DCP Manual 8500-0014 Revision 010

3DM-GX3®-35 Data Communications Protocol

MicroStrain® Little Sensors, Big Ideas®
www.microstrain.com

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

REVISED: Oct 1, 2013

Contents

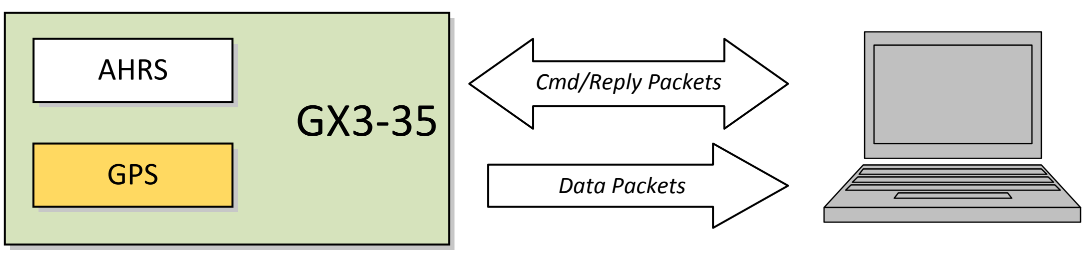
3DM-GX3-35 API

Introduction

The 3DM-GX3-35 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 3 command sets and 2 data sets corresponding to the internal architecture of the device. The three command sets consist of a set of "Base" commands (a set that is common across many types of devices), a set of unified "3DM" (3D Motion) commands that are specific to the MicroStrain inertial product line, and a set of "System" commands that are specific to sensor systems comprised of more than one internal sensor block. The two data sets represent the two types of data that the 3DM-GX3-35 is capable of producing: "AHRS" (Attitude and Heading Reference System) data and "GPS" (Global Positioning Sensor) data.

Base commands	*Ping, Idle, Resume, Get ID Strings, etc.*
3DM commands	*Poll AHRS Data, Poll GPS Data, etc.*
System commands	*Switch Communications Mode, etc.*
AHRS data	*Acceleration Vector, Gyro Vector, Euler Angles, etc.*
GPS data	*Latitude, Longitude, UTC, Satellites in view, etc.*

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



The 3DM-GX3-35 has an advanced mode switch that allows the device to switch into direct "AHRS" or "GPS" mode. In those modes, the device responds to the native protocols of the 3DM-GX3-25 AHRS or the u-blox5 GPS devices which are imbedded in the 3DM-GX3-35. These modes can be used to access advanced or specialized features of these devices or just use them as stand-alone AHRS or GPS devices (see the Advanced Programming section).

Command and Data Summary

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

Commands

Base Command Set (0x01)

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

3DM Command Set (0x0C)

- Poll AHRS Data (0x0C, 0x01)
- Poll GPS Data (0x0C, 0x02)
- Get AHRS Data Rate Base (0x0C, 0x06)
- Get GPS Data Rate Base (0x0C, 0x07)
- AHRS Message Format (0x0C, 0x08)
- GPS Message Format (0x0C, 0x09)
- Enable/Disable Device Continuous Data Stream (0x0C, 0x11)
- Save Device Startup Settings (0x0C, 0x30)
- GPS Dynamics Mode (0x0C, 0x34)
- AHRS Signal Conditioning Settings (0x0C, 0x35)
- AHRS Timestamp (0x0C, 0x36)
- IMU/AHRS Accel Bias*† (0x0C, 0x37)
- IMU/AHRS Gyro Bias*† (0x0C, 0x38)
- IMU/AHRS Capture Gyro Bias† (0x0C, 0x39)
- AHRS Hard Iron Offset*† (0x0C, 0x3A)
- AHRS Soft Iron Matrix*† (0x0C, 0x3B)
- IMU/AHRS Realign Up*† (0x0C, 0x3C)
- AHRS Realign North*† (0x0C, 0x3D)
- UART BAUD rate (0x0C, 0x40)
- Device Data Stream Format* (0x0C, 0x60)
- Device Power States* (0x0C, 0x61)
- Save/Restore Advanced GPS Startup Settings* (0x0C, 0x62)
- Device Status* (0x0C, 0x64)

System Command Set (0x7F)

- Communication Mode* (0x7F, 0x10)

**Advanced Commands*
† Firmware version 1.1.19 and higher

Data

AHRS Data Set (set 0x80)

