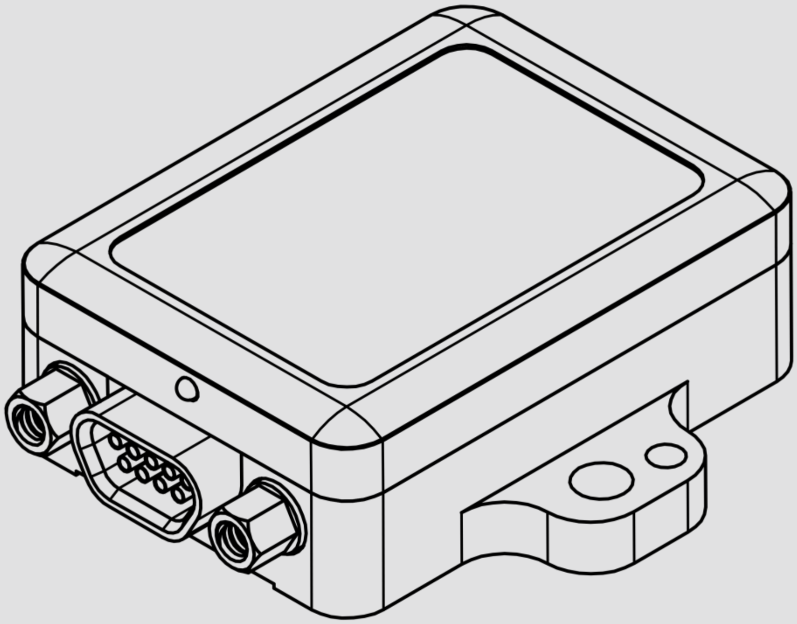
LORD MANUAL

3DM-GX4™-25

Data Communications Protocol



LORD MicroStrain®
SENSING SYSTEMS

MicroStrain® Sensing Systems
459 Hurricane Lane
Suite 102
Williston, VT 05495
United States of America
Phone: 802-862-6629
Fax: 802-863-4093

www.microstrain.com
sensing_support@LORD.com

REVISED: April 29, 2014

Table of Contents

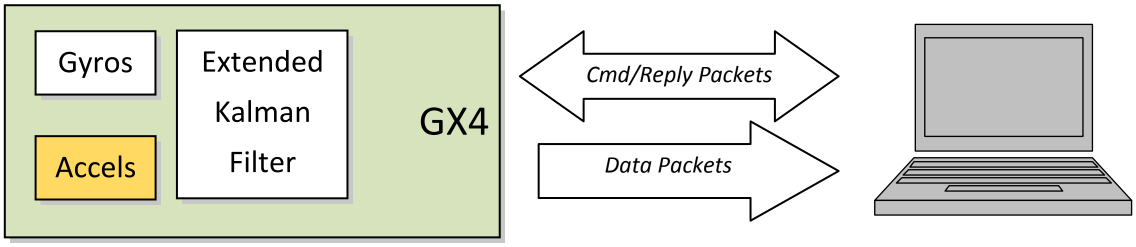
3DM-GX4 API

API Introduction

The 3DM-GX4 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 2 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 linea set of "Estimation Filter" commands that are specific to MicroStrain navigation and advanced AHRS devices, and a set of "System" commands that are specific to sensor systems comprised of more than one internal sensor block. The data set represent the types of data that the 3DM-GX4 is capable of producing: "IMU" (Inertial Measurement Unit) and "Estimation Filter" data.

Base commands	*Ping, Idle, Resume, Get ID Strings, etc.*
3DM commands	*Poll IMU Data, Poll GPS Data, etc.*
Estimation Filter commands	*Reset Estimation Filter, etc.*
System commands	*Switch Communications Mode, etc.*
IMU data	*Acceleration Vector, Gyro Vector, Euler Angles, etc.*
Estimation Filter data	*Attitude, Acceleration 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, or IMU data.



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, IMU 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)
GPS Time Update	(0x01, 0x72)
Device Reset	(0x01, 0x7E)

3DM Command Set (0x0C)

Poll IMU Data	(0x0C, 0x01)
Poll Estimation Filter Data	(0x0C, 0x03)
Get IMU Data Base Rate	(0x0C, 0x06)
Get Estimation Filter Data Base Rate	(0x0C, 0x0B)
IMU Message Format	(0x0C, 0x08)
Estimation Filter Message Format	(0x0C, 0x0A)
Enable/Disable Device Continuous Data Stream	(0x0C, 0x11)
Device Startup Settings	(0x0C, 0x30)
IMU Hard Iron Offset	(0x0C, 0x3A)
IMU Soft Iron Matrix	(0x0C, 0x3B)
Accel Bias	(0x0C, 0x37)
Gyro Bias	(0x0C, 0x38)
Capture Gyro Bias	(0x0C, 0x39)
Coning and Sculling Enable	(0x0C, 0x3E)
Change UART BAUD rate	(0x0C, 0x40)
Advanced Low-Pass Filter Settings	(0x0C, 0x50)
Complementary Filter Settings	(0x0C, 0x51)
Device Status*	(0x0C, 0x64)

Estimation Filter Command Set (0x0D)

Reset Filter	(0x0D, 0x01)
Set Initial Attitude	(0x0D, 0x02)
Set Initial Heading	(0x0D, 0x03)

System Command Set (0x7F)

**Advanced Commands* —

Data

IMU Data Set (set 0x80)

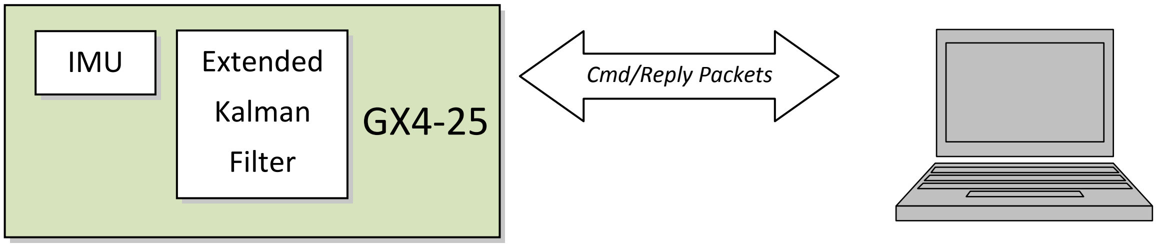
Scaled Accelerometer Vector	(0x80, 0x04)
Scaled Gyro Vector	(0x80, 0x05)
Scaled Magnetometer Vector	(0x80, 0x06)
Scaled Ambient Pressure	(0x80, 0x17)
Delta Theta Vector	(0x80, 0x07)
Delta Velocity Vector	(0x80, 0x08)
IMU GPS Correlated Timestamp	(0x80, 0x12)

Filter Data Set (set 0x82)

Filter Status	(0x82, 0x10)
Filter GPS Timestamp	(0x82, 0x11)
Estimated Quaternion	(0x82, 0x03)
Estimated Orientation Matrix	(0x82, 0x04)
Estimated Euler Angles	(0x82, 0x05)
Estimated Gyro Bias	(0x82, 0x06)
Estimated Attitude Uncertainty (Euler Angles)	(0x82, 0x0A)
Estimated Attitude Uncertainty (Quaternion Elements)	(0x82, 0x12)
Estimated Gyro Bias Uncertainty	(0x82, 0x0B)
Estimated Linear Acceleration	(0x82, 0x0D)
Estimated Angular Rate	(0x82, 0x0E)
WGS84 Local Gravity Magnitude	(0x82, 0x0F)
Estimated Gravity Vector	(0x82, 0x13)
Heading Update Source State	(0x82, 0x14)
Magnetic Model Solution	(0x82, 0x15)
Pressure Altitude	(0x82, 0x21)

Basic Programming

The 3DM-GX4-25 is designed to stream IMU and Estimation Filter 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-GX4-25 either through a COM utility or as a template for software development.

MIP Packet Overview

This is an overview of the 3DM-GX4-25 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"*	*Descript or 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:

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**

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

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 IMU 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 IMU 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 IMU data OR and combination of Estimation Filter data– you cannot combine IMU data and Estimation Filter 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 IMU and Estimation Filter 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 IMU 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-GX4-25 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-GX4-25 sending a continuous data stream. In the following example, the IMU data format is set, followed by the Estimation Filter data format. 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 IMUand Estimation Filter 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 IMU data-stream format

Send a "Set IMU Message Format" command (reply is ACK/NACK). This example requests scaled gyro, scaled accelerometer, and GPS Correlation Timestamp information at 1000 Hz (1000Hz base rate, with a rate decimation of 1 on the 3DM-GX4-25 = 1000 Hz.) This will result in a single IMU data packet sent at 1000 Hz containing the scaled gyro field followed by the scaled accelerometer field followed by the IMU 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 IMU data would begin stream immediately.

Please note, this command will not append the requested descriptors to the current IMU 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 IMU 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 00011200 012A 35"

Step 3: Configure the Estimation Filter data-stream format

The following configuration command requests the Estimated LLH Position, Estimated NED Velocity, Estimated Orientation in Quaternion form, and Filter Status at 100 Hz (500Hz base rate, with a rate decimation of 5 = 100 Hz.) This will result in a single Estimation Filter packet sent at 100 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 Estimation Filter data would begin stream immediately.

Please note, this command will not append the requested descriptors to the current Estimation Filter 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 Estimation Filter 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	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0xF1**	Cmd echo: **0x0A**	**0xE9**	**0xBE**

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

Copy-Paste version of the command: "7565 0C10100A 0104 0100 0502 0005 0300 0510 00053F31"

Step 4: Save the IMU and Estimation Filter MIP Message format

To save the IMU and Estimation Filter MIP Message format, use the "Save" function selector (0x03) in the IMU and Estimation Filter 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 IMU Message Format	**0x75**	**0x65**	**0x0C**	**0x08**	**0x04**	**0x08**	Function: **0x03** Desc count: **0x00**		
Command field 2 Save Current Estimation Filter 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 IMU and Estimation Filter data-streams

Send an "Enable/Disable Continuous Stream" command to enable the IMU and Estimation Filter 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 IMU Message	**0x75**	**0x65**	**0x0C**	**0x0A**	**0x05**	**0x11**	Fctn: **0x01** IMU: **0x01** ON: **0x01**		
Command field 2 Enable Continuous					**0x05**	**0x11**	Fctn: **0x01** Estimation Filter:	**0x24**	**0xCC**

Estimation Filter Message							**0x03** ON: **0x01**		
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"

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 IMU, GPS, and Estimation Filter data-streams)

Same as continuous streaming. See above.

Step 2: Configure the IMU data-stream format

Same as continuous streaming. See above.

Step 3: Configure the Estimation Filter data-stream format

Same as continuous streaming. See above.

Step 4: Save the IMU and Estimation Filter MIP Message format

Same as continuous streaming. See above.

Step 5: Resume the Device

Same as continuous streaming step 6. See above.

