Sparton Navigation Modules

Software Interface User’s Manual

Applicable to Sparton DC-4E, GEDC-6E, and AHRS-8.

REVISION HISTORY:

|  |  |  |
| --- | --- | --- |
| DATE | DESCRIPTION OF CHANGE | INITIALS |
| 5/11/11 | Original | CMN |
| 9/22/11 | Re-factored, NorthTek Moved to standalone document. | CMN |
| 10/5/11 | Review comments finalized. | CMN |
| 10/25/11 | Added PN, additional tuning variables documented. | CMN |
| 9/24/12 | Rev. B – Add AHRS-8 and make corrections | MJB |
| 1/2/13 | Rev. C - Update for GEDC-6 2.3.1 changes | MJB |
| 5/2/13 | Rev. D – Update for releases DC-4 2.3.6, GEDC-6 2.4.10, AHRS-8 2.2.2 | MJB |
| 9/12/13 | Rev. E – Update for releases up to 4.0.2 | MJB |
| 10/1/2013 | Rev. F – Remove Legacy Binary In-Field Calibration Commands | RSW |
| 10/15/2013 | Rev. G – Correct gyrop units | MJB |
| 11/26/13 | Rev. H – Update for E versions | RSW/MJB |
| 11/17/14 | Rev. I – Fix InvokeGyroOffsetCal description | RSW |
| 3/9/15 | Rev. J – Formatting changes | RSW |
| 9/10/15 | Rev K – Modified NorthTek script section to appropriately call out the 5ms delay per line and fixed the cal3D script. | RSW |

Table of Contents

[1 Introduction 8](#_Toc413662762)

[1.1 Scope 8](#_Toc413662763)

[1.2 References 8](#_Toc413662764)

[1.3 Document Conventions 8](#_Toc413662765)

[1.4 Applicability 9](#_Toc413662766)

[2 Overview 9](#_Toc413662767)

[2.1 Protocol Introduction 9](#_Toc413662768)

[2.2 Interface Paradigm 10](#_Toc413662769)

[2.3 Data Organization 11](#_Toc413662770)

[2.3.1 Basic Data Types 11](#_Toc413662771)

[2.3.1.1 RFS BitFields 11](#_Toc413662772)

[2.4 User Accessible inertial system Variables 12](#_Toc413662773)

[2.4.1 Variable Summary Table 12](#_Toc413662774)

[2.5 World Magnetic Model 16](#_Toc413662775)

[2.6 In-Field Calibration 16](#_Toc413662776)

[2.7 Mount Orientation Accommodation 18](#_Toc413662777)

[2.8 Low power mode 18](#_Toc413662778)

[3 Protocols 18](#_Toc413662779)

[3.1 RFS 19](#_Toc413662780)

[3.1.1 RFS Get/Get\_Response Example 20](#_Toc413662781)

[3.1.2 RFS BitField Example 20](#_Toc413662782)

[3.2 NMEA 25](#_Toc413662783)

[3.2.1 Raw Magnetics 26](#_Toc413662784)

[3.2.2 Heading - Magnetic 27](#_Toc413662785)

[3.2.3 Heading - TRUE 27](#_Toc413662786)

[3.2.4 Magnetic Variation 27](#_Toc413662787)

[3.2.5 Automatic Magnetic Variation 28](#_Toc413662788)

[3.2.6 Magnetic Vector 29](#_Toc413662789)

[3.2.7 Magnetic In-Field Calibration 29](#_Toc413662790)

[3.2.8 Gyro Calibration 32](#_Toc413662791)

[3.2.9 Magnetic Adaption Error 32](#_Toc413662792)

[3.2.10 Raw Acceleration 32](#_Toc413662793)

[3.2.11 Acceleration Vector 32](#_Toc413662794)

[3.2.12 Raw Gyro 33](#_Toc413662795)

[3.2.13 Gyro Vector 33](#_Toc413662796)

[3.2.14 Pitch and Roll Output 33](#_Toc413662797)

[3.2.15 Quaternion Vector 34](#_Toc413662798)

[3.2.16 Temperature 34](#_Toc413662799)

[3.2.17 BAUD RATE 34](#_Toc413662800)

[3.2.18 Mounting Configuration 35](#_Toc413662801)

[3.2.19 Read Transducers 35](#_Toc413662802)

[3.2.20 NMEA RFS Access 36](#_Toc413662803)

[3.2.21 NMEA/RFS RPT and GLOM 37](#_Toc413662804)

[3.3 Legacy Binary 37](#_Toc413662805)

[3.3.1 Raw Magnetics 38](#_Toc413662806)

[3.3.2 Heading - TRUE 38](#_Toc413662807)

[3.3.3 Heading - Magnetic 39](#_Toc413662808)

[3.3.4 Magnetic variation 39](#_Toc413662809)

[3.3.5 Automatic Magnetic Variation 39](#_Toc413662810)

[3.3.6 Latitude 40](#_Toc413662811)

[3.3.7 Longitude 40](#_Toc413662812)

[3.3.8 Altitude 40](#_Toc413662813)

[3.3.9 Day 41](#_Toc413662814)

[3.3.10 Magnetic Vector 41](#_Toc413662815)

[3.3.11 Raw Acceleration 42](#_Toc413662816)

[3.3.12 Pitch and Roll Output 42](#_Toc413662817)

[3.3.13 Acceleration Vector 42](#_Toc413662818)

[3.3.14 Temperature 42](#_Toc413662819)

[3.3.15 BAUD RATE 43](#_Toc413662820)

[3.3.16 Mounting Configuration 43](#_Toc413662821)

[3.4 NorthTek 45](#_Toc413662822)

[3.4.1 Displaying Database Elements 45](#_Toc413662823)

[3.4.2 Outputting values in User Defined Format 47](#_Toc413662824)

[3.4.3 Streaming Sensor Data 47](#_Toc413662825)

[3.4.3.1 Accelerometer 48](#_Toc413662826)

[3.4.3.2 Gyro 48](#_Toc413662827)

[3.4.3.3 Magnetometer 49](#_Toc413662828)

[3.4.3.4 All raw sensor data 49](#_Toc413662829)

[3.4.3.5 Computed Roll, Pitch and Yaw 49](#_Toc413662830)

[3.4.3.6 Computed Quaternions 49](#_Toc413662831)

[3.4.3.7 Custom Combination Print Streaming (User Port only) 49](#_Toc413662832)

[3.4.4 Reset 52](#_Toc413662833)

[3.4.5 Restart 52](#_Toc413662834)

[3.4.6 Canceling NorthTek Programs 52](#_Toc413662835)

[3.4.7 Inhibit NorthTek Echoed Characters and “OK”/”Huh” Responses 52](#_Toc413662836)

[3.4.8 Setting Database Elements 53](#_Toc413662837)

[3.4.9 Saving the database 53](#_Toc413662838)

[The database gets automatically saved every ten minutes or every time a “set” command is sent to the compass. If the user wishes to specifically to save the database at a known state they may do so by sending the appropriate command: db.save. Once the save has been initiated it will take from 2-5 seconds before completion. Simply sending a <CR> the user may know if the system is done saving. 53](#_Toc413662839)

[3.4.10 Calibration Data Control 53](#_Toc413662840)

[3.4.10.1 Restore In-field calibration data 53](#_Toc413662841)

[3.4.10.2 Restore Factory calibration data 54](#_Toc413662842)

[3.4.10.3 Run the Gyro Offset Calibration 54](#_Toc413662843)

[3.4.11 3D Compass In-Field Calibration 54](#_Toc413662844)

[3.4.12 2D Compass In-Field Calibration 57](#_Toc413662845)

[3.4.13 User Boot Program 57](#_Toc413662846)

[3.4.14 Accessing an RFS VID Using NorthTek – dvid@ 58](#_Toc413662847)

[A Variable Detailed Descriptions 59](#_Toc413662848)

[A.1 Variables by Name 59](#_Toc413662849)

[A.1.1 vidCount 59](#_Toc413662850)

[A.1.2 name 59](#_Toc413662851)

[A.1.3 quiet 59](#_Toc413662852)

[A.1.4 VERSION 59](#_Toc413662853)

[A.1.5 REVISION 59](#_Toc413662854)

[A.1.6 serialnumber 60](#_Toc413662855)

[A.1.7 orientation 60](#_Toc413662856)

[A.1.8 baud 60](#_Toc413662857)

[A.1.9 pitch 61](#_Toc413662858)

[A.1.10 roll 61](#_Toc413662859)

[A.1.11 yaw 61](#_Toc413662860)

[A.1.12 yawt 61](#_Toc413662861)

[A.1.13 quaternion 61](#_Toc413662862)

[A.1.14 lat 62](#_Toc413662863)

[A.1.15 lonG 62](#_Toc413662864)

[A.1.16 alt 62](#_Toc413662865)

[A.1.17 day 62](#_Toc413662866)

[A.1.18 restorefactorycal 62](#_Toc413662867)

[A.1.19 set\_wmm 63](#_Toc413662868)

[A.1.20 magvar 63](#_Toc413662869)

[A.1.21 wmmGD 63](#_Toc413662870)

[A.1.22 wmmHasRun 63](#_Toc413662871)

[A.1.23 magr 63](#_Toc413662872)

[A.1.24 magp 64](#_Toc413662873)

[A.1.25 accelr 64](#_Toc413662874)

[A.1.26 accelp 64](#_Toc413662875)

[A.1.27 gyror 64](#_Toc413662876)

[A.1.28 gyrop 64](#_Toc413662877)

[A.1.29 gyroSampleRate 65](#_Toc413662878)

[A.1.30 positionrate 65](#_Toc413662879)

[A.1.31 position 65](#_Toc413662880)

[A.1.32 composite2 66](#_Toc413662881)

[A.1.33 composite3 66](#_Toc413662882)

[A.1.34 calmode 66](#_Toc413662883)

[A.1.35 calCommand 67](#_Toc413662884)

[A.1.36 calNumPoints 67](#_Toc413662885)

[A.1.37 magErr 68](#_Toc413662886)

[1.1.38 wlim 68](#_Toc413662887)

[1.1.39 alim 68](#_Toc413662888)

[1.1.40 mlim 68](#_Toc413662889)

[1.1.41 km0 69](#_Toc413662890)

[1.1.42 ka0 69](#_Toc413662891)

[1.1.43 emlim 69](#_Toc413662892)

[1.1.44 ealim 69](#_Toc413662893)

[1.1.45 kgyrooffset 70](#_Toc413662894)

[A.1.46 magOffset 70](#_Toc413662895)

[A.1.47 magCalTemperature 70](#_Toc413662896)

[A.1.48 magOffsetTCX, magOffsetTCY, magOffsetTCZ 70](#_Toc413662897)

[A.1.49 magLinearityXTC0 ,magLinearityXTC1, magLinearityXTC2, magLinearityYTC0, magLinearityYTC1, magLinearityYTC2, magLinearityZTC0, magLinearityZTC1, magLinearityZTC2 70](#_Toc413662898)

[A.1.50 magLinearityX ,magLinearityY, magLinearityZ 71](#_Toc413662899)

[A.1.51 magCrossAxisCorrectionX, magCrossAxisCorrectionY, magCrossAxisCorrectionZ 71](#_Toc413662900)

[A.1.52 magFieldCalX, magFieldCalY, magFieldCalZ 71](#_Toc413662901)

[A.1.53 accelrange 72](#_Toc413662902)

[A.1.54 accelOffset 72](#_Toc413662903)

[A.1.55 accelGyroCalTemperature 72](#_Toc413662904)

[A.1.56 accelOffsetTCX, accelOffsetTCY, accelOffsetTCZ 72](#_Toc413662905)

[A.1.57 accelLinearityXTC0, accelLinearityXTC1 , accelLinearityXTC2, accelLinearityYTC0, accelLinearityYTC1, accelLinearityYTC2, accelLinearityZTC0, accelLinearityZTC1, accelLinearityZTC2 72](#_Toc413662906)

[A.1.58 accelLinearityX ,accelLinearityY, accelLinearityZ 73](#_Toc413662907)

[A.1.59 accelCrossAxisCorrectionX, accelCrossAxisCorrectionY , accelCrossAxisCorrectionZ 73](#_Toc413662908)

[A.1.60 gyroOffset 73](#_Toc413662909)

[1.1.61 gyroMaxRate 73](#_Toc413662910)

[1.1.62 gyroMaxReturnCnt 74](#_Toc413662911)

[A.1.63 gyroOffsetTCX, gyroOffsetTCY, gyroOffsetTCZ 74](#_Toc413662912)

[A.1.64 gyroLinearityXTC0, gyroLinearityXTC1 , gyroLinearityXTC2, gyroLinearityYTC0 , gyroLinearityYTC1, gyroLinearityYTC2, gyroLinearityZTC0, gyroLinearityZTC1, gyroLinearityZTC2 74](#_Toc413662913)

[A.1.65 gyroLinearityX, gyroLinearityY, gyroLinearityZ 74](#_Toc413662914)

[A.1.66 gyroFieldCalX, gyroFieldCalY , gyroFieldCalZ 74](#_Toc413662915)

[A.1.67 gyroCrossAxisCorrectionX, gyroCrossAxisCorrectionY, gyroCrossAxisCorrectionZ 75](#_Toc413662916)

[A.1.68 InvokeGyroOffsetCal 75](#_Toc413662917)

[A.1.69 boresightMatrixX, boresightMatrixY , boresightMatrixZ 75](#_Toc413662918)

[A.1.70 lowpower 75](#_Toc413662919)

[A.1.71 temperature 76](#_Toc413662920)

[A.1.72 kgyroOffsetCalFactor 76](#_Toc413662921)

[A.1.73 nmeaecho 76](#_Toc413662922)

[A.1.74 nmeaignorechecksum 76](#_Toc413662923)

[A.1.75 restoreFieldCal 76](#_Toc413662924)

[A.1.76 pMatrix0, pMatrix1, pMatrix2 76](#_Toc413662925)

[A.1.77 printmask 77](#_Toc413662926)

[A.1.78 printtrigger 77](#_Toc413662927)

[A.1.79 printmodulus 77](#_Toc413662928)

[A.1.80 centripetalCorrection 77](#_Toc413662929)

[A.1.81 calPointsNumToAvg 77](#_Toc413662930)

[A.1.82 InvokeTare 78](#_Toc413662931)

[A.1.83 magErr2 78](#_Toc413662932)

[A.1.84 magFieldCalErr 78](#_Toc413662933)

[A.1.85 magFieldCalTot 79](#_Toc413662934)

[A.1.86 magHorzFieldCalTot 79](#_Toc413662935)

[A.1.87 gyroSigma 79](#_Toc413662936)

[A.1.88 infieldCalMu 79](#_Toc413662937)

[A.1.89 infieldCalMu2 79](#_Toc413662938)

[A.1.90 infieldCalMu3 80](#_Toc413662939)

[A.1.91 restartCompassCalcs 80](#_Toc413662940)

[A.1.92 magPointQF 80](#_Toc413662941)

[A.1.93 possibleMagPointQF 80](#_Toc413662942)

[A.1.94 wmmGDMagnitude 81](#_Toc413662943)

[A.1.95 calPointDistribution 81](#_Toc413662944)

[A.1.96 calPointDistSigma 81](#_Toc413662945)

[A.1.97 timeStamp 81](#_Toc413662946)

[A.1.98 currentTime 82](#_Toc413662947)

[A.1.99 calStatus 82](#_Toc413662948)

[B Troubleshooting 83](#_Toc413662949)

[B.1 NMEA commands are not echoed. 83](#_Toc413662950)

[B.2 NMEA commands don’t respond to “enter” key. 83](#_Toc413662951)

[B.3 Repeating output is not appearing. 83](#_Toc413662952)

[B.4 Repeating output is making typing a new command difficult. 83](#_Toc413662953)

[B.5 The compass is providing an inaccurate heading: 83](#_Toc413662954)

[B.6 The compass does not output any response. 83](#_Toc413662955)

# Introduction

This is an interface document. It describes how to communicate with, control and obtain sensor, status and configuration data from the Sparton family of Attitude and Heading Reference (AHRS) products. This document is intended for system engineers, systems integrators or software developers that need to know how to communicate with, utilize and obtain measurements from a Sparton AHRS product. This document applies to the AHRS-8, GEDC-6E and DC-4E devices. In this document, these devices will be referred to as inertial Systems and all references shall apply to all devices except where noted.

## Scope

This document covers the following topics:

* Low level communications interfaces at the software level
* Packet level communications protocol
* Applications layer functions and the relevance to the function of the inertial system
* Examples of commands for user purposes

This document is not a hardware or mechanical data sheet.

## References

This document is an umbrella document in that some of the interface descriptions for the inertial systems are described in external documentation. This document describes the relationship of these documents. The additional documentation needed to complete the entire interface description for the inertial system family is listed below.

* *NorthTek System Programming Manual*
* *Remote Function Select (RFS) Protocol Suite, Interface Design Description*

## Document Conventions

Descriptive text in this document appears in this font (Calibri 11).

Data sent transmitted to the inertial system appears in ***this font (Courier 11, Bold Italic).***

Data sent from the inertial system appears in **this font (Courier 11, Bold, grey background).**

Information that must be supplied in this spot is enclosed in **<>** pairs. The **<contents>** brace pair should be replaced with the user supplied data or will be filled in by the inertial system according to the description contained in the **<>**’s.

* A carriage return character is shown as **<CR>** or **<cr>.**
* A line feed character is shown as **<cr><lf>** or **<cr><lf>.**
* Control sequences are indicated as **<CTRL-S>.**

Non printable characters are spelled out in **<>** pairs, for example **<ESCAPE>** is equal to the escape character (0x1b). Additional text will included when non-printable characters might not be clear to the reader.

## Applicability

This document applies to the Sparton AHRS-8, GEDC-6E and DC-4E Digital Compasses. All portions of this document that refer to gyro functionality only apply to the AHRS-8 and GEDC-6E.

# Overview

This document is intended to be a reference document. Please read through this Overview section and the Programmer’s Quick Start and follow the references to this document in the areas of interest. The Programmer’s Quick Start gives step-by-step examples and points to sections in this document for more information.

This document details the available communication protocols as well as use of the compass from the programmer’s point of view. Also included is a description of a unique capability where the user can program custom output from the compass using NorthTek scripts (see Section 3.4 NorthTek).

When the compass is installed in any application where there are magnetic disturbances that are fixed in position and strength relative to the compass, then the In-field Calibration process should be performed to cancel those disturbances out. The gyro based GEDC-6E and AHRS-8 will automatically ignore any temporary magnetic disturbances but fixed disturbances require in-field calibration. For more information, see Section 2.6 In-Field Calibration. Note that the inertial system cannot compensate for disturbances that exceed its specified limit of 900 mGauss per axis.

If the true North heading capability is desired, then the World Magnetic Model must be run (see Section 2.5 World Magnetic Model). True North is relative to the Earth’s axis of rotation whereas magnetic North is points to the Earth’s magnetic pole.

For cases where the inertial system is mounted in the host application such that the inertial system’s X/Y/Z axes differ from those of the host application, then the “boresight” matrix can be set such that the inertial system’s outputs will be adjusted to the host application’s frame of reference.

## Protocol Introduction

The interface to the inertial system is a serial communication link (User Port). The link is an asynchronous character oriented interface that operates in a bidirectional mode. This serial link transmits and receives asynchronous characters using nominal “UART” type framing with 8 data bits, no parity and a single stop bit, with a default baud rate of 115200. The baud rate may be changed by software command. No error detection is used with the individual characters.