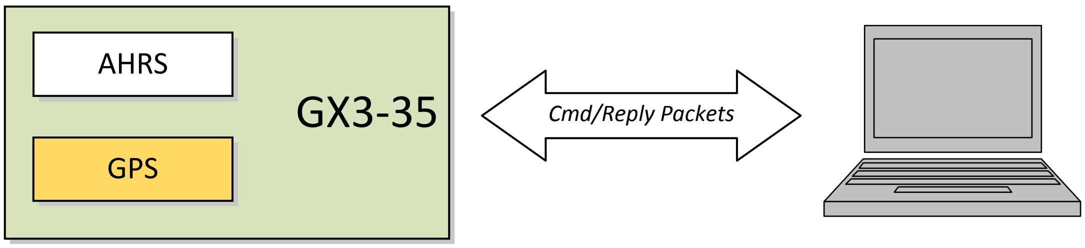
• Raw Accelerometer Vector	(0x80, 0x01)
• Raw Gyro Vector	(0x80, 0x02)
• Raw Magnetometer Vector	(0x80, 0x03)
• Scaled Accelerometer Vector	(0x80, 0x04)
• Scaled Gyro Vector	(0x80, 0x05)
• Scaled Magnetometer Vector	(0x80, 0x06)
• Delta Theta Vector	(0x80, 0x07)
• Delta Velocity Vector	(0x80, 0x08)
• Orientation Matrix	(0x80, 0x09)
• Quaternion	(0x80, 0x0A)
• Orientation Update Matrix	(0x80, 0x0B)
• Euler Angles	(0x80, 0x0C)
• Internal Timestamp	(0x80, 0x0E)
• Beaconed Timestamp	(0x80, 0x0F)
• Stabilized Mag Vector (North)	(0x80, 0x10)
• Stabilized Accel Vector (Up)	(0x80, 0x11)
• GPS Timestamp	(0x80, 0x12)
• Wrapped Raw GX3-25 Data Packet	(0x80, 0x82)

GPS Data Set (set 0x81)

• LLH Position	(0x81, 0x03)
• ECEF Position	(0x81, 0x04)
• NED Velocity	(0x81, 0x05)
• ECEF Velocity	(0x81, 0x06)
• DOP Data	(0x81, 0x07)
• UTC Time	(0x81, 0x08)
• GPS Time	(0x81, 0x09)
• Clock Information	(0x81, 0x0A)
• GPS Fix Information	(0x81, 0x0B)
• Space Vehicle Information	(0x81, 0x0C)
• Hardware Status	(0x81, 0x0D)
• Wrapped Raw NMEA Packet	(0x81, 0x01)
• Wrapped Raw UBX Packet	(0x81, 0x02)

Basic Programming

The 3DM-GX3-35 is designed to stream AHRS and GPS data packets over a common interface as efficiently as possible. To this end, programming the device consists of a configuration stage where the data messages and data rates are configured. The configuration stage is followed by a data streaming stage where the program starts the incoming data packet stream.



In this section there is an overview of the packet, an overview of command and reply packets, an overview of how an incoming data packet is constructed, and then an example setup command sequence that can be used directly with the 3DM-GX3-35 either through a COM utility or as a template for software development.

MIP Packet Overview

This is an overview of the 3DM-GX3-35 packet structure. The packet structure used is the MicroStrain "MIP" packet. A reference to the general packet structure is presented in the MIP Packet Reference section. An overview of the packet is presented here. The MIP packet "wrapper" consists of a four byte header and two byte checksum footer:

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

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

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

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

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

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

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

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

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

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

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

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

Command Overview

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

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

Example "Ping" Command Packet:

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

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

Copy-Paste version: "7565 0102 0201 E0C6"

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

Example "Ping" Reply Packet:

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

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

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

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

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

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

Data Overview

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

Example Data Packet:

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

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

Copy-Paste version: "7565 800E 0E03 3E7A 63A0 BB8E 3B29 7FE5 BF7F 83E1"

The packet header has the "ue" starting sync bytes characteristic of all MIP packets. The descriptor set byte (0x80) identifies the payload field as being from the AHRS data set. The length of the packet payload portion is 14 bytes (0x0E). The payload portion of the packet starts with the length of the field. The field descriptor byte (0x01) identifies the field data as the scaled accelerometer vector from the AHRS data descriptor set. The field data itself is three single precision floating point values of 4 bytes each (total of 12 bytes) representing the X, Y, and Z axis values of the vector. The checksum is a two byte Fletcher checksum (see the MIP Packet Reference for instructions on how to compute a Fletcher two byte checksum).

