DCP Manual 8500-0013 Revision 009

3DM-GX3®-45 Data Communications Protocol

Firmware Version 1.5.38 and higher

MicroStrain® Little Sensors, Big Ideas®

www.microstrain.com

©2013 by LORD MicroStrain
459 Hurricane Lane
Williston, VT 05495
United States of America
Phone: 802-862-6629
Fax: 802-863-4093
www.microstrain.com
support@microstrain.com

REVISED: Sept 12, 2013

Table of Contents

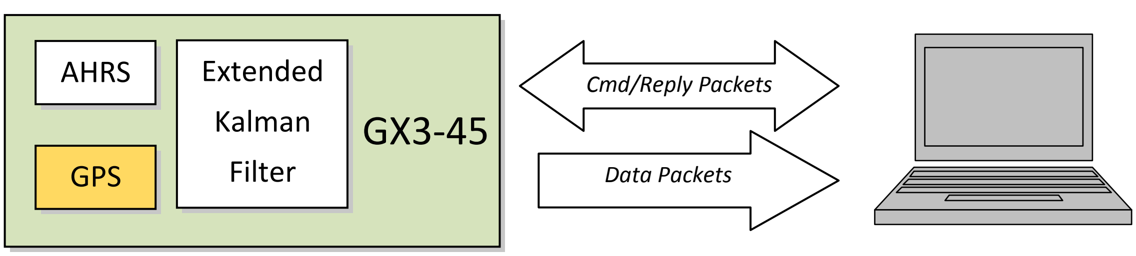
3DM-GX3-45 API

API Introduction

The 3DM-GX3-45 programming interface is comprised of a compact set of setup and control commands and a very flexible user-configurable data output format. The commands and data are divided into 4 command sets and 3 data sets corresponding to the internal architecture of the device. The four command sets consist of a set of "Base" commands (a set that is common across many types of devices), a set of unified "3DM" (3D Motion) commands that are specific to the MicroStrain inertial product line, a set of "NAV" (Navigation) commands that are specific to MicroStrain navigation devices, and a set of "System" commands that are specific to sensor systems comprised of more than one internal sensor block. The three data sets represent the three types of data that the 3DM-GX3-45 is capable of producing: "AHRS" (Attitude and Heading Reference System) data, "GPS" (Global Positioning Sensor) data, and "NAV" (Navigation) data.

Base commands	*Ping, Idle, Resume, Get ID Strings, etc.*
3DM commands	*Poll AHRS Data, Poll GPS Data, etc.*
NAV commands	*Reset Filter, Sensor to Vehicle Frame Transformation, etc.*
System commands	*Switch Communications Mode, etc.*
AHRS data	*Acceleration Vector, Gyro Vector, Euler Angles, etc.*
GPS data	*Latitude, Longitude, UTC, Satellites in view, etc.*
NAV data	*Position, Velocity, Attitude Estimates, etc.*

The protocol is packet based. All commands, replies, and data are sent and received as fields in a message packet. The packets have a descriptor type field based on their contents, so it is easy to identify if a packet contains commands, replies, AHRS data, GPS data, or NAV data.



The 3DM-GX3-45 has an advanced mode switch that allows the device to switch into direct "AHRS" or "GPS" mode. In those modes, the device responds to the native protocols of the 3DM-GX3-25 AHRS or the u-blox5 GPS devices which are imbedded in the 3DM-GX3-45. These modes can be used to access advanced or specialized features of these devices (see the Advanced Programming section).

Command and Data Summary

Below is a summary of the commands and data available in the programming interface. Commands and data are denoted by two values. The first value denotes the “descriptor set” that the command or data belongs to (Base command, 3DM command, AHRS data, or GPS data) and the second value denotes the unique command or data “descriptor” in that set.

Commands

Base Command Set (0x01)

- Ping (0x01, 0x01)
- Set To Idle (0x01, 0x02)
- Get Device Information (0x01, 0x03)
- Get Device Descriptor Sets (0x01, 0x04)
- Device Built-In Test (BIT) (0x01, 0x05)
- Resume (0x01, 0x06)
- Device Reset (0x01, 0x7E)

3DM Command Set (0x0C)

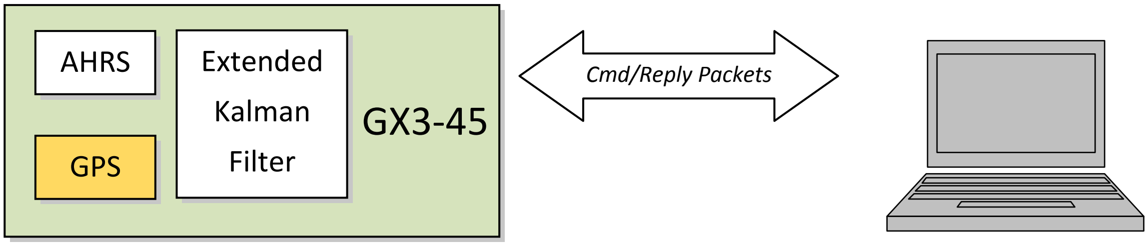
- Poll AHRS Data (0x0C, 0x01)
- Poll GPS Data (0x0C, 0x02)
- Poll NAV Data (0x0C, 0x03)
- Get AHRS Data Rate Base (0x0C, 0x06)
- Get GPS Data Rate Base (0x0C, 0x07)
- Get NAV Data Rate Base (0x0C, 0x0B)
- AHRS Message Format (0x0C, 0x08)
- GPS Message Format (0x0C, 0x09)
- NAV Message Format (0x0C, 0x0A)
- Enable/Disable Device Continuous Data Stream (0x0C, 0x11)
- Device Startup Settings (0x0C, 0x30)
- AHRS Signal Conditioning Settings (0x0C, 0x35)
- Change UART BAUD rate (0x0C, 0x40)
- Device Status* (0x0C, 0x64)

Navigation Filter Command Set (0x0D)

- Reset Filter (0x0D, 0x01)
- Set Initial Attitude (0x0D, 0x02)
- Set Initial Heading (0x0D, 0x03)
- Set Initial Attitude from AHRS (0x0D, 0x04)
- Vehicle Dynamics Mode (0x0D, 0x10)
- Sensor to Vehicle Frame Transformation (0x0D, 0x11)
- Sensor to Vehicle Frame Offset (0x0D, 0x12)
- Antenna Offset (0x0D, 0x13)
- Bias Estimation Control (0x0D, 0x14)
- GPS Source Control (0x0D, 0x15)
- External GPS Update (0x0D, 0x16)

Basic Programming

The 3DM-GX3-45 is designed to stream NAV, AHRS, and GPS data packets over a common interface as efficiently as possible. To this end, programming the device consists of a configuration stage where the data messages and data rates are configured. The configuration stage is followed by a data streaming stage where the program starts the incoming data packet stream.



In this section there is an overview of the packet, an overview of command and reply packets, an overview of how an incoming data packet is constructed, and then an example setup command sequence that can be used directly with the 3DM-GX3-45 either through a COM utility or as a template for software development.

MIP Packet Overview

This is an overview of the 3DM-GX3-45 packet structure. The packet structure used is the MicroStrain "MIP" packet. A reference to the general packet structure is presented in the MIP Packet Reference section. An overview of the packet is presented here.

The MIP packet "wrapper" consists of a four byte header and two byte checksum footer:

Header				Packet Payload			Checksum	
SYNC1 "u"	*SYNC2 "e"*	*Descriptor Set byte*	*Payload Length byte*	*Field Length byte*	*Field Descriptor byte*	*Field Data*	*MSB*	*LSB*
0x75	**0x65**	**0x80**	**0x0E**	**0x0E**	**0x03**	**0x3E 7A 63 A0 0xBB 8E 3B 29 0x7F E5 BF 7F**	**0x83**	**0xE1**

Payload Length byte. This specifies the length of the packet payload. The packet payload may contain one or more fields and thus this byte also represents the sum of the lengths of all the fields in the payload.

Descriptor Set. Descriptors are grouped into different sets. The value 0x80 identifies this packet as an AHRS data packet. Fields in this packet will be from the AHRS data descriptor set.

Start of Packet (SOP) "sync" bytes. These are the same for every MIP packet and are used to identify the start of the packet.

2 byte Fletcher checksum of all the bytes in the packet.

The packet payload section contains one or more fields. Fields have a length byte, descriptor byte, and data. The diagram below shows a packet payload with a single field.

Header				Packet Payload			Checksum	
SYNC1 "u"	*SYNC2 "e"*	*Descriptor Set byte*	*Payload Length byte*	*Field Length byte*	*Field Descriptor byte*	*Field Data*	*MSB*	*LSB*
0x75	0x65	0x80	0x0E	**0x0E**	**0x06**	**0x3E 7A 63 A0 0xBB 8E 3B 29 0x7F E5 BF 7F**	0x86	0x08

Field Length byte. This represents a count of all the bytes in the field including the length byte, descriptor byte and field data.

Descriptor byte. This byte identifies the contents of the field data. This descriptor indicates that the data is a mag vector (set: 0x80, descriptor: 0x06)

Field data. The length of the data is Field Length – 2. This data is 12 bytes long (14 – 2) and represents the floating point magnetometer vector value from the AHRS data set.

Below is an example of a packet payload with two fields (gyro vector and mag vector). Note the payload length byte of 0x1C which is the sum of the two field length bytes 0x0E + 0x0E:

Header				Packet Payload (2 fields)						Checksum	
SYNC1 "u"	*SYNC2 "e"*	*Descrip tor Set*	*Payload Length*	*Field1 Len*	*Field1 Descriptor*	*Field1 Data*	*Field2 Len*	*Field2 Descriptor*	*Field2 Data*	*MSB*	*LSB*
0x75	0x65	0x80	**0x1C**	**0x0E**	0x05	0x3E 7A 63 A0 0xBB 8E 3B 29 0x7F E5 BF 7F	**0x0E**	0x06	0x3E 7A 63 A0 0xBB 8E 3B 29 0x7F E5 BF 7F	0xB1	0x1E

Command Overview

The basic command sequence begins with the host sending a command to the device. A command packet contains a field with the command value and any command arguments.

The device responds by sending a reply packet. The reply contains at minimum an ACK/NACK field. If any additional data is included in a reply, it appears as a second field in the packet.

Example "Ping" Command Packet:

Below is an example of a "Ping" command packet from the Base command set. A "Ping" command has no arguments. Its function is to determine if a device is present and responsive:

Header				Packet Payload			Checksum	
SYNC1 "u"	*SYNC2 "e"*	*Descriptor Set byte*	*Payload Length byte*	*Field Length byte*	*Field Descriptor byte*	*Field Data*	*MSB*	*LSB*
0x75	**0x65**	**0x01**	**0x02**	**0x02**	**0x01**	*N/A*	**0xE0**	**0xC6**

Copy-Paste version: "7565 0102 0201 E0C6"

The packet header has the "ue" starting sync bytes characteristic of all MIP packets. The descriptor set byte (0x01) identifies the data as being from the Base command set. The length of the payload portion is 2 bytes. The payload portion of the packet consists of one field. The field starts with the length of the field which is followed by the descriptor byte (0x01) of the field. The field descriptor value *is* the command value. Here the descriptor identifies the command as the "Ping" command from the Base command descriptor set. There are no parameters associated with the ping command, so the field data is empty. The checksum is a two byte Fletcher checksum (see the MIP Packet Reference for instructions on how to compute a Fletcher two byte checksum).

Example "Ping" Reply Packet:

The "Ping" command will generate a reply packet from the device. The reply packet will contain an ACK/NACK field. The ACK/NACK field contains an "echo" of the command byte plus an error code. An error code of 0 is an "ACK" and a non-zero error code is a "NACK":

Header				Packet Payload			Checksum	
SYNC1 "u"	*SYNC2 "e"*	*Descriptor Set byte*	*Payload Length byte*	*Field Length byte*	*Field Descriptor byte*	*Field Data: 2 bytes*	*MSB*	*LSB*
0x75	**0x65**	**0x01**	**0x04**	**0x04**	**0xF1**	Command echo**: 0x01** Error code**: 0x00**	**0xD5**	**0x6A**

Copy-Paste version: "7565 0104 04F1 0100 D56A"

The packet header has the "ue" starting sync bytes characteristic of all MIP packets. The descriptor set byte (0x01) identifies the payload fields as being from the Base command set. The length of the payload portion is 4 bytes. The payload portion of the packet consists of one field. The field starts with the length of the field which is followed by the descriptor byte (0xF1) of the field. The field descriptor byte identifies the reply as the "ACK/NACK" from the Base command descriptor set. The field data consists of an "echo" of the original command (0x01) followed by the error code for the command (0x00). In this case the error is zero, so the field represents an "ACK". Some examples of non-zero error codes that might be sent are "timeout", "not implemented", and "invalid parameter in command". The checksum is a two byte Fletcher checksum (see the MIP Packet Reference for instructions on how to compute a Fletcher two byte checksum).

The ACK/NACK descriptor value (0xF1) is the same in all descriptor sets. The value belongs to a set of reserved global descriptor values.

The reply packet may have additional fields that contain information in reply to the command. For example, requesting Device Status will result in a reply packet that contains two fields in the packet payload: an ACK/NACK field and a device status information field.

Data Overview

Data packets are generated by the device. When the device is powered up, it may be configured to immediately stream data packets out to the host or it may be "idle" and waiting for a command to either start continuous data or to get data by "polling" (one data packet per request). Either way, the data packet is generated by the device in the same way.

Example Data Packet:

Below is an example of a MIP data packet which has one field that contains the scaled accelerometer vector.

Header				Packet Payload			Checksum	
SYNC1 "u"	*SYNC2 "e"*	*Descriptor Set byte*	*Payload Length byte*	*Field Length byte*	*Field Descriptor byte*	*Field Data: Accel vector (12 bytes, 3 float – X, Y, Z)*	*MSB*	*LSB*
0x75	**0x65**	**0x80**	**0x0E**	**0x0E**	**0x04**	**0x3E 7A 63 A0 0xBB 8E 3B 29 0x7F E5 BF 7F**	**0x92**	**0xC0**

Copy-Paste version: "7565 800E 0E04 3E7A 63A0 BB8E 3B29 7FE5 BF7F 92C0"

The packet header has the "ue" starting sync bytes characteristic of all MIP packets. The descriptor set byte (0x80) identifies the payload field as being from the AHRS data set. The length of the packet payload portion is 14 bytes (0x0E). The payload portion of the packet starts with the length of the field. The field descriptor byte (0x04) identifies the field data as the scaled accelerometer vector from the AHRS data descriptor set. The field data itself is three single precision floating point values of 4 bytes each (total of 12 bytes) representing the X, Y, and Z axis values of the vector. The checksum is a two byte Fletcher checksum (see the MIP Packet Reference for instructions on how to compute a Fletcher two byte checksum).

The format of the field data is fully and unambiguously specified by the descriptor. In this example, the field descriptor (0x04) specifies that the field data holds an array of three single precision IEEE-754 floating point numbers in big-endian byte order and that the values represent units of "g's" and the order of the values is X, Y, Z vector order. Any other specification would require a different descriptor (see the Data Reference section of this manual).

Each packet can contain any combination of data quantities from the same data descriptor set (any combination of GPS data OR any combination of AHRS data OR and combination of NAV data – you cannot combine GPS data, AHRS data, and NAV data in the same packet).

Data polling commands generate two individual reply packets: An ACK/NACK packet and a data packet. Enable/Disable continuous data commands generate an ACK/NACK packet followed by the continuous stream of data packets.

The AHRS, GPS, and NAV data packets can be set up so that each data quantity is sent at a different rate. For example, you can setup continuous data to send the accelerometer vector at 100Hz and the magnetometer vector at 5Hz. This means that packets will be sent at 100Hz and each one will have the accelerometer vector but only every 20th packet will have the magnetometer vector. This helps reduce bandwidth and buffering requirements. An example of this is given in the AHRS Message Format command.

Example Setup Sequence

Setup involves a series of command/reply pairs. The example below demonstrates actual setup sequences that you can send directly to the 3DM-GX3-45 either programmatically or by using a COM utility. In most cases only minor alterations will be needed to adapt these examples for your application.

Continuous Data Example Command Sequence

Most applications will operate with the 3DM-GX3-45 sending a continuous data stream. In the following example, the AHRS data format is set, followed by the NAV data format. GPS data will not be included as we will not be cross-checking against the navigation solution. To reduce the amount of streaming data, if present during the configuration, the device is placed into the idle state while performing the device initialization; when configuration is complete, the required data streams are enabled to bring the device out of idle mode. Finally, the configuration is saved so that it will be loaded on subsequent power-ups, eliminating the need to perform the configuration again.

Step 1: Put the Device in Idle Mode (Disabling the AHRS, GPS, and NAV data-streams)

Send the "Set To Idle" command to put the device in the idle state (reply is ACK/NACK). This is not required but reduces the parsing burden during initialization and makes visual confirmation of the commands easier:

Step 1	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Cmd Desc.*	*Field Data*	*MSB*	*LSB*
Command Set to Idle	**0x75**	**0x65**	**0x01**	**0x02**	**0x02**	**0x02**	N/A	**0xE1**	**0xC7**
Reply ACK/NACK	**0x75**	**0x65**	**0x01**	**0x04**	**0x04**	**0xF1**	Cmd echo: **0x02** Error code: **0x00**	**0xD6**	**0x6C**

Copy-Paste version of the command: "7565 0102 0202 E1C7"

Step 2: Configure the AHRS data-stream format

Send a "Set AHRS Message Format" command (reply is ACK/NACK). This example requests scaled gyro, scaled accelerometer, and GPS Correlation Timestamp information at 100 Hz (100 Hz base rate, with a rate decimation of 1 on the GX3-45 = 100 Hz.) This will result in a single AHRS data packet sent at 100 Hz containing the scaled gyro field followed by the scaled accelerometer field followed by the AHRS GPS Correlation Timestamp. This is a very typical configuration for a base level of inertial data. If different rates were requested, then each packet would only contain the data quantities that fall in the same decimation frame (see the Multiple Rate Data section). If the stream was not disabled in the previous step, the AHRS data would begin stream immediately.