The serial link is full-duplex; however in practice the information flow is largely half-duplex in nature. Essentially the client and the server operate in a poll-response fashion. For example, in the above situation, the device that desires to use the inertial system would be the master (client) and the inertial system would be the slave (server).

The inertial system provides the following protocols on the User Port:

* RFS -- A binary protocol called Remote Function Select (RFS). This protocol provides a robust, error checked method of communicating with the inertial system
* NMEA --- A printable ASCII protocol that conforms to the general message formatting used by the National Marine Equipment Manufacturers (NMEA) association guidelines.
* Legacy Binary – An efficient binary protocol that has no error checking support (i.e. no checksum).
* NorthTek -- A free form command line interpreter suitable for human operator usage or custom output requirements.

These protocols are able to operate simultaneously by virtue of the framing used for each protocol type. After the frame for one protocol is complete, another protocol may be used for the next frame. The RFS protocol packets begin with an <SOH> character and ends with an <ETX> character. The NMEA protocol begins with a $ character and ends with a <cr><lf> character. The Legacy Binary protocol begins with a 0xA4 character and ends with a 0xA0 character. Once the first character for a new frame identifies one of these three protocols, all subsequent characters, up to the terminator for that protocol, are considered only for that protocol. All characters that are received outside one of the packet types are passed to the NorthTek interpreter. Each of the three packet oriented protocols expects packets to be completed in a timely manner. If the next character is not entered in a specific time interval (1 second RFS, 5 seconds NMEA, 0.5 seconds for Legacy), the packet is discarded and the receive parser resets to be ready to accept any packet type.

The Legacy and NMEA protocols provide a degree of backwards compatibility with the SP300x product family. The RFS protocol provides some advantages over the other protocols, namely 1) Robust error checking via a 16 bit CRC as part of each packet, 2) User definable messages such that the user can generally put all required data into a single message and 3) A self-descriptive command structure such that new inertial system features are automatically accessible to hosts. The NMEA protocol can utilize some (but not all) of the features of the RFS command set. The Legacy Binary interface is appropriate for systems that are upgrading from the SP300x product family and is not recommended for new designs. The NorthTek interface is an advanced user interface that allows the user to enter and perform small command scripts within the compass in addition to a standard set of interface commands.

## Interface Paradigm

The previously described protocols differ in the formatting of the actual “packets” or “commands” that are sent to or received from the inertial system. The protocols are similar in that they all conform to the same basic paradigm. The host sends a command, the inertial system then responds. All protocols follow this basic rule. Some protocols provide for the inertial system to respond automatically once commanded. As the purpose of the inertial system is, ostensibly, to provide a constant stream of navigation information most of the protocols (excepting the Legacy Binary) provide a method of automatically outputting information without an additional query from the host. The exception to this paradigm is NorthTek. NorthTek, being a programming environment, allows customization of the user interface in an algorithmic way.

## Data Organization

The inertial systems incorporate an internal database. The database contains primitive elements. The primitive elements provide the storage for information that is accessed externally. The elements can be settings, status or values. Regardless of the type of the element, the method of access to the value of the element is similar across all protocols. The internal data storage method is isolated from the data presented to the external interfaces. This resolves issues of bit/byte ordering (“endian-ness”). The specific format of multi-byte values is specified on each interface. Regardless of the method of transmission there are several basic data types as described in the following section.

### Basic Data Types

The internal database elements are one of the following types:

* Int16 – A 16 bit signed integer.
* Int32 – A 32 bit signed integer.
* Float32 – An IEEE format 32 bit floating point value.
* String – A counted string. Byte 0 is the length of the string that follows ***including*** the terminating null.
* Ordinal -- An 8 bit value that represents a selection from a discrete set of 0 based values (0..N).
* ArrayInt32 – An array of Int32 values
* ArrayFloat32 – An array of Float32 values.
* BitField – RFS only. A combined variable that includes multiple other data types in the same object.

#### RFS BitFields

BitFields are user programmable as to content thus allowing the user to configure a single message that contains multiple useful values into a single report (similar to packed structures). BitFields minimize communications required to obtain the necessary data for the application. An overview of BitFields is provided here to frame the context of the variable table which describes the list of variables that follows in a later section. Full details of the BitFields and their usage can be found in the *RFS Protocol Suite* document.

A BitField is a variable type that contains meta-data instead of data. A BitField, for example, may contain a reference to an Int32 variable, an Ordinal Variable and a Float32 Variable. When setting or getting the variable, the application program would be able set or read all three values in the same message. The BitField variable contains the information to describe what three other variables would be set or read in a message to/from the BitField variable. The BitField thus contains not the actual data to be transmitted, but information describing what information will be transmitted. RFS messages may be used to manipulate the BitField meta-data or operate on the actual data referred to by the BitField. Some RFS messages are modal in that they automatically determine if the intended message is for the meta-data or the data itself. Full details are described in the *RFS Protocol Suite* document. BitFields are only used with the RFS protocol. (RFS protocol technical support is available from the manufacturer).

## User Accessible inertial system Variables

The inertial system contains variables that are accessible by the user. In the NorthTek and NMEA protocols, the variables are identifiable by name. In the RFS protocols variables are identified by name but retrieved and manipulated by number. It is important to note that the number associated with a specific variable name may change in the future, however the RFS protocol was designed to accommodate this behavior and resolves the name/number matchups when the inertial system is first interrogated at power up. This strategy allows future inertial system software releases to add additional features while not obsoleting existing interface software.

The variable definitions for the inertial system are available as an XML file. This XML file describes each variable type, the dimensions of array variables, minimum and maximum values, and default values. Contact the manufacturer for availability.

The inertial system variables are listed in the table below. The variable names are the same names that would be entered in NMEA and NorthTek commands. This is a shorthand table to remind the user of the variable names. Detailed descriptions of each variable are found in Appendix A.1, Variable Detailed Descriptions, Variables by Name. In general, all variables are available via the RFS protocol; variables that are available via NorthTek are marked as shown. The Legacy Binary protocol can only access variables that are predefined in its protocol.

### Variable Summary Table

See Appendix A “Variable Detailed Descriptions” for the details on the variables in this table. Notes are given for variables that were introduced after the initial release. If no notation is listed and the variable is indicated to be in a product then the variable existed in the product from the initial release. If a version is given and a product is not specified for that version then all products, that version and later have the variable.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Type | Notes | N M E A | NorthTek | DC-4E | GEDC-6E  / GEDC-6E | AHRS-8 |
| vidCount | Int32 | Variable Count | ● | ● | ● | ● | ● |
| Name | String | Device Name |  | ● | ● | ● | ● |
| Quiet | Ordinal | Debug Suppress | ● | ● | ● | ● | ● |
| VERSION | String | Software version release identifier |  | ● | ● | ● | ● |
| REVISION | String | Subversion revision number of this software |  | ● | ● | ● | ● |
| serialnumber | String | Factory set serial number |  | ● | ● | ● | ● |
| orientation | Ordinal | Physical orientation of compass | ● | ● | ● | ● | ● |
| Baud | Ordinal | User Port Baud rate | ● | ● | ● | ● | ● |
| pitch | Float32 | Pitch angle in degrees, positive up | ● | ● | ● | ● | ● |
| roll | Float32 | Roll angle in degrees, positive right-side down | ● | ● | ● | ● | ● |
| yaw | Float32 | Magnetic heading in degrees | ● | ● | ● | ● | ● |
| yawt | Float32 | True heading in degrees | ● | ● | ● | ● | ● |
| quaternion | ArrayFloat32 | 4 element array representing w, x, y and z | ● | ● | ● | ● | ● |
| lat | Float32 | Latitude (user supplied for WMM, see set\_wmm) | ● | ● | ● | ● | ● |
| lonG | Float32 | Longitude (user supplied for WMM, see set\_wmm) | ● | ● | ● | ● | ● |
| alt | Float32 | Altitude (user supplied for WMM, see set\_wmm) | ● | ● | ● | ● | ● |
| day | Float32 | Estimated date expressed to the tenth of a year (user supplied for WMM, see set\_wmm) | ● | ● | ● | ● | ● |
| restorefactorycal | Command | Restores calibration data to factory settings | ● | ● | ● | ● | ● |
| set\_wmm | Ordinal | Command to run the World Magnetic Model (uses lat, lonG, alt day), computes magvar and wmmGD as the results. | ● | ● | ● | ● | ● |
| magvar | Float32 | Magnetic Variation - difference between true and magnetic | ● | ● | ● | ● | ● |
| wmmGD | ArrayFloat32 | World Magnetic Model geodetic magnetic field X Y Z at Lat/Long. Used by 2-D in-field calibration. | ● | ● | ● | ● | ● |
| wmmHasRun | Ordinal | Indicates whether the World Magnetic Model has been run (implies wmmGD and magvar are valid) | ● | ● | ● | ● | ● |
| magr | ArrayFloat32 | Raw Magnetometer X Y Z Total array | ● | ● | ● | ● | ● |
| magp | ArrayFloat32 | Processed Magnetometer X Y Z array | ● | ● | ● | ● | ● |
| accelr | ArrayFloat32 | Raw Accelerometer X Y Z array | ● | ● | ● | ● | ● |
| accelp | ArrayFloat32 | Processed Accelerometer X Y Z array | ● | ● | ● | ● | ● |
| gyror | ArrayFloat32 | Raw gyro X Y Z array | ● | ● |  | ● | ● |
| gyrop | ArrayFloat32 | Processed gyro X Y Z array | ● | ● |  | ● | ● |
| gyroSampleRate | Float32 | Actual sample rate for the gyro (samples/second) | ● | ● |  | ● | ● |
| positionrate | Int32 | Report rate for the data described in the RFS “position” variable (see next row) | ● | ● | ● | ● | ● |
| position | BitField | Composite object descriptor with pitch, roll, yawt, magErr, temperature, magp X/Y/Z, accelp X/Y/Z, gyrop X/Y/Z,1 |  |  | ● | ● | ● |
| composite2 | BitField | User defined RFS composite object (packed structure) descriptor |  |  | ● | ● | ● |
| composite3 | BitField | User defined RFS composite object (packed structure) descriptor |  |  | ● | ● | ● |
| calmode | Ordinal | Calibration Mode | ● | ● | ● | ● | ● |
| calCommand | Ordinal | Commands the next step for “in field” calibration | ● | ● | ● | ● | ● |
| calNumPoints | Int32 | Number of points collected for 3D Calibration Mode | ● | ● | ● | ● | ● |
| magErr | Float32 | Magnetic error term resulting from calibration calcs | ● | ● | ● | ● | ● |
| wlim | Float32 | Algorithm constant - threshold limit | ● | ● | ● | ● | ● |
| alim | Float32 | Algorithm constant - threshold limit for accel | ● | ● | ● | ● | ● |
| mlim | Float32 | Algorithm constant – threshold limit for mag | ● | ● | ● | ● | ● |
| km0 | Float32 | Algorithm constant - magnetic gain | ● | ● | ● | ● | ● |
| ka0 | Float32 | Algorithm constant - acceleration gain | ● | ● | ● | ● | ● |
| emlim | Float32 | Algorithm constant - magnetic error threshold limit | ● | ● | ● | ● | ● |
| ealim | Float32 | Algorithm constant - accel error threshold limit | ● | ● | ● | ● | ● |
| kgyrooffset | Float32 | Algorithm constant - gyro offset | ● | ● |  | ● | ● |
| magOffset | ArrayFloat32 | Calibration offset for the magnetometer | ● | ● | ● | ● | ● |
| magCalTemperature | Float32 | Base temperature (degrees C) for the magnetometer calibration parameters | ● | ● | ● | ● | ● |
| magOffsetTCX  magOffsetTCY  magOffsetTCZ | ArrayFloat32 | Factory temperature compensated calibration offsets for the magnetometer | ● | ● | ● | ● | ● |
| magLinearityXTC0  magLinearityXTC1  magLinearityXTC2  magLinearityYTC0  magLinearityYTC1  magLinearityYTC2  magLinearityZTC0  magLinearityZTC1  magLinearityZTC2 | ArrayFloat32 | Factory temperature compensated calibration scalars for the magnetometer | ● | ● | ● | ● | ● |
| magLinearityX  magLinearityY  magLinearityZ | ArrayFloat32 | Computed calibration scalars for the magnetometer - is used to “linearize” the magnetic vector after the offset has been applied | ● | ● | ● | ● | ● |
| magCrossAxisCorrectionX  magCrossAxisCorrectionY  magCrossAxisCorrectionZ | ArrayFloat32 | Adjusts for any unorthogonal mounting of the magnetometers | ● | ● | ● | ● | ● |
| magFieldCalX  magFieldCalY  magFieldCalZ | ArrayFloat32 | In-field calibration offset and scalars for the magnetometers | ● | ● | ● | ● | ● |
| accelrange | Ordinal | Accelerometer full scale range setting (4g vs. 8g) | ● | ● | ● | ● | ● |
| accelOffset | ArrayFloat32 | Calibration offset for the accelerometer | ● | ● | ● | ● | ● |
| accelGyroCalTemperature | ArrayFloat32 | Temperature (deg C) at which the Rate Table temperature compensation parameters for both accelerometer and gyro are computed. | ● | ● | ● | ● | ● |
| accelOffsetTCX  accelOffsetTCY  accelOffsetTCZ | ArrayFloat32 | Temperature compensated calibration offsets for the accelerometer | ● | ● | ● | ● | ● |
| accelLinearityXTC0  accelLinearityXTC1  accelLinearityXTC2  accelLinearityYTC0  accelLinearityYTC1  accelLinearityYTC2  accelLinearityZTC0  accelLinearityZTC1  accelLinearityZTC2 | ArrayFloat32 | temperature compensated calibration scalars for the accelerometer | ● | ● | ● | ● | ● |
| accelLinearityX  accelLinearityY  accelLinearityZ | ArrayFloat32 | Temperature compensated linearity coefficients for the accelerometer; these are computed from the accelLinearityXTC0, etc. variables and temperature. | ● | ● | ● | ● | ● |
| accelCrossAxisCorrectionX  accelCrossAxisCorrectionY  accelCrossAxisCorrectionZ | ArrayFloat32 | Corrections to compensate for any unorthogonal mounting of the accelerometers | ● | ● | ● | ● | ● |
| gyroOffset | ArrayFloat32 | Calibration offset for the gyro | ● | ● |  | ● | ● |
| gyroMaxRate | Float32 | The inertial system switches to using just the magnetometer and accelerometer above this rate. | ● | ● |  | ● | ● |
| gyroMaxReturnCnt | Int32 | After the gyro has exceeded the gyroMaxRate, this variable defines the number of in-range gyro values that must occur before switching back to using the gyro data | ● | ● |  | ● | ● |
| gyroOffsetTCX  gyroOffsetTCY  gyroOffsetTCZ | ArrayFloat32 | Temperature compensated calibration offsets for the magnetometer | ● | ● |  | ● | ● |
| gyroLinearityXTC0  gyroLinearityXTC1  gyroLinearityXTC2  gyroLinearityYTC0  gyroLinearityYTC1  gyroLinearityYTC2  gyroLinearityZTC0  gyroLinearityZTC1  gyroLinearityZTC2 | ArrayFloat32 | Temperature compensated calibration scalars for the gyro | ● | ● |  | ● | ● |
| gyroLinearityX  gyroLinearityY  gyroLinearityZ | ArrayFloat32 | Computed calibration scalars for the gyro based on the temperature compensated calibration scalars (e.g. gyroLinearityXTC0, etc.) | ● | ● |  | ● | ● |
| gyroFieldCalX  gyroFieldCalY  gyroFieldCalZ | ArrayFloat32 | In field calibration offset and scalar for the gyro | ● | ● |  | ● | ● |
| gyroCrossAxisCorrectionX  gyroCrossAxisCorrectionY  gyroCrossAxisCorrectionZ | ArrayFloat32 | Calibration cross axis correction for the gyro | ● | ● |  | ● | ● |
| InvokeGyroOffsetCal | Command | Command the gyro offset calibration to start | ● | ● |  | ● | ● |
| boresightMatrixX  boresightMatrixY  boresightMatrixZ | ArrayFloat32 | Rotation matrix that will be applied to transform from the inertial system’s reference frame to the host reference frame | ● | ● | ● | ● | ● |
| lowpower | Command | Command to enter Low Power Mode | ● | ● | ● | ● | ● |
| temperature | Int32 | Temperature in Celsius | ● | ● | ● | ● | ● |
| kgyroOffsetCalFactor | Float32 | Gyro offset field calibration factor | ● | ● |  | ● | ● |
| nmeaecho | Ordinal | Command to turn on echoes of NMEA characters | ● | ● | ● | ● | ● |
| nmeaignorechecksum | Ordinal | Command to ignore received NMEA checksum | ● | ● | ● | ● | ● |
| restoreFieldCal | Ordinal | Command to set magnetic in-field calibration data back to factory default | ● | ● | ● | ● | ● |
| pMatrix0  pMatrix1  pMatrix2 | ArrayFloat32 | Platform matrix (rotation matrix) – provides 3D orientation. Version 3.0.0 | ● | ● | ● | ● | ● |
| printmask | Int32 | Used to select items for printing Version 3.0.0 | ● | ● | ● | ● | ● |
| printtrigger | Int32 | Used to select triggers (data updates) that will cause the printmask selected items to print. Version 3.0.0 | ● | ● | ● | ● | ● |
| printmodulus | Int32 | Print when x number of trigger events occur as defined by printtrigger Version 3.0.0 | ● | ● | ● | ● | ● |
| centripetalCorrection | Ordinal | Enable Centripetal acceleration correction of acceleration Version 3.0.0 | ● | ● |  | ● | ● |
| calPointsNumToAvg | Int32 | Number of samples to be averaged for a 3D/2D Calibration Mode capture point. | ● | ● | ● | ● | ● |
| InvokeTare | Ordinal | Command to update the boresight matrix based on current orientation | ● | ● | ● | ● | ● |
| restartCompassCalcs | Ordinal | Used to resync the magnetic based and gyro based heading/attitude computations (Version 3.0.0) | ● | ● | ● | ● | ● |
| gyroSigma | Float32 | Gyro stability factor (Version 3.0.0) | ● | ● |  | ● | ● |
| magErr2 | Float32 | Magnetic heading error estimate in degrees based on the current (real-time) magnetic environment relative to the magnetic environment during the last in-field calibration (Version 3.0.0) | ● | ● | ● | ● | ● |
| magFieldCalErr | Float32 | Magnetic field calibration heading error estimate, in degrees, as a result of the last in-field calibration (Version 3.0.0) | ● | ● | ● | ● | ● |
| magFieldCalTot | Float32 | Average of magnetic field calibration point vector magnitudes (Version 3.0.0) | ● | ● | ● | ● | ● |
| magHorzFieldCalTot | Float32 | Average of the horizontal component of the magnetic field calibration points (Version 3.0.0) | ● | ● | ● | ● | ● |
| calPointDistribution | Float32 | Factor indicating quality of in-field calibration points (exactly 12 points must be collected) (Version 3.0.0) | ● | ● | ● | ● | ● |
| calPointDistSigma | Float32 | Used in computing the calibration point distribution indicator (Version 3.0.0) | ● | ● | ● | ● | ● |
| infieldCalMu | Float32 | In-field calibration offset convergence factor (Version 3.0.0) | ● | ● | ● | ● | ● |
| infieldCalMu2 | Float32 | In-field calibration scaling convergence factor (Version 3.0.0) | ● | ● | ● | ● | ● |
| magPointQF | Float32 | Quality factor for the field cal captured magnetic calibration points (Version 3.0.0) | ● | ● | ● | ● | ● |
| possibleMagPointQF | Float32 | Possible quality factor if the current magnetic point is added to the captured calibration points during an in-field calibration (Version 3.0.0) | ● | ● | ● | ● | ● |
| wmmGDMagnitude | Float32 | Magnitude of the World Magnetic Model computed Earth Field (used for magPointQF), defaults to 450 milliGauss (Version 3.0.0) | ● | ● | ● | ● | ● |
| timeStamp | Int32 | Processor time (milliseconds from power-up) at the beginning of the pitch/roll/yaw computations (raw data time stamp) (Version 4.0.2) | ● | ● | ● | ● | ● |
| currentTime | Int32 | Processor time (milliseconds from power-up) at the time of the request for this variable (Version 4.0.2) | ● | ● | ● | ● | ● |

## World Magnetic Model

The inertial systems contain a World Magnetic Model (geodetic coordinate system referenced to the WGS 84 ellipsoid) which is needed to determine the magnetic variation (difference between true heading and magnetic heading). This model requires the user to input a physical location and time (lat, lonG, alt and day[[1]](#footnote-2)) and cause the compass compute the magnetic variation (aka declination).

