be stored in volatile memory and immediately applied. Save this coefficient set to non-volatile memory by sending kSave. If you do not want to use this new coefficient set, either reboot the TCM (which will restore the prior coefficients), switch to a different coefficient set, or reload the factory coefficients.
- On powering up, the last saved calibration coefficients will be loaded.

This frame commands the module to start a user calibration. After sending this command, the module ensures a PNI-established stability condition is met, takes the first calibration point, and then responds with kUserCalSampCount. kUserCalSampCount will continue to be sent after each sample is taken. Subsequent samples will be taken when autosampling when the minimum change and stability conditions are met, or manually after the kTakeUserCalSample is sent and the stability condition is met.) See Section 5 for more information on the various calibration procedures.

Note: *The payload needs to be 4 bytes. 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 - magnetic only ($10_d = 0A_h$)

Recommended calibration method when >45° of tilt is possible.

2D Calibration - magnetic only ($20_d = 14_h$)

Recommended when the available tilt range is limited to ≤5°.

Hard-Iron-Only Calibration - magnetic only ($30_d = 1E_h$)

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

Limited Tilt Range Calibration – magnetic only ($40_d = 28_h$)

Recommended calibration method when >5° of tilt calibration is available, but tilt is restricted to <45°. (i.e. Full-Range Calibration is not possible.)

Accelerometer-Only Calibration ($100_d = 64_h$)

Select this when only 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.

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

- Using the kSetConfig command, set kUserCalAutoSampling. FALSE allows for more direct control, but TRUE may be more convenient.
- Using the kSetConfig command, establish the coefficient set number for the new calibration coefficient by setting the value for kMagCoeffSet (value 0-7) and/or kAccelCoeffSet (value 0-2).
- 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, for example Full-Range Calibration or 2D Calibration.
- Follow the appropriate calibration procedure, as discussed in Section 5. If kUserCalAutoSampling was set to FALSE, send kTakeUserCalSample when ready to take a calibration point. If kUserCalAutoSampling was set to TRUE, then look for kUserCalSampCount to confirm when a calibration point has been taken. During the calibration process, heading, pitch, and roll information will be output from the TCM, and this can be monitored using kGetDataResp.
- When the final calibration point is taken, the device will present the calibration score using kCalScore and save the calibration coefficient set and coefficient values to volatile memory, assuming the calibration was not aborted.
- If the calibration was not good, either perform another calibration procedure, reboot to restore the prior coefficients, recall another coefficient set (kMagCoeffSet), or recall the factory coefficients (kFactoryMagCoeff).
- If the calibration was good and you want to save the calibration coefficients to non-volatile memory, send the kSave command.

7.3.11 kStopCal (frame ID 11_d)

This command aborts the calibration process. Assuming the minimum number of sample points for the calibration, as defined in Table 7-5, is not acquired prior to sending kStopCal, the prior calibration results are retained. If the acquired number of sample points prior to sending kStopCal is within the allowable range of kUserCalNumPoints, then new calibration coefficients and a new score will be generated. For instance, if kUserCalNumPoints is set to 32 for a Full-Range Calibration, and kStopCal is sent after taking the 12^{th} sample point, then a new set of coefficients will be generated based on the 12 sample points that were taken. They will not be saved, however, unless the kSave command is sent.

7.3.12 kSetFIRFilters (frame ID 12_d)

The TCM 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, and directly affects the time for the initial sample reading, as all the taps must be populated before data is output.

The TCM can be configured to clear, or flush, the filters after each measurement. 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.

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, where 0 to 32 total Values can be in the payload, and with each Value being a Float64, with suggested values given in Table 7-6.

Table 7-6: 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

7.3.13 kGetFIRFilters (frame ID 13_d)

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.

Payload

Byte 1	Byte 2
UInt8	UInt8

7.3.14 kGetFIRFiltersResp (frame ID 14_d)

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

7.3.15 kPowerDown (frame ID 15_d)

This frame is used to power-down the module, which puts the module in Sleep Mode. 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 TCM will power up when it receives any signal on the native UART Rx line.

7.3.16 kSaveDone (frame ID 16_d)

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.

Payload

Error Code
UInt16

7.3.17 kUserCalSampleCount (frame ID 17_d)

This frame is sent from the TCM after taking a calibration sample point. The payload contains the sample count with the range of 1 to 32.

Payload

SampleCount#
UInt32

7.3.18 kCalScore (frame ID 18_d)

The calibration score is automatically calculated and sent after taking the final calibration point, although it may take >1 minute for the score to be calculated. The payload is defined below, and the payload components are discussed after this.