Please note, this command will not append the requested descriptors to the current AHRS data-stream configuration, it will overwrite it completely:

Step 2	MIP Packet Header	Command/Reply Fields	Checksum

	Sync1	Sync2	Desc Set	Payload Length	Field Length	Cmd Desc.	Field Data	MSB	LSB
Command New AHRS Message Format	**0x75**	**0x65**	**0x0C**	**0x0D**	**0x0D**	**0x08**	Function: **0x01** Desc count: **0x03** 1st Descriptor: **0x04** Rate Dec: **0x0001** 2nd Descriptor:**0x05** Rate Dec: **0x0001** 3rd Descriptor:**0x12** Rate Dec: **0x0001**	**0x2A**	**0x35**
Reply ACK/NACK	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0xF1**	Cmd echo: **0x08** Error code: **0x00**	**0xE7**	**0xBA**

Copy-Paste version of the command: "7565 0C0D 0D08 0103 0400 0105 0001 1200 012A 35"

Step 3: Configure the NAV data-stream format

The following configuration command requests the Estimated LLH Position, Estimated NED Velocity, Estimated Orientation in Quaternion form, and Filter Status at 20 Hz (100Hz base rate, with a rate decimation of 5 = 20 Hz.) This will result in a single NAV packet sent at 20 Hz containing the requested fields in the requested order. If different rates were requested, the each packet would only contain the data quantities that fall in the same data rate frame (see the Multiple Rate Data section). If the stream was not disabled in the previous step, the NAV data would begin stream immediately.

Please note, this command will not append the requested descriptors to the current NAV data-stream configuration, it will overwrite it completely.

Step 3	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	Sync2	Desc Set	Payload Length	Field Length	Cmd Desc.	Field Data	MSB	LSB
Command New NAV Message Format	**0x75**	**0x65**	**0x0C**	**0x10**	**0x10**	**0x0A**	Function: **0x01** Desc Count: **0x04** Est. Pos desc: **0x01** Rate dec: **0x0005** Est. Vel desc: **0x02** Rate dec: **0x0005** Est. Quat desc: **0x03** Rate dec: **0x0005** Filter Status desc: **0x10** Rate dec: **0x0005**	**0x3F**	**0x31**
Reply ACK/NACK	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0xF1**	Cmd echo: **0x0A** Error code: **0x00**	**0xE9**	**0xBE**

Copy-Paste version of the command: "7565 0C10 100A 0104 0100 0502 0005 0300 0510 0005 3F31"

Step 4: Save the AHRS and NAV MIP Message format

To save the AHRS and NAV MIP Message format, use the "Save" function selector (0x03) in the AHRS and NAV Message Format commands. Below we've combined the two commands as two fields in the same packet. Notice that the two reply ACKs comes in one packet also. Alternatively, they could be sent as separate packets.

Step 4	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Cmd Desc.*	*Field Data*	*MSB*	*LSB*
Command field 1 Save Current AHRS Message Format	**0x75**	**0x65**	**0x0C**	**0x08**	**0x04**	**0x08**	Function: **0x03** Desc count: **0x00**		
Command field 2 Save Current NAV Message Format					**0x04**	**0x0A**	Function: **0x03** Desc count: **0x00**	**0x0E**	**0x31**
Reply field 1 ACK/NACK	**0x75**	**0x65**	**0x0C**	**0x08**	**0x04**	**0xF1**	Cmd echo: **0x08** Error code: **0x00**		
Reply field 2 ACK/NACK					**0x04**	**0xF1**	Cmd echo: **0x0A** Error code: **0x00**	**0xEA**	**0x71**

Copy-Paste version of the command: "7565 0C08 0408 0300 040A 0300 0E31"

Step 5: Enable the AHRS and NAV data-streams

Send an "Enable/Disable Continuous Stream" command to enable the AHRS and NAV continuous streams (reply is ACK). These streams may have already been enabled by default, this step is to confirm they are enabled. These streams will begin streaming data immediately.

Step 5	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Cmd Desc.*	*Field Data*	*MSB*	*LSB*
Command field 1 Enable Continuous AHRS Message	**0x75**	**0x65**	**0x0C**	**0x0A**	**0x05**	**0x11**	Fctn: **0x01** AHRS: **0x01** ON: **0x01**		
Command field 2 Enable Continuous NAV Message					**0x05**	**0x11**	Fctn: **0x01** NAV: **0x03** ON: **0x01**	**0x24**	**0xCC**
Reply field 1 ACK/NACK	**0x75**	**0x65**	**0x0C**	**0x08**	**0x04**	**0xF1**	Cmd echo: **0x11** Error code: **0x00**		

Reply field 2 ACK/NACK					**0x04**	**0xF1**	Cmd echo: **0x11** Error code: **0x00**	**0xFA**	**0xB5**

Copy-Paste version of the command: "7565 0C0A 0511 0101 0105 1101 0301 24 CC"

Step 6 (Optional): Resume the Device

Sending the "Resume" command is another method of re-enabling transmission of enabled data streams (reply is ACK/NACK).

Step 6	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Cmd Desc.*	*Field Data*	*MSB*	*LSB*
Command Resume	**0x75**	**0x65**	**0x01**	**0x02**	**0x02**	**0x06**	N/A	**0xE5**	**0xCB**
Reply ACK/NACK	**0x75**	**0x65**	**0x01**	**0x04**	**0x04**	**0xF1**	Cmd echo: **0x06** Error code: **0x00**	**0xDA**	**0x74**

Copy-Paste version of the command: "7565 0102 0206 E5CB"

Step 7: Initialize the Filter

At this point in the set-up, the GX3-45 is streaming data, but the Kalman Filter is not yet initialized. For a successful initialization to occur, the GPS must have a fix and the initial orientation must be known. The orientation may be initialized in 4 different ways: Setting all of the attitude elements manually, setting only the heading and allowing the device to determine pitch and roll, using the internal AHRS solution (which requires the magnetometers) to provide the initial orientation, or via auto-initialization, which uses the chosen heading update source to initialize. In this example, we will assume the magnetometers are available and use the AHRS solution to initialize the Kalman Filter. Once the attitude is initialized and the GPS fix becomes valid, the Kalman Filter estimation will propagate:

Step 7	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Cmd Desc.*	*Field Data*	*MSB*	*LSB*
Command Initialize Filter	**0x75**	**0x65**	**0x0D**	**0x06**	**0x06**	**0x04**	Declination: 0.0 deg 0x00000000	**0xF7**	**0xE9**
Reply ACK/NACK	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	Cmd echo: **0x04** Error code: **0x00**	**0xE4**	**0xB8**

Copy-Paste version of the command: "7565 0D06 0604 0000 0000 F7E9"

Polling Data Example Sequence

Polling for data is less efficient than processing a continuous data stream, but may be more appropriate for certain applications. The main difference from the continuous data example is the inclusion of the Poll data commands in the data loop:

Step 1: Put the Device in Idle Mode (Disabling the AHRS, GPS, and NAV data-streams)

Same as continuous streaming. See above.

Step 2: Configure the AHRS data-stream format

Same as continuous streaming. See above.

Step 3: Configure the NAV data-stream format

Same as continuous streaming. See above.

Step 4: Save the AHRS and NAV MIP Message format

Same as continuous streaming. See above.

Step 5: Resume the Device

Same as continuous streaming step 6. See above.

Step 6: Initialize the Filter

Same as continuous streaming step 7. See above.

Step 7: Send individual data polling commands

Send individual Poll AHRS Data and Poll NAV Data commands in your data collection loop. After the ACK/NACK is sent by the device, a single data packet will be sent according to the settings in the previous steps. Note that the ACK/NACK has the same descriptor set value as the command, but the data packet has the descriptor set value for the type of data (AHRS or NAV):

Step 7	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Cmd Desc.*	*Field Data*	*MSB*	*LSB*
Command Poll AHRS Data	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0x01**	Option: **0x00** Desc Count: **0x00**	**0xEF**	**0xDA**
Reply ACK/NACK	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0xF1**	Cmd echo: **0x01** Error code: **0x00**	**0xE0**	**0xAC**
AHRS Data Packet field 1 (Gyro Vector)	**0x75**	**0x65**	**0x80**	**0x1C**	**0x0E**	**0x04**	**0x3E 7A 63 A0** **0xBB 8E 3B 29** **0x7F E5 BF 7F**		
AHRS Data Packet field 2(Accel Vector)					**0x0E**	**0x03**	**0x3E 7A 63 A0** **0xBB 8E 3B 29** **0x7F E5 BF 7F**	**0xAD**	**0xDC**

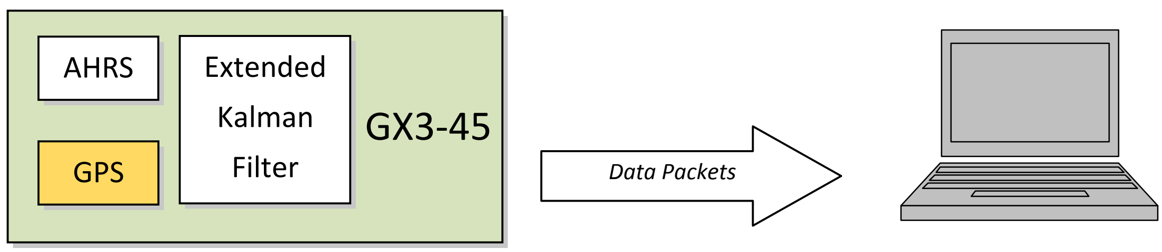
Copy-Paste version of the command: "7565 0C04 0401 0000 EF DA"

You may specify the format of the data packet on a per-polling-command basis rather than using the pre-set data format (see the Poll AHRS Data *and* Poll NAV Data sections*)*

The polling command has an option to suppress the ACK/NACK in order to keep the incoming stream clear of anything except data packets. Set the option byte to 0x01 for this feature.

Parsing Incoming Packets

Setup is usually the easy part of programming the 3DM-GX3-45. Once you start continuous data streaming, parsing and processing the incoming data packet stream will become the primary focus. The stream of data from the AHRS and Kalman Filter (NAV) are usually the dominant source of data since they come in the fastest. Polling for data may seem to be a logical solution to controlling the data flow, and this may be appropriate for some applications, but if your application requires the precise delivery of inertial data, it is often necessary to have the data stream drive the process rather than having the host try to control the data stream through polling.



The “descriptor set” qualifier in the MIP packet header is a feature that greatly aids the management of the incoming packet stream by making it easy to sort the packets into logical sub-streams and route those streams to appropriate handlers. The first step is to parse the incoming character stream into packets.

It is important to take an organized approach to parsing continuous data. The basic strategy is this: parse the incoming stream of characters for the packet starting sequence “ue” and then wait for the entire packet to come in based on the packet length byte which arrives after the “ue” and descriptor set byte. Make sure you have a timeout on your wait loop in case your stream is out of sync and the starting “ue” sequence winds up being a “ghost” sequence. If you timeout, restart the parsing with the first character after the ghost “ue”. Once the stream is in sync, it is rare that you will hit a timeout unless you have an unreliable communications link. After verifying the checksum, examine the “descriptor set” field in the header of the packet. This tells you immediately how to handle the packet.

Based on the value of the descriptor set field in the packet header, pass the packet to either a command handler (if it is a Base command or 3DM command descriptor set) or a data handler (if it is a GPS, AHRS, or NAV data set). Since you know beforehand that the AHRS and NAV data packets will be coming in fastest, you can tune your code to buffer or handle these packets at a high priority. Likewise, you know that the GPS packets will be coming in at a much lower rate but may have much more data to process. Again, you can tune your code to buffer or handle these slower packets appropriately. Replies to commands generally happen sequentially after a command so the incidence of these is under program control.

For multithreaded applications, it is often useful to use queues to buffer packets bound for different packet handler threads. The depth of the queue can be tuned so that no packets are dropped while waiting for their associated threads to process the packets in the queue. See Advanced Programming Models section for more information on this topic.

Once you have sorted the different packets and sent them to the proper packet handler, the packet handler may parse the packet payload fields and handle each of the fields as appropriate for the application. For simple applications, it is perfectly acceptable to have a single handler for all packet types. Likewise, it is perfectly acceptable for a single parser to handle both the packet type and the fields in the packet. The ability to sort the packets by type is just an option that simplifies the implementation of more sophisticated applications.

Multiple Rate Data

The message format commands (AHRS Message Format, GPS Message Format, and NAV Message Format) allow you to set different data rates for different data quantities. This is a very useful feature especially for AHRS data because some data, such as accelerometer and gyroscope data, usually requires higher data rates (100Hz) than other AHRS data such as Magnetometer (20Hz typical) data. The ability to send data at different rates reduces the parsing load on the user program and decreases the bandwidth requirements of the communications channel.

Multiple rate data is scheduled on a common sampling rate clock. This means that if there is more than one data rate scheduled, the schedules coincide periodically. For example, if you request Accelerometer data at 100Hz and Magnetometer data at 50Hz, the magnetometer schedule coincides with the Accelerometer schedule 50% of the time. When the schedules coincide, then the two data quantities are delivered in the same packet. In other words, in this example, you will receive data packets at 100Hz and every packet will have an accelerometer data field and EVERY OTHER packet will also include a magnetometer data field:

Packet 1	*Packet 2*	*Packet 3*	*Packet 4*	*Packet 5*	*Packet 6*	*Packet 7*	*Packet 8*	….
Accel	Accel Mag	Accel	Accel Mag	Accel	Accel Mag	Accel	Accel Mag	Accel

If a timestamp is included at 100Hz, then the timestamp will also be included in every packet in this example. It is important to note that *the data in a packet with a timestamp is always synchronous with the timestamp.* This assures that multiple rate data is always synchronous.

Packet 1	*Packet 2*	*Packet 3*	*Packet 4*	*Packet 5*	*Packet 6*	….
Accel Timestamp	Accel Mag Timestamp	Accel Timestamp	Accel Mag Timestamp	Accel Timestamp	Accel Mag Timestamp	Accel Timestamp

Data Synchronicity

Because the MIP packet allows multiple data fields to be in a single packet, it may be assumed that a single timestamp field in the packet applies to all the data in the packet. In other words, it may be assumed that all the data fields in the packet were sampled at the same time.

AHRS, GPS, and NAV data are generated independently by three systems with different clocks. The importance of time is different in each system and the data they produce. The AHRS data requires precise microsecond resolution and perfectly regular intervals in its timestamps. GPS data produces very precise UTC interval data but it is typically delivered in a 1 second time frame. The Kalman Filter resides on a separate processor and must derive its timing information from the two data sources.

The time base difference is one of the factors that necessitate separation of the GPS, AHRS, and NAV data into separate packets. Conversely, the common time base of the different data quantities within one system is what allows grouping multiple data quantities into a single packet with a common timestamp. In other words, AHRS data is always grouped with a timestamp generated from the AHRS time base, and GPS data is always grouped with a timestamp from the GPS time base, etc.

In many applications, synchronizing the timestamps from the three system time bases is critical. MicroStrain uses an extended timestamp across its product line to allow synchronization of data sampled on different system clocks. This timestamp relies on a pulse per second (PPS) beacon signal. On the 3DM-GX3-45, this PPS signal is generated by the on board GPS. The timestamp of the AHRS data represents the interval in nanoseconds from the last PPS pulse. This allows proper time alignment of the GPS data with the AHRS data. On other systems, the PPS signal is applied externally by a system wide PPS beacon. The 3DM-GX3-45 can be the source of this beacon by picking off the PPS output on the multi-com connector.

A second form of synchronization exists which can be slightly less accurate, but easier to use in practice. All data streams (AHRS, GPS, and NAV) on the GX3-45 output a “GPS Time”-formatted timestamp. This timestamp is synchronized between the 3 devices using the GPS 1PPS hardware beacon. Due to the differences in clocks on each device, the period between two consecutive timestamp values may not be constant; this occurs because periodic corrections are applied to the AHRS and NAV timestamps when the GPS 1PPS signal is asserted.

Communications Bandwidth Management

Because of the large amount and variety of data that is available from the 3DM-GX3-45, it is quite easy to overdrive the bandwidth of the communications channel. This can result in dropped packets. The 3DM-GX3-45 does not do analysis of the bandwidth requirements for any given output data configuration, it will simply drop a packet if its internal serial buffer is being filled faster than it is being emptied. It is up to the programmer to analyze the size of the data packets requested and the available bandwidth of the communications channel.

Often the best way to determine this is empirically by trying different settings and watching for dropped packets. Below are some guidelines on how to determine maximum bandwidth for your application.

UART Bandwidth Calculation

Below is an equation for the maximum theoretical UART BAUD rate for a given message configuration. Although it is possible to calculate the approximate bandwidth required for a given setup, there is no guarantee that the system can support that setup due to internal processing delays. The best approach is to try a setting based on an initial estimate and watch for dropped packets. If there are dropped packets, increase the BAUD rate, reduce the data rate, or decrease the size or number of packets.

$$\mathrm{n}(\mathrm{k} \times f_{mr}) + \mathrm{n}\sum(S_f \times f_{dr})$$

Where

$$S_f = Size\ of\ data\ field\ in\ bytes$$
$$f_{dr} = field\ data\ rate\ in\ Hz$$
$$f_{mr} = maximum\ data\ rate\ in\ Hz$$
$$\mathrm{n} = size\ of\ UART\ word = 10 bits$$
$$\mathrm{k} = Size\ of\ MIP\ wrapper = 6\ bytes$$

which becomes

$$60 f_{mr} + 10\sum(S_f \times f_{dr})$$

Example:

For an AHRS message format of Accelerometer Vector (14 byte data field) + Internal Timestamp (6 byte data field), both at 100 Hz, the theoretical minimum BAUD rate would be:

$$= 60 \times 100 + 10((14 \times 100) + (6 \times 100))$$

$$= 26000\ \mathrm{BAUD}$$

In practice, if you set the BAUD rate to 115200 the packets come through without any packet drops. If you set the BAUD rate to the next available lower rate of 19200, which is lower than the calculated minimum, you get regular packet drops. The only way to determine a packet drop is by observing a timestamp in sequential packets. The interval should not change from packet to packet. If it does change then packets were dropped.

USB vs. UART

The 3DM-GX3-45 has a dual communication interface: USB or UART. There is an important difference between USB and UART communication with regards to data bandwidth. The USB “virtual COM port” that the 3DM-GX3-45 implements runs at USB “full-speed” setting of 12Mbs (megabits per second). However, USB is a polled master-slave system and so the slave (3DM-GX3-45) can only communicate when polled by the master. This results in inconsistent data streaming – that is, the data comes in spurts rather than at a constant rate and, although rare, sometimes data can be dropped if the host processor fails to poll the USB device in a timely manner.

With the UART the opposite is true. The 3DM-GX3-45 operates without UART handshaking which means it streams data out at a very consistent rate without stopping. Since the host processor has no handshake method of pausing the stream, it must instead make sure that it can process the incoming packet stream non-stop without dropping packets.

In practice, USB and UART communications behave similarly on a Windows based PC, however, UART is the preferred communications system if consistent, deterministic communications timing behavior is required. USB is preferred if you require more data than is possible over the UART and you can tolerate the possibility of variable latency in the data delivery and very occasional packet drops due to host system delays in servicing the USB port.

Command Reference

Base Commands

The Base command set is common to many MicroStrain devices. With the Base command set it is possible to identify many properties and do basic functions on a device even if you do not recognize its specialized functionality or data. The commands work the same way on all devices that implement this set.