The world magnetic model is valid for 5 years at which point it must be updated. *NOTE: TO RETAIN MAGNETIC VARIATION ACCURACY, THE MAGNETIC MODEL MUST BE UPDATED EVERY FIVE YEARS ON THE NOAA SCHEDULE (see http://www.ngdc.noaa.gov/geomag/WMM/DoDWMM.shtml). A SEPARATE PROGRAM IS AVAILABLE ON THE SUPPLIED CD WHICH WILL ASSIST IN DOWNLOADING NEW COEFFICIENTS INTO THE DIGITAL COMPASS USING RFS. THIS ONLY AFFECTS THE CALCULATION OF TRUE HEADING AND DOES NOT AFFECT MAGNETIC HEADING ACCURACY*

Users who have a product lifetime expectation that exceeds 5 years, that need true heading output, and wish to field update the world magnetic model data, must implement, at a minimum, the file transfer portion of the RFS protocol. Hardware designers should refer to the reference design to ensure that the proper software interface is available to the compass for performing this update.

This model is invoked by the user (using the NMEA $PSPA,AUTOVAR or by setting set\_wmm using NorthTek or RFS) and should be rerun with updated position and date data whenever they change enough to affect the resulting magnetic variation (magvar). NOTE: Resetting the magnetic parameters to the factory default does not affect the magnetic variation information. To clear out the magnetic variation, manually set magvar to zero.

## In-Field Calibration

The compass is already factory calibrated in an environment free from magnetic distortions. When the compass is first used in the application, it must learn the local magnetic distortions in order to cancel them out. The compass can only remove distortions that are fixed relative to the compass. The two types of in-field calibration, depending on the movement capability of the host device, are 2-D and 3-D.

Note that a poor selection of points (lack of magnetic diversity) can result in erroneous performance. Use the “restoreFieldCal” command to clear the effects of an in-field calibration.

The 3-D calibration procedure is used for applications that move outside of a plane. It does **not** require full 360 deg pitch and roll capability for this calibration but the quality of the calibration improves with the variety in the points captured.

The 2-D calibration procedure is used for applications that stay within a plane containing two of the navigation module’s axes and rotating about the third axis. The point capture positions should be chosen to be distributed well about the circle of rotation. The 2-D calibration differs from the 3-D in that it is recommended that the World Magnetic Model has been run previously (see the NMEA $PSPA,AUTOVAR command or the set\_wmm command).

For in-field calibration, the user must follow this procedure:

1. Set the variable calmode to be 3D (ordinal 1) or 2D (ordinal 2).
2. Start the calibration (calCommand set to cal\_start)
3. User maneuvers the unit to capture 4-12 points with the host device in a variety of positions (set calCommand to be cal\_capture at each of the points, observe that calNumPoints increments at each point)
4. Command the end of point capture (set calCommand to be cal\_end\_capture)
5. Allow the calibration algorithm to converge (monitor magErr till it settles to a minimum).
6. Terminate calibration, causing the results to be stored (set calCommand to cal\_end, set calmode to CalModeOff (ordinal 0).

The results are stored in magFieldCalX, magFieldCalY and magFieldCalZ.

For Version 4.0.2 or later, there are five variables that give an indication of the quality of the in-field calibration. Find the variables in Appendix A for more details.

* magErr2 – magnetic heading error estimator in degrees (real-time)
* magFieldCalErr – magnetic calibration heading error estimate in degrees (once at end of in-field calibration)
* magPointQF – quality factor for magnetic diversity of captured points (computed after each point captured)
* possibleMagPointQF – quality factor if the current magnetic point is added to the captured calibration points (computed real-time during point capture only)
* calPointDistribution – quality factor for orientation diversity of captured points (computed once after exactly 12 points are captured)

The in-field calibration procedure is capable of averaging the magnetic data collected for each capture point. The variable “calPointsNumToAvg” determines how many magnetic values will be averaged at each capture point. The host device and magnetic distortions must be static during each capture period or set calPointsNumToAvg to 1 if a static position cannot be achieved.

## Mount Orientation Accommodation

The inertial system’s orientation output (pitch/roll/yaw, pMatrix and quaternions) can be adjusted to be that of the host application by changing the “boresightMatrix”. There are a number of ways this can be accomplished.

* Set the orientation variable to one of the predefined 90 degree orientations, see A.1.7 Orientation and see the Sparton Navigation Sensor Product Guide available in the web site downloads for physical orientation diagrams.
* Orient the host device towards magnetic North in a level condition and set the InvokeTare command to 1 using NorthTek, NMEA $PSRFS or RFS.
* Use the mechanical design of the host device to determine the boresight matrix and set the boresightMatrixX/Y/Z directly using NorthTek, NMEA $PSRFS or RFS.
* Run a NorthTek tare script (see Application Note AN 1003 and a sample script is in the downloads section on the website).

## Low power mode

The inertial system has a low power mode. When in lowpower the only thing operating is the serial port. Activity on the serial port will cause the inertial system to exit low power mode. Upon exiting low power mode, the inertial system resets itself to the power on state. Any temporary settings prior to low power mode will be lost, but permanent settings will be retained. The low power command is detailed in the NorthTek section of the document and also in the detailed variable descriptions section in the appendix.

# Protocols

All the serial port protocols co-exist on the same serial port at all times. The user may use whichever protocol is most desirable. The suggested protocols for future growth are the RFS protocol for automated binary interfaces and NorthTek for human controlled interactive interfaces. The NMEA protocol will continue to be supported but cannot by its nature support some of the RFS/NorthTek current and future capabilities. Note that NMEA or Legacy protocols can be implemented in NorthTek for custom applications.

The factory default baud rate is 115200 baud but may be changed to any of the allowable baud rates. Note that the same baud rate applies to all protocols simultaneously. Once changed, the inertial system will maintain the baud rate setting through a power cycle. To change the baud rate to 9600 for example, using NMEA enter the following string (see variable description A.1.8 for allowable values, <cr><lf> is ctrl-j on the keyboard):

$PSPA,BAUD=4<cr><lf>

To change the baud rate using NorthTek enter:

baud 4 set drop<cr>

Note that NMEA does not echo characters by default. NMEA echo can be turned on (setting is lost with a power cycle or lowpower cycle) with the following command:

nmeaecho 1 set drop<cr>

This command can be sent in NMEA format as follows:

$PSRFS,nmeaecho,set,1<cr><lf>

The NMEA processor ignores the checksum’s absence by default, however for users that desire to have an additional level of error protection in sending commands to the inertial system, the checksum may be enabled (ignore disabled) for receive. The following command (effect is for the current power on cycle) enables (disables the ignoring of) the NMEA checksum. Commands which fail the checksum test are not performed.

nmeaignorechecksum 0 set drop<cr>

Note that this command can be entered in NMEA format as follows:

$PSRFS,nmeaignorechecksum,set,0<cr><lf>

## RFS

The RFS protocol suite is described in a separate document titled Remote Function Select (RFS) Protocol Suite.

RFS assigns a unique Variable Identifier (VID) for each database element (see Appendix A for the list of available database elements). Other than the reserved VIDs[[2]](#footnote-3) the user application should not depend on a particular numeric for a particular VID for the database elements from one release of the inertial system software to another. The string names will remain the same from release to release and can be used to resolve the same variable to a VID number.

To be software revision independent, the user application should use VID 0 to obtain the number of elements, and then obtain the names of all of the database elements by looping through all of the VIDS and using the “Get\_Next” commands. The user application can then obtain the current VID for the desired named element listed in Appendix A. From then on, with that VID, the user application may use the “Set” command or the “Get\_Value” command and the inertial system will respond with “Value\_Is”.

An alternative method to obtain a VID is to the NorthTek command, dvid@. The syntax is “<variableName> dvid@ .” (note there is a dot at the end to display the result).

The RFS protocol is layered on top of a packet transfer mechanism called Standard Asynchronous Packet Protocol (SAPP). SAPP is also described in the Remote Function Select (RFS) Protocol Suite’s Appendix E. Please refer to the referenced document for more information.

### RFS Get/Get\_Response Example

Below is an inertial system specific example for which the user will need to have the RFS Protocol Suite document open for reference. In this example, the user application is obtaining the serial number of the inertial system unit. All figure numbers in parenthesis below refer to those found in the *RFS Protocol Suite document*.

Application creates a command to get information about the variable with Variable ID (VID) = 4:

RFS Command: **get** VID = 4, payload size = 0, message sequence = 216 (0xD8)

which translates to RFS payload (Figure 4-15 RFS Outer Layer Protocol Diagram):

01 00 00 00 00 01 D8 04

then the application must add the SAPP layer for error protection (Figures E.4-34 – Packet Frame and E.4-35 – Packet Body), note that “10 81” is substituted for “01” in 2 places to prevent “01” from being confused with the start character (SOH) (reference E.3.4 Packet Frame):

01 0B 40 10 81 00 00 00 00 10 81 D8 04 EB 42 03

The inertial system creates the response message:

RFS Response: **getResponse** VID = 4, payload size=23, seq=216,

(see Figure 4-16 Data Description Object) Scalar 0x1X,

(see Figure 4-17 Scalar Description Field) Type 2=String, Field Size = 20 (0x14),

Name=”serialnumber”, String Length Max = 20,

(see Figure 4-8 String Data Format) String Length=4, String Value=”S10”

01 00 00 00 17 00 D8 04 10 02 14 0D 73 65 72 69 61 6C 6E 75 6D 62 65 72 00 14 04 53 31 30 00

The inertial system than adds the SAPP layer for framing and error detection resulting in the following output (RFS Protocol Suite Figures 4-34 – Packet Frame and 4-35 – Packet Body):

01 22 60 10 81 00 00 00 17 00 D8 04 10 90 02 14 0D 73 65 72 69 61 6C 6E 75 6D 62 65 72 00 14 04 53 31 30 00 87 4F 03

### RFS BitField Example

The Bit Field variables named position, composite2 and composite3 are user customizable using the RFS Construct command. The position variable is special in that the positionrate variable can be used to specify the repetition rate for the position variable. The composite2 and composite3 variables must be obtained using command/response (Get\_Value/Value\_Is).

Below is an example that uses the Construct command to define the value object reported by the position variable. The change to position is confirmed using the Show command with the corresponding Format message response. Then the data is obtained using the Get\_Value command and the corresponding Value\_Is response.

The message bytes for RFS are in hexadecimal form. These bytes are expected to be sent by the host application in binary form without the comments denoted by “//”.

// Use NorthTek ASCII command to get the position VID (DC-4E 2.3.5, GEDC-6E 2.4.9, AHRS-8 2.2.1

// or later)

position dvid@ .

// response is 30 => 1E hex

// BitField Construct command

// defines the position variable to have

// - quaternion

// - pitch

// - roll

// - yaw

// - yawt

// - temperature

// SAPP Layer - see RFSProtocolSuite Appendix E

01 // SOH - SAPP packet frame

26 // SAPP packet body size

40 // Error Options – No ack/RFS Protocol

// RFS Layer - see RFSProtocolSuite Figure RFS Outer Layer Protocol Diagram

02 // Revision Level (Construct command is listed as RFS2)

0000001D // RFS payload size

0C // Construct command

1081 // DLE for 01 // Sequence number

1E // VID for position variable

81 // BitField - Read Only

// see RFSProtocolSuite Figure BitField Constructor Object

19 // Field Size

1086 // DLE for 06 // Field Count

0008000C // 0 - quaternion start bit number - first 12 bits of 0008000C

// 128 - quaternion field size (bits) - second 8 bits of 0008000C

// 12 - quaternion VID - right-most 12 bits of 0008000C

08020008 // 128 - pitch start bit number

// 32 - pitch bit field size

// 8 - pitch VID

0A020009 // 160 -roll start bit number

// 32 - roll field size

// 9 - roll VID

0C02000A // 192 - yaw start bit number

// 32 - yaw field size

// 10 - yaw VID

0E02000B // 224 - yawt start bit number

// 32 - yawt field size

// 11 - yawt VID

1090 // DLE for 10

020078 // 256 - temperature start bit number

// 32 - temperature field size

// 120 - temperature VID

// end of RFS Layer, back to SAPP layer

A73A // CRC

03 // ETX

// -----------------------------------------------

// Issue a Show command to confirm the Construct worked. The response

// is a Format message (see RFSProtocolSuite Command Table)

// SAPP Layer - see RFSProtocolSuite Appendix E

01 // SOH - SAPP packet frame

0C // SAPP packet body size

40 // Error Options - No Ack - RFS Protocol

// RFS Layer - see RFSProtocolSuite Figure RFS Outer Layer Protocol Diagram

1081 // DLE - 01 // Revision Level (Show command is RFS1)

00000000 // payload size

05 // Show command

02 // Sequence number

1E // VID for position variable

// RFSProtocolSuite Figure RFS Outer Layer Protocol Diagram says "-no object-"

7E // first byte of CRC

1095 // DLE for 15 – second byte of CRC

03 // ETX

// -----------------------------------------------

// This is the Format response to the Show command

01 // SOH - SAPP packet frame

58 // SAPP packet body size

60 // No Ack/Nak - RFS Protocol

1081 // DLE - 01 // Revision Level (Format command is RFS1)

0000004D // payload size

1086 // DLE - 06 Format response

02 // Sequence number of corresponding Show command

1E // VID - "position" variable

// RFSProtocolSuite Figure RFS Outer Layer Protocol Diagram says Format has DDO

// RFSProtocolSuite Figure Data Description Object (Top Level)

80 // SAP Byte - BitField - Temporary Read/Write

// RFSProtocolSuite Figure BitField DDO

4B // Field Size

09 // Name Field Size

706F736974696F6E00 // Name Field "position"

1090 // DLE - 10 (16 decimal) Field Count

0008000C // 0 - quat start bit number - 12 bits

// 128 - quat field size - 8 bits

// 12 - quat VID - 12 bits

08020008 // 128 - pitch start bit number

// 32 - pitch bit field size

// 8 - pitch VID

0A020009 // 160 -roll start bit number

// 32 - roll field size

// 9 - roll VID

0C02000A // 192 - yaw start bit number

// 32 - yaw field size

// 10 - yaw VID

0E02000B // 224 - yawt start bit number

// 32 - yawt field size

// 11 - yawt VID

1090 // DLE for 10

020078 // 256 - temperature start bit number

// 32 - temperature field size

// 120 - temperature VID

00000000 // unused Cross Reference Fields

00000000

00000000

00000000

00000000

00000000

00000000

00000000

00000000

00000000

// end of RFS Layer, back to SAPP layer

1090 // DLE for 10 - First byte of CRC

43 // Second byte of CRC

03 // ETX

// -----------------------------------------------

// Use the Get\_Value command to obtain the data. The response will return in

// the Value\_Is message.

01 // SOH - SAPP packet frame

0B // SAPP packet body size

40 // No Ack - RFS Protocol

1081 // DLE - 01 // Revision Level (Get\_Value command is RFS1)

00000000 // payload size

08 // Get\_Value command

1083 // DLE for 03 Sequence number

1E // VID - "position" variable

// RFSProtocolSuite Figure RFS Outer Layer Protocol Diagram says "-no object-"

// end of RFS Layer, back to SAPP layer

0F75 // CRC

03 // ETX

// -----------------------------------------------

// This is the Value\_Is response to the Get\_Value command

01 // SOH - SAPP packet frame

32 // SAPP packet body size

60 // No Ack/Nak - RFS Protocol

1081 // DLE - 01 // Revision Level (Value\_Is command is RFS1)

00000027 // RFS payload size

09 // Value\_Is response ID

1083 // DLE 03 - sequence number for the corresponding Get\_Value

1E // VID - "position" variable

// RFSProtocolSuite Figure RFS Outer Layer Protocol Diagram says Value\_Is

// returns a Value Object.

// RFSProtocolSuite Figure Value Object (VO)

80 // SAP Byte - BitField

// RFSProtocolSuite Figure BitField Value Field

25 // Field Size

09 // Count

// The following values are in floating point format

3F4DA46A // quaternion - w

BBAF2B45 // quaternion - x

3C837585 // quaternion - y

BF1868B8 // quaternion - z

3F8E674D // pitch

BFCB2E52 // roll

438F72E4 // yaw

438F72E4 // yawt

42 1081 C000 // Has DLE - 4201C000 temperature

// end of RFS Layer, back to SAPP layer

F121 // CRC

03 // ETX

// To cause position to output on a periodic basis, set the positionrate variable via RFS or NorthTek

// NorthTek ASCII text:

positionrate 2000 set // output position every 2 seconds

// <ctrl-s> inhibits the output, <ctrl-q> re-enables it, set positionrate back to 0 to stop

## NMEA

NMEA commands use ASCII text strings to communicate with the digital compass through the User Port. All commands should follow the command syntax exactly. No white space should be included in NMEA command strings. Data is formatted with the first character being “***$***” to signify the start of a NMEA command and include a ***“\*”*** to signify the start of the checksum.

It is not necessary to include any checksum characters in commands sent to the digital compass. Strictly speaking, this is not standard NMEA protocol but manually entering checksums is not practical. However an option exists to enable NMEA checksums if additional error protection is desired. This option is off by default and may be enabled via a NorthTek or a NMEA command string (see the nmeaignorechecksum explanation in the Protocols section). All NMEA responses from the compass will contain a checksum followed by a carriage return and line feed.

The inertial system by default at power up does not echo the NMEA character strings, however a NorthTek command exists that will enable echo of NMEA command strings as they are typed (***nmeaecho 1 set drop<cr>***). This mode is provided for evaluators that wish to use a serial terminal to hand type inertial system commands. Each command sent must be terminated with a line feed ***<cr><lf>*** (entered as ***<CTRL-J>*** in some terminal programs such as Tera Term). You may use the standard <cr><cr><lf> as well To turn off the character echo, use the command: ***nmeaecho 0 set drop<cr>***

The checksum value is the result of XORing the ASCII bytes between, but not including, the “***$***” and “***\****” characters. This one byte value is reported in the output word by two ASCII characters representing two hex digits, with the most significant nibble first.