Payload

MagCalScore	Reserved	AccelCalScore	DistError	TiltError	TiltRange
Float32	Float32	Float32	Float32	Float32	Float32

MagCalScore:

MagCalScore provides an over-riding quality indicator 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 DistError and TiltError and still have a rather high MagCalScore value. The most likely reason for this is the TCM is close to a source of local magnetic distortion that is not fixed with respect to the device. In the event of an aborted calibration the score will be 179.8d, or in the event of an accel-only calibration the score will be 99.99d.

AccelCalScore:

This score represents the over-riding quality of the accelerometer calibration. An acceptable score is ≤1. In the event of an aborted calibration the score will be 179.8_d, or in the event of a mag-only calibration the score will be 99.99_d.

DistError:

For a magnetic calibration, this score indicates if the distribution of sample points is sufficient, 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. In the event of an aborted calibration the score will be 179.8_d, or in the event of an accel-only calibration the score will be 99.99_d.

TiltError:

This score indicates if the TCM experienced sufficient tilt during a magnetic calibration, taking into account the calibration method. The score should be 0**.** In the event of an aborted calibration the score will be 179.8_d, or in the event of an accel-only calibration the score will be 99.99_d.

TiltRange:

For a magnetic calibration, this reports the larger of either half the full-pitch range or half the full-roll range of sample points. For example, if the device is pitched +10° to -20º, and rolled +25º to -15º, the TiltRange value would be 20º, which represents half the roll range. For Full-Range Calibration and Hard-Iron-Only Calibration, this should be ≥45°. For 2D Calibration, ideally this should be ~2°. For Limited Tilt Range Calibration the value should be as large a possible given the user's constraints. In the event of an aborted calibration the score will be 179.8_d, or in the event of an accel-only calibration the score will be 99.99_d.

7.3.19 kSetConfigDone (frame ID 19_d)

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

7.3.20 kSetFIRFiltersDone (frame ID 20_d)

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

7.3.21 kStartContinuousMode (frame ID 21_d)

If the TCM is configured to operate in Continuous Acquisition Mode, as defined by 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. The response is kGetDataResp.

7.3.22 kStopContinuousMode (frame ID 22_d)

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

7.3.23 kPowerUpDone (frame ID 23_d)

This frame confirms the TCM received a command to power up. The TCM will power up when it receives any signal on the native UART Rx line. The frame has no payload. Since the module was previously powered down which drives the RS-232 driver TX line low (break signal), it is recommended to disregard the first byte.

7.3.24 kSetAcqParams (frame ID 24_d)

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

Payload

AcquisitionMode	FlushFilter	AquireDelay	SampleDelay
UInt8	UInt8	Float32	Float32

AcquisitionMode

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

FlushFilter

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.

AcquireDelay

When operating in Continuous Acquisition Mode, the AcquireDelay sets the time between samples taken by the module, in seconds. The default is 0.0 seconds, which means the module will reacquire data immediately after the last acquisition. This is an internal setting that is NOT tied to the time with which the module transmits data to the host system. Generally speaking, the AcquireDelay is either set to 0, in which case the TCM is constantly sampling, or set to equal the SampleDelay value. The advantage of running with an AcquireDelay of 0 is the FIR filter can run with a relatively high FIR Tap value to provide stable and timely data. The advantage of using a greater AcquireDelay is power consumption can be reduced, assuming the SampleDelay is no less than the AcquireDelay.

SampleDelay

The SampleDelay is relevant when the Continuous Acquisition Mode is selected. It is the time delay, in seconds, between completion of the TCM sending one set of data and the start of sending the next data set. The default is 0 seconds, which means the TCM 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.

7.3.25 kGetAcqParams (frame ID 25_d)

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

7.3.26 kSetAcqParamsDone (frame ID 26_d)

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

7.3.27 kGetAcqParamsResp (frame ID 27_d)

This frame is the response to kGetAcqParams frame. The payload has the same structure as kSetAcqParams.

7.3.28 kPowerDownDone (frame ID 28_d)

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

7.3.29 kFactoryMagCoeff (frame ID 29_d)

For the current designated kMagCoeffSet, 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.

7.3.30 kFactoryMagCoeffDone (frame ID 30_d)

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

7.3.31 kTakeUserCalSample (frame ID 31_d)

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

7.3.32 kFactoryAccelCoeff (frame ID 36_d)

For the current designated kAccelCoeffSet, 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.