Ping (0x01, 0x01)

Description	Send a "Ping" command		
Notes	Device responds with ACK/NACK packet if present.		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x02	0x01	N/A
Reply ACK/NACK	0x04	0xF1	U8 – echo the command byte U8 – error code (0:ACK, non-zero: NACK)

Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command Ping	**0x75**	**0x65**	**0x01**	**0x02**	**0x02**	**0x01**		**0xE0**	**0xC6**
Reply ACK/NACK	**0x75**	**0x65**	**0x01**	**0x04**	**0x04**	**0xF1**	Command echo: **0x01** Error code: **0x00**	**0xD5**	**0x6A**

Copy-Paste version of the command: "7565 0102 0201 E0C6"

Set To Idle (0x01, 0x02)

Description	Place device into idle mode.		
Notes	Command has no parameters. Device responds with ACK if successfully placed in idle mode. This command will suspend streaming (if enabled) or wake the device from sleep (if sleeping) to allow it to respond to status and setup commands. You may restore the device mode by issuing the Resume command.		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x02	0x02	N/A
Reply ACK/NACK	0x04	0xF1	U8 – echo the command byte U8 – error code (0: ACK, non-zero: NACK)

Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command Set To Idle	**0x75**	**0x65**	**0x01**	**0x02**	**0x02**	**0x02**		**0xE1**	**0xC7**
Reply ACK/NACK	**0x75**	**0x65**	**0x01**	**0x04**	**0x04**	**0xF1**	Command echo: **0x02** Error code: **0x00**	**0xD6**	**0x6C**

Copy-Paste version of the command: "7565 0102 0202 E1C7"

Resume (0x01, 0x06)

Description	Place device back into the mode it was in before issuing the Set To Idle command. If the Set To Idle command was not issued, then the device is placed in default mode.		
Notes	Command has no parameters. Device responds with ACK if stream successfully enabled.		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x02	0x06	N/A
Reply ACK/NACK	0x04	0xF1	U8 – echo the command byte U8 – error code (0: ACK, non-zero: NACK)

Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command Set To Idle	**0x75**	**0x65**	**0x01**	**0x02**	**0x02**	**0x06**		**0xE5**	**0xCB**
Reply ACK/NACK	**0x75**	**0x65**	**0x01**	**0x04**	**0x04**	**0xF1**	Command echo: **0x06** Error code: **0x00**	**0xDA**	**0x74**

Copy-Paste version of the command: "7565 0102 0206 E5CB"

Get Device Information (0x01, 0x03)

Description	Get the device ID strings and firmware version					
Notes	Reply has two fields: "ACK/NACK" and "Device Info Field"					
Field Format	*Field Length*	*Field Descriptor*	*Field Data*			
Command	0x02	0x03	N/A			
Reply field 1 ACK/NACK	0x04	0xF1	U8 – echo the command byte U8 – error code (0: ACK, non-zero: NACK)			
Reply field 2 Device Info Field	0x54	0x81	*Binary Offset*	*Description*	*Data Type*	*Units*
			0	Firmware Version	U16	N/A
			2	Model Name	String(16)	N/A
			18	Model Number	String(16)	N/A
			34	Serial Number	String(16)	N/A
			50	Lot Number	String(16)	N/A
			66	Device Options	String(16)	N/A

Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command Get Device Info	**0x75**	**0x65**	**0x01**	**0x02**	**0x02**	**0x03**		**0xE2**	**0xC8**
Reply Field 1 ACK/NACK	**0x75**	**0x65**	**0x01**	**0x58**	**0x04**	**0xF1**	Command echo: **0x03** Error code: **0x00**		
Reply Field 2 Device Info Field					**0x54**	**0x81**	FW Version: **0x05FE** " 3DM-GX3-45" " 6226-4220" " 6226-01319" " I042Y" " 5g, 300d/s"	**0x##**	**0x##**

Copy-Paste version of the command: "7565 0102 0203 E2C8"

Get Device Descriptor Sets (0x01, 0x04)

<table>
<tr><td>Description</td><td colspan="5">Get the set of descriptors that this device supports</td></tr>
<tr><td>Notes</td><td colspan="5">Reply has two fields: “ACK/NACK” and “Descriptors”. The “Descriptors” field is an array of 16 bit values. The MSB specifies the descriptor set and the LSB specifies the descriptor.</td></tr>
<tr><td>Field Format</td><td>Field Length</td><td>Field Descriptor</td><td colspan="3">Field Data</td></tr>
<tr><td>Command</td><td>0x02</td><td>0x04</td><td colspan="3">N/A</td></tr>
<tr><td>Reply field 1
ACK/NACK</td><td>0x04</td><td>0xF1</td><td colspan="3">U8 – echo the command byte
U8 – error code (0: ACK, non-zero: NACK)</td></tr>
<tr><td rowspan="4">Reply field 2
Array of Descriptors</td><td rowspan="4">2 x <Number of descriptors> + 2</td><td rowspan="4">0x82</td><td>Binary Offset</td><td>Description</td><td>Data Type</td></tr>
<tr><td>0</td><td>MSB: Descriptor Set
LSB: Descriptor</td><td>U16</td></tr>
<tr><td>1</td><td>MSB: Descriptor Set
LSB: Descriptor</td><td>U16</td></tr>
<tr><td>…</td><td><etc></td><td>…</td></tr>
</table>

<table>
<tr><td rowspan="2">Example</td><td colspan="4">MIP Packet Header</td><td colspan="3">Command/Reply Fields</td><td colspan="2">Checksum</td></tr>
<tr><td>Sync1</td><td>Sync2</td><td>Desc Set</td><td>Payload Length</td><td>Field Length</td><td>Field Desc.</td><td>Field Data</td><td>MSB</td><td>LSB</td></tr>
<tr><td>Command
Get Device Info</td><td>0x75</td><td>0x65</td><td>0x01</td><td>0x02</td><td>0x02</td><td>0x04</td><td></td><td>0xE3</td><td>0xC9</td></tr>
<tr><td>Reply Field 1
ACK/NACK</td><td>0x75</td><td>0x65</td><td>0x01</td><td>0x04</td><td>0x04</td><td>0xF1</td><td>Command echo: 0x04
Error code: 0x00</td><td></td><td></td></tr>
<tr><td>Reply Field 2
Array of Descriptors</td><td></td><td></td><td></td><td></td><td><n*2> +2</td><td>0x82</td><td>0x0101
0x0102
0x0103
…
0x0C01
0x0C02
…
nth descriptor:
0x0C72</td><td>0x##</td><td>0x##</td></tr>
</table>

Copy-Paste version of the command: “7565 0102 0204 E3C9”

Device Built-In Test (0x01, 0x05)

Description	Run the device Built-In Test (BIT). The Built-In Test command always returns a 32 bit value. A value of 0 means that all tests passed. A non-zero value indicates that not all tests passed. The failure flags are device dependent. The flags for the 3DM-GX3-45 are defined below.
Notes	The BIT will take approximately 5 seconds to complete on the 3DM-GX3-45. The GPS power will be cycled during the test resulting in the temporary loss and subsequent recalculation of the position solution.

3DM-GX3-45 BIT Error Flags:

Byte Device	Byte 1 (LSB) AP-1 Processor	Byte 2 AHRS	Byte 3 GPS	Byte 4 (MSB) Reserved
Bit 1 (LSB)	I2C Hardware Error	Communication Error	Communication Error	Reserved
Bit 2	I2C EEPROM Error	Reserved	1PPS Signal Error	Reserved
Bit 3	Reserved	Reserved	1 PPS Inhibit Error	Reserved
Bit 4	Reserved	Reserved	Power Control Error	Reserved
Bit 5	Reserved	Reserved	Reserved	Reserved
Bit 6	Reserved	Reserved	Reserved	Reserved
Bit 7	Reserved	Reserved	Reserved	Reserved
Bit 8 (MSB)	Reserved	Reserved	Reserved	Reserved

Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x02	0x05	N/A
Reply field 1 ACK/NACK	0x04	0xF1	U8 – echo the command byte U8 – error code (0:ACK, non-zero: NACK)
Reply field 2 BIT Error Flags	0x06	0x83	U32 – BIT Error Flags

Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command Built-In Test	**0x75**	**0x65**	**0x01**	**0x02**	**0x02**	**0x05**	N/A	**0xE4**	**0xCA**
Reply field 1 ACK/NACK	**0x75**	**0x65**	**0x01**	**0x0A**	**0x04**	**0xF1**	*Echo cmd:* **0x05** *Error code:* **0x00**		
Reply field 2 BIT Error Flags					**0x06**	**0x83**	*BIT Error Flags:* **0x00000000**	**0x68**	**0x7D**

Copy-Paste version of the command: "7565 0102 0205 E4CA"

Device Reset (0x01, 0x7E)

Description	Resets the 3DM-GX3-45.		
Notes	Device responds with ACK if it recognizes the command and then immediately resets.		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x02	0x7E	N/A
Reply ACK/NACK	0x04	0xF1	U8 – echo the command descriptor U8 – error code (0: ACK, non-zero: NACK)

Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command Set Reset	**0x75**	**0x65**	**0x01**	**0x02**	**0x02**	**0x7E**	*N/A*	**0x5D**	**0x43**
Reply ACK/NACK	**0x75**	**0x65**	**0x01**	**0x04**	**0x04**	**0xF1**	Command echo: **0x7E** Error code: **0x00**	**0x52**	**0x64**

Copy-Paste version of the command: "7565 0102 027E 5D43"

3DM Commands

The 3DM command set is common to the MicroStrain Inertial sensors that support the MIP packet protocol. Because of the unified set of commands, it is easy to migrate code from one inertial sensor to another.

Poll AHRS Data (0x0C, 0x01)

Description	Poll the 3DM-GX3-45 for an AHRS message with the specified format		
Notes	This function polls for an AHRS message using the provided format. The resulting message will maintain the order of descriptors sent in the command and any unrecognized descriptors are ignored. If the format is not provided, the device will attempt to use the stored format (set with the Set AHRS Message Format command.) If no format is provided and there is no stored format, the device will respond with a NACK. The reply packet contains an ACK/NACK field. The polled data packet is sent separately as an AHRS Data packet. *Possible Option Selector Values:* 0x00 – Normal ACK/NACK Reply. 0x01 – Suppress the ACK/NACK reply.		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	4 + 3*N	0x01	U8 – Option Selector U8 – Number of Descriptors (N), N*(U8 – Descriptor, U16 Reserved)
Reply ACK/NACK	0x04	0xF1	U8 – echo the command byte U8 – error code (0:ACK, not 0:NACK)

Examples	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command Poll AHRS data (use stored format)	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0x01**	Option: **0x00** Desc count: **0x00**	**0xEF**	**0xDA**
Command Poll AHRS data (use specified format)	**0x75**	**0x65**	**0x0C**	**0x0A**	**0x0A**	**0x01**	Option: **0x00** Desc count: **0x02** 1st Descriptor: **0x04** Reserved: **0x0000** 2nd Descriptor:**0x05** Reserved: **0x0000**	**0x06**	**0x27**
Reply ACK/NACK (Data packet is sent separately if ACK)	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x01** *Error code:* **0x00**	**0xE0**	**0xAC**

Copy-Paste versions of the commands:
Stored format: “7565 0C04 0401 0000 EFDA”
Specified format: “7565 0C0A 0A01 0002 0400 0005 0000 0627”

Poll GPS Data (0x0C, 0x02)

Description	Poll the 3DM-GX3-45 for a GPS message with the specified format		
Notes	This function polls for a GPS message using the provided format. The resulting message will maintain the order of descriptors sent in the command and any unrecognized descriptors are ignored. If the format is not provided, the device will attempt to use the stored format (set with the Set GPS Message Format command.) If no format is provided and there is no stored format, the device will respond with a NACK. The reply packet contains an ACK/NACK field. The polled data packet is sent separately as a GPS Data packet. *Possible Option Selector Values:* 0x00 – Normal ACK/NACK Reply. 0x01 – Suppress the ACK/NACK reply.		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	4 + 3*N	0x02	U8 – Option Selector U8 – Number of Descriptors (N), N*(U8 – Descriptor, U16 Reserved)
Reply ACK/NACK	0x04	0xF1	U8 – echo the command byte U8 – error code (0:ACK, not 0:NACK)

Examples	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command Poll GPS data (use stored format)	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0x02**	Option: **0x00** Desc count: **0x00**	**0xF0**	**0xDD**
Command Poll GPS data (use specified format)	**0x75**	**0x65**	**0x0C**	**0x0A**	**0x0A**	**0x02**	Option: **0x00** Desc count: **0x02** 1st Descriptor: **0x03** Reserved: **0x0000** 2nd Descriptor:**0x05** Reserved: **0x0000**	**0x06**	**0x2A**
Reply ACK/NACK (Data packet is sent separately if ACK)	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x02** *Error code:* **0x00**	**0xE1**	**0xAE**

Copy-Paste versions of the commands:
Stored format: "7565 0C04 0402 0000 F0DD"
Specified format: "7565 0C0A 0A02 0002 0300 0005 0000 062A"

Poll NAV Data (0x0C, 0x03)

Description	Poll the 3DM-GX3-45 for a NAV message with the specified format		
Notes	This function polls for a NAV message using the provided format. The resulting message will maintain the order of descriptors sent in the command and any unrecognized descriptors are ignored. If the format is not provided, the device will attempt to use the stored format (set with the Set NAV Message Format command.) If no format is provided and there is no stored format, the device will respond with a NACK. The reply packet contains an ACK/NACK field. The polled data packet is sent separately as a NAV Data packet. *Possible Option Selector Values:* 0x00 – Normal ACK/NACK Reply. 0x01 – Suppress the ACK/NACK reply.		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	4 + 3*N	0x03	U8 – Option Selector U8 – Number of Descriptors (N), N*(U8 – Descriptor, U16 Reserved)
Reply ACK/NACK	0x04	0xF1	U8 – echo the command byte U8 – error code (0:ACK, not 0:NACK)

Examples	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command Poll NAV data (use stored format)	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0x03**	Option: **0x00** Desc count: **0x00**	**0xF1**	**0xE0**
Command Poll NAV data (use specified format)	**0x75**	**0x65**	**0x0C**	**0x0A**	**0x0A**	**0x03**	Option: **0x00** Desc count: **0x02** 1st Descriptor: **0x01** Reserved: **0x0000** 2nd Descriptor:**0x02** Reserved: **0x0000**	**0x02**	**0x1E**
Reply ACK/NACK (Data packet is sent separately if ACK)	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x03** *Error code:* **0x00**	**0xE2**	**0xB0**

Copy-Paste versions of the commands:
Stored format: "7565 0C04 0403 0000 F1E0"
Specified format: "7565 0C0A 0A03 0002 0100 0002 0000 021E"

Get AHRS Data Rate Base(0x0C, 0x06)

Description	Get the decimation base for the AHRS Data rate calculations		
Notes	Returns the value used for data rate calculations. See the AHRS Message Format command. *Most models of 3DM-GX3-45 have an AHRS Base Data Rate of 100. This is used for all the examples in this document. For a given device, this value stays constant.*		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x02	0x06	*none*
Reply field 1 ACK/NACK Field	0x04	0xF1	U8 – echo the command byte U8 – error code (0:ACK, not 0:NACK)
Reply field 2 Communications Mode	0x04	0x83	U16-AHRS data rate decimation base

Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	MSB	LSB
Command Get Communications Mode	**0x75**	**0x65**	**0x0C**	**0x02**	**0x02**	**0x06**		**0xF0**	**0xF7**
Reply field 1 ACK/NACK	**0x75**	**0x65**	**0x0C**	**0x08**	**0x04**	**0xF1**	*Echo cmd:* **0x06** *Error code:* **0x00**		
Reply field 2 Communication Mode					**0x04**	**0x83**	Rate decimation base: **0x0064**	**0xD4**	**0x6B**

Copy-Paste version of the command: “7565 0C02 0206 F0F7”

Get GPS Data Rate Base(0x0C, 0x07)

Description	Get the decimation base for the GPS Data rate calculations		
Notes	Returns the value used for data rate calculations. See the GPS Message Format command. *Most models of 3DM-GX3-45 have a GPS Base Data Rate of 4. This is used for all the examples in this document. For a given device, this value stays constant.*		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x02	0x07	*none*
Reply field 1 *ACK/NACK Field*	0x04	0xF1	U8 – echo the command byte U8 – error code (0:ACK, not 0:NACK)
Reply field 2 *Communications Mode*	0x04	0x84	U16-GPS data rate decimation base

Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	MSB	LSB
Command *Get Communications Mode*	**0x75**	**0x65**	**0x0C**	**0x02**	**0x02**	**0x07**		**0xF1**	**0xF8**
Reply field 1 *ACK/NACK*	**0x75**	**0x65**	**0x0C**	**0x08**	**0x04**	**0xF1**	*Echo cmd:* **0x07** *Error code:* **0x00**		
Reply field 2 *Communication Mode*					**0x04**	**0x84**	Rate decimation base: **0x0004**	**0x76**	**0x14**

Copy-Paste version of the command: "7565 0C02 0207 F1F8"

Get NAV Data Rate Base (0x0C, 0x0B)

Description	Get the decimation base for the NAV Data rate calculations		
Notes	Returns the value used for data rate calculations. See the NAV Message Format command. *Most models of 3DM-GX3-45 have a NAV Base Data Rate of 100. This is used for all the examples in this document. For a given device, this value stays constant.*		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x02	0x0B	*none*
Reply field 1 ACK/NACK Field	0x04	0xF1	U8 – echo the command byte U8 – error code (0:ACK, not 0:NACK)
Reply field 2 Communications Mode	0x04	0x8A	U16- NAV data rate decimation base

Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	MSB	LSB
Command Get Communications Mode	**0x75**	**0x65**	**0x0C**	**0x02**	**0x02**	**0x0B**	*N.A.*	**0xF5**	**0xFC**
Reply field 1 ACK/NACK	**0x75**	**0x65**	**0x0C**	**0x08**	**0x04**	**0xF1**	*Echo cmd:* **0x0B** *Error code:* **0x00**		
Reply field 2 Communication Mode					**0x04**	**0x8A**	Rate decimation base: **0x0064**	**0xE0**	**0x9E**

Copy-Paste version of the command: "7565 0C02 020B F5FC"

AHRS Message Format (0x0C, 0x08)

Description	Set, read, or save the format of the AHRS message packet. This command sets the format for the AHRS data packet when in standard mode. The resulting data messages will maintain the order of descriptors sent in the command. The command has a function selector and a descriptor array as parameters.
Notes	*Possible function selector values:* 0x01 – Use new settings 0x02 – Read back current settings. 0x03 – Save current settings as startup settings 0x04 – Load saved startup settings 0x05 – Reset to factory default settings The rate decimation field is calculated as follows for AHRS messages: Data Rate = 100Hz / Rate Decimation The GX3-45 checks that all descriptors are valid prior to executing this command. If any of the descriptors are invalid for the AHRS descriptor set, a NACK will be returned and the message format will be unchanged. The descriptor array only needs to be provided if the function selector is = 1 (Use new settings). For all other functions it may be empty (Number of Descriptors = 0).

Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	4 + 3*N	0x08	U8 - Function Selector U8 – Number of Descriptors (N), N*(U8 – Descriptor, U16 – Rate Decimation)
Reply ACK/NACK	0x04	0xF1	U8 – echo the command descriptor U8 – error code (0:ACK, not 0:NACK)
Reply field 2 (function = 2)	3 + 3*N	0x80	U8 – Number of Descriptors (N), N*(U8 – Descriptor, U16 – Rate Decimation)

Examples	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command AHRS Message Format (use new settings)	**0x75**	**0x65**	**0x0C**	**0x0A**	**0x0A**	**0x08**	Function: **0x01** Desc count: **0x02** 1st Descriptor: **0x04** Rate Dec: **0x000A** 2nd Descriptor:**0x05**	**0x22**	**0xA0**

							Rate Dec: **0x000A**		
Reply ACK/NACK	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x08** *Error code:* **0x00**	**0xE7**	**0xBA**
Command AHRS Message Format (read back current settings)	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0x08**	Function: **0x02** Desc count: **0x00**	**0xF8**	**0xF3**
Reply field 1 ACK/NACK	**0x75**	**0x65**	**0x0C**	**0x0D**	**0x04**	**0xF1**	*Echo cmd:* **0x08** *Error code:* **0x00**		
Reply field 2 Current AHRS Message Format					**0x09**	**0x80**	Desc count: **0x02** 1st Descriptor: **0x03** Rate Dec: **0x000A** 2nd Descriptor:**0x04** Rate Dec: **0x000A**	**0x98**	**0x0F**

Copy-Paste version of the commands:
Use New Settings: ”7565 0C0A 0A08 0102 0400 0A05 000A 22A0”
Read Current Settings: “7565 0C04 0408 0200 F8F3“

GPS Message Format (0x0C, 0x09)

Description	Set, read, or save the format of the GPS message packet. This function sets the format for the GPS MIP data packet when in standard mode. The resulting message will maintain the order of descriptors sent in the command. The command has a function selector and a descriptor array as parameters.		
Notes	*Possible function selector values:* 0x01 – Use new settings 0x02 – Read back current settings. 0x03 – Save current settings as startup settings 0x04 – Load saved startup settings 0x05 – Reset to factory default settings The rate decimation field is calculated as follows for GPS messages: *Data Rate = 4Hz / Rate Decimation* The GX3-45 checks that all descriptors are valid prior to executing this command. If any of the descriptors are invalid for the GPS data descriptor set, a NACK will be returned and the message format will be unchanged. The descriptor array only needs to be provided if the function selector is = 1 (Use new settings). For all other functions it may be empty (Number of Descriptors = 0).		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	4 + 3*N	0x09	U8 - Function Selector U8 – Number of Descriptors (N), N*(U8 – Descriptor, U16 – Rate Decimation)
Reply field 1 ACK/NACK	0x04	0xF1	U8 – echo the command descriptor U8 – error code (0:ACK, not 0:NACK)
Reply field 2 (function = 2)	3 + 3*N	0x81	U8 – Number of Descriptors (N), N*(U8 – Descriptor, U16 – Rate Decimation)

Examples	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command GPS Message Format (use new settings)	**0x75**	**0x65**	**0x0C**	**0x0A**	**0x0A**	**0x09**	Function: **0x01** Desc count: **0x02** 1st Descriptor: **0x03** Data rate: **0x0004** 2nd Descriptor:**0x05**	**0x16**	**0x85**

							Data rate: **0x0004**		
Reply ACK/NACK	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x09** *Error code:* **0x00**	**0xE8**	**0xBC**
Command GPS Message Format (read back current settings)	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0x09**	Function: **0x02** Desc count: **0x00**	**0xF9**	**0xF6**
Reply field 1 ACK/NACK	**0x75**	**0x65**	**0x0C**	**0x0D**	**0x04**	**0xF1**	*Echo cmd:* **0x09** *Error code:* **0x00**		
Reply field 2 Current GPS Message Format					**0x09**	**0x81**	Desc count: **0x02** 1st Descriptor: **0x03** Data rate: **0x0004** 2nd Descriptor:**0x05** Datarate: **0x0004**	**0x8D**	**0xFE**

Copy-Paste version of the commands:
Use New Settings: "7565 0C0A 0A09 0102 0300 0405 0004 1685"
Read Current Settings: "7565 0C04 0409 0200 F9F6"

NAV Message Format (0x0C, 0x0A)

Description	Set, read, or save the format of the NAV message packet. This function sets the format for the NAV MIP data packet when in standard mode. The resulting message will maintain the order of descriptors sent in the command. The command has a function selector and a descriptor array as parameters.		
Notes	*Possible function selector values:* 0x01 – Use new settings 0x02 – Read back current settings. 0x03 – Save current settings as startup settings 0x04 – Load saved startup settings 0x05 – Reset to factory default settings The rate decimation field is calculated as follows for NAV messages: *Data Rate = 100Hz / Rate Decimation* The GX3-45 checks that all descriptors are valid prior to executing this command. If any of the descriptors are invalid for the NAV data descriptor set, a NACK will be returned and the message format will be unchanged. The descriptor array only needs to be provided if the function selector is = 1 (Use new settings). For all other functions it may be empty (Number of Descriptors = 0).		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	4 + 3*N	0x0A	U8 - Function Selector U8 – Number of Descriptors (N), N*(U8 – Descriptor, U16 – Rate Decimation)
Reply field 1 ACK/NACK	0x04	0xF1	U8 – echo the command descriptor U8 – error code (0:ACK, not 0:NACK)
Reply field 2 (function = 2)	3 + 3*N	0x82	U8 – Number of Descriptors (N), N*(U8 – Descriptor, U16 – Rate Decimation)

Examples	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command NAV Message Format (use new settings)	**0x75**	**0x65**	**0x0C**	**0x0A**	**0x0A**	**0x0A**	Function: **0x01** Desc count: **0x02** 1st Descriptor: **0x01** Data rate: **0x0001** 2nd Descriptor:**0x02**	**0x0C**	**0x6A**

							Data rate: **0x0001**		
Reply ACK/NACK	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x0A** *Error code:* **0x00**	**0xE9**	**0xBE**
Command NAV Message Format (read back current settings)	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0x0A**	Function: **0x02** Desc count: **0x00**	**0xFA**	**0xF9**
Reply field 1 ACK/NACK	**0x75**	**0x65**	**0x0C**	**0x0D**	**0x04**	**0xF1**	*Echo cmd:* **0x0A** *Error code:* **0x00**		
Reply field 2 Current NAV Message Format					**0x09**	**0x82**	Desc count: **0x02** 1st Descriptor: **0x01** Data rate: **0x0001** 2nd Descriptor:**0x02** Datarate: **0x0001**	**0x84**	**0xED**

Copy-Paste version of the commands:
Use New Settings: "7565 0C0A 0A0A 0102 0100 0102 0C6A"
Read Current Settings: "7565 0C04 040A 0200 FAF9"

Enable/Disable Continuous Data Stream (0x0C, 0x11)

Description	Control the streaming of AHRS, GPS, and NAV data. If disabled, the data from the selected device is not continuously transmitted. Upon enabling, the most current data will be transmitted (i.e. no stale data is transmitted.) The default for the device is all streams enabled. For all functions except 0x01 (use new setting), the new enable flag value is ignored.		
Notes	Possible function selector values: 0x01 – Apply new settings 0x02 – Read back current settings. 0x03 – Save current settings as startup settings 0x04 – Load saved startup settings 0x05 – Load factory default settings *The device selector can be:* 0x01 – AHRS 0x02 – GPS 0x03 – NAV *The enable flag can be either:* 0x00 – disable the selected stream. 0x01 – enable the selected stream. *(default)*		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x05	0x11	U8 – Function Selector U8 – Device Selector U8 – New Enable Flag
Reply field 1 ACK/NACK	0x04	0xF1	U8 – echo the command descriptor U8 – error code (0:ACK, not 0:NACK)
Reply field 2 (function = 2)	0x04	0x85	U8 – Device Selector U8 – Current Device Enable Flag

Examples	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command AHRS Stream ON	**0x75**	**0x65**	**0x0C**	**0x05**	**0x05**	**0x11**	Function(Apply):**0x01** Device (AHRS): **0x01** Stream (ON): **0x01**	**0x04**	**0x1A**
Command AHRS Stream OFF	**0x75**	**0x65**	**0x0C**	**0x05**	**0x05**	**0x11**	Function(Apply):**0x01** Device (AHRS): **0x01** Stream (OFF): **0x00**	**0x03**	**0x19**

Command *GPS Stream ON*	**0x75**	**0x65**	**0x0C**	**0x05**	**0x05**	**0x11**	Function(Apply):**0x01** Device (GPS): **0x02** Stream (ON): **0x01**	**0x05**	**0x1C**
Command *GPS Stream OFF*	**0x75**	**0x65**	**0x0C**	**0x05**	**0x05**	**0x11**	Function(Apply):**0x01** Device (GPS): **0x02** Stream (OFF): **0x00**	**0x04**	**0x1B**
Command *NAV Stream ON*	**0x75**	**0x65**	**0x0C**	**0x05**	**0x05**	**0x11**	Function(Apply):**0x01** Device (NAV): **0x03** Stream (ON): **0x01**	**0x06**	**0x1E**
Command *NAV Stream OFF*	**0x75**	**0x65**	**0x0C**	**0x05**	**0x05**	**0x11**	Function(Apply):**0x01** Device (NAV): **0x03** Stream (OFF): **0x00**	**0x05**	**0x1D**
Reply *ACK/NACK*	**0x75**	**0x65**	**0x0C**	**0x05**	**0x05**	**0xF1**	*Echo cmd:* **0x11** *Error code:* **0x00**	**0xEF**	**0xCA**

Copy-Paste version of the 1st command: "7565 0C05 0511 0101 0104 1A"

Device Startup Settings (0x0C, 0x30)

<table>
<tr><td rowspan="2">Description</td><td colspan="9">Save, Load, or Reset to Default the values for all device settings. This is the equivalent of sending the same function selector to each of the following settings commands:

AHRS Message Format
GPS Message Format
NAV Message Format

Enable/Disable Continuous Data Stream
UART BAUD Rate
Communications Mode
AHRS Signal Conditioning Settings

Vehicle Dynamics Mode
Sensor to Vehicle Rotation
Sensor to Vehicle Offset
Antenna Offset
Bias Estimation Control
GPS Source Control
Heading Update Control
Auto-Initialization Control
Accel White Noise
Gyro White Noise
Gyro Bias Model</td></tr>
<tr></tr>
<tr><td>Notes</td><td colspan="9">Possible function selector values:

0x03 – Save current settings as startup settings
0x04 – Load saved startup settings
0x05 – Load factory default settings</td></tr>
<tr><td>Field Format</td><td colspan="2">Field Length</td><td colspan="3">Field Descriptor</td><td colspan="4">Field Data</td></tr>
<tr><td>Command</td><td colspan="2">0x02</td><td colspan="3">0x30</td><td colspan="4">U8 –Function Selector</td></tr>
<tr><td>Reply
ACK/NACK</td><td colspan="2">0x04</td><td colspan="3">0xF1</td><td colspan="4">U8 – echo the command byte
U8 – error code (0:ACK, not 0:NACK)</td></tr>
<tr><td rowspan="2">Example</td><td colspan="4">MIP Packet Header</td><td colspan="3">Command/Reply Fields</td><td colspan="2">Checksum</td></tr>
<tr><td>Sync1</td><td>Sync2</td><td>Desc Set</td><td>Payload Length</td><td>Field Length</td><td>Field Desc.</td><td>Field Data</td><td>MSB</td><td>LSB</td></tr>
<tr><td>Command
Startup Settings
(Save All)</td><td>0x75</td><td>0x65</td><td>0x0C</td><td>0x03</td><td>0x03</td><td>0x30</td><td>Fctn(Save):0x03</td><td>0x1F</td><td>0x45</td></tr>
</table>

Reply ACK/NACK	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x30** *Error code:* **0x00**	**0x0F**	**0x0A**

Copy-Paste version of the command: "7565 0C03 0330 031F 45"

AHRS Signal Conditioning Settings (0x0C, 0x35)

Description	Set, read, or save the AHRS signal conditioning parameters. This function sets the AHRS signal conditioning parameters for all communications and streaming modes. For all functions except 0x01 (use new settings), the new parameter values are ignored.
Notes	*Possible function selector values:* 0x01 – Apply new settings 0x02 – Read back current settings. 0x03 – Save current settings as startup settings 0x04 – Load saved startup settings 0x05 – Load factory default settings *Possible Orientation Calculation Decimation values:* 0x0002 to 0x03E8 (2 to 1000): This value divided into 1000 will determine the rate at which coning & sculling integration, and orientation calculations are made (including Matrix, Euler, and Quaternion). For example, a value of 10 results in 1000/10 = 100Hz calculation rate. **Always o*verwritten to"0x000A" (10) on the GX3-45.*** *Possible Data Conditioning Flags:* 0x0001 – Enables Orientation Calculation (Matrix/Euler). **Always o*verwritten to"1"on the GX3-45.*** 0x0002 – Enables Coning & Sculling. *Default is "1".* **Always o*verwritten to"1"on the GX3-45.*** 0x0040 – Enables finite size correction. *Default is "0"* 0x0100 – Disables Magnetometer. *Default is "0"* 0x0400 – Disables "North" compensation. *Default is "0"* 0x0800 – Disables "Up" compensation. *Default is "0"* 0x1000 – Enables Quaternion calculation. **Always o*verwritten to"1"on the GX3-45.*** *Possible Gyro/Accel and Mag Filter Width values:* 0x01 to 0x20 (1 to 32): This value divided into 1000 determines the bandwidth of the adjustable filter. See the section on "AHRS Filtering" for more information. *Default is 15 for Accel/Gyro, 17 for Mag.*

	Possible Up and North compensation values: 0x0001 to 0x03E8 (1 to 1000): This value represents how quickly (in seconds) the gravitational /magnetometer vectors correct the inertial attitude/yaw orientation results. *Default is 10 (seconds) for both values.* *Possible Mag Power/Bandwidth values:* 0: High bandwidth, highest power consumption 1: Bandwidth is coupled to Data Rate; low power consumption. *Default is "1"*

Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x10	0x35	U8 – Function Selector U16 – New Orientation Calc Decimation Value U16 – New Data Conditioning Flags U8 – New Accel/Gyro Filter Width U8 – New Mag Filter Width U16 – New Up Compensation U16 – New North Compensation U8 – New Mag Bandwidth/Power U16 - *Reserved*
Reply field 1 ACK/NACK	0x04	0xF1	U8 – echo the command descriptor U8 – error code (0:ACK, not 0:NACK)
Reply field 2 (function = 2)	0x0F	0x86	U16 – Current Orientation Decimation Value U16 – Current Data Conditioning Flags U8 – Current Accel/Gyro Filter Width U8 – Current Mag Filter Width U16 – Current Up Compensation U16 – Current North Compensation U8 – Current Mag Bandwidth/Power U16 - *Reserved*

Example	*MIP Packet Header*				*Fields*			*Checksum*	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command GPS Settings	**0x75**	**0x65**	**0x0C**	**0x10**	**0x10**	**0x35**	*Fctn (Apply):* **0x01** *Calc Decimation (100Hz):* **0x000A** *Flags(def):***0x0003** *Acc/GyroFilt:***0x0E** *Mag Filter:* **0x11** *Up Comp:* **0x000A** *N Comp:* **0x000A** *Mag BW:***0x01** *Reserved:***0x0000**	**0x7D**	**0xB7**

Reply *ACK/NACK*	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x35** *Error code:* **0x00**	**0x14**	**0x14**

Copy-Paste version of the command: "7565 0C10 1035 0100 0A00 030E 1100 0A00 0A01 0000 7DB7"

UART BAUD Rate (0x0C, 0x40)

Description	Change, read, or save the BAUD rate of the main communication channel (UART1). For all functions except 0x01 (use new settings), the new BAUD rate value is ignored.		
Notes	*Possible function selector values:* 0x01 – Use new settings 0x02 – Read back current settings. 0x03 – Save current settings as startup settings 0x04 – Load saved startup settings 0x05 – Reset to factory default settings *Supported BAUD rates are:* 9600, 19200, 115200 *(default)*, 230400, 460800, 921600		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x07	0x40	U8 – Function Selector U32 –New BAUD rate
Reply field 1 ACK/NACK	0x04	0xF1	U8 – echo the command descriptor U8 – error code (0:ACK, not 0:NACK)
Reply field 2 (function = 2)	0x06	0x87	U32 – Current BAUD rate

Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command Set BAUD Rate Command	**0x75**	**0x65**	**0x0C**	**0x07**	**0x07**	**0x40**	*Fctn(USE):***0x01** *BAUD (115200)*: **0x0001C200**	**0xF8**	**0xDA**
Reply ACK/NACK	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x40** *Error code:* **0x00**	**0x1F**	**0x2A**

Copy-Paste version of the command: "7565 0C07 0740 0100 01C2 00F8 DA"

Device Status (0x0C, 0x64)

<table>
<tr><td>Description</td><td colspan="3">Get the device-specific status for the 3DM-GX3-45</td></tr>
<tr><td>Notes</td><td colspan="3">Reply has two fields: “ACK/NACK” and “Device Status Field”. The device status field may be one of two selectable formats – basic and diagnostic.