Step 6: Send individual data polling commands

Send individual Poll IMU Data and Poll Estimation Filter 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 (IMU or Estimation Filter):

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 IMU 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**
IMU Data Packet field 1 (Gyro Vector)	**0x75**	**0x65**	**0x80**	**0x1C**	**0x0E**	**0x04**	**0x3E 7A 63 A0** **0xBB 8E 3B 29** **0x7F E5 BF 7F**		
IMU Data Packet field 2(Accel Vector)					**0x0E**	**0x03**	**0x3E 7A 63 A0** **0xBB 8E 3B 29** **0x7F E5 BF 7F**	**0xAD**	**0xDC**

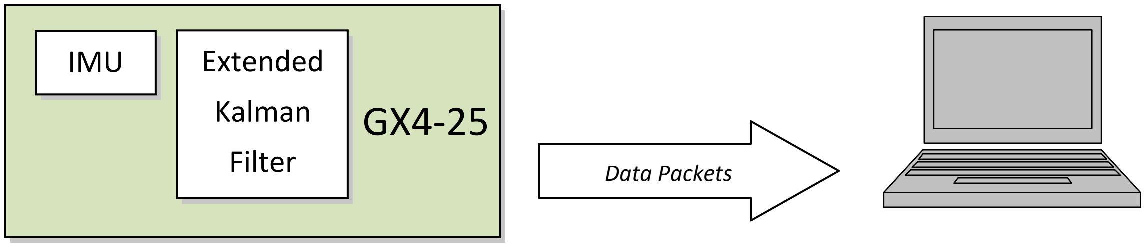
Copy-Paste version of the command: "7565 0C04 0401 0000EF DA"

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 IMU Data and Poll Estimation Filter 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-GX4-25. 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 IMU and Kalman Filter (Estimation Filter) 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 an IMU or Estimation Filter data set). Since you know beforehand that the IMU and Estimation Filter data packets will be coming in fastest, you can tune your code to buffer or handle these packets at a high priority. 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 (IMU Message Format and Estimation Filter Message Format) allow you to set different data rates for different data quantities. This is a very useful feature especially for IMU data because some data, such as accelerometer and gyroscope data, usually requires higher data rates (>100Hz) than other IMU 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.

IMU and Estimation Filter 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 IMU data requires precise microsecond resolution and perfectly regular intervals in its timestamps. The Kalman Filter resides on a separate processor and must derive its timing information from the IMU data.

The time base difference is one of the factors that necessitate separation of the IMU and Estimation Filter 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, IMU data is always grouped with a timestamp generated from the IMU time base, and Estimatiopn Filter data is always grouped with a timestamp from the Estimation Filter time base, etc.

All data streams (IMU and Estimation Filter) on the 3DM-GX4-25 output a “GPS Time”-formatted timestamp. This allows a precise common time base for all data. 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 IMU and Estimation Filter timestamps when the GPS Time Update Command is applied.

Communications Bandwidth Management

Because of the large amount and variety of data that is available from the 3DM-GX4-25, it is quite easy to overdrive the bandwidth of the communications channel. This can result in dropped packets. The 3DM-GX4-25 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

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

which becomes

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

Example:

For an IMU 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.

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)

<table>
<tr><td>Description</td><td colspan="6">Get the device ID strings and firmware version</td></tr>
<tr><td>Notes</td><td colspan="6">Reply has two fields: “ACK/NACK” and “Device Info Field”</td></tr>
<tr><td>Field Format</td><td>Field Length</td><td>Field Descriptor</td><td colspan="4">Field Data</td></tr>
<tr><td>Command</td><td>0x02</td><td>0x03</td><td colspan="4">N/A</td></tr>
<tr><td>Replyfield 1
ACK/NACK</td><td>0x04</td><td>0xF1</td><td colspan="4">U8 – echo the command byte
U8 – error code (0: ACK, non-zero: NACK)</td></tr>
<tr><td rowspan="7">Reply field 2
Device Info Field</td><td rowspan="7">0x52</td><td rowspan="7">0x81</td><td>Binary Offset</td><td>Description</td><td>Data Type</td><td>Units</td></tr>
<tr><td>0</td><td>Firmware Version</td><td>U16</td><td>N/A</td></tr>
<tr><td>2</td><td>Model Name</td><td>String(16)</td><td>N/A</td></tr>
<tr><td>18</td><td>Model Number</td><td>String(16)</td><td>N/A</td></tr>
<tr><td>34</td><td>Serial Number</td><td>String(16)</td><td>N/A</td></tr>
<tr><td>50</td><td>Lot Number</td><td>String(16)</td><td>N/A</td></tr>
<tr><td>66</td><td>Device Options</td><td>String(16)</td><td>N/A</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>0x03</td><td></td><td>0xE2</td><td>0xC8</td></tr>
<tr><td>ReplyField 1
ACK/NACK</td><td>0x75</td><td>0x65</td><td>0x01</td><td>0x58</td><td>0x04</td><td>0xF1</td><td>Command echo:
0x03
Error code: 0x00</td><td></td><td></td></tr>
<tr><td>Reply Field 2
Device Info Field</td><td></td><td></td><td></td><td></td><td>0x54</td><td>0x81</td><td>FW Version: 0x03F1
“ 3DM-GX4-25”
“ 6234-4220”
“ 6243-00009”
“ I042Y”
“ 5g, 300dps”</td><td>0x##</td><td>0x##</td></tr>
</table>

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

Get Device Descriptor Sets (0x01, 0x04)

Description	Get the set of descriptors that this device supports				
Notes	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.				
Field Format	*Field Length*	*Field Descriptor*	*Field Data*		
Command	0x02	0x04	N/A		
Replyfield 1 *ACK/NACK*	0x04	0xF1	U8 – echo the command byte U8 – error code (0: ACK, non-zero: NACK)		
Reply field 2 *Array of Descriptors*	2 x <Number of descriptors> + 2	0x82	*Binary Offset*	*Description*	*Data Type*
			0	MSB: Descriptor Set LSB: Descriptor	U16
			1	MSB: Descriptor Set LSB: Descriptor	U16
			…	*<etc>*	…

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**	**0x04**		**0xE3**	**0xC9**
ReplyField 1 *ACK/NACK*	**0x75**	**0x65**	**0x01**	**0x04**	**0x04**	**0xF1**	Command echo: **0x04** Error code: **0x00**		
Reply Field 2 *Array of Descriptors*					***<n*2>***	**0x82**	**0x0101** **0x0102** **0x0103** **…** **0x0C01** **0x0C02** **…** *nth descriptor:* **0x0C72**	**0x##**	**0x##**

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-GX4-25 are defined below.

Notes

3DM-GX4-25 BIT Error Flags:

Byte	Byte 1 (LSB)	Byte 2	Byte 3	Byte 4 (MSB)
Device	Processor Board	Sensor Board	Reserved	Kalman Filter
Bit 1 (LSB)	WDT Reset (Latching, Reset after first commanded BIT)	IMU Communication Fault	Reserved	Solution Fault
Bit 2	Reserved	Magnetometer Fault (if applicable)	Reserved	Reserved
Bit 3	Reserved	Pressure Sensor Fault (if applicable)	Reserved	Reserved
Bit 4	Reserved	Reserved	Reserved	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"

GPS Time Update (0x01, 0x72)

Description	This message updates the internal GPS Time as reported in the Filter Timestamp.
Notes	This command enables synchronization of IMU/AHRS Timestamps with an external GPS receiver. When combined with a PPS input applied to pin 7 of the i/o connector, the GPS Correlation Timestamp in the inertial data output is synchronized with the external GPS clock. It is recommended that this update command be sent once per second. See the GPS Correlation Timestamp for more information. *Possible function selector values:* 0x01 – Apply new settings 0x02 – Read back current settings. 0x06 – Apply new settings with no ACK/NACK Reply *Possible field selector values:* 0x01 – GPS Week Number. 0x02 – GPS Seconds.

Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x08	0x72	U8 – Function Selector U8 – GPS Time Field Selector U32 – New Time Value
Reply ACK/NACK	0x04	0xF1	U8 – echo the command descriptor U8 – error code (0: ACK, non-zero: NACK)
Reply field 2 (function = 2 selector = 1)	0x06	0x84	U32 – Current GPS Week Value
Reply field 2 (function = 2 selector = 2)	0x06	0x85	U32 – Current GPS Seconds Value

Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command GPS Time Update	**0x75**	**0x65**	**0x01**	**0x08**	**0x08**	**0x72**	*Fctn(Apply):***0x01** *Field (Week):* **0x01** *Val:***0x00000698**	**0xFD**	**0x32**
Reply ACK/NACK	**0x75**	**0x65**	**0x01**	**0x04**	**0x04**	**0xF1**	Cmd echo: **0x72** Error code: **0x00**	**0x46**	**0x4C**

Copy-Paste version of the command: "7565 0108 0872 0101 0000 0698 FD32"

Device Reset (0x01, 0x7E)

Description	Resets the 3DM-GX4.		
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 IMU Data (0x0C, 0x01)

Description	Poll the 3DM-GX4 for an IMU message with the specified format		
Notes	This function polls for an IMU 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 IMU 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 Estimation Filter Data (0x0C, 0x03)

Description	Poll the device for a Estimation Filter message with the specified format		
Notes	This function polls for a Estimation Filter 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 Estimation Filter 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 Estimation Filter 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 Estimation Filter data (use stored format)	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0x03**	Option: **0x00** Desc count: **0x00**	**0xF1**	**0xE0**
Command Poll Estimation Filter 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 IMU Data Base Rate (0x0C, 0x06)

Description	Get the base rate for the IMU data in Hz.		
Notes	Returns the value used for data rate calculations. See the IMU Message Format command.		
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 - IMU data base rate (Hz)

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 Estimation Filter Data Base Rate (0x0C, 0x0B)

Description	Get the base rate for the Estimation Filter data in Hz.		
Notes	Returns the value used for data rate calculations. See the Estimation Filter Message Format command.		
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 *Estimation Filter*	0x04	0x8A	U16 – Filter data base rate (Hz)

Base Rate			

Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	MSB	LSB
Command *Get Base Rate*	**0x75**	**0x65**	**0x0C**	**0x02**	**0x02**	**0x0B**		**0xF5**	**0xFC**
Reply field 1 *ACK/NACK*	**0x75**	**0x65**	**0x0C**	**0x08**	**0x04**	**0xF1**	*Echo cmd:* **0x0B** *Error code:* **0x00**		
Reply field 2 *Estimation Filter* *Base Rate*					**0x04**	**0x8A**	Base rate (Hz): **0x0064**	**0xE0**	**0x9E**

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

IMU Message Format (0x0C, 0x08)

Description	Set, read, or save the format of the IMU message packet. This command sets the format for the IMU 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 IMU messages: Data Rate = 1000 Hz / Rate Decimation The GX4 checks that all descriptors are valid prior to executing this command. If any of the descriptors are invalid for the IMU 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 IMU 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 *IMU 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 IMU 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“

Estimation Filter Message Format (0x0C, 0x0A)

Description	Set, read, or save the format of the Estimation Filter message packet. This function sets the format for the Estimation Filter 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 Estimation Filter messages: *Data Rate = 1000 Hz / Rate Decimation* The device checks that all descriptors are valid prior to executing this command. If any of the descriptors are invalid for the Estimation Filter 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 Estimation Filter 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** Data rate: **0x0001**	**0x0C**	**0x6A**

Reply ACK/NACK	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0xF1**	*Echo cmd:***0x0A** *Error code:***0x00**	**0xE9**	**0xBE**
Command Estimation Filter 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 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 IMU and Estimation Filter 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 – IMU 0x03 – Estimation Filter *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 IMU Stream ON	**0x75**	**0x65**	**0x0C**	**0x05**	**0x05**	**0x11**	Function(Apply):**0x01** Device (IMU): **0x01** Stream (ON): **0x01**	**0x04**	**0x1A**
Command IMU Stream OFF	**0x75**	**0x65**	**0x0C**	**0x05**	**0x05**	**0x11**	Function(Apply):**0x01** Device (IMU): **0x01** Stream (OFF): **0x00**	**0x03**	**0x19**
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)