NMEA commands must be completed in a timely manner. The terminating character must be received within 5 seconds after the reception of the last character. If the command is not received in the time allotted all characters received will be discarded and the command ignored. This is to prevent the inertial system from getting stuck in the NMEA protocol.

Response example:

*$HCHDM,300.4,M\*2E<cr><lf>*

“$” = Start of NMEA text message

“HCHDM” = Response header from compass. (HC=Magnetic Compass, HDM = Magnetic Heading)

“300.4” = Heading in degrees

“M” = Magnetic Heading

“\*” = Start of checksum field

“2E” = Hexadecimal checksum value

Any NMEA command can be repeated continuously by adding the repeat (RPT) instruction to the end of the command string (before the checksum). The repetition rate is given in seconds and should be in the range 0.01 to 500.0 seconds. For example, to request magnetic heading continuously at a 0.5second (2Hz) rate, the following command should be issued:

$xxHDM,RPT=0.5<cr><lf>

The compass will respond by continuously sending magnetic heading information every 0.5 seconds. To discontinue RPT function, simply send another NMEA command. Repeat commands are not maintained through a power cycle or low power mode.

In the following command descriptions, <int#> represents an integer number (the <> will not be in the response). The symbols such as ###.# with decimal points represent floating point numbers with a fixed format. The symbol float# represents floating numbers without a fixed format.

### Raw Magnetics

Description: Reads current magnetics directly from magnetometers (Mx, My, and Mz). These are raw sensor readings and do not yet have any calibration parameters applied.

Send: ***$PSPA,MR<cr><lf>***

Response: **$PSPA,MRx=<int#>,MRy=<int#>,MRz=<int#>\*<checksum in hex>**

Response Example:

**$PSPA,MRx=1553,MRy=-1669,MRz=-1419\*60**

### Heading - Magnetic

Description: Reads the current magnetic heading. The heading is compensated for platform tilt. The xx in xxHDM represent “don’t care” characters and so are not used.

Send: ***$xxHDM<cr><lf>***

Response: **$HCHDM,<###.# >,M\*<checksum in hex>**

<###.#> = Heading in range 000.0 to 359.9

Response Example:

**$HCHDM,300.4,M\*2E**

### Heading - TRUE

Description: Reads the current true heading. The heading is compensated for platform tilt. True heading is the magnetic heading corrected for magnetic variation.

Send: ***$xxHDT<cr><lf>***

Response: **$HCHDT,<###.#>,T\*<checksum in hex>**

<###.#> = Heading in range 000.0 to 359.9

Response Example:

**$HCHDT,295.9,T\*2B**

### Magnetic Variation

Description: Set the magnetic variation angle. The variation angle is used in adjusting magnetic heading to true heading. This will override any previous internally computed World Magnetic value.

Send: ***$xxVAR,<###.#>,<E or W><cr><lf>***

Response: **$HCVAR,<###.#>,<E or W>\*<checksum in hex>**

<###.#> = Variation in range 000.0 to 180.0

<E or W> = Direction of variation East(+) or West(-)

Send Example:

***$xxVAR,4.2,W<cr><lf>***

Response Example:

**$HCVAR,004.2,W\*31**

### Automatic Magnetic Variation

Description: Computes the local magnetic variation based on the device’s current geographical location using the World Magnetic Model. Latitude and longitude are entered in degrees with + being north and east respectively. Altitude is entered in meters above sea level. The day is entered as a fractional year based on the current day of the year (i.e. February 15 is the 46th day of the 2010. In fractional terms, this would be 46/365 = 0.126. The Day value for February 15, 2010 would then be entered as 2010.1 (resolution beyond a tenth causes negligible change in variation) Once the computation is complete, the magnetic variation will be updated in the compass. Since the World Magnetic Model is not valid at the North and South poles, the Latitude is limited to -89.9 to 89.9. See the World Magnetic Model section for more information.

Send:

***$PSPA,AUTOVAR,<#Latitude>,<#Longitude>,<#Altitude>,<#Day><cr><lf>***

Response: **$PSPA,AutoVar=<###.#>,\*<checksum in hex>**

#Latitude = Geodetic latitude in degrees. North(+) or South(-) (-89.9 .. 89.9)

#Longitude = Geodetic longitude in degrees. East(+) or West(-). (-180.0 .. 180.0)

#Altitude = Geodetic altitude in meters from sea-level (-10000.0 .. 1000000.0)

#Day = Approximate date in terms of 1/10th of a year (2010.0 .. 3000.0)

Send Example:   
***$PSPA,AUTOVAR,29.12,-81.35,100,2011.6<cr><lf>***

Response Example:   
**$PSPA,AutoVar=-005.9\*66**

### Magnetic Vector

Description: Measures the magnetic field strength along each axis (X, Y, and Z) and total absolute field strength (Mt) in milligauss.

Send: ***$PSPA,M<cr><lf>***

Response:

**$PSPA,Mx=<int#>,My=<int#>,Mz=<int#>,Mt=<int#>\*<checksum in hex>**

Response Example:

**$PSPA,Mx=63,My=-261,Mz=-262,Mt=376\*29**

### Magnetic In-Field Calibration

Description: Controls the compass in field calibration process. Field calibration involves multiple steps:

1. Select the mode, 3D or 2D.
2. Ensure the orientation is set to the default horizontal state
3. Start the calibration
4. Capture Points
5. End point capture
6. Allow the calibration algorithm to converge.
7. Terminate calibration, causing values to be stored.

The compass is already factory calibrated in an environment free from magnetic distortions. When the compass is first used in the application, it must learn the local magnetic distortions.

3-D Field Calibration

The 3-D calibration procedure is used for applications that move outside of a plane. It does **not** require full 360 deg pitch and roll capability for this calibration but will perform better given more range of movement.

The results are stored in magFieldCalX, magFieldCalY and magFieldCalZ with the first value being offset and the second being a divisor.

If using the NMEA command set, you should place 300ms after receipt of each response before sending the next command. The exception to thise rule is when the unit is capturing points. Since there is averaging at each point outlined by the variable “calPointsNumToAvg”, you should delay 10ms\*calPointsNumToAvg. Alternatively, you can also query the variable calNumPoints to determine when that variable has increased by 1 before sending the next command.

Send: ***$PSPA,CAL=3D<cr><lf>***

***$PSPA,CAL\_CMD=START\_CAL<cr><lf>***

Change position and repeat CAPTURE\_POINT totaling 4 to 12 points

***$PSPA,CAL\_CMD=CAPTURE<cr><lf>  
.  
.  
.  
$PSPA,CAL\_CMD=CAPTURE<cr><lf>***Stop the capture after a minimum of 4 points***.  
$PSPA,CAL\_CMD=END\_CAPTURE<cr><lf>  
$PSPA,MAGERR,RPT=0.5<cr><lf>***Watch for the magnetic error quality indicator to settle to a minimum value  
***$PSPA,CAL\_CMD=END\_CAL<cr><lf>***

***$PSPA,CAL=OFF<cr><lf>***

Response:

**$PSPA,CAL=3D\*3A  
$PSPA,CAL\_CMD=START\_CAL\*09  
$PSPA,CAL\_CMD=CAPTURE,POINTS=<int#>\*<checksum in hex>**

**.  
.  
.  
$PSPA,CAL\_CMD=CAPTURE,POINTS=<int#>\*<checksum in hex>  
$PSPA,CAL\_CMD=END\_CAPTURE\*0C  
$PSPA,MagErr=<float#>\*<checksum in hex>  
$PSPA,MagErr=<float#>\*<checksum in hex>**

**.  
.  
.  
$PSPA,MagErr=<float#>\*<checksum in hex>   
$PSPA,MagErr=<float#>\*<checksum in hex>  
$PSPA,CAL\_CMD=END\_CAL\*06**

**$PSPA,CAL=OFF\*02**

2-D Field Calibration

The 2-D calibration procedure is used for applications that stay within a plane containing two of the navigation module’s axes and rotating about the third axis. The point capture positions should be chosen to be distributed well about the circle of rotation. The 2-D calibration differs from the 3-D in that it requires that the World Magnetic Model to have been run previously (see the $PSPA,AUTOVAR command).

The results are stored in magFieldCalX, magFieldCalY and magFieldCalZ with the first value being offset and the second being a divisor.

Send: ***$PSPA,CAL=2D<cr><lf>***

***$PSPA,CAL\_CMD=START\_CAL<cr><lf>***

Change position and repeat CAPTURE\_POINT totaling 4 to 12 points

***$PSPA,CAL\_CMD=CAPTURE<cr><lf>  
.  
.  
.  
$PSPA,CAL\_CMD=CAPTURE<cr><lf>***Stop the capture after a minimum of 4 points. ***$PSPA,CAL\_CMD=END\_CAPTURE<cr><lf>  
$PSPA,MAGERR,RPT=0.5<cr><lf>***Watch for the magnetic error quality indicator to settle to a minimum value  
***$PSPA,CAL\_CMD=END\_CAL<cr><lf>***

***$PSPA,CAL=OFF<cr><lf>***

Response:

**$PSPA,CAL=2D\*3B  
$PSPA,CAL\_CMD=START\_CAL\*09  
$PSPA,CAL\_CMD=CAPTURE,POINTS=<int#>\*<checksum in hex>**

**.  
.  
.  
$PSPA,CAL\_CMD=CAPTURE,POINTS=<int#>\*<checksum in hex>  
$PSPA,CAL\_CMD=END\_CAPTURE\*0C  
$PSPA,MagErr=<float#>\*<checksum in hex>  
$PSPA,MagErr=<float#>\*<checksum in hex>**

**.  
.  
.  
$PSPA,MagErr=<float#>\*<checksum in hex>   
$PSPA,MagErr=<float#>\*<checksum in hex>  
$PSPA,CAL\_CMD=END\_CAL\*06**

**$PSPA,CAL=OFF\*02**

### Gyro[[3]](#footnote-4) Calibration

For this procedure, the navigation unit must be in a static position. The gyro calibration procedure will complete automatically. The results are stored in gyroFieldCalX, gyroFieldCalY and gyroFieldCalZ with the first value being offset and the next value being a divisor.

Send: ***$PSPA,GYROCAL<cr><lf>***

Response:

**$PSPA,GYROCAL\*73**

### Magnetic Adaption Error

Description: Indicates quality of the adaptive magnetic calibration process. Smaller values represent lower error corresponding to better magnetic calibration.

Send: ***$PSPA,MAGERR<cr><lf>***

Response: **$PSPA,MagErr=<float#>\*<checksum in hex>**

Response Example:

$PSPA,MagErr=0.876963#16

### Raw Acceleration

Description: Reads current acceleration directly from accelerometers (AccelX, AccelY, AccelZ). These are raw sensor readings and do not yet have any calibration parameters applied.

Send: ***$PSPA,AR<cr><lf>***

Response: **$PSPA,ARx=<int#>,ARy=<int#>,ARz=<int#>\*<checksum in hex>**

Response Example:

$PSPA,ARx=2052,ARy=1991,ARz=1284\*61

### Acceleration Vector

Description: Measures the acceleration along each axis (X, Y, and Z) and total magnitude (At = Atotal) in milli-g.

Send: ***$PSPA,A<cr><lf>***

Response:

**$PSPA,Ax=<int#>,Ay=<int#>,Az=<int#>,At=<int#>\*<checksum in hex>**

Example Response:

$PSPA,Ax=-70,Ay=76,Az=995,At=1000\*02

### Raw Gyro

Description: Reads current gyro directly from gyrometers (GyroX, GyroY, GyroZ). These are raw sensor readings and do not yet have any calibration parameters applied.

Send: ***$PSPA,GR<cr><lf>***

Response: **$PSPA,GRx=<int#>,GRy=<int#>,GRz=<int#>\*<checksum in hex>**

Response Example:

$PSPA,GRx=133,GRy=93,GRz=80\*5E

### Gyro Vector

Description: Provides the calibrated gyro values along each axis (X, Y, and Z) in milliDegrees per second.

Send: ***$PSPA,G<cr><lf>***

Response: **$PSPA,Gx=<float#>,Gy=<float#>,GRz=<float#>\*<checksum in hex>**

Response Example:

$PSPA,Gx=165.974,Gy=285.613,Gz=-168.670\*31

### Pitch and Roll Output

Description: Reads the current platform orientation (Pitch and Roll).

Send: ***$PSPA,PR<cr><lf>***

Response: **$PSPA,Pitch=<##.#>,Roll=<###.#>\*<checksum in hex>**

Pitch range (degrees): -90.0 to +90.0

Roll range (degrees): -180.0 to +180.0

Example Response:

$PSPA,Pitch=+18.2,Roll=-042.4\*56

### Quaternion Vector

Description: Provides the attitude in the form of a quaternion. The x, y, z define the axis of rotation and the w provides the angle (theta) of rotation about that axis where theta = arccos(w) \* 2

Send: ***$PSPA,QUAT<cr><lf>***

Response:

**$PSPA,QUATw=<#.######>,x=<#.######>, y=<#.######>,z=<#.######>\*<checksum in hex>**

Example Response:

$PSPA,QUATw=0.314214,x=0.007481,y=-0.034541,z=-0.948694\*0D

### Temperature

Description: Reads the internal temperature channel of the on-board microcontroller and converts to degrees C. This measurement is calibrated at the factory for general use even though it is not required by the compass in determining an accurate heading.

Send: ***$PSPA,TEMP<cr><lf>***

Response: **$PSPA,Temp=<##.#>,C\*<checksum in hex>**

<##.#> = Variation in range -20.0 to +70.0C

Response Example:

$PSPA,Temp=+24.1,C\*72

### BAUD RATE

Description: The factory default BAUD setting is 115200 Baud (Baud value = 8). When the baud rate command is issued, the compass will echo back the command once at the current baud rate and then again at the new baud rate. The baud rate will be stored in EEPROM and will become the new operating communication rate for the UART. The baud rate will apply to both Legacy, RFS, NorthTek and NMEA commands issued on the UART.

Send: ***$PSPA,BAUD=<# Baud value><cr><lf>***

Response: **$PSPA,Baud=<# Baud value>\*<checksum in hex>**

Send Example:   
***$PSPA,BAUD=4<cr><lf>***

Response Example:

$PSPA,BAUD=4\*25

Acceptable Baud Rate Values:

0= 300 Baud

1 = 1200 Baud

2 = 2400 Baud

3 = 4800 Baud

4 = 9600 Baud

5 = 19.2 kBaud

6 = 38.4 kBaud

7 = 57.6 kBaud

8 = 115.2 kBaud

### Mounting Configuration

Description: Sets the mounting orientation of the compass platform. The default orientation is horizontal (see Figure 4 in the Navigation Sensors Product Guide). To determine the orientation setting, read the acceleration vector. When in a static level condition, Az should be approximately +1000mg and Ax and Ay should be close to zero.

Send: ***$PSPA,MOUNT=<’H’ for horizontal or ‘V’ for vertical><cr><lf>***

Response: **$PSPA,Mount=<’H or V’>\*<checksum in hex>**

Send Example:

$PSPA,MOUNT=V<cr><lf>

Response Example:

$PSPA,Mount=V\*18

### Read Transducers

Description: Reads current magnetic heading, true heading, pitch, roll, temperature, and magnetic error. This command provides the most frequently used information in one command string.

Send: ***$xxXDR<cr><lf>***

Response: **$HCXDR, <Transducer string as described below>\*<checksum in hex>**

<Transducer String> =

A, - Angular Displacement Measurement

###.#, - Magnetic Heading

D, - Units of Degrees for Magnetic Heading

A, - Second Angular Displacement Measurement

###.#, - True Heading

D, - Units of Degrees for True Heading

A, - Third Angular Displacement Measurement

+##.#, - Pitch

D, - Units of Degrees for Pitch

A, - Fourth Angular Displacement Measurement

+###.# - Roll

D, - Units of Degrees for Roll

C, - Temperature Measurement

+##.#, Temperature

C, - Units of Degrees C for Temperature

G, - Generic Measurement

### - Magnetic Error (measurement is unitless)

Send Example:

$xxXDR<cr><lf>

Response Example:

$HCXDR,A,281.3,D,A,281.3,D,A,+07.9,D,A,-000.8,D,C,+21.1,C,G,0216\*2C

### NMEA RFS Access

Most of the RFS variables are accessible via the NMEA protocol. Strings and BitFields are not supported (see the NMEA column in the 2.4.1 Variable Summary Table). This section describes the method used to set or get these variables. Note: NMEA and RFS variable names are case sensitive and must be matched exactly.

NMEA access to RFS variables (see Variable Summary Table, 2.4.1 for variable availability) is with one of two commands, with options. The first command is to “get” or read a variable. The syntax is:

***$PSRFS,<rfs variable name>,get<cr><lf>***

The inertial system will respond in kind with the variable name and the values of the variable. For Example the following command produces:

***$PSRFS,yaw,get<cr><lf>***

**$PSRFS,yaw,286.672424\*38<cr><lf>**

Variables that may be set, are “settable” by a similar mechanism except by replacing the get command with a set command and providing the new values separated by commas. For example, the sequence below indicates how to use the NMEA/RFS protocol to get the inertial system orientation and then set it to another value:

***$PSRFS,orientation,get<cr><lf>***

**$PSRFS,orientation,0\*02<cr><lf>**

***$PSRFS,orientation,set,1<cr><lf>***

**$PSRFS,orientation,1\*03<cr><lf>**

### NMEA/RFS RPT and GLOM

NMEA/RFS commands may have the RPT=# option applied to make an output repeat at the specified rate. In addition that NMEA/RFS commands allow additional (up to 8) NMEA/RFS commands to be “GLOM”d onto the current repeating output. This is best described by the following example that shows how to output the magnetic heading (yaw) at a 5 second rate and to GLOM on the true heading (yawt) at the same rate.

***$PSRFS,yaw,get,RPT=5.0<cr><lf>***

**$PSRFS,yaw,287.384308\*3C** << immediate response to get command

**$PSRFS,yaw,287.273376\*3C** << occurring at 5 second rate

**$PSRFS,yaw,287.244049\*37** << occurring at 5 second rate

***$PSRFS,yawt,get,GLOM<cr><lf>***

**$PSRFS,yawt,287.167603\*49** << immediate response to get command

**$PSRFS,yaw,287.301758\*30** << occurring at 5 second rate

**$PSRFS,yawt,287.301758\*44** << output at same time as line above

**$PSRFS,yaw,287.294495\*3F** << occurring at 5 second rate

**$PSRFS,yawt,287.294495\*4B** << output at same time as line above

**$PSRFS,yaw,287.294983\*35** << occurring at 5 second rate

**$PSRFS,yawt,287.294983\*41** << output at same time as line above

## Legacy Binary

Legacy Binary is intended to be only used by users of the older Sparton compasses such as the SP3002, SP3003, and SP3004. As a result, it does not have access to any of the newer data such as the gyro data and so is not recommended for new designs.

All Legacy Binary commands sent to the compass begin with a header byte (0xA4) and end with a termination byte (0xA0). Since the I-Field calibration routines have been modified and improved, the In-Field Calibration commands of Legacy Binary are no longer supported.

If the compass does not recognize a sequence of bytes as being a legacy command, no response is produced. If the compass recognizes the command but, for some reason, cannot execute it, it will respond with an error code. All error codes begin with a header byte (0xAE) and end with a termination byte (0xA0). Commands which commence and do not conclude within a timely manner will be rejected. All multi-byte values are sent most significant byte (MSByte) first.