7.3.33 kFactoryAccelCoeffDone (frame ID 37_d)

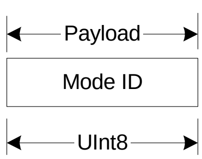
This frame is the response to kFactoryAccelCoeff frame. The frame has no payload.

7.3.34 kSetSyncMode (frame ID 46_d)

When the TCM operates in Sync Mode the module will stay in Sleep Mode until the user’s system sends a trigger to report data. When so triggered, the TCM will wake up, report data once, then return to Sleep Mode. One application of this is to reduce power consumption. Another use of the Sync Mode is to trigger a reading during an interval when local magnetic sources are well understood. For instance, if a system has considerable magnetic noise due to nearby motors, the Synch Mode can be used to take measurements when the motors are turned off

Note: *When Sync Mode is selected, the TCM will acknowledge the change in mode and immediately trigger the Sync Mode and send a data frame.*

This frame allows the module to be placed in Sync Mode. The payload contains the Mode ID requested, as given below.



Mode ID: Normal Mode = 0
Sync Mode = 100

If the module is in Sync Mode and the user desires to switch back to Normal Mode, an “FFh” string first must be sent, followed by some minimum delay time prior to sending the kSetSyncMode frame. The minimum delay time is dependent on the baud rate, and for a baud rate equal to or slower than 9600 there is no delay. For baud rates greater than 9600 the minimum delay is equal to:

Minimum delay after sending “FF_h” (in seconds) = 7E-3 – (10/baud rate)

Example: With a baud rate of 38400, the minimum delay after sending FF_h is:

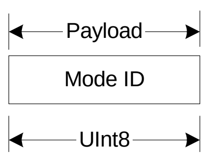
Minimum delay at 38400 baud = 7E-4 – (10/38400) = 4.4E-4 seconds = 440 μs

Sync Mode generally is intended for applications in which sampling does not occur frequently. For applications where Sync Mode sampling will be at a frequency of 1 Hz or higher, there is a minimum allowable delay between taking samples. This minimum delay between samples (approximately inverse to the maximum sample rate) varies from 100 msec to 1.06 second and is a function of the number of FIR filter taps, as defined by the following formula:

Minimum Delay between Samples (in seconds) = 0.1 + 0.03*(number of Taps)

7.3.35 kSetSyncModeResp (frame ID 47 $_d$)

This frame is the response to kSetSyncMode frame. The payload contains the Mode ID requested.



7.3.36 kSyncRead (frame ID 49 $_d$)

If the TCM is configured to operate in Sync Mode, as defined by kSetSyncMode, then this frame wakes up the module, requests a measurement, outputs the results, then powers down again. This frame has no payload. The response is kGetDataResp, with heading, pitch, and roll automatically set as the data component IDs.

Prior to sending the kSyncRead frame, the user's system must first send an "FF_h" string which wakes up the system, then wait some minimum delay time before sending the kSyncRead frame. The minimum delay time is dependent on the baud rate, and for a baud rate equal to or slower than 9600 there is no delay. The minimum delay is defined by the same formula given for switching from Sync Mode to Normal Mode in kSetSyncMode.

7.4 Using Multiple Coefficient Sets

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

Table 7-7 Multiple Coefficient Command List

Magnetometer Calibration					
kSetConfig (frame ID)	kCoeffCopySet (config ID)	Value (UInt32)	Examples	Command Bytes	TCM 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	TCM Response
0x07	0x12		get kCoeffCopySet value which is currently used in TCM	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	TCM 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**	**TCM Response**
0x07	0x13		get kAccelCoeffCopySet value which is currently used in TCM	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

7.5 Code Examples

The following example files, CommProtocol.h, CommProtocol.cp, TCM.h and TCM.cp would be used together for proper communication with a TCM 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.*

7.5.1 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;
}
```

7.5.2 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();
};
```

7.5.3 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 ;
        }
}
```

7.5.4 TCM.h File

```
#pragma once

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


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

      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();
};
```

7.5.5 TCM.cpp File

```
#include "TCM.h"
#include "TickGenerator.h"

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

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

      mTime = 0;
      mStep = 1;
}

TCM::~TCM()
{
}

//
// Called by the CommProtocol object when a frame is completely //
received
//
void TCM::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 TCM::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 TCM::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
R08	Added Revision Control Block. Corrected Section 7.3.3 example to indicate Pitch ID is 24, not 79.	Mar. 18, 2014	A. Leuzinger
R09	Added in section on using Multiple Calibration Coefficient	January 21, 2015	D. McKenzie