Description	Save, Load, or Reset to Default the values for all device settings.		
Notes	*Possible function selector values:* 0x03 – Save current settings as startup settings 0x04 – Load saved startup settings 0x05 – Load factory default settings		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x02	0x30	U8 –Function Selector
Reply ACK/NACK	0x04	0xF1	U8 – echo the command byte U8 – error code (0:ACK, not 0:NACK)

Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command Startup Settings (Save All)	**0x75**	**0x65**	**0x0C**	**0x03**	**0x03**	**0x30**	*Fctn(Save):***0x03**	**0x1F**	**0x45**
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"

IMU Hard Iron Offset (0x0C, 0x3A)

Description	This command will read or write values to the magnetometer Hard Iron Offset Vector. For all functions except 0x01 and 0x06 (apply new settings), the new vector value is ignored. The offset value is subtracted from the scaled Mag value prior to output.
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 0x06 – Apply new settings with no ACK/NACK Reply
Notes	*Default values:*

	Hard Iron Offset: [0,0,0]		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x0F	0x3A	U8 – Function Selector float – X Hard Iron Offset float – Y Hard Iron Offset float – Z Hard Iron Offset
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)	0x0E	0x9C	float – current X Hard Iron Offset float – current Y Hard Iron Offset float – current Z Hard Iron Offset

Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command Hard Iron Offset	**0x75**	**0x65**	**0x0C**	**0x0F**	**0x0F**	**0x3A**	*Fctn(Apply):***0x01** *Offset Vector:* **0x00000000** **0x00000000** **0x00000000**	**0x3F**	**0x9F**
Reply field 1 ACK/NACK	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x3A** *Error code:* **0x00**	**0x19**	**0x1E**

Copy-Paste version of the command: "7565 0C0F 0F3A 0100 0000 0000 0000 0000 0000 003F 9F"

IMU Soft Iron Matrix (0x0C, 0x3B)

Description	This command will read or write values to the magnetometer Soft Iron Compensation Matrix. The values for this matrix are determined empirically by external software algorithms based on calibration data taken after the device is installed in its application. These values can be obtained and set by using the MicroStrain "MIP Iron Calibration" application. The matrix is applied to the scaled magnetometer vector prior to output.		
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 0x06 – Apply new settings with no ACK/NACK Reply *Default values:* Soft Iron Compensation Matrix (identity matrix; row order): [1,0,0][0,1,0][0,0,1]		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x27	0x3B	U8 – Function Selector float – $m_{1,1}$ float – $m_{1,2}$ float – $m_{1,3}$ float – $m_{2,1}$ float – $m_{2,2}$ float – $m_{2,3}$ float – $m_{3,1}$ float – $m_{3,2}$ float – $m_{3,3}$
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)*	0x26	0x9D	float – $m_{1,1}$ float – $m_{1,2}$ float – $m_{1,3}$ float – $m_{2,1}$ float – $m_{2,2}$ float – $m_{2,3}$ float – $m_{3,1}$ float – $m_{3,2}$ float – $m_{3,3}$

Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command *Soft Iron Matrix*	**0x75**	**0x65**	**0x0C**	**0x27**	**0x27**	**0x3B**	*Fctn(Apply):***0x01** *Comp Matrix:* **0x3F800000** **0x00000000** **0x00000000** **0x00000000**	**0xAD**	**0x59**

							0x3F800000 **0x00000000** **0x00000000** **0x00000000** **0x3F800000**		
Reply field 1 *ACK/NACK*	**0x75**	**0x65**	**0x0C**	**0x12**	**0x04**	**0xF1**	*Echo cmd:* **0x3B** *Error code:* **0x00**	**0x1A**	**0x20**

Copy-Paste version of the command: "7565 0C27 273B 013F 8000 0000 0000 0000 0000 0000 0000 003F 8000 0000 0000 0000 0000 0000 0000 003F 8000 00AD 59"

Accel Bias (0x0C, 0x37)

Advanced

Description	Set or read the current value of the IMU Accelerometer Bias Vector. For all functions except 0x01 and 0x06 (apply new settings), the new vector value is ignored. The bias value is subtracted from the scaled accelerometer value prior to output.		
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 0x06 – Apply new settings with no ACK/NACK Reply		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x0F	0x37	U8 – Function Selector float – X Accel Bias Value float – Y Accel Bias Value float – Z Accel Bias Value
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)	0x0E	0x9A	float – current X Accel Bias Value float – current Y Accel Bias Value float – current Z Accel Bias Value

Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command Accel Bias	**0x75**	**0x65**	**0x0C**	**0x0F**	**0x0F**	**0x37**	*Fctn(Apply):***0x01** *Field (Bias):* **0x00000000** **0x00000000** **0x00000000**	**0x3C**	**0x75**
Reply ACK/NACK	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0xF1**	*Echo cmd:***0x37** *Error code:***0x00**	**0x16**	**0x18**

Copy-Paste version of the command: "7565 0C0F 0F37 0100 0000 0000 0000 0000 0000 003C 75"

Gyro Bias (0x0C, 0x38)

Advanced

Description	Set or read the current value of the IMU Gyro Bias Vector. For all functions except 0x01 and 0x06 (apply new settings), the new vector value is ignored. The bias value is subtracted from the scaled Gyro value prior to output.		
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 0x06 – Apply new settings with no ACK/NACK Reply		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x0F	0x38	U8 – Function Selector float – X Gyro Bias Value float – Y Gyro Bias Value float – Z Gyro Bias Value
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)	0x0E	0x9B	float – current X Gyro Bias Value float – current Y Gyro Bias Value float – current Z Gyro Bias Value

Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command Gyro Bias	**0x75**	**0x65**	**0x0C**	**0x0F**	**0x0F**	**0x38**	*Fctn(Apply):***0x01** *Field (Bias):* **0x00000000** **0x00000000** **0x00000000**	**0x3D**	**0x83**
Reply ACK/NACK	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0xF1**	*Echo cmd:***0x38** *Error code:***0x00**	**0x17**	**0x1A**

Copy-Paste version of the command: "7565 0C0F 0F38 0100 0000 0000 0000 0000 0000 003D 83"

Capture Gyro Bias (0x0C, 0x39)

Description	This command will cause the IMU to sample its gyro sensors for the specified number of milliseconds. The resulting data will be used estimate its gyro bias error. The estimated gyro bias error will be automatically written to the Gyro Bias vector. The bias vector is not saved as a startup value. If you wish to save this vector, use the Gyro Bias command.		
Notes	Possible Sampling Time values: 1000 to 30000 milliseconds. (1 to 30 sec) Note: The IMU must be stationary during the execution of the Capture Gyro Bias Operation.		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x04	0x39	U16 – Sampling Time (milliseconds)
Reply ACK/NACK	0x04	0xF1	U8 – echo the command descriptor U8 – error code (0:ACK, not 0:NACK)
Reply field 2 (function = 2)	0x0E	0x9B	float – current X Gyro Bias Value float – current Y Gyro Bias Value float – current Z Gyro Bias Value

Example	*MIP Packet Header*				*Fields*			*Checksum*	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0x39**	Sampling Time: **0x2710**	**0x5E**	**0xE0**
Reply field 1 ACK/NACK	**0x75**	**0x65**	**0x0C**	**0x12**	**0x04**	**0xF1**	Echo cmd: **0x39** Error code: **0x00**		
Reply field 2 Bias Vector					**0x0E**	**0x9B**	Field (Bias): **0x00000000** **0x00000000** **0x00000000**	**0xCF**	**0x19**

Copy-Paste version of the command: "7565 0C04 0439 2710 5EE0"

Coning and Sculling Enable (0x0C, 0x3E)

Description	Set, read, or save the Coning and Sculling compensation Enable. This function sets the Coning and Sculling compensation Enable. For all functions except 0x01 (use new setting), the new parameter values are ignored.
Notes	*Possible function selector values:* 0x01 – Apply new setting 0x02 – Read back current setting 0x03 – Save current settings as startup setting 0x04 – Load saved startup setting 0x05 – Load factory default setting *The enable flag can be either:* 0x00 – disable the Coning and Sculling compensation. 0x01 – enable the Coning and Sculling compensation. *(default)*

Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x10	0x3E	U8 – Function Selector U8 – New Coning and Sculling enable setting
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)	0x03	0x9E	U8 – Current Coning and Sculling enable setting

Example	*MIP Packet Header*				*Fields*			*Checksum*	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command Enable Settings	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0x3E**	*Fctn (Apply):* **0x01** *Enable:* **0x01**	**2E**	**94**
Reply ACK/NACK	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x3E** *Error code:***0x00**	**1D**	**26**

Copy-Paste version of the command: "7565 0C04 043E 0101 2E94"

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"

Complementary Filter Settings (0x0C, 0x51)

Description	Configuration for the AHRS complementary filter. The Complementary Filter data outputs are supported in the IMU/AHRS Data set (0x80) to provide compatibility with the 3DM-GX3.		
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 up/north compensation enable values:* 0x00 – Disable 0x01 – Enable (default) *Range of up/north compensation time constants:* 1-1000 seconds, default = 10 seconds Values outside of the specified range for up/north compensation will be NACK'd. ⚠ The Complementary Filter provides attitude outputs (Matrix, Euler, Quaternion, Up, and North) that are independent of the Estimation Filter outputs. The CF outputs are calculated using the same algorithm as the 3DM-GX3 series of Inertial Devices. This provides drop-in compatibility that duplicates the performance of the 3DM-GX3. It is highly recommended that you transition to the EF outputs as they will provide better performance as well as compatibility with higher grade devices such as the 3DM-RQ1.		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x0D	0x51	U8 – Function selector U8 – Up compensation enable U8 – North compensation enable float – Up compensation time constant (sec) float – North compensation time constant (sec)
Reply ACK/NACK	0x04	0xF1	U8 – echo the command descriptor U8 – error code (0:ACK, not 0:NACK)
Reply field 2 (function = 2)	0x0C	0x97	U8 – Up compensation enable U8 – North compensation enable float – Up compensation time constant (sec) float – North compensation time constant (sec)

Examples	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command IMU Message Format (use new settings)	**0x75**	**0x65**	**0x0C**	**0x0D**	**0x0D**	**0x51**	Function Selector: **0x01** (Write) Up Compensation Enable: **0x01 (enable)** North Compensation Enable: **0x01 (enable)** Up Compensation Time Constant: **5.0 (sec)** North Compensation Time Constant: **5.0 (sec)**	**0xXX**	**0xXX**
Reply ACK/NACK	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0xF1**	*Echo cmd:***0x51** *Error code:***0x00**	**0x**	**0x**

Copy-Paste version of the commands: "7565 0C09 0951 0104 0100 0000 00"

Low-Pass Filter Settings (0x0C, 0x50)