Error Code Format: 3 Bytes (***0xAE,<8-bit error code>,0xA0***)

8-bit Error Codes:

0xFF = Improper command termination (i.e. no 0xA0 found)

0xFE = Receive buffer overflow

0xFD = Invalid parameter associated with given command

### Raw Magnetics

Description: Reads current magnetics directly from magnetometers (Mx, My, and Mz). These are raw sensor readings and do not yet have any calibration parameters applied.

Send: 3 Byte (***0xA4,0x01,0xA0***)

Response: 9 Bytes   
(**0xA4,0x01,<Mx>,<My>,<Mz as 16-bit integers MS byte first>,0xA0**)

### Heading - TRUE

Description: Reads the current true heading. The heading is compensated for platform tilt. True heading is the magnetic heading corrected for magnetic variation.

Send: 3 Byte (***0xA4,0x02,0xA0***)

Response: 5 Bytes   
(**0xA4,0x02,<Heading as a 16-bit signed integer>,0xA0**)

Heading (degrees) = (16-bit Heading value)\*360/4096

Heading Range = 0.0 to +359.9

### Heading - Magnetic

Description: Reads the current magnetic heading. The heading is compensated for platform tilt.

Send: 3 Bytes (***0xA4,0x09,0xA0***)

Response: 5 Bytes   
(**0xA4,0x09,<Heading as a 16-bit signed integer>,0xA0**)

Heading (degrees) = (16-bit Heading value)\*360/4096

Heading Range = 0.0 to +359.9

### Magnetic variation

Description: Set the magnetic variation angle. The heading will be adjusted to indicate true north instead of magnetic north. Magnetic variation angles >+180 and <-180 will be limited to +180 and –180 respectively.

Send: 5 Bytes (***0xA4,0x83,<16-bit signed integer value MSB first>,0xA0***)

Response: 5 Bytes   
(**0xA4,0x83,<16-bit signed Variation>,0xA0**)

16-bit signed integer = (Magnetic Variation)\*10.0

### Automatic Magnetic Variation

Description: Latitude, Longitude, Altitude, and Day should be programmed separately using their respective commands before issuing this command. Automatic variation will compute the local magnetic variation based on the device’s current geographical location (geodetic coordinate system referenced to the WGS 84 ellipsoid). Once the computation is complete, the magnetic variation will be updated in the compass. *NOTE: TO RETAIN MAGNETIC VARIATION ACCURACY, THE MAGNETIC MODEL MUST BE UPDATED EVERY FIVE YEARS. A SEPARATE* PROGRAM *IS AVAILABLE ON THE SUPPLIED CD WHICH WILL ASSIST IN DOWNLOADING NEW COEFFICIENTS INTO THE DIGITAL COMPASS. THIS ONLY AFFECTS THE CALCULATION OF TRUE HEADING AND DOES NOT AFFECT MAGNETIC HEADING ACCURACY.* NOTE: Resetting the magnetic parameters to the factory default does not affect the magnetic variation information. To clear out the magnetic variation, manually set the magnetic variation to zero

Send: 3 Byte (***0xA4,0x0F,0xA0***)

Response: 5 Bytes   
**(0xA4,0x0F,<Variation as a 16-bit signed integer>,0xA0**)

16-bit signed integer = (Magnetic Variation)\*10.0

### Latitude

Description: Set the geodetic latitude angle in degrees (geodetic coordinate system referenced to the WGS 84 ellipsoid). The magnetic variation will not change until latitude, longitude, altitude, and day have been programmed and the Automatic Variation command is issued. Latitude >+90 or <-90 will be limited to +90 and –90 respectively.

Send: 5 Bytes   
(***0xA4,0x8B,<16-bit signed integer value MSB first>,0xA0***)

Response: 5 Bytes   
(**0xA4,0x8B,<16-bit Latitude>,0xA0**)

16-bit signed integer = (North(+) or South(-) Latitude in degrees)\*100.0

### Longitude

Description: Set the geodetic longitude angle in degrees (geodetic coordinate system referenced to the WGS 84 ellipsoid). The magnetic variation will not change until latitude, longitude, altitude, and day have been programmed and the Automatic Variation command is issued. . Latitude >+180 or <-180 will be limited to +180 and –180 respectively.

Send: 5 Bytes   
(***0xA4,0x8C, 16-bit signed integer value MSB first,0xA0***)

Response: 5 Bytes   
(**0xA4,0x8C, 16-bit signed Longitude,0xA0**)

16-bit signed integer = (East(+) or West(-) Longitude in degrees)\*100.0

### Altitude

Description: Set the geodetic altitude in meters above sea level (geodetic coordinate system referenced to the WGS 84 ellipsoid). The magnetic variation will not change until latitude, longitude, altitude, and day have been programmed and the Automatic Variation command is issued. Altitude >+32767 or <-32767 will be limited to +32767 and –32767 respectively.

Send: 5 Bytes   
(***0xA4,0x8D, 16-bit signed integer value MSB first,0xA0***)

Response: 5 Bytes   
(**0xA4,0x8D, 16-bit signed Altitude,0xA0**)

16-bit signed integer = +/- Altitude in meters

### Day

Description: The day is entered as a fractional year based on the current day of the year (i.e. February 15 is the 46th day of the 2008. In fractional terms, this would be 46/365 = 0.126. The Fractional Day value for February 15, 2008 would then be 2008.1 (resolution beyond a tenth causes negligible change in variation). The magnetic variation will not change until latitude, longitude, altitude, and day have been programmed and the Automatic Variation command is issued. Day < 2005 will be limited to 2005.

Send: 5 Bytes   
(***0xA4,0x8E, 16-bit unsigned integer value MSB first,0xA0***)

Response: 5 Bytes   
(**0xA4,0x8E, 16-bit unsigned Day,0xA0**)

16-bit unsigned integer = (Fractional Day)\*10.0

### Magnetic Vector

Description: Measures the magnetic field strength along each axis (X, Y, and Z) and total absolute field strength (MAtotal) in milligauss.

Send: 3 Bytes  
(***0xA4,0x04,0xA0)***

Response: 11 Bytes   
(**0xA4,0x04, MAx, MAy, MAz, MAtotal as 16-bit integers,0xA0**)

### Raw Acceleration

Description: Reads current acceleration directly from accelerometers (AccelX, AccelY, AccelZ). These are raw sensor readings and do not yet have any calibration parameters applied.

Send: 3 Bytes   
(***0xA4,0x05,0xA0***)

Response: 9 Bytes   
(**0xA4,0x05, AccelX, AccelY, AccelZ as 16-bit integers,0xA0**)

### Pitch and Roll Output

Description: Reads the current platform orientation (Pitch and Roll).

Send: 3 Byte   
(***0xA4,0x06,0xA0***)

Response: 7 Bytes   
(**0xA4,0x06, Pitch, Roll as 16-bit signed integers,0xA0**)

Pitch (in degrees) = (Response Value)\*90/4096

Pitch Range = -90 to +90

Roll (in degrees) = (Response Value)\*180/4096

Acceleration Vector Roll Range = -180 to +180

### Acceleration Vector

Description: Measures the acceleration along each axis (X, Y, and Z) and total absolute strength (Atotal) in milli-g.

Send: 3 Byte   
(***0xA4,0x07,0xA0***)

Response: 11 Bytes   
(**0xA4,0x07, Ax, Ay, Az, Atotal as 16-bit integers,0xA0**)

### Temperature

Description: Reads the internal temperature channel of the on-board microcontroller. This measurement is calibrated at the factory, though not required by the compass in determining an accurate heading.

Send: 3 Bytes   
(***0xA4,0x11,0xA0***)

Response: 5 Bytes  
(**0xA4,0x11, Temperature as 16-bit signed integer MSB first,0xA0**)

Temperature\_C = (Temperature\_MSB\*256 + Temperature\_LSB)/10.0

### BAUD RATE

Description: The factory default BAUD setting is 115200 Baud (0x08). When the baud rate command is issued on the UART, the compass will echo back the command once at the current baud rate and then again at the new baud rate. The baud rate will be stored in EEPROM and will become the new operating communication rate for the UART. The baud rate will apply to Legacy, RFS, NorthTek and NMEA commands issued on the UART.

Send: 4 Bytes   
(***0xA4,0x57, 8-bit BAUD value MSB first,0xA0***)

Response: 4 Bytes   
(**0xA4,0x57, 8-bit BAUD value,0xA0**)

Acceptable Baud Rate Values:

0x01 = 1200 Baud

0x02 = 2400 Baud

0x03 = 4800 Baud

0x04 = 9600 Baud

0x05 = 19.2 kBaud

0x06 = 38.4 kBaud

0x07 = 57.6 kBaud

0x08 = 115.2 kBaud

### Mounting Configuration

Description: Sets the mounting orientation of the compass platform. The default orientation is horizontal. For vertical orientations consult the particular device’s data sheet. To determine the orientation setting, read the acceleration vector. When in a static level condition, Az should be approximately +1000mg and Ax and Ay should be close to zero.

Send: 4 Bytes   
(***0xA4,0x4A, 8-bit orientation {0x00=Horizontal,0x01=Vertical},0xA0***)

Response: 4 Bytes   
(**0xA4,0x4A, 8-bit orientation,0xA0**)

## NorthTek

NorthTek is not a protocol per se. NorthTek is a programming language, a command interpreter, and an execution environment. NorthTek is described in detail in a separate manual. NorthTek provides the user with a command line interface that allows direct interaction with the inertial system. NorthTek also allows the user to load custom programs into the inertial system that will execute a custom user application. The user may create programs that cause some of the standard protocol outputs but at user defined points, or may create custom output depending on the specific need. The user specific algorithms may be used to filter the output data, control the reporting rate, create unusual mounting configurations, or select multiple calibration sets, for example.

Because NorthTek is an environment unto itself, an entire manual is dedicated to the NorthTek System as listed in the references. The user should refer to that manual for detailed descriptions of the commands being used in this manual.

NorthTek provides a command line access to the internal database variables. NorthTek also provides raw and processed sensor data streamed at the acquisition rate. The sections that follow illustrate some of the NorthTek functionality that can be used on the inertial system. The reader should refer to the *NorthTek System Programming Manual* for detailed descriptions on syntax and semantics of the commands used in the examples that follow.

### Displaying Database Elements

The internal database is available to the NorthTek command line. See Appendix A for a list of the database elements. All database elements may be printed to the output using the NorthTek command “di.”. This command expects a reference to a data element on the stack, and then prints out the current value in a human readable form. The syntax is the name of the data element (case sensitive) followed by a space followed by “di.” (also case sensitive).

Some specific values are typically more interesting than others to the user. Some of the common values are:

Compass Heading (magnetic)

***yaw di.<CR>***

NorthTek prints:

yaw = 4.433077e+01

OK

pitch di.

pitch = 1.655078

OK

roll di.

roll = -5.246325e-01

OK

yaw di.

yaw = 12.752021

OK

yawt di.

yawt = 12.639648

OK

help pitch

Pitch angle in degrees, positive up

OK

help roll

Roll angle in degrees, positive right-side down

OK

orientation di.

orientation = Horizontal

OK

help magr

Raw Magnetometer X,Y,Z,Total array

OK

help magp

Processed Magnetometer X,Y,Z array

OK

help accelr

Raw Accelerometer X,Y,Z array

OK

help accelp

Processed Accelerometer X,Y,Z array

OK

magr di. magp di. accelr di. accelp di.

magr = 00-- 5424.000000

01-- 1232.000000

02-- 6600.000000

03-- 8651.347656

magp = 00-- 290.135895

01-- -40.983604

02-- 338.315552

accelr = 00-- 0.000000e+00

01-- 0.000000e+00

02-- 0.000000e+00

accelp = 00-- -26.452639

01-- -7.922364

02-- 999.194458

This example also shows that multiple commands can be entered on a single input line (up to 80).

Note that help is available for the database variables. The NorthTekSystem (NTS) outputs a longer, more informative string for the variable when

**help <variable>**

is entered.

### Outputting values in User Defined Format

The user can control the output formatting if desired. The example below shows how to first print the default formatted output for a variable, then print the output in a fixed format (equivalent to %8.3f in “C”).

pitch di.

pitch = 1.509415

OK

roll di.

roll = -4.413672e-01

OK

yaw di.

yaw = 6.415520

OK

8 3 pitch di@ ff. 1.510OK

8 3 roll di@ ff. -0.444OK

8 3 yaw di@ ff. 6.514OK

### Streaming Sensor Data

The NTS allows the user to stream the raw sensor data, the processed sensor data and the computed heading information. Note that once the command is entered the data streams continuously. The user can suspend the output of all streaming data temporarily by using the <CTRL-S> keystroke. Data can be resumed with the <CTRL-Q> keystroke. Various output streams are demonstrated in the example below:

**1 magp.pOK**

**MP:954362,5473,1193,6571,293.256195,-38.316196,336.443970,7.061350**

**MP:954462,5505,1198,6608,295.269257,-38.649624,338.835449,6.935188**

**MP:954561,5472,1216,6609,293.155518,-39.872185,338.939423,6.863805**

**MP:954661,5494,1235,6625,294.564667,-41.205887,339.979187,6.923146**

**MP:954761,5460,1192,6555,292.450928,-38.205055,335.404205,7.046950** <<< ctrl-s hit here

**0 magp.pOK**

**OK**

**OK**

**1 accelp.pOK** <<< ctrl-q hit here

**AP:1140739,-19,8,2027,-26.391603,-8.007813,999.548462,1.498477,-0.441541**

**AP:1140839,-19,8,2025,-26.367191,-7.925182,998.910889,1.504051,-0.448529**

**AP:1140938,-19,8,2024,-26.513676,-7.824708,998.474243,1.507221,-0.450943**

**AP:1141038,-19,8,2027,-26.428226,-7.995606,999.524048,1.512748,-0.450165**

**AP:1141138,-18,8,2026,-25.966551,-7.837541,999.211365,1.513467,-0.453428**

**AP:1141236,-18,8,2026,-26.171877,-7.971193,999.133423,1.503507,-0.451818** <<< ctrl-s hit here

**0 accelp.p**

**1 accelp.p 1 magp.pOK**<<< ctrl-q hit here

**AP:1211835,-19,8,2027,-26.513676,-7.983399,999.755981,1.500116,-0.440688**

**MP:1211854,5480,1220,6580,293.658783,-40.205616,337.067810,7.080771**

**AP:1211935,-19,7,2028,-26.562504,-7.641603,1000.036743,1.507705,-0.447420**

**MP:1211953,5456,1222,6628,292.148987,-40.316753,340.187134,7.076100**

**AP:1212034,-19,8,2027,-26.467350,-8.025342,999.837341,1.513208,-0.443575**

**MP:1212054,5448,1232,6580,291.645721,-40.983604,337.067810,7.101585**

**AP:1212135,-18,7,2026,-26.159672,-7.739259,999.353149,1.514449,-0.450098**

**MP:1212154,5464,1220,6611,292.652252,-40.205616,339.043396,7.164870** <<< ctrl-s hit here

**0 magp.p 0 accelp.pOK**

The format of the XXX.p commands is shown below.

#### Accelerometer

1 accel.p // enables print of raw accelerometer readings

// Output format is

// A:%6d,%6d,%6d,%6d

// printing timestamp(ms), x, y and z accelerometer values, 2048 = 1g for the 4g accelrange setting (default) and 1024=1g for the 8g accelrange setting

0 accel.p // disables

1 accelp.p // enables print of raw/processed accelerometer readings

// Output format is

// AP:%d,%d,%d,%d,%f,%f,%f

// printing timestamp(ms), x, y and z accelerometer raw values and x,y and z linearized values, the raw

// values are scaled such that 2048 = 1g for the 4g accelrange setting (default) and 1024=1g for the 8g accelrange setting and the linearized values are in the units of milli-gs.

0 accelp.p // disables

#### Gyro

1 gyro.p // enables print of raw gyroscope readings

// Output format is

// G:%6d,%6d,%6d,%6d

// printing timestamp(ms), x, y, z values

0 gyro.p // disables

1 gyrop.p // enables print of raw and processed gyro readings

// Output format is

// GP:%d,%d,%d,%d,%.3e,%.3e,%.3e

// printing timestamp(ms), x, y and z gyro raw values and x,y and z linearized values

// with units of radians/sample, use the value of gyroSampleRate to convert

0 gyrop.p // disables

#### Magnetometer

1 mag.p // enables print of raw magnetometer readings

// Output format is

// M:%6d,%6d,%6d,%6d

// printing timestamp(ms), x, y and z magnetometer values

0 mag.p // disables

1 magp.p // enables print of raw and processed magnetometer readings

// Output format is

// MP:%d,%d,%d,%d,%f,%f,%f

// printing timestamp(ms), x, y and z magnetometer raw values and x,y and z linearized values

0 magp.p // disables

#### All raw sensor data

1 s.p // enables print of all raw sensor data

// Output format is

// %d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d

// mag x,y,z, accel x,y,z, gyro x,y,z and raw temp (1/10 deg C) respectively

0 s.p // disables

#### Computed Roll, Pitch and Yaw

1 compass.p // enables print of computed attitude and heading

// Output format is

// C,%d,%7.2f,%7.2f,%7.2f

// timestamp(ms), pitch, roll and yaw respectively

0 compass.p // disables

#### Computed Quaternions

1 quat.p // enables print of computed quaternions

// Output format is

// QUAT:%f,%f,%f,%f

// quaternion vector (w,x,y,z) respectively

0 quat.p // disables

#### Custom Combination Print Streaming (User Port only)

The customizable print streaming commands allows selection of the data to be streamed and provides control of when the data is streamed. This capability is available in units DC-4E 4.2.1, GEDC-6E 4.0.2, AHRS-8 2.2.1 or later.

For this capability, there are three configuration variables: printmodulus, printtrigger and printmask plus the following set of constants:

magr\_trigger

magp\_trigger

accelr\_trigger

accelp\_trigger

gyror\_trigger

gyrop\_trigger

yaw\_trigger

yawt\_trigger

quat\_trigger

temp\_trigger

pitch\_trigger

roll\_trigger

time\_trigger

Each of the above trigger definitions is a single bit. If a bit is set in the printmask, then when a print trigger event occurs, the data that is associated with that bit will print in the output. Note that, although the triggers occur in interrupt handlers or high priority tasks, the actual printing occurs in a low priority task. This may cause a small delay in the print from the point of the trigger but is necessary to avoid interfering with the data acquisition and computations.

The printmodulus indicates how many trigger events must take place before the printing occurs.

For example, suppose we want to print the raw and processed magnetometer data

and the temperature and we want to print on every 100th processed magnetometer

calculation. To accomplish this, we would set the print mask using this sequence of NorthTek:

>>>   printmask

>>>   magr\_trigger magp\_trigger or temp\_trigger or set drop

The “drop” is needed because the “set” operator puts a status result on the top of the stack. If the “drop” is not used, then eventually, the stack may overflow.

To cause it to print on every 100th trigger event:

>>>   printmodulus 100 set drop

The trigger condition is set

>>>   printtrigger magp\_trigger set drop

Note that this kind of printing is on by default, but can be disabled with

printer\_mask d.off or suspended with CTRL-S, or by setting the printtrigger to 0

>>>   printtrigger 0 set drop

Note that we can cause the print to be output with the word "p.”.  With the “p.” command, we can write our own trigger condition in NorthTek and then evoke this printout.