The format of the field data is fully and unambiguously specified by the descriptor. In this example, the field descriptor (0x04) from the descriptor set (0x80) specifies that the field data holds an array of three single precision IEEE-754 floating point numbers in big-endian byte order and that the values represent units of "g's" and the order of the values is X, Y, Z vector order. Any other specification would require a different descriptor (see the Data Reference section of this manual).

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

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

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

Startup Settings Overview

The startup settings for the 3DM-GX3-35 may be changed by utilizing a function selector that is included with all settings commands. This selector allows you to apply, read, save, load, or reset to factory defaults. The selector actions are uniform across the command set and are summarized here:

Selector Value	Action	Description
1	Apply	Applies the new settings (passed in with the command) immediately. New settings are now the "current" settings. Unless these settings are saved, they will be lost when the device is reset or turned off.
2	Read	Reads the current settings and places them in a data field in the reply packet.
3	Save	Saves current settings to EEPROM. These are now the new startup settings.
4	Load	Loads the last saved settings from EEPROM and makes them the current settings.
5	Load Default	Loads the factory settings and makes them the current settings. (Does not erase the "saved" settings.)

The last saved settings become the new startup settings.

Note: When using any other selector other than "Apply" new settings values may be omitted from the command. If any settings are passed in by the user, they will be ignored.

Note: It is important to be aware that the “Save” function may only be used reliably to a maximum of 100,000 times. This is due to the finite cycle life of the memory used in these devices. For that reason, you should not use the “Save” function as a frequent operation. Typically you only need to use the “Save” function once to set the device startup characteristics. During operation you will usually only use “Apply”.

Example Setup Sequence

Setup involves a series of command/reply pairs. The example below demonstrates actual setup sequences that you can send directly to the 3DM-GX3-35 either programmatically or by using a COM utility. In most cases only minor alterations will be needed to adapt these examples for your application.

Continuous Data Example Command Sequence

Most applications will operate with the 3DM-GX3-35 sending a continuous data stream. In the following example, the AHRS data format is set, followed by the GPS data format. To reduce the amount of streaming data, if present, during the configuration the AHRS and GPS data-streams are disabled while performing the device initialization; they are re-enabled when the set-up is complete. These are not required steps, but are shown here for reference. 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: Disable the AHRS and GPS data-streams

Send "Enable/Disable Continuous Stream" commands to disable continuous streams (reply is ACK/NACK). This is not required but reduces the parsing burden during initialization and makes visual confirmation of the commands easier. This packet combines the AHRS and GPS disable as two fields in a single packet:

Step 1	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Cmd Desc.*	*Field Data*	*MSB*	*LSB*
Command field 1 Disable AHRS Stream	**0x75**	**0x65**	**0x0C**	**0x0A**	**0x05**	**0x11**	Action(APPLY):**0x01** Device (AHRS): **0x01** Stream (OFF): **0x00**		
Command field 2 Disable GPS Stream					**0x05**	**0x11**	Action(APPLY):**0x01** Device (GPS): **0x02** Stream (OFF): **0x00**	**0x21**	**0xC3**
Reply field 1 ACK AHRS disable	**0x75**	**0x65**	**0x0C**	**0x08**	**0x04**	**0xF1**	Cmd echo: **0x11** Error code: **0x00**		
Reply field 2 ACK GPS disable					**0x04**	**0xF1**	Cmd echo: **0x11** Error code: **0x00**	**0xFA**	**0xB5**

Copy-Paste version of the command: "7565 0C0A 0511 0101 0005 1101 0200 21C3"

Step 2: Configure the AHRS data-stream format

Send a "Set AHRS Message Format" command (reply is ACK/NACK). This example requests scaled gyro, scaled accelerometer, and timestamp information at 100 Hz (1000Hz base rate, with a rate decimation of 10 = 100 Hz.) This will result in a single AHRS data packet sent at 100 Hz containing the scaled gyro field followed by the scaled accelerometer field followed by the device's native 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 Data Rate section). If the stream was not disabled in the previous step, the AHRS data would begin stream immediately.

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

Step 2	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Cmd Desc.*	*Field Data*	*MSB*	*LSB*
Command New AHRS Message Format	**0x75**	**0x65**	**0x0C**	**0x0D**	**0x0D**	**0x08**	Action(APPLY): **0x01** Desc count: **0x03** 1st Descriptor: **0x04** Rate Dec: **0x000A** 2nd Descriptor:**0x05** Rate Dec: **0x000A** 3rd Descriptor:**0x0E** Rate Dec: **0x000A**	**0x41**	**0x95**
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 0A05 000A 0E00 0A41 95"

Step 3: Configure the GPS data-stream format

The following configuration command requests ECEF position and velocity information at 1 Hz (4Hz base rate, with a rate decimation of 4 = 1 Hz.) This will result in a single GPS packet sent at 1 Hz containing the ECEF position field followed by the ECEF velocity field. 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 Data Rate section). If the stream was not disabled in the previous step, the GPS data would begin stream immediately.