The reply data for this command is device specific. The reply is specified by two parameters in the command. The first parameter is the model number (which for the 3DM-GX3-45 is always = 6228 (0x1854)). That is followed by a status selector byte which determines the type of data structure returned. In the case of the 3DM-GX3-45, there are two selector values – one to return a basic status structure and a second to return an extensive diagnostics status structure. A list of available values for the selector values and specific fields in the data structure are as follows:

Possible Status Selector Values:

0x01 – Basic Status Structure

Possible Communication Mode Values:

0x01 – Standard MIP Mode
0x02 – Advanced AHRS Direct Mode
0x03 – Advanced GPS Direct Mode

Possible Communication Device Values:

0x01 - Com1 (Serial)
0x02 - USB

Possible Settings Flags:

0x00000001 – AHRS Continuous Stream Enabled
0x00000100 – GPS Continuous Stream Enabled
0x00010000 – NAV Continuous Stream Enabled

Possible Com1 State, USB State, GPS Driver State, GPS Port State, AHRS Driver State, AHRS Port State Values:

0x00 – Not Initialized
0x01 – Initialized</td></tr>
<tr><td>Field Format</td><td>Field Length</td><td>Field Descriptor</td><td>Field Data</td></tr>
<tr><td>Command</td><td>0x02</td><td>0x64</td><td>U16-Device Model Number: set = 6228 (0x1854)
U8-Status Selector</td></tr>
<tr><td>Reply field 1</td><td>0x04</td><td>0xF1</td><td>U8 – echo the command byte</td></tr>
</table>

ACK/NACK Field			U8 – error code (0:ACK, not 0:NACK)			
	0x11	0x90	*Binary Offset*	*Description*	*Data Type*	*Units*
			0	Echo of the Device Model Number	U16	N/A
			2	Echo of the selector byte	U8	N/A
			3	Communication Mode	U8	See Notes
			4	Communication Device	U8	See Notes
			5	Settings Flags	U32	See Notes
			9	Com 1 State	U16	See Notes
			11	Com1 Baudrate	U32	Baud

Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	MSB	LSB
Command Get Device Status (return Basic Status structure: selector = 1)	**0x75**	**0x65**	**0x0C**	**0x05**	**0x05**	**0x64**	Model # (6228): **0x1854** Status Selector (basic status): **0x01**	**0xC1**	**0x51**
Reply field 1 ACK/NACK	**0x75**	**0x65**	**0x0C**	**0x15**	**0x04**	**0xF1**	*Echo cmd:* **0x64** *Error code:* **0x00**		
Reply field 2 Device Status (Basic Status structure)					**0x11**	**0x90**	Echo Model#: **0x1854** Echo Selector: **0x01** U8: U8: U32: U16: U32:	**0x##**	**0x##**

Copy-Paste version of the command: “7565 0C05 0564 1852 01BF 4D”

Navigation Filter Commands

The Navigation Filter command set is specific to MicroStrain Inertial Navigation sensors.

Reset Filter (0x0D, 0x01)

Description	Reset the filter to the initialize state.		
Notes	If the auto-initialization feature is disabled, the initial attitude or heading must be set in order to enter the run state after a reset.		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x02	0x01	N/A
Reply ACK/NACK	0x04	0xF1	U8 – echo the command byte U8 – error code (0:ACK, non-zero: NACK)

Example	*MIP Packet Header*				*Fields*			*Checksum*	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command	**0x75**	**0x65**	**0x0D**	**0x02**	**0x02**	**0x01**		**0xEC**	**0xF6**
Reply ACK/NACK	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x01** *Error code:* **0x00**	**0xE1**	**0xB2**

Copy-Paste version of the command: "7565 0D02 0201 ECF6"

Set Initial Attitude (0x0D, 0x02)

Description	Set the initial attitude.		
Notes	This command can only be issued in the "INIT" state and should be used with a good estimate of the vehicle attitude. The Euler Angles are the sensor body frame with respect to the local NED frame.		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x0E	0x02	Float – Roll (radians) Float – Pitch (radians) Float – Heading (radians)
Reply ACK/NACK	0x04	0xF1	U8 – echo the command byte U8 – error code (0:ACK, not 0:NACK)

Example	*MIP Packet Header*				*Fields*			*Checksum*	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command	**0x75**	**0x65**	**0x0D**	**0E**	**0E**	**02**	*Roll:* 0x00000000 (**0.0f**) *Pitch:* 0x00000000 (**0.0f**) *Heading:* 0x00000000 (**0.0f**)	**0x05**	**0x6F**
Reply ACK/NACK	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x02** *Error code:* **0x00**	**0xE2**	**0xB4**

Copy-Paste version of the command: "7565 0D0E 0E02 0000 0000 0000 0000 0000 0000 0000 0000 056F"

Set Initial Heading (0x0D, 0x03)

Description	Set the initial heading angle.		
Notes	This command can only be issued in the "INIT" state and should be used with a good estimation of Heading. The GX3-45 will use this value in conjunction with the output of the accelerometers to determine the initial attitude estimate. The Euler Angles are the sensor body frame with respect to the local NED frame.		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x06	0x03	Float – Heading (radians)
Reply ACK/NACK	0x04	0xF1	U8 – echo the command byte U8 – error code (0:ACK, not 0:NACK)

Example	*MIP Packet Header*				*Fields*			*Checksum*	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command	**0x75**	**0x65**	**0x0D**	**0x06**	**0x06**	**0x03**	*Heading:* 0x00000000 (**0.0f)**	**0xF6**	**0xE4**
Reply ACK/NACK	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x03** *Error code:* **0x00**	**0xE3**	**0xB6**

Copy-Paste version of the command: "7565 0D06 0603 0000 0000 F6E4"

Set Initial Attitude From AHRS (0x0D, 0x04)

Description	Set the initial attitude using the embedded AHRS.		
Notes	This command can only be issued in the "INIT" state. The GX3-45 will use the on-board AHRS unit to initialize the attitude. The user must supply a declination angle for the local magnet field conditions ***Special Note:*** The AHRS uses a magnetometer to determine heading. In the presence of significant magnetic interference, this value can be wildly off, causing the filter to go unstable.		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x06	0x04	Float – Declination Angle (radians)
Reply ACK/NACK	0x04	0xF1	U8 – echo the command byte U8 – error code (0:ACK, not 0:NACK)

Example	*MIP Packet Header*				*Fields*			*Checksum*	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command	**0x75**	**0x65**	**0x0D**	**0x06**	**0x06**	**0x04**	*Declination:* 0x00000000 (**0.0f**)	**0xF7**	**0xE9**
Reply ACK/NACK	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x04** *Error code:* **0x00**	**0xE4**	**0xB8**

Copy-Paste version of the command: "7565 0D06 0604 0000 0000 F7E9"

Vehicle Dynamics Mode (0x0D, 0x10)

Description	Set, read, or save the vehicle dynamics mode. For all functions except 0x01 (use new settings), the new dynamics mode value is ignored.									
Notes	*Possible function selector values:* 0x01 – Apply new settings 0x02 – Read back current settings. 0x03 – Save current settings as startup settings 0x04 – Load saved startup settings 0x05 – Load factory default settings *Possible Modes:* **Mode** / **Use** / **Altitude Limits** / **Velocity Limits** 0x01 – Portable (default) / Applications with low acceleration / 12,000 m / Horizontal - 310 m/s, Vertical - 50 m/s 0x02 – Automotive / Low vertical acceleration, wheeled-vehicle dynamics / 6000 m / Horizontal - 84 m/s, Vertical - 15 m/s 0x03 – Airborne / Typical airborne application / 50,000 m / Horizontal - 250 m/s, Vertical - 100 m/s									
Field Format	*Field Length*	*Field Descriptor*	*Field Data*							
Command	0x04	0x10	U8 – Function Selector U8 – New Dynamics Mode							
Reply ACK/NACK	0x04	0xF1	U8 – echo the command descriptor U8 – error code (0:ACK, not 0:NACK)							
Reply field 2 (function = 2)	3	0x80	U8 –Current Dynamics Mode							
Example	*MIP Packet Header*				*Fields*			*Checksum*		
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*	
Command Dynamics Mode	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0x10**	*Fctn (Apply):* **0x01** *Mode (Portable):* **0x01**	**0x01**	**0x10**	
Reply ACK/NACK	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x10** *Error code:* **0x00**	**0xF0**	**0xD0**	

Possible Modes:

Mode	Use	Altitude Limits	Velocity Limits
0x01 – Portable (default)	Applications with low acceleration	12,000 m	Horizontal - 310 m/s Vertical - 50 m/s
0x02 – Automotive	Low vertical acceleration, wheeled-vehicle dynamics	6000 m	Horizontal - 84 m/s Vertical - 15 m/s
0x03 – Airborne	Typical airborne application	50,000 m	Horizontal - 250 m/s Vertical - 100 m/s

Copy-Paste version of the command: "7565 0C04 0410 0101 0110"

Sensor to Vehicle Frame Transformation (0x0D, 0x11)

<table>
<tr><td>Description</td><td colspan="3">Set the sensor to vehicle frame transformation matrix using Roll, Pitch, and Yaw Euler angles. These angles define the rotation from the sensor body frame to the fixed vehicle frame. Please reference the GX3-45 Theory of Operation for more information.</td></tr>
<tr><td>Notes</td><td colspan="3">Possible function selector values:

0x01 – Use new settings
0x02 – Read back current settings.
0x03 – Save current settings as startup settings
0x04 – Load saved startup settings
0x05 – Reset to factory default settings

This transformation affects the following output quantities:

NAV:
Estimated Orientation, Quaternion
Estimated Orientation, Matrix
Estimated Orientation, Euler Angles
Estimated Linear Acceleration
Estimated Angular Rate
Estimated Gravity Vector</td></tr>
<tr><td>Field Format</td><td>Field Length</td><td>Field Descriptor</td><td>Field Data</td></tr>
<tr><td>Command</td><td>0x0F</td><td>0x11</td><td>U8 – Function Selector
Float – Roll Angle (radians)
Float – Pitch Angle (radians)
Float – Yaw Angle (radians)</td></tr>
<tr><td>Reply
ACK/NACK</td><td>0x04</td><td>0xF1</td><td>U8 – echo the command descriptor
U8 – error code (0:ACK, not 0:NACK)</td></tr>
<tr><td>Reply field 2
(function = 2)</td><td>0x0E</td><td>0x81</td><td>Float – Roll Angle (radians)
Float – Pitch Angle (radians)
Float – Yaw Angle (radians)</td></tr>
</table>

<table>
<tr><td rowspan="2">Example</td><td colspan="4">MIP Packet Header</td><td colspan="3">Fields</td><td colspan="2">Checksum</td></tr>
<tr><td>Sync1</td><td>Sync2</td><td>Desc Set</td><td>Payload Length</td><td>Field Length</td><td>Field Desc.</td><td>Field Data</td><td>MSB</td><td>LSB</td></tr>
<tr><td>Command</td><td>0x75</td><td>0x65</td><td>0x0D</td><td>0x0F</td><td>0x0F</td><td>0x11</td><td>Fctn (Apply): 0x01
Roll: 0x00000000 (0.0f)
Pitch: 0x00000000</td><td>0x17</td><td>0x72</td></tr>
</table>

							(**0.0f)** *Yaw:* 0x00000000 (**0.0f)**		
Reply *ACK/NACK*	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x11** *Error code:* **0x00**	**0xF1**	**0xD2**

Copy-Paste version of the command: "7565 0D0F 0F11 0100 0000 0000 0000 0000 0000 0017 72"

Sensor to Vehicle Frame Offset (0x0D, 0x12)

Description	Set the sensor to vehicle frame offset, expressed in the sensor frame. Please reference the GX3-45 Theory of Operation for more information.								
Notes	*Possible function selector values:* 0x01 – Use new settings 0x02 – Read back current settings. 0x03 – Save current settings as startup settings 0x04 – Load saved startup settings 0x05 – Reset to factory default settings This offset affects the following output quantities: Estimated LLH Position The maximum value for any axis is +-100 meters.								
Field Format	*Field Length*			*Field Descriptor*		*Field Data*			
Command	0x0F			0x12		U8 – Function Selector Float – X (meters, sensor body frame) Float – Y (meters, sensor body frame) Float – Z (meters, sensor body frame)			
Reply ACK/NACK	0x04			0xF1		U8 – echo the command descriptor U8 – error code (0:ACK, not 0:NACK)			
Reply field 2 (function = 2)	0x0E			0x82		Float – X (meters, sensor body frame) Float – Y (meters, sensor body frame) Float – Z (meters, sensor body frame)			
Example	*MIP Packet Header*				*Fields*			*Checksum*	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command	**0x75**	**0x65**	**0x0D**	**0x0F**	**0x0F**	**0x12**	*Fctn (Apply):* **0x01** *X:* 0x00000000 (**0.0f)** *Y:* 0x00000000 (**0.0f)** *Z:* 0x00000000 (**0.0f)**	**0x18**	**0x80**
Reply ACK/NACK	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x12** *Error code:* **0x00**	**0xF2**	**0xD4**

Copy-Paste version of the command: "7565 0D0F 0F12 0100 0000 0000 0000 0000 0000 0018 80"

Antenna Offset (0x0D, 0x13)

Description	Set the sensor to antenna offset, expressed in the sensor frame.		
Notes	*Possible function selector values:* 0x01 – Use new settings 0x02 – Read back current settings. 0x03 – Save current settings as startup settings 0x04 – Load saved startup settings 0x05 – Reset to factory default settings This offset affects the following output quantities: Estimated LLH Position A maximum value of +-10 meters for any axis is recommended.		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x0F	0x13	U8 – Function Selector Float – X (meters, sensor body frame) Float – Y (meters, sensor body frame) Float – Z (meters, sensor body frame)
Reply ACK/NACK	0x04	0xF1	U8 – echo the command descriptor U8 – error code (0:ACK, not 0:NACK)
Reply field 2 (function = 2)	0x0E	0x83	Float – X (meters, sensor body frame) Float – Y (meters, sensor body frame) Float – Z (meters, sensor body frame)

Example	*MIP Packet Header*				*Fields*			*Checksum*	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command	**0x75**	**0x65**	**0x0D**	**0x0F**	**0x0F**	**0x13**	*Fctn (Apply):* **0x01** *X:* 0x00000000 (**0.0f)** *Y:* 0x00000000 (**0.0f)** *Z:* 0x00000000 (**0.0f)**	**0x19**	**0x8E**
Reply ACK/NACK	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x13** *Error code:* **0x00**	**0xF3**	**0xD6**

Copy-Paste version of the command: "7565 0D0F 0F13 0100 0000 0000 0000 0000 0000 0019 8E"

Bias Estimation Control (0x0D, 0x14)

Description	Control the calculation of sensor biases.		
Notes	*Possible function selector values:* 0x01 – Use new settings 0x02 – Read back current settings. 0x03 – Save current settings as startup settings 0x04 – Load saved startup settings 0x05 – Reset to factory default settings *Possible control bitfield values:* Bit 0 (0x00000001) – Gyro bias estimate (1 – enable, 0 – disable)		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x05	0x14	U8 – Function Selector U16 – Control Bitfield
Reply ACK/NACK	0x04	0xF1	U8 – echo the command descriptor U8 – error code (0:ACK, not 0:NACK)
Reply field 2 (function = 2)	0x04	0x84	U16 – Control Bitfield

Example	*MIP Packet Header*				*Fields*			*Checksum*	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command	**0x75**	**0x65**	**0x0D**	**0x05**	**0x05**	**0x14**	*Fctn (Apply):* **0x01** *X:* 0x0001 (**Enable Gyro Bias Estimation)**	**0x07**	**0x2B**
Reply ACK/NACK	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x14** *Error code:* **0x00**	**0xF4**	**0xD8**

Copy-Paste version of the command: "7565 0D05 0514 0100 0107 2B"

GPS Source Control (0x0D, 0x15)

Description	Control the source of GPS information used to update the Kalman Filter.		
Notes	*Possible function selector values:* 0x01 – Use new settings 0x02 – Read back current settings. 0x03 – Save current settings as startup settings 0x04 – Load saved startup settings 0x05 – Reset to factory default settings *Possible GPS Source values:* 0x01 – Internal GPS 0x02 – External GPS (Requires user to provide GPS information via the "External GPS Update" command) Changing the GPS source while the sensor is in the "running" state will temporarily place it back in the "init" state until the new source of GPS data is received.		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x04	0x15	U8 – Function Selector U8 – GPS Source
Reply ACK/NACK	0x04	0xF1	U8 – echo the command descriptor U8 – error code (0:ACK, not 0:NACK)
Reply field 2 (function = 2)	0x03	0x86	U8 – GPS Source

Example	*MIP Packet Header*				*Fields*			*Checksum*	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0x15**	*Fctn (Apply):* **0x01** *Source:* 0x02 (**External GPS)**	**0x07**	**0x20**
Reply ACK/NACK	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x15** *Error code:* **0x00**	**0xF5**	**0xDA**

Copy-Paste version of the command: "7565 0D04 0415 0102 0720"

External GPS Update (0x0D, 0x16)

Description	Trigger a filter update step using external GPS information.		
Notes	GPS source control must be set to external for this command to update the filter; it will be ignored/NACK'd otherwise.		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x48	0x16	Double – GPS Time of Week (seconds) U16 – GPS Week Number Double – Latitude (deg) Double – Longitude (deg) Double – Altitude above WGS84 Ellipsoid (m) Float – North Velocity (m/s) Float – East Velocity (m/s) Float – Down Velocity (m/s) Float – North Position Uncertainty (m, 1-sigma) Float – East Position Uncertainty (m, 1-sigma) Float – Down Position Uncertainty (m, 1-sigma) Float – North Velocity Uncertainty (m/s, 1-sigma) Float – East Velocity Uncertainty (m/s, 1-sigma) Float – Down Velocity Uncertainty (m/s, 1-sigma)
Reply ACK/NACK	0x04	0xF1	U8 – echo the command byte U8 – error code (0:ACK, not 0:NACK)

Example	*MIP Packet Header*				*Fields*			*Checksum*	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command	**0x75**	**0x65**	**0x0D**	**48**	**48**	**16**	*GPS Tow:* **0.0d** *GPS Week: 0x0000* *Latitude:* **0.0d** *Longitude:* **0.0d** *Height:* **0.0d** *Vel North:* **0.0f** *Vel East:* **0.0f** *Vel Down:* **0.0f** *Pos Sigma (N)* **0.0f** *Pos Sigma (E)* **0.0f** *Pos Sigma (D)* **0.0f** *Vel Sigma (N)* **0.0f** *Vel Sigma (E)* **0.0f** *Vel Sigma (D)* **0.0f**	**0xXX**	**0xXX**
Reply ACK/NACK	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x16** *Error code:* **0x00**	**0xF6**	**0xDC**

Copy-Paste version of the command: N/A

External Heading Update (0x0D, 0x17)

Description	Trigger a filter update step using external heading information ***The heading must be the sensor frame with respect to the NED frame.***								
Notes	Angle uncertainties of 0.0 will be NACK'd. Possible Heading Type Commands: 0x01 – True Heading 0x02 – Magnetic Heading* *Note: if the GPS is unavailable, magnetic heading updates will be NACK'd. This happens because the on-board magnetic model cannot be run without GPS updates.								
Field Format	*Field Length*	*Field Descriptor*	*Field Data*						
Command	0x0B	0x16	Float – Heading Angle (radians, true north, +- PI) Float – Heading Angle Uncertainty (radians, 1-sigma) U8 – Heading type (1 – true, 2 – magnetic)						
Reply ACK/NACK	0x04	0xF1	U8 – echo the command byte U8 – error code (0:ACK, not 0:NACK)						
Example	*MIP Packet Header*				*Fields*			*Checksum*	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command	**0x75**	**0x65**	**0x0D**	**0B**	**0B**	**17**	*Angle:* **0.0f** *Angle Sigma:* **0.01f** Heading Type: **0x01** (True)	**0xXX**	**0xXX**
Reply ACK/NACK	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x17** *Error code:* **0x00**	**0xF7**	**0xDE**