Given the above configuration, the output is:

P:,659539,mr,5890,-712,7323,mp,327.566681,94.818001,385.820099,T,33.06

P:,660481,mr,5968,-695,7342,mp,314.355835,93.428726,386.015045,T,33.06

P:,661423,mr,5850,-661,7316,mp,330.460510,93.706581,385.495178,T,33.06

P:,662365,mr,5995,-742,7337,mp,314.104187,93.914970,386.534943,T,33.00

P:,663306,mr,5965,-699,7328,mp,324.169617,93.150871,385.625153,T,33.00

P:,664248,mr,5902,-694,7337,mp,320.206360,92.803551,386.210022,T,33.00

P:,665190,mr,5900,-677,7326,mp,320.080536,91.622673,385.495178,T,33.00

The interpretation of this output is P:,TIME,mr(indicates mag raw),x,y,z,mp(indicates mag processed),x,y,z,T(temp),value.

The printmodulus for accelr and gyror is applied to the RAW sample rates, i.e. 400 Hz on gyro/accel, NOT on the divided down values.

This is what we get when we set printmask to all of the trigger constants:

P:,878979,mr,6174,-751,7403,mp,337.317566,96.762985,390.499084,ar,22,38,2052,ap,-6.347656,-22.460938,1011.718750,gr,-8,-4,26,gp,0.000010,0.000039,-0.000102,y,342.53,yt,342.53,q,0.988351,-0.009816,0.004613,-0.151807,T,32.19

P:,879920,mr,5985,-763,7416,mp,327.126343,95.165321,389.914185,ar,21,34,2049,ap,-6.591797,-20.874023,1010.375977,gr,-3,-16,9,gp,-0.000003,0.000004,-0.000050,y,342.53,yt,342.53,q,0.988343,-0.009981,0.004853,-0.151841,T,32.19

P:,880862,mr,6045,-735,7397,mp,334.172119,97.874405,391.148926,ar,21,35,2046,ap,-6.835938,-21.118164,1009.033203,gr,6,-25,6,gp,-0.000035,-0.000023,-0.000042,y,342.54,yt,342.54,q,0.988360,-0.010000,0.004931,-0.151726,T,32.19

The time\_trigger is triggered by the 1msec clock tick, so a time based output can be created if necessary.

Note that if too much data is specified to print for the baud rate, the inertial system will drop whole lines of output. The algorithm will not let another trigger occur until the previous output has completed. Dropping whole lines avoids the problem of erroneous data due to dropped digits.

Here is a sample script which is similar to gyrop.p but with temperature added:

printmodulus 4 set drop // every 4th raw gyro

printmask           // variable that gets set in the next line

gyror\_trigger gyrop\_trigger or temp\_trigger or set drop

printtrigger printmask set drop

### Reset

The NorthTek “reset” command will invoke a reset of the processor. Allow a few seconds for the processor to come back up. All variables will be reset to their default value or read from non-volatile memory if the variable is marked “permanent”. Any NorthTek script which has been loaded will be lost unless the script has been loaded into EEPROM using the userOpen and userClose commands (see the User Boot Program section). Note that if an EEPROM update is in progress, then the reset will be delayed until the update completes. Also consider that the power up sequence takes time to reload configuration from EEPROM, so the reset command has a delay that is dependent on the amount of data in EEPROM.

### Restart

The NorthTek “restart” command can be used when a magnetic interference is removed. Without the restart, the navigation module may take a while to readjust to the new magnetic field because the gyro sensors are not detecting movement. With the “restart” command, the navigation module will immediately adjust to the changed magnetic environment.

After Version 3.0.0, a restart can also be accomplished by setting the variable named “restartCompassCalcs” to 1. This variable can be set using RFS and NMEA’s $PSRFS as well as NorthTek. This variable will automatically reset to 0 once the computations are reset.

A restart is much faster than a reset command which resets the processor and so takes longer to start producing data. No data cycles will be missed with a restart.

### Canceling NorthTek Programs

A <CTRL-C> will cancel a NorthTek program.

### Inhibit NorthTek Echoed Characters and “OK”/”Huh” Responses

NorthTek responses (echoed characters, OK, Huh) can be inhibited on the user port. The requested data will still be returned. Use “0 echo!” to turn off responses (they are on by default). Use “1 echo!” to turn them back on.

### Setting Database Elements

Integers and Ordinals can be set using the following syntax. Note that ordinals are set with an integer but display with the ordinal name.

***<variableName> <integerValue> set***

Example:

***orientation 2 set drop***

***orientation di.***

***orientation = Left Edge***

Floating point values can be set using the following syntax:

***<variableName> F<floatValue> set***

Example:

***magvar F5.9 set drop***

Arrays can be set using the following syntax:

***<variableName> array[ <startIndex> <endIndex> <sequence of numbers> ]array set***

Example:

***boresightMatrixX array[ 0 2 F1.0 F0.0 F0.0 ]array set drop***

### Saving the database

### The database gets automatically saved every ten minutes or every time a “set” command is sent to the compass. If the user wishes to specifically to save the database at a known state they may do so by sending the appropriate command: db.save. Once the save has been initiated it will take from 2-5 seconds before completion. Simply sending a <CR> the user may know if the system is done saving.

Normal operation will continue through the save since this occurs normally throughout the compass lifecycle. It should be noted that the system has a double-buffered database, so shutting of the system while a write is occurring will simply allow the compass to use the previously save database. To keep this from happening, before shutting down the system, send a carriage return. If the compass is amidst a database save, it will respond as follows at the rate of *gyroSampleRate*:

***EEPROM update in progress!***

### Calibration Data Control

The commands in this section can be set using RFS as well as NorthTek.

#### Restore In-field calibration data

The inertial system comes with a nominal field calibration setup. The user may perform an in-field calibration as described in the section titled “3D Compass Calibration” below or using one of the cal3D NorthTek scripts provided on the website. Performing the in-field calibration stores internal coefficients. If in-field calibration has been performed and the magnetic environment has changed, the in-field calibration may be in error. The in-field calibration can be re-performed or the in-field calibration data can be reset to the factory values. To reset the in-field calibration to the factory values:

***restoreFieldCal 1 set drop***

#### Restore Factory calibration data

Each inertial system is individually internally calibrated at the factory. Because of the unique ability to configure the inertial system, the user may choose to modify the internal calibration (for example magOffset). If at some time the user wishes to return the inertial system to the factory calibration, it can be restored with the following command:

***restorefactorycal 1 set drop***

#### Run the Gyro Offset Calibration

The GEDC-6E and AHRS-8 both use an advanced algorithm to continuously track the gyro offsets during normal use. Therefore, calibration of the gyro offsets should not be necessary. If offsets exist in the gyros when the inertial system is stationary, the user would encounter a slow, constant change in orientation (i.e. drifting) over time. The following steps describe how to successfully execute an in-field calibration of the gyros:

1. Insure that the inertial system is stationary. Orientation of the compass does not matter.
2. Invoke the Gyro Offset Calibration option using the command:

***InvokeGyroOffsetCal 1 set drop***

1. Once activated, keep the inertial system stationary for 30 seconds to insure that the offsets measurements are complete.
2. The gyro calibration will end automatically and return to normal gyro offset tracking after about 30 seconds. The new gyro offset will be saved to internal non-volatile memory.
3. After 30 seconds from the start of calibration, the inertial system can be used.

Note: Gyro offset calibration is not applicable to the Sparton DC-4E Digital Compass.

### 3D Compass In-Field Calibration

The script below performs a 3-D compass in-field calibration.

To perform calibration, use a terminal emulator to send the script below to the inertial system (if a “Huh??” is seen flying by, add a 5 ms delay to each line transmitted or reduce the baud to 38400). Once the script has been loaded, send the command “cal3D”.

The script will then prompt the user to capture calibration points. The user should move the compass around and select between 4 and 12 points that capture, as much as possible, the extremes of the X/Y/Z positions. Hit any key but ESC to capture each point and then hit ESC when the point capture is complete.

The script will then print the magnetic error at a 0.5 Hz rate. Observe the magnetic error (magErr) until it converges sufficiently then hit the spacebar or any other key. The compass will now be field calibrated.

The cal3D command may be re-entered as many times as desired without re-loading the macro until the power is shut off. The macro needs to be loaded only once while the inertial system remains powered up.

The results are stored in magFieldCalX, magFieldCalY and magFieldCalZ, magCrossAxisCorrectionX, magCrossAxisCorrectionY, magCrossAxisCorrectionZ.

**When using scripts this large you must set up the system as follows:**

* **Insert a delay per line of 5ms.**
* **Ensure each line “to be processed” (not comment lines) is less than 60 characters (bytes).**

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// NorthTek Script for 3d calibration.

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// This erases this script should it be reloaded.

forget cal3D

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// This is the actual calibration program.

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

( -- )

: cal3D

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Init the calibration process by setting calmode to 1.

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

calmode 1 set

250 delay // give compass time to process

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Give the user a heads up that we are starting.

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

." Calibration starting" cr

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Tell the calibration logic to start calibration

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

calCommand cal\_start set // cal start

250 delay // give compass time to process

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Give the user some instructions

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

." Press any key to take next point, ESC to finish" cr

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// The code now grabs a point, the user changes the

// compass position and repeats the cal3DState

// from 4-12 times total.

// sit in a loop, taking points until

// the user hits escape.

// User hits spacebar to take a point, ESC to quit.

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

begin

key 27 = 0= // until user enters ESC

while

// capture a point

calCommand cal\_capture set // take another point

250 delay // give compass time to capture

// Print out the current point number

calNumPoints di.

repeat

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Now the points are captured,

// Issue the command to start computing

// the real time cal values

//

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

calCommand cal\_end\_capture set // cal computation

250 delay // give compass time to process

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Some more user instructions

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

." Starting error settling" cr

." Press any key to terminate" cr

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// The user observes magErr to watch it settle

// at a minimum value (EKit can display every sec or so):

// This runs until the user has decided that the value

// has converged. See Software Interface Users Manual

// regarding the calibration process.

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

begin // keep printing magErr at .250 sec intervals

?key 0= // till user hits a keystroke

while

magErr di.

250 delay

repeat

key drop // read and remove the key used

// to stop the loop

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Let the user know that we are all done.

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

." Calibration done!" cr

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Send the cal\_end command.

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

calCommand cal\_end set // cal computation

250 delay // give compass time to process

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// End calibration mode

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

calmode 0 set // terminate calibration mode.

250 delay // give compass time to process

magFieldCalErr di.

250 delay

;

### 2D Compass In-Field Calibration

The 2-D calibration procedure is used for applications that stay within a plane containing two of the **navigation module’s** axes and rotating about the third axis. The point capture positions should be chosen to be distributed well about the circle of rotation.

The 2-D calibration differs from the 3-D in that it performs better if the World Magnetic Model has been run previously (see the set\_wmm command description in the appendix) by using it to obtain information on the axis of rotation. The World Magnetic Model information is not required though.

The 3-D calibration script above could be changed to a 2-D script simply by setting calmode to 2 instead of 1 (see the calmode command description in the appendix).

The results are stored in magFieldCalX, magFieldCalY and magFieldCalZ, magCrossAxisCorrectionX, magCrossAxisCorrectionY, magCrossAxisCorrectionZ.

**When using scripts this large you must set up the system as follows:**

* **Insert a delay per line of 5ms.**
* **Ensure each line “to be processed” (not comment lines) is less than 60 characters (bytes).**

### User Boot Program

Start-up configuration and output can be programmed by the user in the inertial system’s EEPROM. Upon start-up, the inertial system checks an EEPROM sector located at 0x10000 for a NorthTek script and will execute it if finds one. See Application Note AN1002 for a detailed example.

To load a script, enter:

***0x10000 userOpen // Opens the EEPROM section at 0x10000 for writing***

***: put start: userWrite ; // Defines a “put” function to write a subsequent string to EEPROM***

Now take the NorthTek script that is to be run on start-up and insert the string “put ” at the beginning of each line. Each “put” line will load the script line into EEPROM. **Make sure that the script works** before putting it into EEPROM or the inertial system could be rendered unusable! If the script fails on start-up, try <ctrl-c> to stop it. After entering the modified script, enter:

***userClose // completes the EEPROM “file”***

To try it out, cycle power or enter:

***reset***

To remove the script from EEPROM, enter:

***0x10000 eesectorerase***

Here is an example that will print the version number upon start-up:

***0x10000 userOpen***

***: put start: userWrite ;***

***put VERSION di.***

***userClose***

To output a NMEA command, precede it with “put “ and a quote followed by a space. Then append \n quote and “ count nmea!”. Here is a script which will repeat the $xxXDR command response:

***0x10000 userOpen // Opens the EEPROM section at 0x10000 for writing***

***: put start: userWrite ; // Defines a “put” function to write a subsequent string to EEPROM***

***put " $xxXDR,RPT=1.0\n" count nmea!***

***userClose // completes the EEPROM “file”***

### Accessing an RFS VID Using NorthTek – dvid@

An RFS VID (Variable ID) can be accessed using the NorthTek command, dvid@. The syntax is “<variableName> dvid@ .” (note there is a dot at the end to display the result).

Appendices

1. Variable Detailed Descriptions
   1. Variables by Name

In the tables below, “Permanent” indicates that the value is stored in non-volatile memory.

* + 1. vidCount

|  |  |
| --- | --- |
| Type: | Int32 |
| Detailed Description: | Describes how many RFS variables exist in the database. |
| Range: | 3..total number of variables. |
| Default: | n/a |
| Persistence: | Permanent |
| Access | Read Only |

* + 1. name

|  |  |
| --- | --- |
| Type: | String |
| Detailed Description: | Gives the name of the device such as “AHRS-8 (Attitude Heading and Reference System)” |
| Range: | n/a |
| Default: | n/a |
| Persistence: | Constant |
| Access: | Read Only |

* + 1. quiet

|  |  |
| --- | --- |
| Type: | Constant |
| Detailed Description: | Allows the user to suppress some debug output at runtime. |
| Range: | 0..1,  0=Off 1=On |
| Default: | 0 |
| Persistence: | Temporary |
| Access: | read/write |

* + 1. VERSION

|  |  |
| --- | --- |
| Type: | String |
| Detailed Description: | Contains the code version. Format is $ Version: x.x.x |
| Range: | n/a |
| Default: | n/a |
| Persistence: | Constant |
| Access: | Read Only |

* + 1. REVISION

|  |  |
| --- | --- |
| Type: | String |
| Detailed Description: | Contains the code’s configuration revision number. Format is $Revision: DDD $, where DDD is a decimal number. |
| Range: | n/a |
| Default: | n/a |
| Persistence: | Constant |
| Access: | Read Only |

* + 1. serialnumber

|  |  |
| --- | --- |
| Type: | String |
| Detailed Description: | Contains the factory set value for this unit’s serial number. |
| Range: | n/a |
| Default: | n/a |
| Persistence: | Constant |
| Access: | Read Only |

* + 1. orientation

|  |  |
| --- | --- |
| Type: | Ordinal |
| Detailed Description: | Physical orientation of compass relative to the host application. Heading, pitch and roll will be translated from that of the inertial system to that of the application according to the given setting. See the Sparton Digital Compass Product Guide for details |
| Range: | 0..4 0 = horizontal  1 = vertical  2 = left edge  3 = right edge  4 = inverted |
| Default: | 0 |
| Persistence: | Permanent |
| Access: | Read/Write |

* + 1. baud

|  |  |
| --- | --- |
| Type: | Ordinal |
| Detailed Description: | Selects the user port baud rate using an index. |
| Range: | 0..8  0 = 300  1 = 1200  2 = 2400  3 = 4800  4 = 9600  5 = 19200  6 = 38400  7 = 57600  8 = 115200 |
| Default: | 0 |
| Persistence: | Permanent |
| Access: | Read/Write |

* + 1. pitch

|  |  |
| --- | --- |
| Type: | Float32 |
| Detailed Description: | Gives the computed pitch angle, positive is nose up, units are degrees |
| Range: | -180.0 .. 180.0 |
| Default: | n/a |
| Persistence: | Temporary |
| Access: | Read Only |

* + 1. roll

|  |  |
| --- | --- |
| Type: | Float32 |
| Detailed Description: | Gives the computed roll angle, positive direction is right wing down, units are degrees |
| Range: | -180.0 .. 180.0 |
| Default: | n/a |
| Persistence: | Temporary |
| Access: | Read Only |

* + 1. yaw

|  |  |
| --- | --- |
| Type: | Float32 |
| Detailed Description: | Gives the computed magnetic heading angle, positive value starting at 0.0 = magnetic North, units are degrees |
| Range: | 0.00 .. 360.0 |
| Default: | n/a |
| Persistence: | Temporary |
| Access: | Read Only |

* + 1. yawt

|  |  |
| --- | --- |
| Type: | Float32 |
| Detailed Description: | Gives the computed true heading angle in degrees, positive value starting at 0.0 = true North. Only valid if a declination has been set manually or by execution of the world magnetic model with a valid position and time. |
| Range: | 0.00 .. 360.0 |
| Default: | n/a |
| Persistence: | Temporary |
| Access: | Read Only |

* + 1. quaternion

|  |  |
| --- | --- |
| Type: | ArrayFloat32 |
| Detailed Description: | 4 element quaternion array representing w, x, y, and z respectively. Quaternions represent the inertial system’s attitude similar to pitch/roll/heading but without the problem of Gimbal lock. Thus quaternions are recommended over pitch/roll/heading for any device that has full 3D range of rotation. Note that this quaternion is relative to magnetic North rather than true North. |
| Range: | n/a |
| Default: | n/a |
| Persistence: | Temporary |
| Access: | Read Only |

* + 1. lat

|  |  |
| --- | --- |
| Type: | Float32 |
| Detailed Description: | This variable needs to be set by the user prior to executing the world magnetic model. Value is latitude in degrees and fractions of a degree. |
| Range: | +/- 90 degrees. |
| Default: | 0.0 |
| Persistence: | Permanent |
| Access: | Read/Write |

* + 1. lonG

|  |  |
| --- | --- |
| Type: | Float32 |
| Detailed Description: | This variable needs to be set by the user prior to executing the world magnetic model. Value is longitude in degrees and fractions of a degree. |
| Range: | +/- 180.0 degrees |
| Default: | 0.0 |
| Persistence: | Permanent |
| Access: | Read/Write |

* + 1. alt

|  |  |
| --- | --- |
| Type: | Float32 |
| Detailed Description: | The altitude needs to be set by the user prior to executing the World Magnetic Model to compute magnetic variation. Units are in meters. |
| Range: | +- 10000 meters |
| Default: | 0.0 |
| Persistence: | Permanent |
| Access: | Read/Write |

* + 1. day

|  |  |
| --- | --- |
| Type: | Float32 |
| Detailed Description: | The date which needs to by set by the user prior to executing the World Magnetic Model. Units are in years. Precision is to 0.1 of a year. Accuracy beyond 0.1 year does not improve the World Magnetic Model estimate. |
| Range: | 2010.0 to 3000.0 |
| Default: | 2010.0 |
| Persistence: | Permanent |
| Access: | Read/Write |

* + 1. restorefactorycal

|  |  |
| --- | --- |
| Type: | Ordinal |
| Detailed Description: | The user can command the calibration data to be reset to the factory settings  by setting this variable to one (will automatically be reset to 0). |
| Range: | 0..1 |
| Default: | 0 |
| Persistence: | Temporary |
| Access: | Read/Write |