<table>
<tr><td><b>Description</b></td><td colspan="3">Configuration for low-pass filter settings. The scaled gyro, scaled accel, scaled mag, and scaled pressure data quantities are <b>by default</b> filtered through a single-pole IIR low-pass filter which is configured with a -3dB cutoff frequency of half the reporting frequency (set by decimation factor in the IMU Message Format command) to prevent aliasing on a per data quantity basis. This advanced configuration command allows for the cutoff frequency to be configured independently of the data reporting frequency as well as allowing for a complete bypass of the digital low-pass filter for either or both scaled data quantities.</td></tr>
<tr><td><b>Notes</b></td><td colspan="3"><i>Possible function selector values:</i>
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
<i>Possible data type specifier:</i>
0x04 – Scaled accel data
0x05 – Scaled gyro data
0x06 – Scaled mag data (if applicable)
0x17 – Scaled pressure data (if applicable)
<i>Possible filter typevalues:</i>
0x01 – Single pole IIR low-pass filter
0x00 – Do not apply low-pass filter
<i>Manual filter bandwidth configuration:</i>
0x01 – Use user specified -3 dB cutoff frequency
0x00 – Automatically configure -3 dB cutoff frequency to half reporting rate
<i>-3 dB Cutoff Frequency:</i>
Cutoff Frequency value specified must be no greater than 500 Hz (0x01f4).
<i>**This value in a write command is ignored if Automatic Bandwidth is selected.</i>
<i>Reserved Byte:</i>
This byte is reserved for internal use and should be left in the 0x00 state</td></tr>
<tr><td><b>Field Format</b></td><td><i>Field Length</i></td><td><i>Field Descriptor</i></td><td><i>Field Data</i></td></tr>
</table>

Command	0x09	0x50	U8 – Function Selector U8 – Data Descriptor (0x04: Scaled Accel, 0x05 Scaled Gyro) U8 – Low-Pass Filter Type Type (0x01: IIR, 0x00 None) U8 – Manual/Auto -3 dB Cutoff Frequency Configuration U16 – -3 dB Cutoff Frequency U8 – Reserved Byte
Reply ACK/NACK	0x04	0xF1	U8 – echo the command descriptor U8 – error code (0:ACK, not 0:NACK)
Reply field 2 (function = 2)	0x08	0x8B	U8 – Data Descriptor (Scaled Accel, Scaled Gyro or Scaled Magnetometer) U8 – Filter (0x01: IIR Filter, 0x00: No Filter) U8 – Cutoff Frequency (0x00: Auto, 0x01: Manual) U16 – -3 dB Cutoff Frequency Hz U8 – Reserved

Examples	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command IMU Message Format (use new settings)	**0x75**	**0x65**	**0x0C**	**0x09**	**0x09**	**0x50**	Function: **0x01** Scaled Accel: **0x04** Enable Filter: **0x01** Automatic Cutoff Configuration: **0x00** -3 dB Cutoff Frequency: **0x0000** (ignored for automatic cutoff configuration) Reserved: **0x00**	**0x4C**	**0x6D**
Reply ACK/NACK	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0xF1**	*Echo cmd:***0x50** *Error code:***0x00**	**0x2F**	**0x4A**

Copy-Paste version of the commands: "7565 0C09 0950 0104 0100 0000 004E 80"

Device Status (0x0C, 0x64)

<table>
<tr><td><b>Description</b></td><td colspan="6">Get the device-specific status for the 3DM-GX4-25</td></tr>
<tr><td><b>Notes</b></td><td colspan="6">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-GX4-25 is always = 6234 (0x185A)). That is followed by a status selector byte which determines the type of data structure returned. In the case of the 3DM-GX4-25, 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:

<i>Possible Status Selector Values:</i>

0x01 – Basic Status Structure
0x02 – Diagnostic Status Structure</td></tr>
<tr><td><b>Field Format</b></td><td><i>Field Length</i></td><td><i>Field Descriptor</i></td><td colspan="4"><i>Field Data</i></td></tr>
<tr><td><i>Command</i></td><td>0x02</td><td>0x64</td><td colspan="4">U16-Device Model Number: set = 6234 (0x185A)
U8-Status Selector</td></tr>
<tr><td><i>Reply field 1
ACK/NACK Field</i></td><td>0x04</td><td>0xF1</td><td colspan="4">U8 – echo the command byte
U8 – error code (0:ACK, not 0:NACK)</td></tr>
<tr><td rowspan="5"><i>Reply field 2
Basic Device Status Field</i></td><td rowspan="5">0x0F</td><td rowspan="5">0x90</td><td><i>Binary Offset</i></td><td><i>Description</i></td><td><i>Data Type</i></td><td><i>Units</i></td></tr>
<tr><td>0</td><td>Echo of the Device Model Number</td><td>U16</td><td>N/A</td></tr>
<tr><td>2</td><td>Echo of the selector byte</td><td>U8</td><td>N/A</td></tr>
<tr><td>3</td><td>Status Flags (Reserved)</td><td>U32</td><td>N/A</td></tr>
<tr><td>7</td><td>System Timer (since start-up)</td><td>U32</td><td>milliseconds</td></tr>
<tr><td rowspan="7"><i>Reply field 2
Diagnostic Device Status Field</i></td><td rowspan="7">0x4B</td><td rowspan="7">0x90</td><td><i>Binary Offset</i></td><td><i>Description</i></td><td><i>Data Type</i></td><td><i>Units</i></td></tr>
<tr><td>0</td><td>Echo of the Device Model Number</td><td>U16</td><td>N/A</td></tr>
<tr><td>2</td><td>Echo of the selector byte</td><td>U8</td><td>N/A</td></tr>
<tr><td>3</td><td>Status Flags (Reserved)</td><td>U32</td><td>N/A</td></tr>
<tr><td>7</td><td>System Timer (since start-up)</td><td>U32</td><td>milliseconds</td></tr>
<tr><td>11</td><td>Number of 1PPS Pulses</td><td>U32</td><td>Count</td></tr>
<tr><td>15</td><td>Last 1PPS (System Timer)</td><td>U32</td><td>milliseconds</td></tr>
</table>

			19	IMU Stream Enabled	U8	1 – on 0 – off
			20	Estimation Filter Stream Enabled	U8	1 – on 0 – off
			21	Outgoing IMU Stream Dropped Packet Count	U32	Count
			25	Outgoing Estimation Filter Stream Dropped Packet Count	U32	Count
			29	Number of bytes written to com port	U32	Count
			33	Number of bytes read from com port	U32	Count
			37	Number of overruns when writing to com port	U32	Count
			41	Number of overruns when reading from com port	U32	Count
			45	Number of bytes written to USB port	U32	Count
			49	Number of bytes read from USB port	U32	Count
			53	Number of overruns when writing to USB port	U32	Count
			57	Number of overruns when reading from USB port	U32	Count
			61	Number of IMU message parsing errors	U32	Count
			65	Total IMU messages read	U32	Count
			69	Last IMU message read (System Timer)	U32	Millisecond

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 # (6234): **0x185A** Status Selector (basic status): **0x01**	**0xC7**	**0x5D**
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)					**0x0D**	**0x90**	Echo Model#: **0x185A** Echo Selector: **0x01** .... Additional Data	**0x##**	**0x##**

Copy-Paste version of the command: "7565 0C05 0564 185A 01C7 5D"

Estimation Filter Commands

Reset Filter (0x0D, 0x01)

Description	Reset the Estimation 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)

<table>
<tr><td><b>Description</b></td><td colspan="9">Set the initial attitude.</td></tr>
<tr><td><b>Notes</b></td><td colspan="9">This command can only be issued in the “INIT” state and should be used with a good estimate of the sensor’s attitude. The Euler Angles are the sensor body frame with respect to the local NED frame.
The valid input ranges are as follows:
Roll: [-π, π]
Pitch: $[-\frac{\pi}{2}, \frac{\pi}{2}]$
Yaw: [-π, π]</td></tr>
<tr><td><b>Field Format</b></td><td colspan="2"><i>Field Length</i></td><td colspan="2"><i>Field Descriptor</i></td><td colspan="5"><i>Field Data</i></td></tr>
<tr><td><i>Command</i></td><td colspan="2">0x0E</td><td colspan="2">0x02</td><td colspan="5">Float – Roll (radians)
Float – Pitch (radians)
Float – Heading (radians)</td></tr>
<tr><td><i>Reply ACK/NACK</i></td><td colspan="2">0x04</td><td colspan="2">0xF1</td><td colspan="5">U8 – echo the command byte
U8 – error code (0:ACK, not 0:NACK)</td></tr>
<tr><td rowspan="2"><b>Example</b></td><td colspan="4"><i>MIP Packet Header</i></td><td colspan="3"><i>Fields</i></td><td colspan="2"><i>Checksum</i></td></tr>
<tr><td><i>Sync1</i></td><td><i>Sync2</i></td><td><i>Desc Set</i></td><td><i>Payload Length</i></td><td><i>Field Length</i></td><td><i>Field Desc.</i></td><td><i>Field Data</i></td><td><i>MSB</i></td><td><i>LSB</i></td></tr>
<tr><td><i>Command</i></td><td><b>0x75</b></td><td><b>0x65</b></td><td><b>0x0D</b></td><td><b>0E</b></td><td><b>0E</b></td><td><b>02</b></td><td><i>Roll:</i>0x00000000 <b>(0.0f)</b>
<i>Pitch:</i>0x00000000 <b>(0.0f)</b>
<i>Heading:</i>0x00000000 <b>(0.0f)</b></td><td><b>0x05</b></td><td><b>0x6F</b></td></tr>
<tr><td><i>Reply ACK/NACK</i></td><td><b>0x75</b></td><td><b>0x65</b></td><td><b>0x0D</b></td><td><b>0x04</b></td><td><b>0x04</b></td><td><b>0xF1</b></td><td><i>Echo cmd:</i> <b>0x02</b>
<i>Error code:</i> <b>0x00</b></td><td><b>0xE2</b></td><td><b>0xB4</b></td></tr>
</table>

Copy-Paste version of the command: “7565 0D0E0E02 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 device 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. The valid input range for heading is [-π, π].		
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 with Magnetometer (0x0D, 0x04)

Description	Set the initial attitude using the embedded magnetometer.		
Notes	This command can only be issued in the "INIT" state. The device will use the on-board magnetometer to initialize the attitude. The user may supply a declination angle for the local magnet field conditions ***Special Note:*** In the presence of significant magnetic interference, the magnetometer heading value can be wildly off, causing the filter to initialize improperly.		
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"

Tare Orientation (0x0D, 0x21)

Description	This function uses the current device orientation relative to the NED frame as the current sensor to vehicle transformation. This command is provided as a convenient way to set the sensor to vehicle frame transformation.		
Notes	*Possible function selector values:* 0x01 – Use new settings 0x03 – Save current settings as startup settings 0x04 – Load saved startup settings 0x05 – Reset to factory default settings *Possible axis bitfield values:* 0x00 – Reset all axis 0x01 – Tare the roll axis 0x02 – Tare the pitch axis 0x04 – Tare the yaw axis Example Combinations: 0x03 – Tare the roll and pitch axis 0x07 – Tare all 3 axis		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x04	0x21	U8 – Function Selector U8 – Tare Axis Bitfield
Reply ACK/NACK	0x04	0xF1	U8 – echo the command descriptor 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**	**0x04**	**0x04**	**0x21**	*Fctn (Apply):* **0x01** *X*:0x07 (**All axis)**	**0x18**	**0x49**
Reply ACK/NACK	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x21** *Error code:***0x00**	**0x**	**0x**

Copy-Paste version of the command: "7565 0D04 0421 0107 1849"