Copy-Paste version of the command: N/A

Heading Update Control (0x0D, 0x18)

Description	Select the source for heading updates to the Kalman Filter.		
Notes	*Possible function selector values:* 0x01 – Use new settings 0x02 – Read back current settings. 0x03 – Save current settings as startup settings 0x04 – Load saved startup settings 0x05 – Reset to factory default settings *Possible Enable Flag values:* 0x00 – Disable Heading Updates 0x01 – Use the Magnetometer for Heading Updates* 0x02 – Use the Internal GPS Velocity Vector for Heading Updates** 0x03 – Use external heading updates *The magnetometer inclination angle (dip angle) is calculated and tested against the World Magnetic Model value. When an error of 30 degrees or more is detected, the heading is marked as invalid and is not used by the filter. **To use the Internal GPS velocity vector for heading updates, the target application must have no (or minimal) side-slip; this is true in most ground vehicle applications. *Additionally, when this option is selected, the X-axis must be co-aligned with the direction of travel of the vehicle.*		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x04	0x18	U8 – Function Selector U8 – Enable Flag
Reply ACK/NACK	0x04	0xF1	U8 – echo the command descriptor U8 – error code (0:ACK, not 0:NACK)
Reply field 2 (function = 2)	0x03	0x87	U8 – Enable Flag

Example	*MIP Packet Header*				*Fields*			*Checksum*	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0x18**	*Fctn (Apply):* **0x01** *Enable:* 0x01 (**Enable Mag. Updates)**	**0x09**	**0x28**

Reply *ACK/NACK*	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x18** *Error code:* **0x00**	**0xF8**	**0xE0**

Copy-Paste version of the command: "7565 0D04 0418 0101 0928"

Auto-Initialization Control (0x0D, 0x19)

Description	Enable/Disable automatic initialization upon device startup.		
Notes	*Possible function selector values:* 0x01 – Use new settings 0x02 – Read back current settings. 0x03 – Save current settings as startup settings 0x04 – Load saved startup settings 0x05 – Reset to factory default settings *Possible Enable Flag values:* 0x00 – Disable auto-initialization 0x01 – Enable auto-initialization* *A heading update source must be selected in order for the sensor to auto-initialize. The filter will initialize the roll and pitch angles using the AHRS estimation, heading from the heading update source, and position and velocity from the selected GPS source. Attitude initialization takes approximately 5 seconds; whereas, GPS initialization can be shorter or longer, depending on the state of the GPS during initialization.		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x04	0x19	U8 – Function Selector U8 – Enable Flag
Reply ACK/NACK	0x04	0xF1	U8 – echo the command descriptor U8 – error code (0:ACK, not 0:NACK)
Reply field 2 (function = 2)	0x03	0x88	U8 – Enable Flag

Example	*MIP Packet Header*				*Fields*			*Checksum*	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0x19**	*Fctn (Apply):* **0x01** *Enable:* 0x01 (**Enable auto-initialization)**	**0x0A**	**0x2B**
Reply	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x19**	**0xF9**	**0xE2**

ACK/NACK							*Error code:* **0x00**		

Copy-Paste version of the command: "7565 0D04 0419 0101 0A2B"

Accelerometer White Noise Standard Deviation (0x0D, 0x1A)

Description	Set the expected accelerometer white noise 1-sigma values. This function can be used to tune the filter performance in the target application.		
Notes	*Possible function selector values:* 0x01 – Use new settings 0x02 – Read back current settings. 0x03 – Save current settings as startup settings 0x04 – Load saved startup settings 0x05 – Reset to factory default settings Each of the noise values must be greater than 0.0		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x0F	0x1A	U8 – Function Selector Float – X Accel Noise 1-sigma (meters/second^2) Float – Y Accel Noise 1-sigma (meters/second^2) Float – Z Accel Noise 1-sigma (meters/second^2)
Reply ACK/NACK	0x04	0xF1	U8 – echo the command descriptor U8 – error code (0:ACK, not 0:NACK)
Reply field 2 (function = 2)	0x0E	0x89	Float – X Accel Noise 1-sigma (meters/second^2) Float – Y Accel Noise 1-sigma (meters/second^2) Float – Z Accel Noise 1-sigma (meters/second^2)

Example	*MIP Packet Header*				*Fields*			*Checksum*	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command	**0x75**	**0x65**	**0x0D**	**0x0F**	**0x0F**	**0x1A**	*Fctn (Apply):* **0x01** *X:* (**0.02f**) *Y:* (**0.02f**) *Z:* (**0.02f**)	**0x60**	**0xA3**
Reply ACK/NACK	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x1A** *Error code:* **0x00**	**0xFA**	**0xE4**

Copy-Paste version of the command: "7565 0D0F 0F01 1A01 3CA3 D70A 3CA3 D70A 3CA3 D760 A3"

Gyroscope White Noise Standard Deviation (0x0D, 0x1B)

Description	Set the expected gyroscope white noise 1-sigma values. This function can be used to tune the filter performance in the target application.		
Notes	*Possible function selector values:* 0x01 – Use new settings 0x02 – Read back current settings. 0x03 – Save current settings as startup settings 0x04 – Load saved startup settings 0x05 – Reset to factory default settings Each of the noise values must be greater than 0.0		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x0F	0x1B	U8 – Function Selector Float – X Gyro Noise 1-sigma (rad/second) Float – Y Gyro Noise 1-sigma (rad/second) Float – Z Gyro Noise 1-sigma (rad/second)
Reply ACK/NACK	0x04	0xF1	U8 – echo the command descriptor U8 – error code (0:ACK, not 0:NACK)
Reply field 2 (function = 2)	0x0E	0x8A	Float – X Gyro Noise 1-sigma (rad/second) Float – Y Gyro Noise 1-sigma (rad/second) Float – Z Gyro Noise 1-sigma (rad/second)

Example	*MIP Packet Header*				*Fields*			*Checksum*	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command	**0x75**	**0x65**	**0x0D**	**0x0F**	**0x0F**	**0x1B**	*Fctn (Apply):* **0x01** *X:* (**0.0000539f)** *Y:* (**0.0000539f)** *Z:* (**0.0000539f)**	**0xDE**	**0xE8**
Reply ACK/NACK	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x1B** *Error code:* **0x00**	**0xFB**	**0xE6**

Copy-Paste version of the command: "7565 0D0F 0F1B 013A 0D4B AD3A 0D4B AD3A 0D4B ADDE E8"

Gyroscope Bias Model Parameters (0x0D, 0x1D)

Description	Set the gyroscope bias model parameters.		
Notes	*Possible function selector values:* 0x01 – Use new settings 0x02 – Read back current settings. 0x03 – Save current settings as startup settings 0x04 – Load saved startup settings 0x05 – Reset to factory default settings Each of the noise values must be greater than 0.0		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x1B	0x1D	U8 – Function Selector Float – X Gyro Bias Beta (1/second) Float – Y Gyro Bias Beta (1/second) Float – Z Gyro Bias Beta (1/second) Float – X Gyro Bias Noise 1-sigma (rad /second) Float – Y Gyro Bias Noise 1-sigma (rad /second) Float – Z Gyro Bias Noise 1-sigma (rad /second)
Reply ACK/NACK	0x04	0xF1	U8 – echo the command descriptor U8 – error code (0:ACK, not 0:NACK)
Reply field 2 (function = 2)	0x1A	0x8C	Float – X Gyro Bias Beta (1/second) Float – Y Gyro Bias Beta (1/second) Float – Z Gyro Bias Beta (1/second) Float – X Gyro Bias Noise 1-sigma (rad /second) Float – Y Gyro Bias Noise 1-sigma (rad /second) Float – Z Gyro Bias Noise 1-sigma (rad /second)

Example	*MIP Packet Header*				*Fields*			*Checksum*	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command	**0x75**	**0x65**	**0x0D**	**0x0F**	**0x1B**	**0x1D**	*Fctn (Apply):* **0x01** *X Beta:* (**0.01f)** *Y Beta:* (**0.01f)** *Z Beta:* (**0.01f)** *X Noise:* (**0.00016f)** *Y Noise:* (**0. 00016f)** *Z Noise:* (**0.**	**0xXX**	**0xXX**

							00016f)		
Reply *ACK/NACK*	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x1D** *Error code:* **0x00**	**0xFD**	**0xEA**

Copy-Paste version of the command: N/A

System Commands

The System Command set provides a set of advanced commands that are specific to devices such as the 3DM-GX3-35 that have multiple intelligent internal sensor blocks. These commands allow special mode such as talking directly to the native protocols of the embedded sensor blocks. For example, with the 3DM-GX3-35, you may switch into a mode that talks directly to the internal u-blox chip or directly to the embedded 3DM-GX3-25 AHRS. This allows you to use code or utilities written specifically for the native u-blox protocols (NMEA or UBX) and 3DM-GX3-25 protocols (original single byte commands or ASPP packet protocol).

Communication Mode (0x7F, 0x10)

Advanced

<table>
<tr><td>Description</td><td colspan="3">Set, read, or save the device communication mode. This will change the communications protocol to and from “NAV” mode to “AHRS Direct” (3DM-GX3-25 protocols) or “GPS Direct” (u-blox5 protocols). This command is always active, even when switched to the direct modes. This command responds with an ACK/NACK just prior to switching to the new protocol. For all functions except 0x01 (use new settings), the new communications mode value is ignored.</td></tr>
<tr><td>Notes</td><td colspan="3">Possible function selector values:

0x01 – Use new settings
0x02 – Read back current settings.
0x03 – Save current settings as startup settings
0x04 – Load saved startup settings
0x05 – Reset to factory default settings

Possible Communications Modes:

<table><tr><th>Value</th><th>Mode</th><th>Protocol(s)</th></tr><tr><td>0x01</td><td>NAV</td><td>3DM-GX3-45 MIP Packet (default)</td></tr><tr><td>0x02</td><td>AHRS Direct</td><td>3DM-GX3-25 Single Byte, ASPP</td></tr><tr><td>0x03</td><td>GPS Direct</td><td>NMEA, UBX</td></tr></table>
IMPORTANT: GPS message settings are automatically reloaded when switching from direct modes back into standard mode.

Note: Switching to and from GPS Direct Mode takes longer than most commands to complete due to the amount of GPS setup data that needs to be stored/retrieved.</td></tr>
<tr><td>Field Format</td><td>Field Length</td><td>Field Descriptor</td><td>Field Data</td></tr>
<tr><td>Command</td><td>0x04</td><td>0x10</td><td>U8 –Function Selector
U8 –New Communications Mode</td></tr>
<tr><td>Reply field 1
ACK/NACK</td><td>0x04</td><td>0xF1</td><td>U8 – echo the command descriptor
U8 – error code (0:ACK, not 0:NACK)</td></tr>
</table>

Reply field 2 (function = 2)	0x03		0x90		U8 –Current Communications Mode				
Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command COM Mode	**0x75**	**0x65**	**0x7F**	**0x04**	**0x04**	**0x10**	*Fctn(USE):***0x01** New mode (*AHRS Direct)*: **0x02**	**0x74**	**0xBD**
Reply ACK/NACK	**0x75**	**0x65**	**0x7F**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x10** *Error code:* **0x00**	**0x62**	**0x7C**

Copy-Paste version of the command: "7565 7F04 0410 0102 74BD"

Error Codes

Error Name	*Error Value*	*Description*
MIP Unknown Command	0x01	The command descriptor is not supported by this device
MIP Invalid Checksum	0x02	An otherwise complete packet has a bad checksum
MIP Invalid Parameter	0x03	One or more parameters in the packet are invalid. This can refer to a value that is outside the allowed range for a command or a value that is not the expected size or type
MIP Command Failed	0x04	Device could not complete the command
MIP Command Timeout	0x05	Device did not complete the command within the expected time

Data Reference

AHRS Data

Scaled Accelerometer Vector (0x80, 0x04)

<table>
<tr><td>Description</td><td colspan="6">Scaled Accelerometer Vector</td></tr>
<tr><td>Notes</td><td colspan="6">This is a vector quantifying the direction and magnitude of the acceleration that the 3DM-GX3 ® is exposed to. This quantity is derived from Raw Accelerometer, but is fully temperature compensated and scaled into physical units of g (1 g = 9.80665 m/sec^2). It is expressed in terms of the 3DM-GX3®'s local coordinate system.</td></tr>
<tr><td rowspan="5">Field Format</td><td>Field Length</td><td>Data Descriptor</td><td colspan="4">Message Data</td></tr>
<tr><td rowspan="4">14 (0x0E)</td><td rowspan="4">0x04</td><td>Binary Offset</td><td>Description</td><td>Data Type</td><td>Units</td></tr>
<tr><td>0</td><td>X Accel</td><td>float</td><td>g</td></tr>
<tr><td>4</td><td>Y Accel</td><td>float</td><td>g</td></tr>
<tr><td>8</td><td>Z Accel</td><td>float</td><td>g</td></tr>
</table>

Scaled Gyro Vector (0x80, 0x05)

<table>
<tr><td>Description</td><td colspan="6">Scaled Gyro Vector</td></tr>
<tr><td>Notes</td><td colspan="6">This is a vector quantifying the rate of rotation (angular rate) of the 3DM-GX3®. This quantity is derived from the Raw Angular Rate quantities, but is fully temperature compensated and scaled into units of radians/second. It is expressed in terms of the 3DM-GX3®'s local coordinate system in units of radians/second.</td></tr>
<tr><td rowspan="5">Field Format</td><td>Field Length</td><td>Data Descriptor</td><td colspan="4">Message Data</td></tr>
<tr><td rowspan="4">14 (0x0E)</td><td rowspan="4">0x05</td><td>Binary Offset</td><td>Description</td><td>Data Type</td><td>Units</td></tr>
<tr><td>0</td><td>X Gyro</td><td>float</td><td>Radians/second</td></tr>
<tr><td>4</td><td>Y Gyro</td><td>float</td><td>Radians/second</td></tr>
<tr><td>8</td><td>Z Gyro</td><td>float</td><td>Radians/second</td></tr>
</table>

Scaled Magnetometer Vector (0x80, 0x06)

Description	Scaled Mag Vector					
Notes	This is a vector which gives the instantaneous magnetometer direction and magnitude. It is fully temperature compensated and is expressed in terms of the 3DM-GX3®'s local coordinate system in units of Gauss.					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
			Binary Offset	*Description*	*Data Type*	*Units*
	14 (0x0E)	0x06	0	X Mag	float	Gauss
			4	Y Mag	float	Gauss
			8	Z Mag	float	Gauss

Delta Theta Vector (0x80, 0x07)

Description	Time integral of angular rate.					
Notes	This is a vector which gives the time integral of Angular Rate where the limits of integration are the beginning and end of the calculation cycle at 100Hz. It is expressed in terms of the 3DM-GX3®'s local coordinate system in units of radians.					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
			Binary Offset	*Description*	*Data Type*	*Units*
	14 (0x0E)	0x07	0	X Delta Theta	float	radians
			4	Y Delta Theta	float	radians
			8	Z Delta Theta	float	radians

Delta Velocity Vector (0x80, 0x08)

Description	Time integral of velocity.					
Notes	This is a vector which gives the time integral of *Accel* where the limits of integration are the beginning and end of the calculation cycle at 100Hz. It is expressed in terms of the 3DM-GX3®'s local coordinate system in units of g*second where g is the standard gravitational constant. To convert Delta Velocity into the more conventional units of m/sec, simply multiply by the standard gravitational constant, 9.80665 m/sec^2					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
			Binary Offset	*Description*	*Data Type*	*Units*
	14 (0x0E)	0x08	0	X Delta Velocity	float	g*seconds
			4	Y Delta Velocity	float	g*seconds
			8	Z Delta Velocity	float	g*seconds

Orientation Matrix (0x80, 0x09)

<table>
<tr><td>Description</td><td colspan="6">3 x 3 Orientation Matrix M</td></tr>
<tr><td>Notes</td><td colspan="6">This is a 9 component coordinate transformation matrix which describes the orientation of the 3DM-GX3® with respect to the fixed earth coordinate system.

$$M = \begin{bmatrix} M_{1,1} & M_{1,2} & M_{1,3} \\ M_{2,1} & M_{2,2} & M_{2,3} \\ M_{3,1} & M_{3,2} & M_{3,3} \end{bmatrix}$$
M satisfies the following equation:

$V\_IL_i = M_{ij} \cdot V\_E_j$

Where: V_IL is a vector expressed in the 3DM-GX3®'s local coordinate system.
V_E is the same vector expressed in the stationary, earth-fixed coordinate system</td></tr>
<tr><td rowspan="11">Field Format</td><td>Field Length</td><td>Data Descriptor</td><td colspan="4">Message Data</td></tr>
<tr><td rowspan="10">38 (0x26)</td><td rowspan="10">0x09</td><td>Binary Offset</td><td>Description</td><td>Data Type</td><td>Units</td></tr>
<tr><td>0</td><td>M_{11}</td><td>float</td><td>n/a</td></tr>
<tr><td>4</td><td>M_{12}</td><td>float</td><td>n/a</td></tr>
<tr><td>8</td><td>M_{13}</td><td>float</td><td>n/a</td></tr>
<tr><td>12</td><td>M_{21}</td><td>float</td><td>n/a</td></tr>
<tr><td>16</td><td>M_{22}</td><td>float</td><td>n/a</td></tr>
<tr><td>20</td><td>M_{23}</td><td>float</td><td>n/a</td></tr>
<tr><td>24</td><td>M_{31}</td><td>float</td><td>n/a</td></tr>
<tr><td>28</td><td>M_{32}</td><td>float</td><td>n/a</td></tr>
<tr><td>32</td><td>M_{33}</td><td>float</td><td>n/a</td></tr>
</table>

Orientation Quaternion (0x80, 0x0A)

<table>
<tr><td>Description</td><td colspan="6">4 x 1 quaternion Q.</td></tr>
<tr><td>Notes</td><td colspan="6">This is a 4 component quaternion which describes the orientation of the 3DM-GX3 with respect to the fixed earth coordinate quaternion.

$$Q = \begin{bmatrix} q0 \\ q1 \\ q2 \\ q3 \end{bmatrix}$$

Q satisfies the following equation:

V_ILi = Q • V_E • Q-1

Where: V_IL is a vector expressed in the 3DM-GX3®'s local coordinate system.
V_E is the same vector expressed in the stationary, earth-fixed coordinate system</td></tr>
<tr><td rowspan="6">Field Format</td><td>Field Length</td><td>Data Descriptor</td><td colspan="4">Message Data</td></tr>
<tr><td rowspan="5">18 (0x12)</td><td rowspan="5">0x0A</td><td>Binary Offset</td><td>Description</td><td>Data Type</td><td>Units</td></tr>
<tr><td>0</td><td>q_0</td><td>float</td><td>n/a</td></tr>
<tr><td>4</td><td>q_1</td><td>float</td><td>n/a</td></tr>
<tr><td>8</td><td>q_2</td><td>float</td><td>n/a</td></tr>
<tr><td>12</td><td>q_3</td><td>float</td><td>n/a</td></tr>
</table>

Euler Angles (0x80, 0x0C)

<table>
<tr><td>Description</td><td colspan="6">Pitch, Roll, and Yaw (aircraft) values</td></tr>
<tr><td>Notes</td><td colspan="6">This is a 3 component vector containing the Roll, Pitch and Yaw angles in radians. It is computed by the AHRS from the orientation matrix M.
$$Euler = \begin{bmatrix} Roll \\ Pitch \\ Yaw \end{bmatrix} \text{(radians)}$$</td></tr>
<tr><td rowspan="5">Field Format</td><td>Field Length</td><td>Data Descriptor</td><td colspan="4">Message Data</td></tr>
<tr><td rowspan="4">14 (0x0E)</td><td rowspan="4">0x0C</td><td>Binary Offset</td><td>Description</td><td>Data Type</td><td>Units</td></tr>
<tr><td>0</td><td>Roll</td><td>float</td><td>radians</td></tr>
<tr><td>4</td><td>Pitch</td><td>float</td><td>radians</td></tr>
<tr><td>8</td><td>Yaw</td><td>float</td><td>radians</td></tr>
</table>

GPS Correlation Timestamp (0x80, 0x12)

<table>
<tr><td>Description</td><td>GPS correlation timestamp.</td></tr>
<tr><td>Notes</td><td>This timestamp has three fields:

Double GPS TOW
U16 GPS Week number
U16 Timestamp flags

Timestamp Status Flags:
Bit0 – PPS Beacon Good If set, GPS PPS signal is present
Bit1 – GPS Time Refresh (toggles with each refresh)
Bit2 – GPS Time Initialized (set with the first GPS Time Refresh)

This timestamp correlates the AHRS packets with the GPS packets. It is identical to the GPS Time record except the flags are defined specifically for the AHRS. When the GPS Time Initialized flag is asserted, the GPS Time and AHRS GPS Timestamp are correlated. This flag is only set once upon the first valid GPS Time record. After that, each time the GPS Time becomes invalid (from a lack of signal) and then valid again (regains signal) the GPS Time Refresh flag will toggle. The GPS Time Initialized will remain set.