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 GPS Message Format	**0x75**	**0x65**	**0x0C**	**0x0A**	**0x0A**	**0x09**	Action(APPLY): **0x01** Desc Count: **0x02** ECEF pos desc: **0x04** Rate dec: **0x0004** ECEF vel desc: **0x06** Rate dec: **0x0004**	**0x18**	**0x8E**
Reply ACK/NACK	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0xF1**	Cmd echo: **0x09** Error code: **0x00**	**0xE8**	**0xBC**

Copy-Paste version of the command: "7565 0C0A 0A09 0102 0400 0406 0004 188E"

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

Step 4: Save the AHRS and GPS MIP Message format

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

Step 4	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Cmd Desc.*	*Field Data*	*MSB*	*LSB*
Command field 1 Save Current AHRS Message Format	**0x75**	**0x65**	**0x0C**	**0x08**	**0x04**	**0x08**	Action(SAVE): **0x03** Desc count: **0x00**		
Command field 2 Save Current GPS Message Format					**0x04**	**0x09**	Action(SAVE): **0x03** Desc count: **0x00**	**0x0D**	**0x2E**
Reply field 1 ACK/NACK	**0x75**	**0x65**	**0x0C**	**0x08**	**0x04**	**0xF1**	Cmd echo: **0x08** Error code: **0x00**		
Reply field 2 ACK/NACK					**0x04**	**0xF1**	Cmd echo: **0x09** Error code: **0x00**	**0x0F**	**0x0A**

Copy-Paste version of the command: "7565 0C08 0408 0300 0409 0300 0D2E"

Step 5: Re-enable the AHRS and GPS data-streams

Send an "Enable/Disable Continuous Stream" command to enable continuous stream (reply is ACK). This is just the opposite of step one. After the ACK/NACK is sent data packets will immediately start streaming from the device according to the settings in the previous steps:

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 AHRS Stream	**0x75**	**0x65**	**0x0C**	**0x0A**	**0x05**	**0x11**	Action(APPLY):**0x01** Device (AHRS): **0x01** Stream (ON): **0x01**		
Command field 2 Enable GPS Stream					**0x05**	**0x11**	Action(APPLY):**0x01** Device (GPS): **0x02** Stream (ON): **0x01**	**0x23**	**0xCA**
Reply field 1 ACK AHRS enable	**0x75**	**0x65**	**0x0C**	**0x08**	**0x04**	**0xF1**	Cmd echo: **0x11** Error code: **0x00**		
Reply field 2 ACK GPS enable					**0x04**	**0xF1**	Cmd echo: **0x11** Error code: **0x00**	**0xFA**	**0xB5**

Copy-Paste version of the command: “7565 0C0A 0511 0101 0105 1101 0201 23CA”

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: Disable the AHRS and GPS data-streams
Same as continuous streaming. See above.

Step 2: Configure the AHRS data-stream format
Same as continuous streaming. See above.

Step 3: Configure the GPS data-stream format
Same as continuous streaming. See above.

Step 4: Save the AHRS and GPS MIP Message format
Same as continuous streaming. See above.

Step 5: Send individual data polling commands
Send individual Poll AHRS Data and Poll GPS Data commands in your data collection loop. After the ACK/NACK is sent by the device, a single data packet will be sent according to the settings in the previous steps. Note that the ACK/NACK has the same descriptor set value as the command, but the data packet has the descriptor set value for the type of data (AHRS or GPS):

Step 5	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Cmd Desc.*	*Field Data*	*MSB*	*LSB*
Command Poll AHRS Data	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0x01**	Option: **0x00** Desc count: **0x00**	**0xEF**	**0xDA**
Reply ACK/NACK	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0xF1**	Cmd echo: **0x11** Error code: **0x00**	**0xF0**	**0xCC**
AHRS Data Packet field 1 (Gyro Vector)	**0x75**	**0x65**	**0x80**	**0x1C**	**0x0E**	**0x04**	**0x3E 7A 63 A0** **0xBB 8E 3B 29** **0x7F E5 BF 7F**		
AHRS Data Packet field 2(Accel Vector)					**0x0E**	**0x03**	**0x3E 7A 63 A0** **0xBB 8E 3B 29** **0x7F E5 BF 7F**	**0xAD**	**0xDC**