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

Description	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 device 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 transformation affects the following output quantities: IMU: Scaled Acceleration Scaled Gyro Delta Theta Delta Velocity Estimation Filter: Estimated Orientation, Quaternion Estimated Orientation, Matrix Estimated Orientation, Euler Angles Estimated Linear Acceleration Estimated Angular Rate Estimated Gravity Vector		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x0F	0x11	U8 – Function Selector Float – Roll Angle (radians) Float – Pitch Angle (radians) Float – Yaw Angle (radians)
Reply ACK/NACK	0x04	0xF1	U8 – echo the command descriptor U8 – error code (0:ACK, not 0:NACK)
Reply field 2 (function = 2)	0x0E	0x81	Float – Roll Angle (radians) Float – Pitch Angle (radians) Float – Yaw Angle (radians)

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**	**0x11**	*Fctn (Apply):* **0x01** *Roll:*0x00000000 **(0.0f)** *Pitch:*0x00000000 **(0.0f)** *Yaw:*0x00000000 **(0.0f)**	**0x17**	**0x72**
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"

Estimation Control Flags (0x0D, 0x14)

Description	Controls which parameters are estimated by 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 *Available Flags :* 0x0001 – Enable Gyro Bias Estimation (Recommended) *Examples :* 0xFFFF – Enable all 0xFFFE – Disable Gyro Bias Estimation (note: Any bit without a designated function should be set to 1 for future compatibility.)		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x05	0x14	U8 – Function Selector U16 – Estimation Control Flags
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 – Estimation Control Flags

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** *Flags:*0xFFFF **(Enable all states)**	**0x04**	**0x27**
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 01FF FF04 27”

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 internal magnetometer 0x03 – Use external heading updates		
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* *Note: Auto-initialization on the GX4-25 can only take place under one of the following conditions: 1) The heading source is set to external and the user is providing external heading data 2) The heading source is set to internal magnetometer and the magnetometer is producing valid data.		
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 ACK/NACK	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x19** *Error code:***0x00**	**0xF9**	**0xE2**

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

Gyroscope Noise Standard Deviation (0x0D, 0x1B)

Description	Set the expected gyroscope 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 The noise value represents measurement noise in the GX3-25 Estimation Filter. Changing this value modifies how the filter responds to dynamic input and can be used to tune the performance of the filter. Default values provide good performance for most laboratory conditions.		
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. 00016f)**	**0xXX**	**0xXX**
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

Accelerometer Noise Standard Deviation (0x0D, 0x1A)

Description	Set the expected accelerometer 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 The noise value represents measurement noise in the GX3-25 Estimation Filter. Changing this value modifies how the filter responds to dynamic input and can be used to tune the performance of the filter. Default values provide good performance for most laboratory conditions. ⚠ If accelerometer adaptive measurements are enabled, this value will be overridden.

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 1A013CA3D70A3CA3D70A3CA3D760A3”

Magnetometer Noise Standard Deviation (0x0D, 0x42)

Description	Set the expected magnetometer 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 The noise value represents measurement noise in the GX3-25 Estimation Filter. Changing this value modifies how the filter responds to dynamic input and can be used to tune the performance of the filter. Default values provide good performance for most laboratory conditions. ⚠ If magnetometer adaptive measurements are enabled, this value will be overridden.

Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x0F	0x42	U8 – Function Selector Float – X Mag Noise 1-sigma (gauss) Float – Y Mag Noise 1-sigma (gauss) Float – Z Mag Noise 1-sigma (gauss)
Reply ACK/NACK	0x04	0xF1	U8 – echo the command descriptor U8 – error code (0:ACK, not 0:NACK)
Reply field 2 (function = 2)	0x0E	0xB1	Float – X Mag Noise 1-sigma (gauss) Float – Y Mag Noise 1-sigma (gauss) Float – Z Mag Noise 1-sigma (gauss)

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**	**0x42**	*Fctn (Apply):* **0x01** *X:*(**0.02f)** *Y:*(**0.02f)** *Z:*(**0.02f)**	**0x**	**0x**
Reply ACK/NACK	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x42** *Error code:***0x00**	**0x22**	**0x34**

Copy-Paste version of the command: “”

Enable/Disable Measurements (0x0D, 0x41)

Description	Allows the user to control accelerometer and magnetometer measurement updates		
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) – Accelerometer Measurements (1 – enable, 0 – disable) Bit 1 (0x00000010) – Magnetometer Measurements (1 – enable, 0 – disable)		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x05	0x41	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	0xB0	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**	**0x41**	*Fctn (Apply):* **0x01** *X*:0x0003 (**Enable Accel/Mag measurements)**	**0x36**	**0xE1**
Reply *ACK/NACK*	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x41** *Error code:***0x00**	**0x21**	**0x32**

Copy-Paste version of the command: “7565 0D05 0541 0100 0336 E1”

Declination Source (0x0D, 0x43)

Description	Set/Get the local declination angle source. This can be used to correct for the difference in magnetic and true north. Normally, the device reports heading with-respect-to magnetic north, but when an accurate declination angle is provided, the device will report heading with respect to true north.		
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 declination sources:* 0x01 – None 0x02 – World Magnetic Model 0x03 – Manual Option description: None: orientation information will be reported with respect to magnetic north. World Magnetic Model: The declination will be sourced from the device's internal world magnetic model. This requires a reference position be set via the Set Reference Position (0x0D, 0x26) command. For maximum accuracy, the user should also supply the current GPS time via the GPS Time broadcast command (0x01, 0x72). If no time is provided, the device uses the first valid date for the model (Jan 1st, 2010.) Manual: The user provides the declination angle. The device does not validate this angle and it is therefore up to the user to select the correct value.		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x08	0x43	U8 – Function Selector U8 – Declination Source float – Manual Declination angle (radians, only required if source = Manual)
Reply ACK/NACK	0x04	0xF1	U8 – echo the command descriptor U8 – error code (0:ACK, not 0:NACK)
Reply field 2 (function = 2)	0x07	0xB2	U8 – Declination Source float – Declination angle (radians)
Example	*MIP Packet Header*	*Fields*	*Checksum*

	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command	**0x75**	**0x65**	**0x0D**	**0x08**	**0x08**	**0x43**	*Fctn (Apply):* **0x01** *Source (Manual):* **0x03** *Angle:*0x00000000 (**0.0f)**	**0x**	**0x**
Reply ACK/NACK	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x43** *Error code:***0x00**	**0x23**	**0x36**

Copy-Paste version of the command: “”

External Heading Update (0x0D, 0x17)

<table>
<tr><td><b>Description</b></td><td colspan="3">Trigger a filter update step using external heading information

<b><i>The heading must be the sensor frame with respect to the NED frame.</i></b></td></tr>
<tr><td><b>Notes</b></td><td colspan="3">The heading update control must be set to external for this command to update the filter; it will be ignored/NACK'd otherwise.

Angle uncertainties of 0.0 will be NACK'd.

Possible Heading Type Commands:

0x01 – True Heading*
0x02 – Magnetic Heading*

<b><i>*Note:</i></b>

- On the -25 model, if the declination source (0x0D, 0x43) is not valid, <b><i>true heading updates</i></b> will be NACK'd.

On the -25 model, if the declination source is invalid, <b><i>magnetic heading updates</i></b> will be NACK'd.</td></tr>
<tr><td><b>Field Format</b></td><td><i>Field Length</i></td><td><i>Field Descriptor</i></td><td><i>Field Data</i></td></tr>
<tr><td><i>Command</i></td><td>0x0B</td><td>0x17</td><td>Float – Heading Angle (radians, true north, +- PI)
Float – Heading Angle Uncertainty (radians, 1-sigma)
U8 – Heading type (1 – true, 2 – magnetic)</td></tr>
<tr><td><i>Reply
ACK/NACK</i></td><td>0x04</td><td>0xF1</td><td>U8 – echo the command byte
U8 – error code (0:ACK, not 0:NACK)</td></tr>
</table>

Example	*MIP Packet Header*				*Fields*			*Checksum*	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command	**0x75**	**0x65**	**0x0D**	**0x0B**	**0x0B**	**0x17**	*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

External Heading Update with Timestamp (0x0D, 0x1F)

Description	Trigger a filter update step using external heading information that is time-tagged with a specific GPS time if provided, local timer time otherwise. ***The heading must be the sensor frame with respect to the NED frame.***
Notes	This is more accurate than the External Heading Update (0x0D, 0x17) and should be used in applications where the vehicle heading experiences high angular rate, which may cause significant error in the applied measurement due to the sampling, transmission, and processing time required for the command. Accurate time-stamping of the heading information is important. Angle uncertainties of 0.0 will be NACK'd. Possible Heading Type Commands: 0x01 – True Heading* 0x02 – Magnetic Heading* ***Note:*** - On the -25 model, if the declination source (0x0D, 0x43) is not valid, ***true heading updates*** will be NACK'd. - On the -25 model, if the declination source is invalid, ***magnetic heading updates*** will be NACK'd.

Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x15	0x1F	Double – GPS TOW (time-of-week, seconds) U16 – GPS week number 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**	**0x15**	**0x15**	**0x1F**	*GPS TOW:* ***30,000.0*** *GPS Week Number:* ***1700*** *Angle:***0.0f** *Angle Sigma:***0.01f** Heading Type: **0x01**(True)	**0xXX**	**0xXX**
Reply	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x1F**	**0xFF**	**0xEE**

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

Copy-Paste version of the command: N/A

Zero Angular Rate Update Control (0x0D, 0x20)

<table>
<tr><td><b>Description</b></td><td colspan="9">Control the use of zero angular rate updates.</td></tr>
<tr><td><b>Notes</b></td><td colspan="9"><i>Possible function selector values:</i>
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 zero angular rate update is triggered when the scalar magnitude of the angular rate vector is equal-to or less than the threshold value. The device will NACK threshold values that are less than zero (i.e. negative.)</td></tr>
<tr><td><b>Field Format</b></td><td colspan="2"><i>Field Length</i></td><td colspan="2"><i>Field Descriptor</i></td><td colspan="5"><i>Field Data</i></td></tr>
<tr><td><i>Command</i></td><td colspan="2">0x08</td><td colspan="2">0x20</td><td colspan="5">U8 – Function Selector
U8 –Enable Value (0 – disable, 1 – enable)
Float –Threshold (rad/s)</td></tr>
<tr><td><i>Reply
ACK/NACK</i></td><td colspan="2">0x04</td><td colspan="2">0xF1</td><td colspan="5">U8 – echo the command descriptor
U8 – error code (0:ACK, not 0:NACK)</td></tr>
<tr><td><i>Reply field 2
(function = 2)</i></td><td colspan="2">0x07</td><td colspan="2">0x8E</td><td colspan="5">U8– Enable Value
Float – ZUPT threshold (rad/s)</td></tr>
<tr><td rowspan="2"><b>Example</b></td><td colspan="4"><i>MIP Packet Header</i></td><td colspan="3"><i>Fields</i></td><td colspan="2"><i>Checksum</i></td></tr>
<tr><td><i>Sync1</i></td><td><i>Sync2</i></td><td><i>Desc Set</i></td><td><i>Payload Length</i></td><td><i>Field Length</i></td><td><i>Field Desc.</i></td><td><i>Field Data</i></td><td><i>MSB</i></td><td><i>LSB</i></td></tr>
<tr><td><i>Command</i></td><td><b>0x75</b></td><td><b>0x65</b></td><td><b>0x0D</b></td><td><b>0x08</b></td><td><b>0x08</b></td><td><b>0x20</b></td><td><i>Fctn (Apply):</i> <b>0x01</b>
<i>Enable:</i>0x01 <b>(Enable)</b>
Threshold: 0x00000000 <b>(0.0f)</b></td><td><b>0x19</b></td><td><b>0xC8</b></td></tr>
<tr><td><i>Reply
ACK/NACK</i></td><td><b>0x75</b></td><td><b>0x65</b></td><td><b>0x0D</b></td><td><b>0x04</b></td><td><b>0x04</b></td><td><b>0xF1</b></td><td><i>Echo cmd:</i> <b>0x20</b>
<i>Error code:</i><b>0x00</b></td><td><b>0x00</b></td><td><b>0xF0</b></td></tr>
</table>