The “PPS Beacon Good” flag in the Timestamp flags byte indicates if the PPS beacon coming from the GPS is present. If this flag is not asserted, it means that the AHRS internal clock is being used for the PPS. The fractional portion of the GPS TOW represents the amount of time that has elapsed from the last PPS.

If the GPS loses signal, the GPS and AHRS timestamps become free running and will slowly drift</td></tr>
</table>

<table>
<tr><td></td><td colspan="6">away from each other. If the timestamp clocks have drifted apart, then there will be a jump in the timestamp when the PPS Beacon Good reasserts, reflecting the amount of drift of the clocks.

See the Data Synchronicity section of this manual for more information on timestamps.</td></tr>
<tr><td rowspan="4">Field Format</td><td>Field Length</td><td>Data Descriptor</td><td colspan="4">Message Data</td></tr>
<tr><td rowspan="4">14 (0x0E)</td><td rowspan="4">0x12</td><td>Binary Offset</td><td>Description</td><td>Data Type</td><td>Units</td></tr>
<tr><td>0</td><td>GPS Time of Week</td><td>Double</td><td>Seconds</td></tr>
<tr><td>8</td><td>GPS Week Number</td><td>U16</td><td></td></tr>
<tr><td></td><td>10</td><td>Timestamp Flags</td><td>U16</td><td>See Notes</td></tr>
</table>

GPS Data

LLH Position (0x81, 0x03)

Description	Position Data in the Geodetic Frame					
Notes	Valid Flag Mapping: 0x0001 – Latitude & Longitude Valid 0x0002 – Ellipsoid Height Valid 0x0004 – MSL Height Valid 0x0008 – Horizontal Accuracy Valid 0x0010 – Vertical Accuracy Valid					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
			Binary Offset	*Description*	*Data Type*	*Units*
	44 (0x2C)	0x03	0	Latitude	Double	Decimal Degrees
			8	Longitude	Double	Decimal Degrees
			16	Height above Ellipsoid	Double	Meters
			24	Height above MSL	Double	Meters
			32	Horizontal Accuracy	Float	Meters
			36	Vertical Accuracy	Float	Meters
			40	Valid Flags	U16	See Notes

NED Velocity (0x81, 0x05)

Description	Velocity Data in the North-East-Down Frame					
Notes	Valid Flag Mapping: 0x0001 – NED Velocity Valid 0x0002 – Speed Valid 0x0004 – Ground Speed Valid 0x0008 – Heading Valid 0x0010 – Speed Accuracy Valid 0x0020 – Heading Accuracy Valid					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
			Binary Offset	*Description*	*Data Type*	*Units*
	36(0x24)	0x05	0	North	Float	Meters / Sec
			4	East	Float	Meters / Sec
			8	Down	Float	Meters / Sec
			12	Speed	Float	Meters / Sec
			16	Ground Speed	Float	Meters / Sec
			20	Heading	Float	Decimal Degrees
			24	Speed Accuracy	Float	Meters / Sec
			28	Heading Accuracy	Float	Decimal Degrees
			32	Valid Flags	U16	See Notes

UTC Time (0x81, 0x08)

Description	Coordinated Universal Time Data					
Notes	Valid Flag Mapping: 0x0001 – Date Valid 0x0002 – Time Valid					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
			Binary Offset	*Description*	*Data Type*	*Units*
	15 (0x0F)	0x08	0	Year	U16	Years (1999-2099)
			2	Month	U8	Months (1-12)
			3	Day	U8	Days (1-31)
			4	Hour	U8	Hours (0-23)
			5	Minute	U8	Minutes (0-59)
			6	Second	U8	Seconds (0-59)
			7	Millisecond	U32	Milliseconds
			11	Valid Flags	U16	See Notes

GPS Time (0x81, 0x09)

Description	Global Positioning System Time Data					
Notes	Valid Flag Mapping: 0x0001 – TOW Valid 0x0002 – Week Number Valid					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
			Binary Offset	*Description*	*Data Type*	*Units*
	14 (0x0E)	0x09	0	Time of Week	Double	Seconds
			8	Week Number	U16	
			10	Valid Flags	U16	See Notes

Hardware Status (0x81, 0x0D)

<table>
<tr><td>Description</td><td colspan="6">GPS Hardware Status Information</td></tr>
<tr><td>Notes</td><td colspan="6">Hardware status is only available at 1 Hz. Setting the rate higher than 1 Hz has no effect.

Valid Flag Mapping:

0x0001 – Sensor State Valid
0x0002 – Antenna State Valid
0x0004 – Antenna Power Valid

Possible Sensor State values:

0x00 – Sensor Off
0x01 – Sensor On
0x02 – Sensor State Unknown

Possible Antenna State values:

0x01 – Antenna Init
0x02 – Antenna Short
0x03 – Antenna Open
0x04 – Antenna Good
0x05 – Antenna State Unknown.

Possible Antenna Power values:

0x00 – Antenna Off
0x01 – Antenna On
0x02 – Antenna Power Unknown</td></tr>
<tr><td rowspan="6">Field Format</td><td>Field Length</td><td>Data Descriptor</td><td colspan="4">Message Data</td></tr>
<tr><td rowspan="5">7(0x07)</td><td rowspan="5">0x0D</td><td>Binary Offset</td><td>Description</td><td>Data Type</td><td>Units</td></tr>
<tr><td>0</td><td>Sensor State</td><td>U8</td><td>See Notes</td></tr>
<tr><td>1</td><td>Antenna State</td><td>U8</td><td>See Notes</td></tr>
<tr><td>2</td><td>Antenna Power</td><td>U8</td><td>See Notes</td></tr>
<tr><td>3</td><td>Valid Flags</td><td>U16</td><td>See Notes</td></tr>
</table>

NAV Data

Filter Status (0x82, 0x10)

<table>
<tr><td>Description</td><td colspan="6">Kalman Filter Status</td></tr>
<tr><td>Notes</td><td colspan="6">Possible Filter States:

0x00 – Startup
0x01 – Initialization (see status flags)
0x02 – Running, Solution Valid
0x03 – Running, Solution Error (see status flags)

Possible Dynamics Modes:

0x01 – Portable
0x02 – Automotive
0x03 – Airborne

Possible Status Flags:

Filter State = Initialization:

0x1000 – Attitude not initialized
0x2000 – Position & Velocity not initialized

Filter State = Running:

0x0001 – IMU Unavailable
0x0002 – GPS Unavailable
0x0008 – Matrix Singularity in calculation

0x0010 – Position Covariance High Warning*
0x0020 – Velocity Covariance High Warning*
0x0040 – Attitude Covariance High Warning*
0x0080 – NAN in Solution

*Note: The covariance high warnings are triggered when any axis of the covariance vector exceeds normal operating limits. If more information is required, please inspect the relevant uncertainty packet to determine which axis is in error.</td></tr>
<tr><td rowspan="5">Field Format</td><td>Field Length</td><td>Data Descriptor</td><td colspan="4">Message Data</td></tr>
<tr><td rowspan="4">08 (0x08)</td><td rowspan="4">0x10</td><td>Binary Offset</td><td>Description</td><td>Data Type</td><td>Units</td></tr>
<tr><td>0</td><td>Filter State</td><td>U16</td><td>See Notes</td></tr>
<tr><td>2</td><td>Dynamics Mode</td><td>U16</td><td>See Notes</td></tr>
<tr><td>4</td><td>Status Flags</td><td>U16</td><td>See Notes</td></tr>
</table>

GPS Timestamp (0x82, 0x11)

Description	Kalman Filter Calculated Value Timestamp Data					
Notes	Valid Flag Mapping: 0x0000 – Time Invalid 0x0001 – Time Valid					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
	14 (0x0E)	0x11	*Binary Offset*	*Description*	*Data Type*	*Units*
			0	Time of Week	Double	Seconds
			8	Week Number	U16	
			10	Valid Flags	U16	See Notes

Estimated LLH Position (0x82, 0x01)

Description	INS Estimated Position Data expressed in the Geodetic Frame					
Notes	Valid Flag Mapping: 0x0000 – Latitude, Longitude, & Height are Invalid 0x0001 – Latitude, Longitude, & Height Valid					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
	28 (0x1C)	0x01	*Binary Offset*	*Description*	*Data Type*	*Units*
			0	Latitude	Double	Decimal Degrees
			8	Longitude	Double	Decimal Degrees
			16	Height above Ellipsoid	Double	Meters
			24	Valid Flags	U16	See Notes

Estimated NED Velocity (0x82, 0x02)

<table>
<tr><td>Description</td><td colspan="6">INS Estimated Velocity Data expressed in the Local-Level Frame</td></tr>
<tr><td>Notes</td><td colspan="6">Valid Flag Mapping:

0x0000 – NED Velocity is Invalid
0x0001 – NED Velocity Valid</td></tr>
<tr><td rowspan="6">Field Format</td><td>Field Length</td><td>Data Descriptor</td><td colspan="4">Message Data</td></tr>
<tr><td rowspan="5">16 (0x10)</td><td rowspan="5">0x02</td><td>Binary Offset</td><td>Description</td><td>Data Type</td><td>Units</td></tr>
<tr><td>0</td><td>North</td><td>Float</td><td>Meters / Sec</td></tr>
<tr><td>4</td><td>East</td><td>Float</td><td>Meters / Sec</td></tr>
<tr><td>8</td><td>Down</td><td>Float</td><td>Meters / Sec</td></tr>
<tr><td>12</td><td>Valid Flags</td><td>U16</td><td>See Notes</td></tr>
</table>

Estimated Orientation, Quaternion (0x82, 0x03)

Description	INS Estimated Orientation in quaternion form.					
Notes	This is a 4 component quaternion which describes the orientation of the 3DM-GX3-45 with respect to the fixed earth coordinate quaternion. $$Q = \begin{bmatrix} q0 \\ q1 \\ q2 \\ q3 \end{bmatrix}$$ Q satisfies the following equation: V_E = Q • V_IL • Q-1 *Where: V_IL is a vector expressed in the 3DM-GX3®'s local coordinate system. V_E is the same vector expressed in the stationary, earth-fixed coordinate system* Valid Flag Mapping: 0x0000 – Quaternion is Invalid 0x0001 – Quaternion Valid					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
			Binary Offset	*Description*	*Data Type*	*Units*
	20 (0x14)	0x03	0	q_0	float	*n/a*
			4	q_1 * i	float	*n/a*
			8	q_2 * j	float	*n/a*
			12	q_3 * k	float	*n/a*
			16	Valid Flags	U16	See Notes

Estimated Orientation, Matrix (0x82, 0x04)

<table>
<tr><td>Description</td><td colspan="6">INS Estimated Orientation in Matrix form.</td></tr>
<tr><td>Notes</td><td colspan="6">This is a 9 component coordinate transformation matrix which describes the orientation of the 3DM-GX3® with respect to the fixed earth coordinate system.

$$M = \begin{bmatrix} M_{1,1} & M_{1,2} & M_{1,3} \\ M_{2,1} & M_{2,2} & M_{2,3} \\ M_{3,1} & M_{3,2} & M_{3,3} \end{bmatrix}$$

M satisfies the following equation:

$V\_IL_i = M_{ij} \cdot V\_E_j$

Where: V_IL is a vector expressed in the 3DM-GX3®'s local coordinate system.
V_E is the same vector expressed in the stationary, earth-fixed coordinate system

Valid Flag Mapping:

0x0000 – Orientation Matrix is Invalid
0x0001 – Orientation Matrix Valid</td></tr>
<tr><td rowspan="12">Field Format</td><td>Field Length</td><td>Data Descriptor</td><td colspan="4">Message Data</td></tr>
<tr><td rowspan="11">40 (0x28)</td><td rowspan="11">0x04</td><td>Binary Offset</td><td>Description</td><td>Data Type</td><td>Units</td></tr>
<tr><td>0</td><td>M_{11}</td><td>float</td><td>n/a</td></tr>
<tr><td>4</td><td>M_{12}</td><td>float</td><td>n/a</td></tr>
<tr><td>8</td><td>M_{13}</td><td>float</td><td>n/a</td></tr>
<tr><td>12</td><td>M_{21}</td><td>float</td><td>n/a</td></tr>
<tr><td>16</td><td>M_{22}</td><td>float</td><td>n/a</td></tr>
<tr><td>20</td><td>M_{23}</td><td>float</td><td>n/a</td></tr>
<tr><td>24</td><td>M_{31}</td><td>float</td><td>n/a</td></tr>
<tr><td>28</td><td>M_{32}</td><td>float</td><td>n/a</td></tr>
<tr><td>32</td><td>M_{33}</td><td>float</td><td>n/a</td></tr>
<tr><td>36</td><td>Valid Flags</td><td>U16</td><td>See Notes</td></tr>
</table>

Estimated Orientation, Euler Angles (0x82, 0x05)

Description	Pitch, Roll, and Yaw (aircraft) values
Notes	This is a 3 component vector containing the Roll, Pitch and Yaw angles in radians. It is computed by the INS from the orientation quaternion Q. $Euler = \begin{bmatrix} Roll \\ Pitch \\ Yaw \end{bmatrix}$ (radians) Valid Flag Mapping: 0x0000 – Euler Angles are Invalid 0x0001 – Euler Angles Valid

Field Format

Field Length	Data Descriptor	Message Data			
		Binary Offset	Description	Data Type	Units
16 (0x10)	0x05	0	Roll	float	radians
		4	Pitch	float	radians
		8	Yaw	float	radians
		12	Valid Flags	U16	See Notes

Estimated Gyro Bias (0x82, 0x06)

Description	Estimated Gyro Biases expressed in the Sensor Body Frame.
Notes	Valid Flag Mapping: 0x0000 – Gyro Bias are Invalid 0x0001 – Gyro Bias Valid

Field Format

Field Length	Data Descriptor	Message Data			
		Binary Offset	Description	Data Type	Units
16 (0x10)	0x06	0	X Gyro Bias	float	radians/sec
		4	Y Gyro Bias	float	radians/sec
		8	Z Gyro Bias	float	radians/sec
		12	Valid Flags	U16	See Notes

Estimated LLH Position Uncertainty (0x82, 0x08)

Description	INS Estimated Position Data expressed in the Geodetic Frame					
Notes	Valid Flag Mapping: 0x0000 – Position Uncertainties are Invalid 0x0001 – Position Uncertainties Valid					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
			Binary Offset	*Description*	*Data Type*	*Units*
	16 (0x10)	0x08	0	Local-Level, 1-Sigma Position Uncertainty (North)	Float	Meters
			4	Local-Level, 1-Sigma Position Uncertainty (East)	Float	Meters
			8	Local-Level, 1-Sigma Position Uncertainty (Down)	Float	Meters
			12	Valid Flags	U16	See Notes

Estimated NED Velocity Uncertainty (0x82, 0x09)

Description	INS Estimated Velocity Data expressed in the Local-Level Frame					
Notes	Valid Flag Mapping: 0x0000 – NED Velocity Uncertainties are Invalid 0x0001 – NED Velocity Uncertainties Valid					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
			Binary Offset	*Description*	*Data Type*	*Units*
	16 (0x10)	0x09	0	Local-Level, 1-Sigma Velocity Uncertainty (North)	Float	Meters / Sec
			4	Local-Level, 1-Sigma Velocity Uncertainty (East)	Float	Meters / Sec

			8	Local-Level, 1-Sigma Velocity Uncertainty (Down)	Float	Meters / Sec
			12	Valid Flags	U16	See Notes

Estimated Attitude Uncertainty, Euler Angles (0x82, 0x0A)

Description	1-sigma attitude uncertainty expressed in Pitch, Roll, and Yaw (aircraft) elements.					
Notes	This is a 3 component vector containing the Roll, Pitch and Yaw angle uncertainties in radians. **IMPORTANT:** These values are derived from the quaternion elements and become increasingly inaccurate as the pitch angle approaches +-90 degrees. To compensate for this limitation, these values will be marked as invalid when the pitch angle exceeds +-70 degrees. Valid Flag Mapping: 0x0000 – Attitude Uncertainties are Invalid 0x0001 – Attitude Uncertainties Valid					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
			Binary Offset	*Description*	*Data Type*	*Units*
	16 (0x10)	0x0A	0	1-Sigma Attitude Uncertainty (Roll)	float	radians
			4	1-Sigma Attitude Uncertainty (Pitch)	float	radians
			8	1-Sigma Attitude Uncertainty (Yaw)	float	radians
			12	Valid Flags	U16	See Notes

Estimated Gyro Bias Uncertainty (0x82, 0x0B)

Description	Estimated Gyro Bias Uncertainty expressed in the Sensor Body Frame.					
Notes	Valid Flag Mapping: 0x0000 – Gyro Bias Uncertainties are Invalid 0x0001 – Gyro Bias Uncertainties Valid					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
			Binary Offset	*Description*	*Data Type*	*Units*
	16 (0x10)	0x0B	0	1-Sigma Gyro Bias Uncertainty (X)	float	radians/sec
			4	1-Sigma Gyro Bias Uncertainty (Y)	float	radians/sec
			8	1-Sigma Gyro Bias Uncertainty (Z)	float	radians/sec
			12	Valid Flags	U16	See Notes

Estimated Linear Acceleration (0x82, 0x0D)

Description	INS Estimated Linear Acceleration Data expressed in: 1) The Sensor Frame, if no sensor to body rotation has been defined. 2) The Vehicle Frame, if a sensor to body rotation has been defined.					
Notes	Valid Flag Mapping: 0x0000 – Linear Accelerations are Invalid 0x0001 – Linear Accelerations are Valid					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
			Binary Offset	*Description*	*Data Type*	*Units*
	16 (0x10)	0x0D	0	X	Float	Meters / Sec^2
			4	Y	Float	Meters / Sec^2
			8	Z	Float	Meters / Sec^2
			12	Valid Flags	U16	See Notes

Estimated Angular Rate (0x82, 0x0E)

<table>
<tr><td>Description</td><td colspan="6">INS Estimated Angular Rate Data expressed in:
1) The Sensor Frame, if no sensor to body rotation has been defined.
2) The Vehicle Frame, if a sensor to body rotation has been defined.</td></tr>
<tr><td>Notes</td><td colspan="6">The estimated gyro bias has been removed from these angular rate values.