* + 1. set\_wmm

|  |  |
| --- | --- |
| Type: | Ordinal |
| Detailed Description: | Command to update the World Magnetic Model offset (magvar) in accordance with the current location in the world. Uses the variables lat, lonG, alt and day which must be previously provided by the user. Set this variable to 1 to invoke the World Magnetic Model. Reading this variable has no meaning. The result is stored in magvar and as seen in the difference between yawt and yaw. See the World Magnetic Model section for more information. |
| Range: | 0..1 (automatically resets to 0 when set to 1) |
| Default: | 0 |
| Persistence: | Temporary |
| Access: | Read/Write |

* + 1. magvar

|  |  |
| --- | --- |
| Type: | Float32 |
| Detailed Description: | Contains the difference between the magnetic heading and the true heading. This value can be computed indirectly using set\_wmm or can be directly written to the inertial system by the user. |
| Range: | +/- 360.0 degrees |
| Default: | 0.0 |
| Persistence: | Permanent |
| Access: | Read/Write |

* + 1. wmmGD

|  |  |
| --- | --- |
| Type: | ArrayFloat32 |
| Detailed Description: | World Magnetic Model geodetic magnetic field X Y Z at Lat/Long |
| Range: | N/A |
| Default: | <value>0.521215</value>  <value>0.0</value>  <value>0.853426</value>  <value>1.0 |
| Persistence: | Permanent |
| Access: | Read Only |

* + 1. wmmHasRun

|  |  |
| --- | --- |
| Type: | Ordinal |
| Detailed Description: | Indicates whether the World Magnetic Model (WMM) has been run (implies wmmGD and magvar are valid), a value of 1 means the WMM has been run. |
| Range: | 0..1 |
| Default: | 0 |
| Persistence: | Permanent |
| Access: | Read Only |

* + 1. magr

|  |  |
| --- | --- |
| Type: | ArrayFloat32 |
| Detailed Description: | Raw Magnetometer X Y Z array |
| Range: | n/a |
| Default: | n/a |
| Persistence: | Temporary |
| Access: | Read Only |

* + 1. magp

|  |  |
| --- | --- |
| Type: | ArrayFloat32 |
| Detailed Description: | Processed Magnetometer X Y Z array, this value takes magr and applies the factory calibration, the field calibration data and the cross axis correction (magCrossAxisCorrection) |
| Range: | n/a |
| Default: | n/a |
| Persistence: | Temporary |
| Access: | Read Only |
| Units | milli-gauss |

* + 1. accelr

|  |  |
| --- | --- |
| Type: | ArrayFloat32 |
| Detailed Description: | Raw Magnetometer X Y Z array, 2048=1g when accelrange=4g, 1024=1g when accelrange=8g |
| Range: | n/a |
| Default: | n/a |
| Persistence: | Temporary |
| Access: | Read Only |

* + 1. accelp

|  |  |
| --- | --- |
| Type: | ArrayFloat32 |
| Detailed Description: | Processed Accelerometer X Y Z array, this value takes accelr and applies the factory calibration and the cross axis correction (accelCrossAxisCorrection). Units are in milli-gs. |
| Range: | n/a |
| Default: | n/a |
| Persistence: | Temporary |
| Access: | Read Only |
| Units | milli-g |

* + 1. gyror

|  |  |
| --- | --- |
| Type: | ArrayFloat32 |
| Detailed Description: | Applies to GEDC-6E and AHRS-8 only. Raw gyro X Y Z array |
| Range: | n/a |
| Default: | n/a |
| Persistence: | Temporary |
| Access: | Read Only |

* + 1. gyrop

|  |  |
| --- | --- |
| Type: | ArrayFloat32 |
| Detailed Description: | Applies to GEDC-6E and AHRS-8 only. Processed gyro X Y Z array, this value takes gyror and applies the factory calibration. Units are radians/sample. To convert gyrop vector to millidegrees per second, get the sample rate using the command “gyroSampleRate di.” (note the dot at the end), then multiply each vector element with gyroSampleRate \* (180.0/PI)\*1000.0. |
| Range: | n/a |
| Default: | n/a |
| Persistence: | Temporary |
| Access: | Read Only |
| Units | Radians/sample |

* + 1. gyroSampleRate

|  |  |
| --- | --- |
| Type: | Float32 |
| Detailed Description: | Applies to GEDC-6E and AHRS-8 only. Actual sample rate for the gyro (samples/second) as measured once per second. |
| Range: | n/a |
| Default: | 100.0 |
| Persistence: | Temporary |
| Access: | Read Only |
| Units | samples/second |

* + 1. positionrate

|  |  |
| --- | --- |
| Type: | Int32 |
| Detailed Description: | This variable determines the interval of the automatic output of the RFS variable “position”. It may be 0 indicating that there is no automatic output, or up to 180000 msecs indicating 180 seconds (every 3 minutes). |
| Range: | 0..180000 (0 = off) (msecs) |
| Default: | 0 |
| Persistence: | Temporary |
| Access: | Read/Write |

* + 1. position

|  |  |
| --- | --- |
| Type: | Bit Field |
| Detailed Description: | RFS only. This is a user modifiable aggregate bit field (packed structure) descriptor. It may be redefined to contain up to 16 variables. The user may define which variables are carried in this bit field and the format is stored permanently (survives a power cycle).  Note that running with the NDS-1 or NDS-2 GUI may change these settings.  This variable can be output at a repeating rate using the positionrate variable. |
| Range: | n/a |
| Default: | pitch  roll  yawt  magErr  temperature  magp  accelp  gyrop  currentTime (Version 4.0.2 or later) |
| Persistence: | Permanent |
| Access: | Read/Write |

* + 1. composite2

|  |  |
| --- | --- |
| Type: | Bit Field |
| Detailed Description: | RFS only. A user defined composite variable (packed structure), similar to “position”. The user may define up to 16 fields in this variable and query it with RFS. |
| Range: | n/a |
| Default: | pitch  roll  yaw  lat  lonG  alt  temperature |
| Persistence: | Permanent |
| Access: | Read/Write |

* + 1. composite3

|  |  |
| --- | --- |
| Type: | Bit Field |
| Detailed Description: | RFS only. A user defined composite variable (packed structure), similar to “position”. The user may define up to 16 fields in this variable and query it with RFS. |
| Range: | n/a |
| Default: | pitch  roll  yaw  lat  lonG  alt  temperature |
| Persistence: | Permanent |
| Access: | Read/Write |

* + 1. calmode

|  |  |
| --- | --- |
| Type: | Ordinal |
| Detailed Description: | In-Field Calibration Mode – see the “In-Field Calibration” section for an overview and see the “Magnetic In-Field Calibration” for the NMEA protocol. Or see the “3D Compass In-Field Calibration” or “2D Compass In-Field Calibration” sections for the NorthTek protocol.  A NorthTek example is “***calmode 1 set drop***” and a NMEA command example is “$PSPA,CAL=3D”. Or use an RFS set command to set calmode to 1.  Use the numbers for NorthTek and RFS and use the all-caps names for NMEA.  The User mode is a legacy carry-over and has no functionality associated with it. |
| Range: | 0..3  0 = OFF  1 = 3D  2 = 2D  3 = USER |
| Default: | 0 |
| Persistence: | Temporary |
| Access: | Read/Write |

* + 1. calCommand

|  |  |
| --- | --- |
| Type: | Ordinal |
| Detailed Description: | Commands the next step in the “in field” calibration procedure after the calmode has been set to 2D or 3D.  See the “In-Field Calibration” section for an overview. For the NMEA protocol, see the “Magnetic In-Field Calibration”. For the NorthTek protocol, see the “3D Compass In-Field Calibration” or “2D Compass In-Field Calibration” sections. RFS can be used to follow the same sequence using the “Set” command.  The commands are ignored unless calmode is 2D or 3D.  The cal\_reset command sets the magnetic field calibration values back to the default settings of (0.0, 1.0).  The lower case names are used by NorthTek and the upper case names are used by NMEA. The numeric values are used by RFS and can be used in NorthTek. |
| Range: | 0..4  0 = cal\_start, START\_CAL  1 = cal\_capture, CAPTURE  2 = cal\_end\_capture,END\_CAPTURE  3 = cal\_end,END\_CAL  4 = cal\_reset,CAL\_RESET |
| Default: | n/a |
| Persistence: | Temporary |
| Access: | Read/Write |

* + 1. calNumPoints

|  |  |
| --- | --- |
| Type: | Int32 |
| Detailed Description: | Number of points collected for the magnetic calibration procedure - see the Magnetic In-Field Calibration section in which it is reported by the $PSPA,CAL\_CMD=CAPTURE,POINTS= response, but can also be read directly via RFS or NorthTek, see the NorthTek section for the display command (di.), see also the 3D Compass Calibration subsection in the NorthTek section for a script that uses these variables. |
| Range: | 0..12 |
| Default: | 0 |
| Persistence: | Temporary |
| Access: | Read/Write |

* + 1. magErr

|  |  |
| --- | --- |
| Type: | Float32 |
| Detailed Description: | Magnetic error term resulting from the magnetic calibration procedure – this is not the actual error but is a quality factor that is a function of the points collected in the “in field” calibration procedure, see the Magnetic In-Field Calibration section in which it is reported by the $PSPA,MAGERR,RPT=0.5 command, but can also be viewed directly using RFS or NorthTek, see the NorthTek section for the display command (di.), see also the 3D Compass Calibration subsection in the NorthTek section for a script that uses these variables |
| Range: | N/A |
| Default: | 0 |
| Persistence: | Temporary |
| Access: | Read Only |

* + 1. wlim

|  |  |
| --- | --- |
| Type: | Float32 |
| Detailed Description: | Algorithm constant – Controls the reliance on gyros that is motion dependent. Rotations that exceed wlim will tend to favor the gyro estimates. |
| Range: | N/A |
| Default: | N/A |
| Persistence: | Permanent |
| Access: | Read/Write |

* + 1. alim

|  |  |
| --- | --- |
| Type: | Float32 |
| Detailed Description: | Algorithm constant – Controls estimator sensitivity to transient acceleration anomalies |
| Range: | N/A |
| Default: | N/A |
| Persistence: | Permanent |
| Access: | Read/Write |

* + 1. mlim

|  |  |
| --- | --- |
| Type: | Float32 |
| Detailed Description: | Algorithm constant – Controls estimator sensitivity to transient magnetic anomalies |
| Range: | N/A |
| Default: | N/A |
| Persistence: | Permanent |
| Access: | Read/Write |

* + 1. km0

|  |  |
| --- | --- |
| Type: | Float32 |
| Detailed Description: | Algorithm constant - magnetic gain. This variable sets an overall reliance on the magnetometers versus the gyros that are independent of motion. Lower values will force the compass to rely more heavily on the gyro estimates rather than the absolute magnetometer values. CAUTION: Setting this value too low may cause the heading orientation to drift due to accumulated errors in the estimation process. |
| Range: | N/A |
| Default: | N/A |
| Persistence: | Permanent |
| Access: | Read/Write |

* + 1. ka0

|  |  |
| --- | --- |
| Type: | Float32 |
| Detailed Description: | Algorithm constant - acceleration gain. This variable sets an overall reliance on the accelerometers versus the gyros that are independent of motion. Lower values will force the compass to rely more heavily on the gyro estimates rather than the absolute accelerometer values. CAUTION: Setting this value too low may cause the heading orientation to drift due to accumulated errors in the estimation process. |
| Range: | N/A |
| Default: | N/A |
| Persistence: | Permanent |
| Access: | Read/Write |

* + 1. emlim

|  |  |
| --- | --- |
| Type: | Float32 |
| Detailed Description: | Algorithm constant – Controls the convergence between actual magnetometer sensor orientation and the internal estimator during periods of excessive magnetic disturbances. |
| Range: | N/A |
| Default: | N/A |
| Persistence: | Permanent |
| Access: | Read/Write |

* + 1. ealim

|  |  |
| --- | --- |
| Type: | Float32 |
| Detailed Description: | Algorithm constant – Controls the convergence between actual accelerometer sensor orientation and the internal estimator during periods of excessive acceleration disturbances. |
| Range: | N/A |
| Default: | N/A |
| Persistence: | Permanent |
| Access: | Read/Write |

* + 1. kgyrooffset

|  |  |
| --- | --- |
| Type: | Float32 |
| Detailed Description: | Applies to GEDC-6E and AHRS-8 only. Algorithm constant – Controls the sensitivity of the gyro offset tracking. |
| Range: | 0 .. 1000.0 |
| Default: | 0.1 |
| Persistence: | Permanent |
| Access: | Read/Write |

* + 1. magOffset

|  |  |
| --- | --- |
| Type: | ArrayFloat32 |
| Detailed Description: | Calibration offset for the magnetometer – is subtracted from magr as part of computing magp, this is a computed value based on the magOffsetTC variables |
| Range: | n/a |
| Default: | n/a |
| Persistence: | Temporary |
| Access: | Read Only |

* + 1. magCalTemperature

|  |  |
| --- | --- |
| Type: | Float32 |
| Detailed Description: | Base temperature for the magnetometer temperature compensations. |
| Range: | -45 .. 90 |
| Default: | n/a |
| Persistence: | Permanent |
| Access: | Read/Write |

* + 1. magOffsetTCX, magOffsetTCY, magOffsetTCZ

|  |  |
| --- | --- |
| Type: | ArrayFloat32 |
| Detailed Description: | Factory temperature compensated calibration offsets for the magnetometer - is used to compute the offset compensation based on current temperature (magOffset). |
| Range: | n/a |
| Default: | n/a |
| Persistence: | Permanent |
| Access: | Read/Write |
|  |  |

* + 1. magLinearityXTC0 ,magLinearityXTC1, magLinearityXTC2, magLinearityYTC0, magLinearityYTC1, magLinearityYTC2, magLinearityZTC0, magLinearityZTC1, magLinearityZTC2

|  |  |
| --- | --- |
| Type: | ArrayFloat32 |
| Detailed Description: | Factory temperature compensated calibration scalars for the magnetometer - is used to compute the linearity compensation factors based on current temperature (magLinearityX, magLinearityY, magLinearityZ). |
| Range: | n/a |
| Default: | n/a |
| Persistence: | Permanent |
| Access: | Read/Write |

* + 1. magLinearityX ,magLinearityY, magLinearityZ

|  |  |
| --- | --- |
| Type: | ArrayFloat32 |
| Detailed Description: | Computed calibration scalars for the magnetometer - is used to “linearize” the magnetic vector after the offset has been applied. Computation is based on the current temperature and the factory temperature compensated calibration scalars for each axis (e.g. magLinearityXTC0). In each array, the first component is the scalar that multiplies the X (or Y or Z), then the second multiplies the X (or Y or Z) squared and the third multiplies the X (or Y or Z) cubed and then they are summed to yield the processed mag X (or Y or Z). |
| Range: | n/a |
| Default: | n/a |
| Persistence: | Temporary |
| Access: | Read Only |

* + 1. magCrossAxisCorrectionX, magCrossAxisCorrectionY, magCrossAxisCorrectionZ

|  |  |
| --- | --- |
| Type: | ArrayFloat32 |
| Detailed Description: | Factory calibration cross axis correction – these are the 3 rows of a matrix that is multiplied (dot product) by the calibrated magnetometer vector to get the final (magp) vector |
| Range: | n/a |
| Default: | n/a |
| Persistence: | Permanent |
| Access: | Read Only |

* + 1. magFieldCalX, magFieldCalY, magFieldCalZ

|  |  |
| --- | --- |
| Type: | ArrayFloat32 |
| Detailed Description: | In field calibration offset and scalars. This is computed by the 2D or 3D calibration procedure (see In-Field Calibration section). The first component is subtracted from the factory calibrated mag X/Y/Z and the second component is a scaler. |
| Range: | n/a |
| Default: | n/a |
| Persistence: | Permanent |
| Access: | Read/Write |

* + 1. accelrange

|  |  |
| --- | --- |
| Type: | Ordinal |
| Detailed Description: | Accelerometer full scale range setting. The higher range is a trade-off with less resolution. A reset or power cycle must be performed for this change to take effect in the accelerometer. |
| Range: | 0..1  0 = 4g (default)  1 = 8g |
| Default: | 0 |
| Persistence: | Permanent |
| Access: | Read/Write |

* + 1. accelOffset

|  |  |
| --- | --- |
| Type: | ArrayFloat32 |
| Detailed Description: | Calibration offset for the accelerometer - is subtracted from accelr as part of computing accelp, this is computed based on the accelOffsetTC variables. |
| Range: | n/a |
| Default: | n/a |
| Persistence: | Temporary |
| Access: | Read Only |

* + 1. accelGyroCalTemperature

|  |  |
| --- | --- |
| Type: | ArrayFloat32 |
| Detailed Description: | Temperature (deg C) at which the Rate Table temperature compensation parameters for both accelerometer and gyro are computed. |
| Range: | -45 .. 90 |
| Default: | n/a |
| Persistence: | Permanent |
| Access: | Read/Write |

* + 1. accelOffsetTCX, accelOffsetTCY, accelOffsetTCZ

|  |  |
| --- | --- |
| Type: | ArrayFloat32 |
| Detailed Description: | Temperature compensated calibration offsets for the accelerometer which are used to compute the offset compensation based on current temperature (accelOffset). |
| Range: | n/a |
| Default: | n/a |
| Persistence: | Permanent |
| Access: | Read/Write |

* + 1. accelLinearityXTC0, accelLinearityXTC1 , accelLinearityXTC2, accelLinearityYTC0, accelLinearityYTC1, accelLinearityYTC2, accelLinearityZTC0, accelLinearityZTC1, accelLinearityZTC2

|  |  |
| --- | --- |
| Type: | ArrayFloat32 |
| Detailed Description: | Factory temperature compensated calibration scalars for the accelerometer -used to compute the linearity compensation factors based on current temperature (accelLinearityX, accelLinearityY, accelLinearityZ). |
| Range: | n/a |
| Default: | n/a |
| Persistence: | Permanent |
| Access: | Read/Write |

* + 1. accelLinearityX ,accelLinearityY, accelLinearityZ

|  |  |
| --- | --- |
| Type: | ArrayFloat32 |
| Detailed Description: | Computed calibration scalars for the accelerometer - used to “linearize” the accelerometer vector after the offset has been applied. Computation is based on the current temperature and the factory temperature compensated calibration scalars for each axis (e.g. accelLinearityXTC0). In each array, the first component is the scalar that multiplies the X (or Y or Z), then the second multiplies the X (or Y or Z) squared and the third multiplies the X (or Y or Z) cubed and then they are summed to yield the processed accel X (or Y or Z). |
| Range: | n/a |
| Default: | n/a |
| Persistence: | Temporary |
| Access: | Read Only |

* + 1. accelCrossAxisCorrectionX, accelCrossAxisCorrectionY , accelCrossAxisCorrectionZ

|  |  |
| --- | --- |
| Type: | ArrayFloat32 |
| Detailed Description: | Factory calibration cross axis correction – these are the 3 rows of a matrix that is multiplied (dot product) by the calibrated accelerometer vector to get the final (accelp) vector |
| Range: | n/a |
| Default: | n/a |
| Persistence: | Permanent |
| Access: | Read Only |

* + 1. gyroOffset

|  |  |
| --- | --- |
| Type: | ArrayFloat32 |
| Detailed Description: | Applies to GEDC-6E and AHRS-8 only. Calibration offset for the gyro is subtracted from gyror as part of computing gyrop. For AHRS-8, this value is computed from the accelOffsetTC variables. |
| Range: | n/a |
| Default: | n/a |
| Persistence: | Temporary |
| Access: | Read Only |