Copy-Paste version of the command: “7565 0D08 0820 0101 00000000 19C8”

Commanded Zero-Angular Rate Update (0x0D, 0x23)

Description	Perform a commanded zero-angular rate update.		
Notes			
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x02	0x23	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**	**0x23**		**0x0E**	**0x18**
Reply ACK/NACK	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x23** *Error code:* **0x00**	**0x03**	**0xF6**

Copy-Paste version of the command: "7565 0D02 0222 0D17"

Accelerometer Magnitude Error Adaptive Measurement (0x0D, 0x44)

Description	Set the accelerometer magnitude error adaptive measurement parameters. 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 Adaptive measurements can be enabled/disabled without the need for providing the additional parameters. In this case, only the function selector and enable value are required; all other parameters will remain at their previous values.		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x1C (28)	0x44	U8 – Function Selector U8 – Enable (0 – Disable, 1 – Enable) Float – Low-pass filter cutoff frequency (Hz) Float – Low Limit (meters/second^2) Float – High Limit (meters/second^2) Float – Low Limit Uncertainty, 1-Sigma (meters/second^2) Float – High Limit Uncertainty, 1-Sigma (meters/second^2) Float – Minimum Uncertainty, 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)	0x1B (27)	0xB3	U8 – Enable (0 – Disable, 1 – Enable) Float – Low-pass filter cutoff frequency (Hz) Float – Low Limit (meters/second^2) Float – High Limit (meters/second^2) Float – Low Limit Uncertainty, 1-Sigma (meters/second^2) Float – High Limit Uncertainty, 1-Sigma (meters/second^2) Float – Minimum Uncertainty, 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**	**0x1C**	**0x1C**	**0x44**	*Fctn (Apply):* **0x01** *Enable:* **0x01** *Freq (Hz):*(**1.0f)** *Low Limit:*(**-0.2)** *High Limit:*(**0.2f)** *Low Limit 1-sigma:*(**0.2f)** *High Limit 1-sigma:*(**0.2f)** *Min 1-sigma:*(**0.004f)**	**0x**	**0x**
Reply ACK/NACK	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x44** *Error code:***0x00**	**0x**	**0x**

Copy-Paste version of the command: “”

Magnetometer Magnitude Error Adaptive Measurement (0x0D, 0x45)

Description	Set the magnetometer magnitude error adaptive measurement parameters. 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 Adaptive measurements can be enabled/disabled without the need for providing the additional parameters. In this case, only the function selector and enable value are required; all other parameters will remain at their previous values. ⚠ The local magnetic field strength must be known to use this feature. This can be obtained by setting the reference position for the device (0x0D, 0x26). It is also important to do an accurate Soft & Hard Iron calibration with the device mounted as it will be used in normal operation.		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x1C (28)	0x45	U8 – Function Selector U8 – Enable (0 – Disable, 1 – Enable) Float – Low-pass filter cutoff frequency (Hz) Float – Low Limit (Gauss) Float – High Limit (Gauss) Float – Low Limit Uncertainty, 1-Sigma (Gauss) Float – High Limit Uncertainty, 1-Sigma (Gauss) Float – Minimum Uncertainty, 1-Sigma (Gauss)
Reply ACK/NACK	0x04	0xF1	U8 – echo the command descriptor U8 – error code (0:ACK, not 0:NACK)
Reply field 2 (function = 2)	0x1B (27)	0xB4	U8 – Enable (0 – Disable, 1 – Enable) Float – Low-pass filter cutoff frequency (Hz) Float – Low Limit (Gauss) Float – High Limit (Gauss) Float – Low Limit Uncertainty, 1-Sigma (Gauss) Float – High Limit Uncertainty, 1-Sigma (Gauss) Float – Minimum Uncertainty, 1-Sigma (Gauss)

Example	*MIP Packet Header*				*Fields*			*Checksum*	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*

Command	**0x75**	**0x65**	**0x0D**	**0x1C**	**0x1C**	**0x45**	*Fctn (Apply):* **0x01** *Enable:* **0x01** *Freq (Hz):*(**1.0f)** *Low Limit:*(**-0.2)** *High Limit:*(**0.2f)** *Low Limit 1-sigma:*(**0.2f)** *High Limit 1-sigma:*(**0.2f)** *Min 1-sigma:*(**0.01f)**	**0x**	**0x**
Reply ACK/NACK	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x45** *Error code:***0x00**	**0x**	**0x**

Copy-Paste version of the command: “”

Magnetometer Dip Angle Error Adaptive Measurement (0x0D, 0x46)

Description	Set the magnetometer dip angle error adaptive measurement parameters. 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 The adaptive function is linear, taking inputs from 0 to the stated high limit (in radians) and outputs from the minimum uncertainty to the high-limit uncertainty (in Gauss.) Adaptive measurements can be enabled/disabled without the need for providing the additional parameters. In this case, only the function selector and enable value are required; all other parameters will remain at their previous values. ⚠ The local magnetic field inclination (dip angle) must be known to use this feature. This can be obtained by setting the reference position for the device (0x0D, 0x26). It is also important to do an accurate Soft & Hard Iron calibration with the device mounted as it will be used in normal operation.

Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x14 (20)	0x46	U8 – Function Selector U8 – Enable (0 – Disable, 1 – Enable) Float – Low-pass filter cutoff frequency (Hz) Float – High Limit (Radians) Float – High Limit Uncertainty, 1-Sigma (Gauss) Float – Minimum Uncertainty, 1-Sigma (Gauss)
Reply ACK/NACK	0x04	0xF1	U8 – echo the command descriptor U8 – error code (0:ACK, not 0:NACK)
Reply field 2 (function = 2)	0x13 (19)	0xB5	U8 – Enable (0 – Disable, 1 – Enable) Float – Low-pass filter cutoff frequency (Hz) Float – High Limit (Radians) Float – High Limit Uncertainty, 1-Sigma (Gauss) Float – Minimum Uncertainty, 1-Sigma (Gauss)

Example	*MIP Packet Header*				*Fields*			*Checksum*	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*

Command	**0x75**	**0x65**	**0x0D**	**0x14**	**0x14**	**0x46**	*Fctn (Apply):* **0x01** *Enable:* **0x01** *Freq (Hz):*(**10.0f)** *High Limit (rad):*(**0.3f)** *High Limit 1-sigma:*(**0.2f)** *Min 1-sigma:*(**0.01f)**	**0x**	**0x**
Reply ACK/NACK	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x46** *Error code:***0x00**	**0x**	**0x**

Copy-Paste version of the command: “”

Set Reference Position (0x0D, 0x26)

Description	Set the Lat/Long/Alt reference position for the sensor.		
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 position is used by the sensor to calculate the WGS84 gravity and WMM2010 magnetic field parameters.		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x1C (28)	0x26	U8 – Function Selector U8 – Enable (0 – disable, 1 – enable) Double – Latitude (decimal degrees) Double – Longitude (decimal degrees) Double – Altitude (meters)
Reply ACK/NACK	0x04	0xF1	U8 – echo the command descriptor U8 – error code (0:ACK, not 0:NACK)
Reply field 2 (function = 2)	0x1B (27)	0x90	U8 – Enable (0 – disable, 1 – enable) Double – Latitude (decimal degrees) Double – Longitude (decimal degrees) Double – Altitude (meters)

Example	*MIP Packet Header*				*Fields*			*Checksum*	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command	**0x75**	**0x65**	**0x0D**	**0x1C**	**0x1C**	**0x26**	*Fctn (Apply):* **0x01** *Enable:* **0x01** *Latitude (deg):*(**44.437f)** *Longitude (deg):*(**-73.106)** *Altitude (m):*(**155.0f)**	**0x**	**0x**
Reply ACK/NACK	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x26** *Error code:***0x00**	**0x**	**0x**

Copy-Paste version of the command: “”

System Commands

The System Command set provides a set of advanced commands that are specific to devices such as the 3DM-GX4-25 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-GX4-25, you may switch into a mode that talks directly to the internal 3DM-GX4-10 IMU.

Communication Mode (0x7F, 0x10)

Advanced

<table>
<tr><td><b>Description</b></td><td colspan="3">Set, read, or save the device communication mode. This will change the communications protocol to and from “Estimation Filter” mode to “IMU Direct” (MIP IMU protocol.) 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><b>Notes</b></td><td colspan="3"><i>Possible function selector values:</i>
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
<i>Possible Communications Modes:</i>
<table><tr><th>Value</th><th>Mode</th><th>Protocol(s)</th></tr><tr><td>0x01</td><td>Standard</td><td>3DM-GX4-25 MIP Packet <i>(default)</i></td></tr><tr><td>0x02</td><td>IMU Direct</td><td>MIP IMU</td></tr></table></td></tr>
<tr><td><b>Field Format</b></td><td><i>Field Length</i></td><td><i>Field Descriptor</i></td><td><i>Field Data</i></td></tr>
<tr><td><i>Command</i></td><td>0x04</td><td>0x10</td><td>U8 –Function Selector
U8 –New Communications Mode</td></tr>
<tr><td><i>Reply field 1 ACK/NACK</i></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><i>Reply field 2 (function = 2)</i></td><td>0x03</td><td>0x90</td><td>U8 –Current Communications Mode</td></tr>
<tr><td><b>Example</b></td><td colspan="3"><table><tr><td colspan="4">MIP Packet Header</td><td colspan="3">Command/Reply Fields</td><td colspan="2">Checksum</td></tr><tr><td><i>Sync1</i></td><td><i>Sync2</i></td><td><i>Desc Set</i></td><td><i>Payload Length</i></td><td><i>Field Length</i></td><td><i>Field Desc.</i></td><td><i>Field Data</i></td><td><i>MSB</i></td><td><i>LSB</i></td></tr></table></td></tr>
</table>

Command COM Mode	**0x75**	**0x65**	**0x7F**	**0x04**	**0x04**	**0x10**	*Fctn(USE)*:**0x01** New mode (*IMU 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"

Data Reference

IMU Data

Scaled Accelerometer Vector (0x80, 0x04)

<table>
<tr><td><b>Description</b></td><td colspan="6">Scaled Accelerometer Vector</td></tr>
<tr><td><b>Notes</b></td><td colspan="6">This is a vector quantifying the direction and magnitude of the acceleration that the 3DM-GX4 ® is exposed to. This quantity is derived from Raw Accelerometer, but is fully temperature compensated and scaled into physical units of <i>g</i> (1 <i>g</i> = 9.80665 m/sec^2). It is expressed in the sensor coordinate frame.</td></tr>
<tr><td rowspan="5"><b>Field Format</b></td><td><i>Field Length</i></td><td><i>Data Descriptor</i></td><td colspan="4"><i>Message Data</i></td></tr>
<tr><td rowspan="4">14 (0x0E)</td><td rowspan="4">0x04</td><td><i>Binary Offset</i></td><td><i>Description</i></td><td><i>Data Type</i></td><td><i>Units</i></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><b>Description</b></td><td colspan="6">Scaled Gyro Vector</td></tr>
<tr><td><b>Notes</b></td><td colspan="6">This is a vector quantifying the rate of rotation (angular rate) of the 3DM-GX4®. 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 the sensor coordinate frame.</td></tr>
<tr><td rowspan="5"><b>Field Format</b></td><td><i>Field Length</i></td><td><i>Data Descriptor</i></td><td colspan="4"><i>Message Data</i></td></tr>
<tr><td rowspan="4">14 (0x0E)</td><td rowspan="4">0x05</td><td><i>Binary Offset</i></td><td><i>Description</i></td><td><i>Data Type</i></td><td><i>Units</i></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. This quantity is derived from the Raw Magnetometer quantities, but is fully temperature compensated and scaled into units of Gauss. It is expressed in the sensor coordinate frame.					
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