Valid Flag Mapping:

0x0000 – Angular Rates are not Valid
0x0001 – Angular Rates are Valid</td></tr>
<tr><td rowspan="6">Field Format</td><td>Field Length</td><td>Data Descriptor</td><td colspan="4">Message Data</td></tr>
<tr><td rowspan="5">16 (0x10)</td><td rowspan="5">0x0E</td><td>Binary Offset</td><td>Description</td><td>Data Type</td><td>Units</td></tr>
<tr><td>0</td><td>X</td><td>Float</td><td>Radians / Sec</td></tr>
<tr><td>4</td><td>Y</td><td>Float</td><td>Radians / Sec</td></tr>
<tr><td>8</td><td>Z</td><td>Float</td><td>Radians / Sec</td></tr>
<tr><td>12</td><td>Valid Flags</td><td>U16</td><td>See Notes</td></tr>
</table>

WGS84 Local Gravity Magnitude (0x82, 0x0F)

<table>
<tr><td>Description</td><td colspan="6">Local Magnitude of Earth's gravity using the WGS84 gravity model.</td></tr>
<tr><td>Notes</td><td colspan="6">The GX3-45 implements the WGS84 gravity model, valid for altitudes of 20km or less.

Valid Flag Mapping:

0x0000 – Gravity value is Invalid
0x0001 – Gravity value is Valid</td></tr>
<tr><td rowspan="4">Field Format</td><td>Field Length</td><td>Data Descriptor</td><td colspan="4">Message Data</td></tr>
<tr><td rowspan="3">08 (0x08)</td><td rowspan="3">0x0F</td><td>Binary Offset</td><td>Description</td><td>Data Type</td><td>Units</td></tr>
<tr><td>0</td><td>Gravity Magnitude</td><td>Float</td><td>meters / sec^2</td></tr>
<tr><td>4</td><td>Valid Flags</td><td>U16</td><td>See Notes</td></tr>
</table>

Estimated Attitude Uncertainty, Quaternion Elements (0x82, 0x12)

<table>
<tr><td>Description</td><td colspan="6">1-sigma attitude uncertainty expressed in quaternion components.</td></tr>
<tr><td>Notes</td><td colspan="6">This is a 4 component vector containing the attitude uncertainty expressed in quaternion elements.

Valid Flag Mapping:

0x0000 – Attitude uncertainties are Invalid
0x0001 – Attitude uncertainties are Valid</td></tr>
<tr><td rowspan="7">Field Format</td><td>Field Length</td><td>Data Descriptor</td><td colspan="4">Message Data</td></tr>
<tr><td rowspan="6">20 (0x14)</td><td rowspan="6">0x12</td><td>Binary Offset</td><td>Description</td><td>Data Type</td><td>Units</td></tr>
<tr><td>0</td><td>1-Sigma Attitude Uncertainty (q0)</td><td>float</td><td></td></tr>
<tr><td>4</td><td>1-Sigma Attitude Uncertainty (q1)</td><td>float</td><td></td></tr>
<tr><td>8</td><td>1-Sigma Attitude Uncertainty (q2)</td><td>float</td><td></td></tr>
<tr><td>12</td><td>1-Sigma Attitude Uncertainty (q3)</td><td>float</td><td></td></tr>
<tr><td>16</td><td>Valid Flags</td><td>U16</td><td>See Notes</td></tr>
</table>

Estimated Gravity Vector (0x82, 0x13)

<table>
<tr><td rowspan="2">Description</td><td colspan="6">INS Estimated Gravity Data expressed in:
1) The Sensor Frame, if no sensor to body rotation has been defined.
2) The Vehicle Frame, if a sensor to body rotation has been defined.</td></tr>
<tr></tr>
<tr><td>Notes</td><td colspan="6">Valid Flag Mapping:

0x0000 – Gravity vector is Invalid
0x0001 – Gravity vector is Valid</td></tr>
<tr><td rowspan="6">Field Format</td><td>Field Length</td><td>Data Descriptor</td><td colspan="4">Message Data</td></tr>
<tr><td rowspan="5">16 (0x10)</td><td rowspan="5">0x13</td><td>Binary Offset</td><td>Description</td><td>Data Type</td><td>Units</td></tr>
<tr><td>0</td><td>X</td><td>Float</td><td>Meters / Sec^2</td></tr>
<tr><td>4</td><td>Y</td><td>Float</td><td>Meters / Sec^2</td></tr>
<tr><td>8</td><td>Z</td><td>Float</td><td>Meters / Sec^2</td></tr>
<tr><td>12</td><td>Valid Flags</td><td>U16</td><td>See Notes</td></tr>
</table>

Heading Update Source State (0x82, 0x14)

Description	Heading Update Source information expressed in the sensor frame.
Notes	Heading updates can be applied from a number of sources (listed below.) The heading value is always relative to true north. Possible Sources: 0x0000 – No source, heading updates disabled 0x0001 – Internal Magnetometer 0x0002 – Internal GPS Velocity Vector 0x0003 – External Heading Update Command Valid Flag Mapping: 0x0000 – No heading update received in 2 seconds. 0x0001 – The heading update source has provided data within 2 seconds.

Field Format

Field Length	*Data Descriptor*	*Message Data*			
		Binary Offset	*Description*	*Data Type*	*Units*
14 (0x0E)	0x14	0	Heading (True)	Float	Radians
		4	Heading 1-sigma Uncertainty	Float	Radians
		8	Source	U16	See Notes
		10	Valid Flags	U16	See Notes

Magnetic Model Solution (0x82, 0x15)

<table>
<tr><td>Description</td><td colspan="6">Magnetic model solution expressed in the NED frame.</td></tr>
<tr><td>Notes</td><td colspan="6">The World Magnetic Model 2010 is used. A valid GPS location is required for the model to be valid.

Valid Flag Mapping:

0x0000 – Magnetic model solution is invalid (note: this will be the state when the magnetic model is recalculating for the current time and location as well as when GPS is unavailable)
0x0001 – Magnetic model solution is valid</td></tr>
<tr><td rowspan="8">Field Format</td><td>Field Length</td><td>Data Descriptor</td><td colspan="4">Message Data</td></tr>
<tr><td rowspan="7">24 (0x18)</td><td rowspan="7">0x15</td><td>Binary Offset</td><td>Description</td><td>Data Type</td><td>Units</td></tr>
<tr><td>0</td><td>Intensity (North)</td><td>Float</td><td>Gauss</td></tr>
<tr><td>4</td><td>Intensity (East)</td><td>Float</td><td>Gauss</td></tr>
<tr><td>8</td><td>Intensity (Down)</td><td>Float</td><td>Gauss</td></tr>
<tr><td>12</td><td>Inclination</td><td>Float</td><td>Radians</td></tr>
<tr><td>16</td><td>Declination</td><td>Float</td><td>Radians</td></tr>
<tr><td>20</td><td>Valid Flags</td><td>U16</td><td>See Notes</td></tr>
</table>

MIP Packet Reference

Structure

Commands and Data are sent and received as fields in the MicroStrain "MIP" packet format. Below is the general definition of the structure:

Header				Payload			Checksum	
SYNC1 "u"	SYNC2 "e"	Descriptor Set byte	Payload Length byte	Fields			MSB	LSB
0x75	0x65	<desc set selector>	$k_1+k_2+\ldots k_n$	MIP Field 1 length = k_1	...	MIP Field *n* length = k_n	0x*MM*	0x*LL*

Field Header		Field Data
Field Length byte	Field Descriptor byte	Field Data
k_n	<descriptor>	<k_n-2 bytes of data>

The packet always begins with the start-of-packet sequence "ue" (0x75, 0x65). The "Descriptor Set" byte in the header specifies which command or data set is contained in fields of the packet. The payload length byte specifies the sum of all the field length bytes in the payload section.

Payload Length Range

Packet Header				Payload	Checksum	
SYNC 1	SYNC 2	Descriptor Set	Payload Length	MIP Data Fields	MSB	LSB
				<-------------*Payload Length Range* --------------->		

The payload section can be empty or can contain one or more fields. Each field has a length byte and a descriptor byte. The field length byte specifies the length of the entire field including the field length byte and field descriptor byte. The descriptor byte specifies the command or data that is contained in the field data. The descriptor can only be from the set of descriptors specified by the descriptor set byte in the header. The field data can be anything but is always rigidly defined. The definition of a descriptor is fundamentally described in a ".h" file that corresponds to the descriptor set that the descriptor belongs to.

MicroStrain provides a “Packet Builder” functionality in the “MIP Monitor” software utility to simplify the construction of a MIP packet. Most commands will have a single field in the packet, but multiple field packets are possible. Extensive examples complete with checksums are given in the command reference section.

Checksum Range

The checksum is a 2 byte Fletcher checksum and encompasses all the bytes in the packet:

Packet Header				Payload	Checksum	
SYNC 1	SYNC 2	Descriptor Set	Payload Length	MIP Data Fields	MSB (byte1)	LSB (byte2)
<--- *Checksum Range* -->						

16-bit Fletcher Checksum Algorithm (C language)

```
for(i=0; i<checksum_range; i++)
{
 checksum_byte1 += mip_packet[i];
 checksum_byte2 += checksum_byte1;
}

checksum = ((u16) checksum_byte1 << 8) + (u16) checksum_byte2;
```

Advanced Programming

Multiple Commands in a Single Packet

MIP packets may contain one or more individual commands. In the case that multiple commands are transmitted in a single MIP packet, the GX3-45 will respond with a single packet containing multiple replies. As with any packet, all commands must be from the same descriptor set (you cannot mix Base commands with 3DM commands in the same packet).

Below is an example that shows how you can combine the commands from step 2 and 3 of the Example Setup Sequence into a single packet. The commands are from the 3DM set. The command packet has two fields as does the reply packet (the fields are put on separate rows for clarity):

Step 2 and 3	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Cmd Desc.*	*Field Data*	*MSB*	*LSB*
Command field 1 Set AHRS Message Format	**0x75**	**0x65**	**0x0C**	**0x14**	**0x0A**	**0x08**	Function: **0x00** Desc count: **0x02** 1st Descriptor: **0x03** Rate Dec: **0x000A** 2nd Descriptor:**0x04** Rate Dec: **0x000A**		
Command field 2 Set GPS Message Format					**0x0A**	**0x09**	Function: **0x00** Desc Count: **0x02** ECEF pos desc: **0x04** Rate dec: **0x0004** ECEF vel desc: **0x06** Rate dec: **0x0004**	**0x50**	**0x98**
Reply field 1 ACK/NACK	**0x75**	**0x65**	**0x0C**	**0x08**	**0x04**	**0xF1**	Cmd echo: **0x08** Error code: **0x00**		
Reply field 2 ACK/NACK					**0x04**	**0xF1**	Cmd echo: **0x09** Error code: **0x00**	**0xE9**	**0x6F**

Copy-Paste version of the command: "7565 0C14 0A08 0002 0300 0A04 000A 0A09 0002 0400 0406 0004 5098"

Note that the only difference in the packet headers of the single command packets compared to the multiple command packets is the payload length. Parsing multiple fields in a single packet involves subtracting the field length of the next field from the payload length until the payload length is less than or equal to zero.

Direct Modes

The GX3-45 has special “direct” modes that switch the device into a “GX3-25” AHRS or a “u-blox” GPS device. The Device Communications Mode command is used to switch between modes. When in these modes, the GX3-45 acts just like a GX3-25 AHRS or a u-blox GPS sensor respectively. Any code or tools developed for these devices may be used in these modes. For example, when in the “u-blox” direct mode, the u-blox “u-center” application works perfectly with the GPS chip embedded in the GX3-45.

These modes can be used to access advanced (native) data of the individual sensors, data that isn’t represented in the 3DM command sets of the GX3-45.

IMPORTANT: When you switch modes, you are switching to a new device protocol EXCEPT for two commands: the Device Communications Mode and Device Status commands. Those commands are always available regardless of which mode you are in. For example, if you switch to GPS direct mode, then the protocol recognized by the device is NMEA and UBX protocol, however the GX3-45 is still “listening” for mode switch or device status commands and will respond to them. It will not respond to any other 3DM-GX3-45 Basic or 3DM commands until switched back to the ”Standard Mode”.

IMPORTANT: The GPS message settings required for Kalman Filter execution are automatically reloaded when switching from direct modes back in to standard mode.

Internal Diagnostic Functions

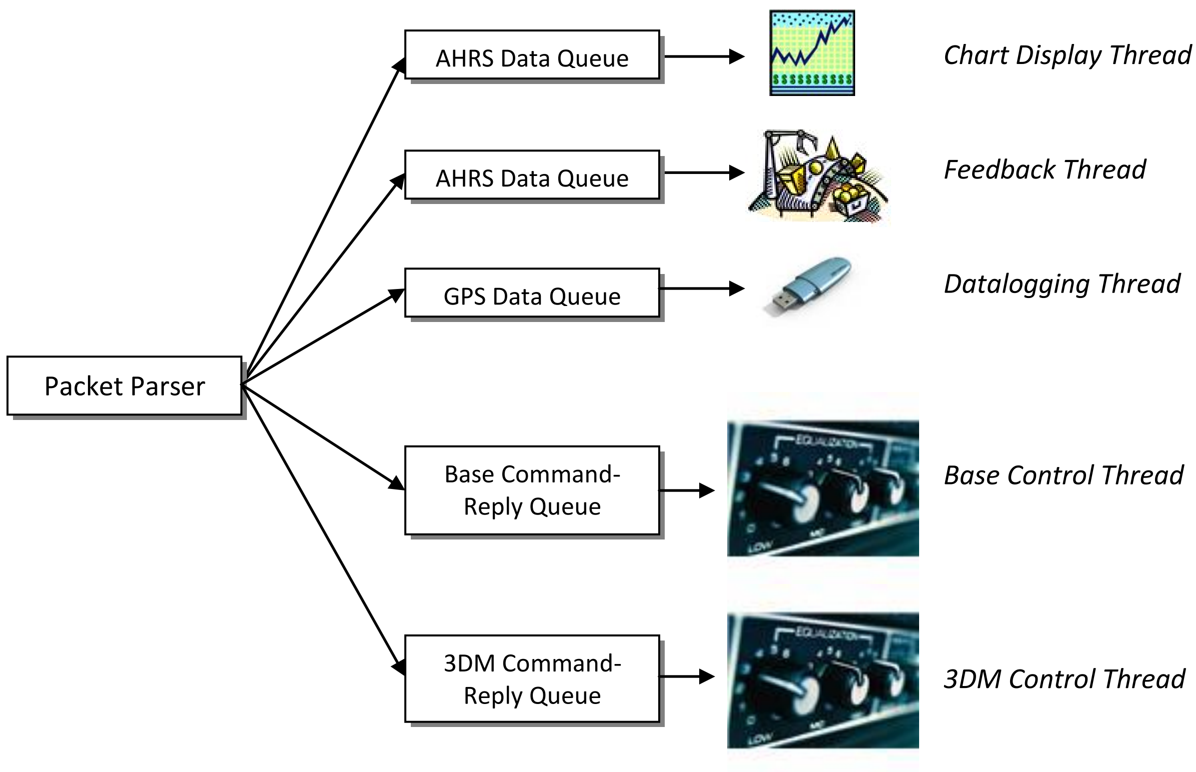
The 3DM-GX3-45 supports two device specific internal functions used for diagnostics and system status. These are Device Built In Test and Device Status. These commands are defined generically but the implementation is very specific to the hardware implemented on this device. Other MicroStrain devices will have their own implementations of these functions depending on the internal hardware of the devices.

3DM-GX3-45 INTERNAL DIAGNOSTIC COMMANDS

- Device Built In Test (0x01, 0x05)
- Device Status (0x0C, 0x64)

Advanced Programming Models

Many applications will only require a single threaded programming model which is simple to implement using a single program loop that services incoming packets. In other applications, advanced techniques such as multithreading or event based processes are required. The MIP packet design simplifies implementation of these models. It does this by limiting the packet size to a maximum of 261 bytes and it provides the “descriptor set” byte in the header. The limited packet size makes scalable packet buffers possible even with limited memory space. The descriptor set byte aids in sorting an incoming packet stream into one or more command-reply packet queues and/or data packet queues. A typical multithreaded environment will have a command/control thread and one or more data processing threads. Each of these threads can be fed with individual incoming packet queues, each containing packets that only pertain to that thread – sorted by descriptor set. Packet queues can easily be created dynamically as threads are created and destroyed. All packet queues can be fed by a single incoming packet parser that runs continuously independent of the queues. The packet queues are individually scaled as appropriate to the process; smaller queues for lower latency and larger queues for more efficient batch processing of packets.



Multithreaded application with multiple incoming packet queues