* + 1. gyroMaxRate

|  |  |
| --- | --- |
| Type: | Float32 |
| Detailed Description: | Applies to GEDC-6E and AHRS-8 only. The gyro full scale range is fixed at 500 degrees per second. This variable defines the point at which the inertial system switches to using just the magnetometer and accelerometer and ignores the gyro value. |
| Range: | 0 .. 50000.0 |
| Default: | 450 |
| Persistence: | Permanent |
| Access: | Read/Write |

* + 1. gyroMaxReturnCnt

|  |  |
| --- | --- |
| Type: | Int32 |
| Detailed Description: | Applies to GEDC-6E and AHRS-8 only. After the gyro has exceeded the gyroMaxRate, this variable defines the number of in-range gyro values that must occur before switching back to using the gyro data. |
| Range: | 0 .. 10,000 |
| Default: | 25 |
| Persistence: | Permanent |
| Access: | Read/Write |

* + 1. gyroOffsetTCX, gyroOffsetTCY, gyroOffsetTCZ

|  |  |
| --- | --- |
| Type: | ArrayFloat32 |
| Detailed Description: | Applies to GEDC-6E and AHRS-8 only. Factory temperature compensated calibration offsets for the gyro - is used to compute the offset compensation based on current temperature (gyroOffset). |
| Range: | n/a |
| Default: | n/a |
| Persistence: | Permanent |
| Access: | Read/Write |

* + 1. gyroLinearityXTC0, gyroLinearityXTC1 , gyroLinearityXTC2, gyroLinearityYTC0 , gyroLinearityYTC1, gyroLinearityYTC2, gyroLinearityZTC0, gyroLinearityZTC1, gyroLinearityZTC2

|  |  |
| --- | --- |
| Type: | ArrayFloat32 |
| Detailed Description: | Applies to AHRS-8 only. Factory temperature compensated calibration scalars for the gyro -used to compute the linearity compensation factors based on current temperature (gyroLinearityX, gyroLinearityY, gyroLinearityZ). |
| Range: | n/a |
| Default: | n/a |
| Persistence: | Permanent |
| Access: | Read/Write |

* + 1. gyroLinearityX, gyroLinearityY, gyroLinearityZ

|  |  |
| --- | --- |
| Type: | ArrayFloat32 |
| Detailed Description: | Applies to GEDC-6E and AHRS-8 only. Temperature compensated linearity coefficients for the gyro, these are computed from the gyroLinearityXTC0, etc. variables and temperature. |
| Range: | n/a |
| Default: | n/a |
| Persistence: | Temporary |
| Access: | Read Only |

* + 1. gyroFieldCalX, gyroFieldCalY , gyroFieldCalZ

|  |  |
| --- | --- |
| Type: | ArrayFloat32 |
| Detailed Description: | Applies to GEDC-6E and AHRS-8 only. In field calibration offset and scalars. This is computed by the Gyro calibration command (set InvokeGyroOffsetCal to 1). The first component is subtracted from the factory calibrated gyro X/Y/Z and the second component divides into the difference. |
| Range: | n/a |
| Default: | n/a |
| Persistence: | Permanent |
| Access: | Read Only |

* + 1. gyroCrossAxisCorrectionX, gyroCrossAxisCorrectionY, gyroCrossAxisCorrectionZ

|  |  |
| --- | --- |
| Type: | ArrayFloat32 |
| Detailed Description: | Applies to GEDC-6E and AHRS-8 only. Calibration cross axis correction for the gyro |
| Range: | n/a |
| Default: | n/a |
| Persistence: | Permanent |
| Access: | Read/Write |

* + 1. InvokeGyroOffsetCal

|  |  |
| --- | --- |
| Type: | Ordinal |
| Detailed Description: | Applies to GEDC-6E and AHRS-8 only. Command the gyro offset calibration to start by setting this to 1. The inertial system will set this to zero upon initiation of the process. This variable will not remainat 1 or’ON’ during the process |
| Range: | 0..1 |
| Default: | 0 |
| Persistence: | Temporary |
| Access: | Read/Write |

* + 1. boresightMatrixX, boresightMatrixY , boresightMatrixZ

|  |  |
| --- | --- |
| Type: | ArrayFloat32 |
| Detailed Description: | Boresight orientation – these arrays make up the rows of the boresight matrix, this rotation matrix is used to transform from the inertial system reference frame to the host reference frame. This affects yaw, yawT, pitch, roll, quat, pMatrix |
| Range: | N/A |
| Default: | Identity matrix |
| Persistence: | Permanent |
| Access: | Read/Write |

* + 1. lowpower

|  |  |
| --- | --- |
| Type: | Ordinal |
| Detailed Description: | Setting this variable to 1 makes the inertial system enter low power mode. All functions stop and the inertial system enters the lowest power state. The inertial system is restarted with any serial port activity. The reset is identical to a power up reset. |
| Range: | 0..1 |
| Default: | 0 |
| Persistence: | Temporary |
| Access: | Read/Write |

* + 1. temperature

|  |  |
| --- | --- |
| Type: | Float32 |
| Detailed Description: | Internal temperature in degrees Celsius |
| Range: | -1000 .. 1000 |
| Default: | 0 |
| Persistence: | Temporary |
| Access: | Read Only |

* + 1. kgyroOffsetCalFactor

|  |  |
| --- | --- |
| Type: | Float32 |
| Detailed Description: | Applies to GEDC-6E and AHRS-8 only. Gyro offset field cal factor |
| Range: | N/A |
| Default: | 0.996 |
| Persistence: | Permanent |
| Access: | Read/Write |

* + 1. nmeaecho

|  |  |
| --- | --- |
| Type: | Ordinal |
| Detailed Description: | Setting this variable to 1 makes the NMEA protocol parser echo input characters and properly handle backspace characters. For human interactive use, set this variable to 1. |
| Range: | 0..1 |
| Default: | 0 |
| Persistence: | Temporary |
| Access: | Read/Write |

* + 1. nmeaignorechecksum

|  |  |
| --- | --- |
| Type: | Ordinal |
| Detailed Description: | Setting this variable to 1 causes the NMEA input parser to ignore incoming checksums. For human interactive use, set this variable to 1(default). |
| Range: | 0..1 |
| Default: | 1 |
| Persistence: | Temporary |
| Access: | Read/Write |

* + 1. restoreFieldCal

|  |  |
| --- | --- |
| Type: | Ordinal |
| Detailed Description: | Command to set magnetic in-field calibration data back to factory default |
| Range: | N/A |
| Default: | 0 |
| Persistence: | Temporary |
| Access: | Read/Write |

* + 1. pMatrix0, pMatrix1, pMatrix2

|  |  |
| --- | --- |
| Type: | ArrayFloat32 |
| Detailed Description: | Platform matrix (rotation matrix) – provides 3D orientation, each array is a row in the platform matrix, the transform from the inertial system reference to the host reference is included as defined by the boresightMatrix. |
| Range: | N/A |
| Default: | 0 |
| Persistence: | Temporary |
| Access: | Read/Write |

* + 1. printmask

|  |  |
| --- | --- |
| Type: | Int32 |
| Detailed Description: | Setting this variable selects items which are to be printed out for the ASCII formatted print statement. Section 3.4.3.7 |
| Range: | N/A |
| Default: | 0 |
| Persistence: | Temporary |
| Access: | Read/Write |

* + 1. printtrigger

|  |  |
| --- | --- |
| Type: | Int32 |
| Detailed Description: | Used to select triggers (data updates) that will cause the printmask selected items to print. |
| Range: | N/A |
| Default: | 0 |
| Persistence: | Temporary |
| Access: | Read/Write |

* + 1. printmodulus

|  |  |
| --- | --- |
| Type: | Int32 |
| Detailed Description: | Setting this variable defines when to print out based on this many triggers as defined by printtrigger. |
| Range: | 0..10000 |
| Default: | 4 |
| Persistence: | Temporary |
| Access: | Read/Write |

* + 1. centripetalCorrection

|  |  |
| --- | --- |
| Type: | Ordinal |
| Detailed Description: | (GEDC-6E and AHRS-8 only) Enable Centripetal acceleration correction of acceleration |
| Range: | 0 = Off  1 = On |
| Default: | On |
| Persistence: | Permanent |
| Access: | Read/Write |

* + 1. calPointsNumToAvg

|  |  |
| --- | --- |
| Type: | Int32 |
| Detailed Description: | Number of magnetic samples to be averaged for a 3D/2D Calibration Mode capture point. Note that the DC-4E collects samples at 10 Hz and GEDC-6E/AHRS-8 collect samples at gyroSampleRate***. The device is required to be static during each point capture sampling period.*** |
| Range: | 1..6000 |
| Default: | 4 (DC-4E)  10 (GEDC-6E and AHRS-8) |
| Persistence: | Permanent |
| Access: | Read/Write |

* + 1. InvokeTare

|  |  |
| --- | --- |
| Type: | Ordinal |
| Detailed Description: | Command to update the boresight matrix based on current orientation. The tare is complete once InvokeTare goes back to 0. The host device must be pointed to North in an accurate and level condition. |
| Range: | 0 = Off  1 = On |
| Default: | Off |
| Persistence: | Temporary |
| Access: | Read/Write |

* + 1. magErr2

|  |  |
| --- | --- |
| Type: | Float32 |
| Detailed Description: | Magnetic heading error estimate in degrees based on the current (real-time) magnetic environment relative to the magnetic environment during the last in-field calibration. This variable will give feedback showing an estimated heading accuracy based upon the real-time magnetic environment and how it relates to the magnetic environment during the last in-field calibration. The firmware computes a real-time error estimate to detect changes in the magnetic environment indicating a need for in-field calibration. Deviations in magnetic field strength are an indication of potential heading errors.  Versions 3.0.0 or later. |
| Range: | 0 - 360 |
| Default: | ~45.0 |
| Persistence: | Temporary |
| Access: | Read |

* + 1. magFieldCalErr

|  |  |
| --- | --- |
| Type: | Float32 |
| Detailed Description: | Magnetic field calibration heading error estimate, in degrees, as a result of the last in-field calibration (smaller is better). The magnetic data collected during field calibration will be analyzed for variability in field strength.  This variability can be related directly to heading error providing feedback as to the quality of calibration. This variable gives an estimate of the error in degrees and is computed once at the end of the in-field calibration process.  Versions 3.0.0 or later. |
| Range: | 0 – 360 |
| Default: | 999.0 |
| Persistence: | Permanent |
| Access: | Read/Write |

* + 1. magFieldCalTot

|  |  |
| --- | --- |
| Type: | Float32 |
| Detailed Description: | Average of magnetic field calibration point vector magnitudes.  Versions 3.0.0 or later. |
| Range: | -3.0E38 .. 3.0E38 |
| Default: | 99999.0 |
| Persistence: | Permanent |
| Access: | Read/Write |

* + 1. magHorzFieldCalTot

|  |  |
| --- | --- |
| Type: | Float32 |
| Detailed Description: | Average of the horizontal component of the magnetic field calibration points.  Versions 3.0.0 or later. |
| Range: | -3.0E38 .. 3.0E38 |
| Default: | 99999.0 |
| Persistence: | Permanent |
| Access: | Read/Write |

* + 1. gyroSigma

|  |  |
| --- | --- |
| Type: | Float32 |
| Detailed Description: | Applies to GEDC-6E and AHRS-8 only. Gyro stability factor. This feature will zero the processed gyro data for a unit that is not moving. Set this variable to 0.0 if you wish to turn off this feature.  Versions 3.0.0 or later. |
| Range: | -3.0E38 .. 3.0E38 |
| Default: | 8.9E-5 |
| Persistence: | Permanent |
| Access: | Read/Write |

* + 1. infieldCalMu

|  |  |
| --- | --- |
| Type: | Float32 |
| Detailed Description: | In-field calibration offset convergence factor.  Versions 3.0.0 or later. |
| Range: | 0..100 |
| Default: | 2.0E-2 |
| Persistence: | Permanent |
| Access: | Read/Write |

* + 1. infieldCalMu2

|  |  |
| --- | --- |
| Type: | Float32 |
| Detailed Description: | In-field calibration scaling convergence factor.  Versions 3.0.0 or later. |
| Range: | 0..100 |
| Default: | 2.0E-7 |
| Persistence: | Permanent |
| Access: | Read/Write |

* + 1. infieldCalMu3

|  |  |
| --- | --- |
| Type: | Float32 |
| Detailed Description: | In-field calibration cross-axis convergence factor.  Versions 4.2.10 or later. |
| Range: | 0..100 |
| Default: | 2.0E-8 |
| Persistence: | Permanent |
| Access: | Read/Write |

* + 1. restartCompassCalcs

|  |  |
| --- | --- |
| Type: | Ordinal |
| Detailed Description: | Used to resync the magnetic based and gyro based heading/attitude computations.  Versions 3.0.0 or later. |
| Range: | 0 = Off  1 = On |
| Default: | 0 |
| Persistence: | Temporary |
| Access: | Read/Write |

* + 1. magPointQF

|  |  |
| --- | --- |
| Type: | Float32 |
| Detailed Description: | Quality factor for the field cal captured magnetic calibration points (higher is better). As each calibration point is selected, magPointQF that indicates how well the magnetic values in the points collected cover the expected possible range for each axis. By default, the expected possible range is 2 times 450 mGauss (world average). If the Auto Variance (magvar) is computed using the World Magnetic Model, then the expected magnetic field strength computed for the given location will be used which should yield a more consistent indicator for different world locations (assuming no external magnetic interference).  Versions 3.0.0 or later. |
| Range: | 0..50000 |
| Default: | 0 |
| Persistence: | Permanent |
| Access: | Read |

* + 1. possibleMagPointQF

|  |  |
| --- | --- |
| Type: | Float32 |
| Detailed Description: | Possible quality factor if the current magnetic point is added to the captured calibration points during an in-field calibration. By using possibleMagPointQF during an in-field calibration, the user can find calibration point positions that increase the quality factor.  Versions 3.0.0 or later. |
| Range: | 0..50000 |
| Default: | 0 |
| Persistence: | Temporary |
| Access: | Read |

* + 1. wmmGDMagnitude

|  |  |
| --- | --- |
| Type: | Float32 |
| Detailed Description: | Magnitude of the World Magnetic Model computed Earth Field (used for magPointQF), defaults to 450 milliGauss.  Versions 3.0.0 or later. |
| Range: | 0.1 .. 50000 |
| Default: | 450 |
| Persistence: | Permanent |
| Access: | Read |

* + 1. calPointDistribution

|  |  |
| --- | --- |
| Type: | Float32 |
| Detailed Description: | Factor indicating magnetic distribution of in-field calibration points (exactly 12 points must be collected, larger is better). The collected magnetic data is analyzed to estimate the level coverage of the magnetic sphere.  Using magnetic samples for calibration that are grouped too closely together will result in a poor overall calibration.  Insuring that the samples are sufficiently spaced will dramatically improve accuracy and increase speed of convergence in the adaptive calibration algorithm.  Versions 3.0.0 or later. |
| Range: | 0 .. 1000 |
| Default: | 0 |
| Persistence: | Permanent |
| Access: | Read |

* + 1. calPointDistSigma

|  |  |
| --- | --- |
| Type: | Float32 |
| Detailed Description: | Used in computing the calibration point distribution indicator (calPointDistribution).  Versions 3.0.0 or later. |
| Range: | -3.0E38 .. 3.0E38 |
| Default: | 30 |
| Persistence: | Permanent |
| Access: | Read/Write |

* + 1. timeStamp

|  |  |
| --- | --- |
| Type: | Int32 |
| Detailed Description: | This variable gives the processor time (milliseconds from power-up) at the beginning of the pitch/roll/yaw computations. So it is basically a time stamp for the raw sensor data going into the P/R/Y computations.  Version 4.0.2 or later. |
| Range: | 0..0x7FFFFFFF (msecs) |
| Default: | 0 |
| Persistence: | Temporary |
| Access: | Read |

* + 1. currentTime

|  |  |
| --- | --- |
| Type: | Int32 |
| Detailed Description: | This variable gives the processor time (milliseconds from power-up) at the time of the variable request.  Version 4.0.2 or later. |
| Range: | 0..0x7FFFFFFF (msecs) |
| Default: | 0 |
| Persistence: | Temporary |
| Access: | Read |

* 1. calStatus

|  |  |
| --- | --- |
| Type: | Int32 |
| Detailed Description: | This variable is stored in non-volatile memory and gives the status of the in-field calibration result data in magFieldCalX, magFieldCalY, magFieldCalZ, magCrossAxisCorrectionX, magCrossAxisCorrectionY and magCrossAxisCorrectionZ.  0 = DEFAULT, 1 = INPROGRESS, 2 = COMPLETED, 3 = ABORTED  Version 4.2.10 or later. |
| Range: | 0..3 (msecs) |
| Default: | 0 |
| Persistence: | Permanent |
| Access: | Read/Write |

1. Troubleshooting
   1. NMEA commands are not echoed.

Choose one of the following options:

* turn on echoes using the command “nmeaecho 1 set drop”
* turn on local echo on your terminal emulator. For example, using HyperTerminal: File->Properties->Settings tab->ASCII Setup->”Echo typed characters locally”. Note that the disadvantage of this option is that you have to turn it back off for NorthTek commands.
  1. NMEA commands don’t respond to “enter” key.
* set your terminal emulator to send <cr><lf> for the “enter” key. For example, using HyperTerminal: File->Properties->Settings tab->ASCII Setup->”Send line ends with line feeds”
  1. Repeating output is not appearing.
* type <ctrl-Q> since you may have previously typed a <ctrl-S>
  1. NMEA commands are taking a long time to respond
* Send the <lf> by itself instead of the <cr><lf>
  1. Repeating output is making typing a new command difficult.
* enter <ctrl-S> to pause the repeat display and type in your new command, restart any repeat displays with <ctrl-Q>
  1. The compass is providing an inaccurate heading:
* check for magnetic interference in the area, electric motors, chairs, door, cables, cell phone
* in-field calibration may need to be reset (restoreFieldCal 1 set drop) or rerun
* gyro calibration may need to be rerun (InvokeGyroOffsetCal 1 set drop)
  + this could also manifest as a spinning pitch/roll/yaw
* if you have modified the calibration data, use the restorefactorycal command (restorefactorycal 1 set drop)
  1. The compass does not output any response.

Check the following (reference the Data Sheet for your product):

* that there is a TTL logic level connection to compass pins USER\_RXD (P2-2) and USER\_TXD (P2-3)
* that the baud rate matches the baud rate setting on the compass (initial factory setting is 115200), note that the NDS-1 GUI has an option to try all baud rates if the baud rate is unknown
* that the other serial interface properties are 8 bit, no parity, 1 stop bit, no flow control
* compass is seated properly (we recommend that it be keyed by blocking the socket for P1-4)
* compass has power on P2-1 (+4V – +10V DC) and ground on P1-7, P2-6, P2-7
* Factory use pins (P2-4 and P2-5) are high

1. These are NorthTek variable names, hence the odd capitalization. [↑](#footnote-ref-2)
2. VID 0 is always the count of the number of VIDs in the device. VID 1 is always the name of the device. Vid 2 is always the major version number. Vid 3 is always the minor version number. [↑](#footnote-ref-3)
3. GEDC-6E and AHRS-8 Only [↑](#footnote-ref-4)