Scaled Ambient Pressure (0x80, 0x17)

Description	Scaled Ambient Pressure					
Notes	This is a scalar which gives the instantaneous ambient pressure reading. This quantity is fully temperature compensated and scaled into units of milliBar.					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
			Binary Offset	*Description*	*Data Type*	*Units*
	06 (0x06)	0x17	0	Ambient Pressure	float	milliBar

Delta Theta Vector (0x80, 0x07)

Description	Time integral of angular rate.					
Notes	This is a vector which gives the time integral of Angular Rate. It is expressed in terms of the sensor frame 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 Acceleration. It is expressed in terms of the sensor frame 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

CF Orientation Matrix (0x80, 0x09)

Description	3 x 3 Orientation Matrix *M* *This value is produced by the Complementary Filter fusion algorithm*

Notes

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

Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
			Binary Offset	*Description*	*Data Type*	*Units*
	38 (0x26)	0x09	0	M_{11}	float	*n/a*
			4	M_{12}	float	*n/a*
			8	M_{13}	float	*n/a*
			12	M_{21}	float	*n/a*
			16	M_{22}	float	*n/a*
			20	M_{23}	float	*n/a*
			24	M_{31}	float	*n/a*
			28	M_{32}	float	*n/a*
			32	M_{33}	float	*n/a*

CF Quaternion (0x80, 0x0A)

<table>
<tr><td><b>Description</b></td><td colspan="6">4 x 1 quaternion <i>Q</i>.
<i>This value is produced by the Complementary Filter fusion algorithm</i></td></tr>
<tr><td><b>Notes</b></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^{-1} • V_E • Q
<i>Where: V_IL is a vector expressed in the 3DM-GX3®'s local coordinate system.</i>
<i>V_E is the same vector expressed in the stationary, earth-fixed coordinate system</i></td></tr>
<tr><td rowspan="6"><b>Field Format</b></td><td><i>Field Length</i></td><td><i>Data Descriptor</i></td><td colspan="4"><i>Message Data</i></td></tr>
<tr><td rowspan="5">18 (0x12)</td><td rowspan="5">0x0A</td><td><i>Binary Offset</i></td><td><i>Description</i></td><td><i>Data Type</i></td><td><i>Units</i></td></tr>
<tr><td>0</td><td>q_0</td><td>float</td><td><i>n/a</i></td></tr>
<tr><td>4</td><td>q_1</td><td>float</td><td><i>n/a</i></td></tr>
<tr><td>8</td><td>q_2</td><td>float</td><td><i>n/a</i></td></tr>
<tr><td>12</td><td>q_3</td><td>float</td><td><i>n/a</i></td></tr>
</table>

CF Euler Angles (0x80, 0x0C)

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

CF Stabilized Mag Vector (North) (0x80, 0x10)

<table>
<tr><td><b>Description</b></td><td colspan="6">Gyro stabilized estimated vector for geomagnetic vector.
<i>This value is produced by the Complementary Filter fusion algorithm</i></td></tr>
<tr><td><b>Notes</b></td><td colspan="6">This is a vector which represents the complementary filter’s best estimate of the geomagnetic field direction (magnetic north). In the absence of magnetic interference, it should be equal to <i>Magnetometer</i>.   When transient magnetic interference is present, <i>Magnetometer</i> will be subject to transient (possibly large) errors.   The IMU/AHRS complementary filter computes <i>Stabilized North</i> which is its estimate of the geomagnetic field vector only, even thought the system may be exposed to transient magnetic interference.   Note that sustained magnetic interference cannot be adequately compensated for by the complementary filter.</td></tr>
<tr><td rowspan="5"><b>Field Format</b></td><td><i>Field Length</i></td><td><i>Data Descriptor</i></td><td colspan="4"><i>Message Data</i></td></tr>
<tr><td rowspan="4">14 (0x0E)</td><td rowspan="4">0x10</td><td><i>Binary Offset</i></td><td><i>Description</i></td><td><i>Data Type</i></td><td><i>Units</i></td></tr>
<tr><td>0</td><td>X Stab Mag</td><td>Float</td><td>Gauss</td></tr>
<tr><td>4</td><td>Y Stab Mag</td><td>Float</td><td>Gauss</td></tr>
<tr><td>8</td><td>Z Stab Mag</td><td>Float</td><td>Gauss</td></tr>
</table>

CF Stabilized Accel Vector (Up) (0x80, 0x11)

Description	Gyro stabilized estimated vector for the gravity vector. *This value is produced by the Complementary Filter fusion algorithm*					
Notes	This is a vector which represents the IMU/AHRS complementary filter's best estimate of the vertical direction. Under stationary conditions, it should be equal to Accel. In dynamic conditions, Accel will be sensitive to both gravitational acceleration as well as linear acceleration. The Complementary filter computes Stab Accel which is its estimate of the gravitation acceleration only, even thought the system may be exposed to significant linear acceleration.					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
			Binary Offset	*Description*	*Data Type*	*Units*
	14 (0x0E)	0x11	0	X Stab Accel	Float	g
			4	Y Stab Accel	Float	g
			8	Z Stab Accel	Float	g

GPS Correlation Timestamp (0x80, 0x12)

Description	GPS correlation timestamp.
Notes	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 IMU packets with the GPS packets. It is identical to the GPS Time record except the flags are defined specifically for the IMU. When the GPS Time Initialized flag is asserted, the GPS Time and IMU 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 update was presented. If this flag is not asserted, it means that the IMU 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 Estimation Filter and IMU timestamps become free running and may slowly drift away from each other. If the timestamp clocks have drifted apart, then there will be a jump in the timestamp when the GPS Time Update is issued, reflecting the amount of drift of the clocks. See the Data Synchronicity section of this manual for more information on timestamps.

	Field Length	*Data Descriptor*	*Message Data*			
			Binary Offset	*Description*	*Data Type*	*Units*
Field Format	14 (0x0E)	0x12	0	GPS Time of Week	Double	Seconds
			8	GPS Week Number	U16	
			10	Timestamp Flags	U16	See Notes

Estimation Filter Data

Estimation Filter Status (0x82, 0x10)

<table>
<tr><td><b>Description</b></td><td colspan="6">Adaptive Kalman Estimation Filter Status</td></tr>
<tr><td><b>Notes</b></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 Status Flags:
Filter State = Running:
0x0001 –Sensors Unavailable
0x0008 – Matrix Singularity in calculation
0x0040 – Attitude Covariance High Warning*
0x0080 – NAN in Solution
0x0100 – Gyro bias estimate high warning
*Note: The covariance high warnings are triggered when <b>any</b> 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="4"><b>Field Format</b></td><td><i>Field Length</i></td><td><i>Data Descriptor</i></td><td colspan="4"><i>Message Data</i></td></tr>
<tr><td rowspan="3">06 (0x06)</td><td rowspan="3">0x10</td><td><i>Binary Offset</i></td><td><i>Description</i></td><td><i>Data Type</i></td><td><i>Units</i></td></tr>
<tr><td>0</td><td>Filter State</td><td>U16</td><td>See Notes</td></tr>
<tr><td>2</td><td>Status Flags</td><td>U16</td><td>See Notes</td></tr>
</table>

GPS Timestamp (0x82, 0x11)

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

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 device 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 device'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 0x0002 – Quaternion is referenced to magnetic north		
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><b>Description</b></td><td colspan="6">INS Estimated Orientation in Matrix form.</td></tr>
<tr><td><b>Notes</b></td><td colspan="6">This is a 9 component coordinate transformation matrix which describes the orientation of the device with respect to the fixed earth coordinate system.
$$M = \begin{bmatrix} M_{1,1} & M_{1,2} & M_{1,3} \\ M_{2,1} & M_{2,2} & M_{2,3} \\ M_{3,1} & M_{3,2} & M_{3,3} \end{bmatrix}$$
<i>M</i> satisfies the following equation:
$V\_IL_i = M_{ij} \cdot V\_E_j$
<i>Where: V_IL is a vector expressed in the device’s local coordinate system.</i>
<i>V_E is the same vector expressed in the stationary, earth-fixed coordinate system</i>
Valid Flag Mapping:
0x0000 – Orientation Matrix is Invalid
0x0001 – Orientation Matrix Valid
0x0002 – Orientation Matrix is referenced to magnetic north</td></tr>
<tr><td rowspan="12"><b>Field Format</b></td><td><i>Field Length</i></td><td><i>Data Descriptor</i></td><td colspan="4"><i>Message Data</i></td></tr>
<tr><td rowspan="11">40 (0x28)</td><td rowspan="11">0x04</td><td><i>Binary Offset</i></td><td><i>Description</i></td><td><i>Data Type</i></td><td><i>Units</i></td></tr>
<tr><td>0</td><td>M_{11}</td><td>float</td><td><i>n/a</i></td></tr>
<tr><td>4</td><td>M_{12}</td><td>float</td><td><i>n/a</i></td></tr>
<tr><td>8</td><td>M_{13}</td><td>float</td><td><i>n/a</i></td></tr>
<tr><td>12</td><td>M_{21}</td><td>float</td><td><i>n/a</i></td></tr>
<tr><td>16</td><td>M_{22}</td><td>float</td><td><i>n/a</i></td></tr>
<tr><td>20</td><td>M_{23}</td><td>float</td><td><i>n/a</i></td></tr>
<tr><td>24</td><td>M_{31}</td><td>float</td><td><i>n/a</i></td></tr>
<tr><td>28</td><td>M_{32}</td><td>float</td><td><i>n/a</i></td></tr>
<tr><td>32</td><td>M_{33}</td><td>float</td><td><i>n/a</i></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 0x0002 – Euler Angles are referenced to magnetic north					
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 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 Attitude Uncertainty, Euler Angles (0x82, 0x0A)

<table>
<tr><td><b>Description</b></td><td colspan="6">1-sigma attitude uncertainty expressed in Pitch, Roll, and Yaw (aircraft) elements.</td></tr>
<tr><td><b>Notes</b></td><td colspan="6">This is a 3 component vector containing the Roll, Pitch and Yaw angle uncertainties in radians.
⚠ 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</td></tr>
<tr><td rowspan="6"><b>Field Format</b></td><td rowspan="2"><i>Field Length</i></td><td rowspan="2"><i>Data Descriptor</i></td><td colspan="4"><i>Message Data</i></td></tr>
<tr><td><i>Binary Offset</i></td><td><i>Description</i></td><td><i>Data Type</i></td><td><i>Units</i></td></tr>
<tr><td rowspan="4">16 (0x10)</td><td rowspan="4">0x0A</td><td>0</td><td>1-Sigma Attitude Uncertainty (Roll)</td><td>float</td><td>radians</td></tr>
<tr><td>4</td><td>1-Sigma Attitude Uncertainty (Pitch)</td><td>float</td><td>radians</td></tr>
<tr><td>8</td><td>1-Sigma Attitude Uncertainty (Yaw)</td><td>float</td><td>radians</td></tr>
<tr><td>12</td><td>Valid Flags</td><td>U16</td><td>See Notes</td></tr>
</table>

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

Description	1-sigma attitude uncertainty expressed in quaternion components.					
Notes	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					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
			Binary Offset	*Description*	*Data Type*	*Units*
	18 (0x12)	0x12	0	1-Sigma Attitude Uncertainty (q0)	float	
			4	1-Sigma Attitude Uncertainty (q1)	float	
			8	1-Sigma Attitude Uncertainty (q2)	float	
			12	1-Sigma Attitude Uncertainty (q3)	float	
			16	Valid Flags	U16	See Notes

Estimated Gyro Bias Uncertainty (0x82, 0x0B)

Description	Estimated Gyro Bias Uncertainty expressed in the Sensor 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	Filter 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><b>Description</b></td><td colspan="6">Filter 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><b>Notes</b></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"><b>Field Format</b></td><td><i>Field Length</i></td><td><i>Data Descriptor</i></td><td colspan="4"><i>Message Data</i></td></tr>
<tr><td rowspan="5">16 (0x10)</td><td rowspan="5">0x0E</td><td><i>Binary Offset</i></td><td><i>Description</i></td><td><i>Data Type</i></td><td><i>Units</i></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><b>Description</b></td><td colspan="6">Local Magnitude of Earth's gravity using the WGS84 gravity model.</td></tr>
<tr><td><b>Notes</b></td><td colspan="6">The device implements the WGS84 gravity model, valid for altitudes of 20km or less. The local reference position (0x0D, 0x26) command must be issued before the WGS84 solution is valid.

Valid Flag Mapping:

0x0000 – Gravity value is Invalid
0x0001 – Gravity value is Valid</td></tr>
<tr><td rowspan="4"><b>Field Format</b></td><td><i>Field Length</i></td><td><i>Data Descriptor</i></td><td colspan="4"><i>Message Data</i></td></tr>
<tr><td rowspan="3">08(0x08)</td><td rowspan="3">0x0F</td><td><i>Binary Offset</i></td><td><i>Description</i></td><td><i>Data Type</i></td><td><i>Units</i></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 Gravity Vector (0x82, 0x13)

Description	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.					
Notes	Valid Flag Mapping: 0x0000 – Gravity vector is Invalid 0x0001 – Gravity vector is Valid					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
			Binary Offset	*Description*	*Data Type*	*Units*
	16 (0x10)	0x13	0	X	Float	Meters / Sec^2
			4	Y	Float	Meters / Sec^2
			8	Z	Float	Meters / Sec^2
			12	Valid Flags	U16	See Notes

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.) Possible Sources: 0x0000 – No source, heading updates disabled 0x0001 – Internal Magnetometer 0x0004 – 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	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)

Description	Magnetic model solution expressed in the NED frame.					
Notes	The World Magnetic Model 2010 is used. The local reference position (0x0D, 0x26) and the GPS Time Update (0x01, 0x72) commands must be issued before the Magnetic Model solution is valid. Valid Flag Mapping: 0x0000 – Magnetic model solution is invalid 0x0001 – Magnetic model solution is valid					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
			Binary Offset	*Description*	*Data Type*	*Units*
	24 (0x18)	0x15	0	Intensity (North)	Float	Gauss
			4	Intensity (East)	Float	Gauss
			8	Intensity (Down)	Float	Gauss
			12	Inclination	Float	Radians
			16	Declination	Float	Radians
			20	Valid Flags	U16	See Notes

Pressure Altitude (0x82, 0x21)

Description	Estimated Pressure Altitude.					
Notes	The US 1976 Standard Atmosphere Model is used to calculate the pressure altitude in meters. A valid pressure sensor reading is required for the pressure altitude to be valid. The minimum pressure reading supported by the model is 0.0037 mBar, corresponding to an altitude of 84,852 meters. Valid Flag Mapping: 0x0000 – Pressure Altitude is Invalid 0x0001 – Pressure Altitude is Valid					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
			Binary Offset	*Description*	*Data Type*	*Units*
	8 (0x08)	0x21	0	Pressure Altitude	float	meters

			4	Valid Flags	U16	See Notes

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	
SYNC1	SYNC2	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 3DM-GX4-25 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 IMU 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 posdesc: **0x04** Rate dec:**0x0004** ECEF veldesc: **0x06** Rate dec: **0x0004**	**0x50**	**0x98**
Replyfield 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 3DM-GX4-25 has special "direct" modes that switch the device into an IMU direct or GPS direct device. The Device Communications Mode command is used to switch between modes. When in these modes, the 3DM-GX4-25 acts like a an “IMU only” 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 3DM-GX4-25.

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 GX4-25.

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 3DM-GX4-25 is still “listening” for mode switch or device status commands and will respond to them. It will not respond to any other 3DM-GX4-25 Base or 3DM commands until switched back to the ”Standard Mode”.*

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

Internal Diagnostic Functions

The 3DM-GX4-25 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-GX4-25 Internal Diagnostic Commands

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

Handling High Rate Data

The size of the data fields from an inertial device is substantially greater than on most other types of sensors. On top of that, in many applications it is desirable to receive that data with the lowest latency possible and thus the highest BAUD rate is selected. The result is that the port servicing requirements in terms of both speed and buffer size can be surprisingly large for inertial data. This can lead to a couple of common problems: runaway latency and dropped packets.

Runaway latency

Most operating systems provide drivers that have ample buffers and take care of port servicing at the hardware level. Dropping packets or losing data is not usually an issue on these systems. What can be an issue is latency, that is, when the buffer is not emptied by the application in a timely manner. In the worst case, the buffer is being filled faster than it is emptied and the application operates with increasingly “old” data – which causes runaway latency. It is important to monitor the incoming data buffer to make sure you do not reach this condition.

Dropped packets

Many applications do not use an operating system but are written from scratch or on top of proprietary application frameworks. These are most often embedded MCUs or small single board microcontrollers. On these systems, port handling is usually done in code at the hardware level. Collecting data from a port requires the use one of three techniques: register polling, hardware interrupts, or direct memory access (DMA). Register polling is very easy to do and is adequate for simple communications where data comes in very small chunks and at reasonable data rates. The problem with register polling is that you either waste time looping while waiting for a byte to come in at the port or you get too busy doing other tasks so that by the time you poll the port, the byte is lost because the next one overwrites it. This causes dropped packets. On these systems, it is imperative to utilize either a hardware interrupt or hardware DMA on the UART receiving data from the 3DM-GX4-25. The DMA or UART interrupt service routine only takes processor time when a byte is ready and as long as the interrupts are preemptive, the processor will fetch every byte received. Using the interrupt routine to fill a ring buffer makes the most efficient use of an MCU and makes it easier to write your application main line code. This is essentially what drivers in operating systems do.

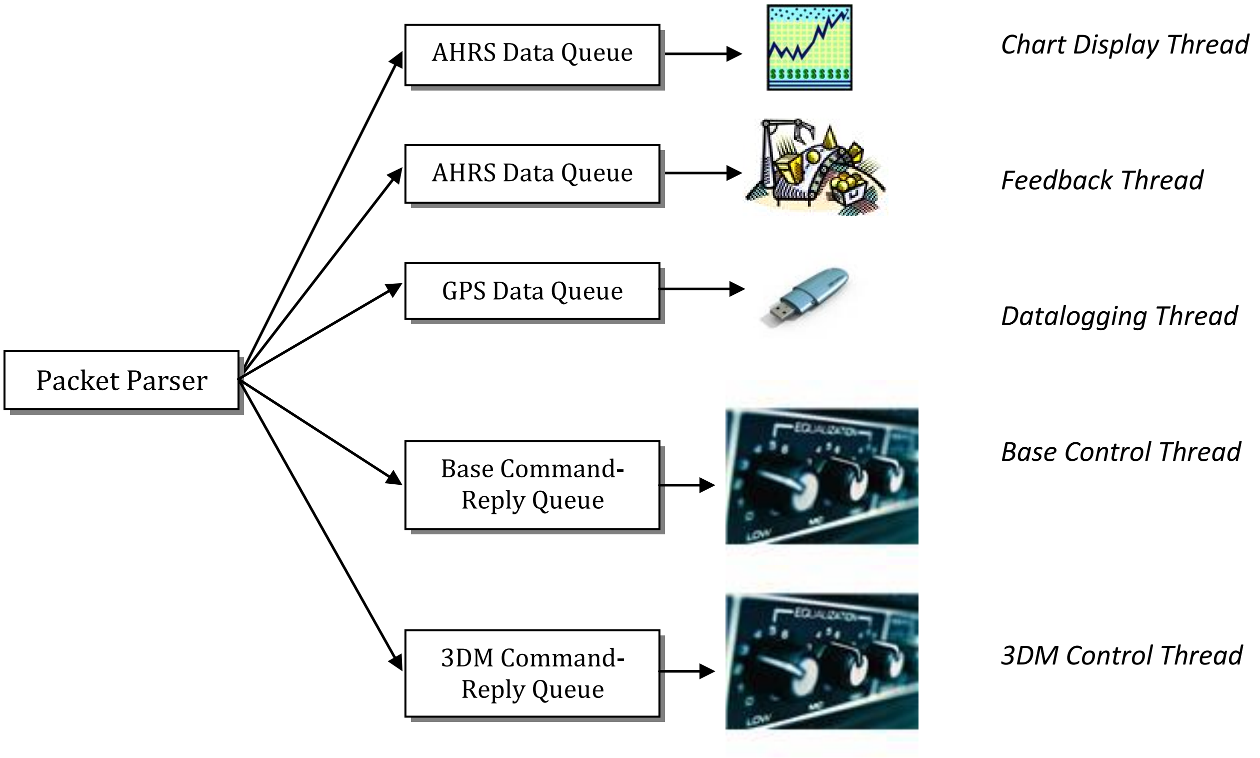
Creating Fixed Data Packet Format

The MIP packet structure and protocol provides a great deal of flexibility to the user for creating a custom data stream. It does this by allowing selectable data fields and individual data rates for each field. The side effect of this feature is that packets vary in size depending on what data is being delivered in any particular time frame. For example, if acceleration data is configured for 100Hz and magnetometer data is configured for 25Hz, every fourth packet is larger than the previous three because of the additional magnetometer data. In some applications, this is undesirable and there may be a requirement for a fixed packet structure so that each data packet is exactly the same. A fixed packet structure allows you to find data fields by fixed offsets rather than parsing the packet for each field.

A fixed packet structure is easily achieved with MIP packet protocol by simply making sure the data rate for each data quantity is the same. The order of the data fields in the packet reflect the order of the fields in the message format command and thus are completely under the control of the user. Once an acceptable data packet structure is determined, and all the rates are set to the same decimation, use the “Save current settings as startup settings” function selector in the message format command, and that format will be saved and used automatically on subsequent device startups. The message formats for each of the data classes (IMU, EF, etc) work the same way, however the available data rates for each class is different, so you will need to create a fixed message format for each one.

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

End of Document

LORD Corporation
MicroStrain® Sensing Systems
459 Hurricane Lane,
Suite 102
Williston, VT 05495 USA
www.microstrain.com

ph: 800-449-3878
fax: 802-863-4093
sensing_support@LORD.co
sensing_sales@LORD.com