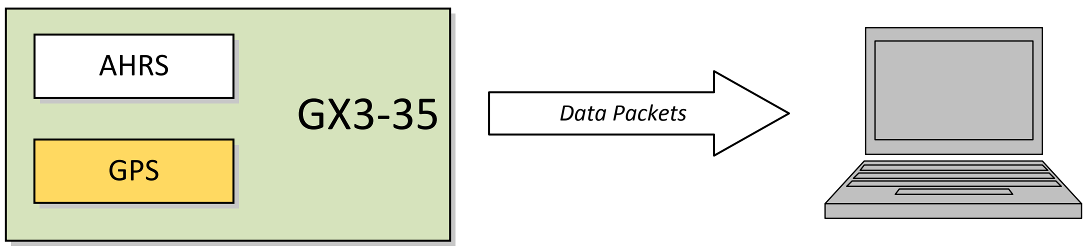
Copy-Paste version of the command: "7565 0C04 0401 0000 EFDA"

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

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

Parsing Incoming Packets

Setup is usually the easy part of programming the 3DM-GX3-35. Once you start continuous data streaming, parsing and processing the incoming data packet stream will become the primary focus. The stream of data from the AHRS is usually the dominant source of data since it comes in the fastest. Although it comes in more slowly, GPS data can contain large structures. Polling for data may seem to be a logical solution to controlling the data flow, and this may be appropriate for some applications, but because timely delivery of data is usually very important in inertial sensor systems, it is often necessary to have the data stream drive the process rather than having the host try to control the data stream through polling.



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

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

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

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

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

MicroStrain supplies examples of parsers for "C", LabVIEW, and Visual Basic in the 3DM-GX3-35 SDK.

Multiple Rate Data

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

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

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

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

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

Data Synchronicity

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

AHRS and GPS data are generated independently by two systems with different clocks. The importance of time is different in each system and the data they produce. The AHRS data requires precise microsecond resolution and perfectly regular intervals in its timestamps. GPS data produces very precise UTC interval data but it is typically delivered in a 1 second time frame.

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

In many applications, synchronizing the timestamps from the two system time bases is critical. MicroStrain uses an extended Beaconed Timestamp across its product line to allow synchronization of data sampled on different system clocks. This timestamp relies on a pulse per second (PPS) beacon signal. On the 3DM-GX3-35, this PPS signal is generated by the on board GPS. The timestamp of the AHRS data represents the interval in nanoseconds from the last PPS pulse. This allows proper time alignment of the GPS data with the AHRS data. On other systems, the PPS signal is applied externally by a system wide PPS beacon. The 3DM-GX3-35 can be the source of this beacon by picking off the PPS output on the multi-com connector.

Another option is to use the GPS Correlation Timestamp for the AHRS data. This timestamp is also synchronized with the hardware PPS signal and in addition it is synchronized with the absolute GPS TOW seconds value and GPS Week number. Flags indicate when the GPS time values have been become synchronized with the GPS module.

Communications Bandwidth Management

Because of the large amount and variety of data that is available from the 3DM-GX3-35, it is quite easy to overdrive the bandwidth of the communications channel, in particular, the RS-232 interface. This can result in dropped packets. The 3DM-GX3-35 does not do any 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. You may detect dropped packet events by including the AHRS System timestamp as one of your AHRS data quantities using the same data rate as the highest data rate in your message format. Use the interval between timestamps to detect packet drop events. Below are some guidelines on how to determine maximum bandwidth for your application.

UART Bandwidth Calculation

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

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

Where

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

which becomes

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

Example:

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

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

$$= 26000 \text{ BAUD}$$

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.

There are three different timestamps available but the shortest and most convenient one for detecting packet drops is the system Internal Timestamp.

USB vs. UART

The 3DM-GX3-35 has a dual communication interface: USB or UART. There is an important difference between USB and UART communication with regards to data bandwidth. The USB "virtual COM port" that the 3DM-GX3-35 implements runs at USB "full-speed" setting of 12Mbs (megabits per second). However, USB is a polled master-slave system and so the slave (3DM-GX3-35) can only communicate when polled by the master. This results in inconsistent data streaming – that is, the data comes in spurts rather than at a constant rate and although rare, sometimes data can be dropped if the host processor fails to poll the USB device in a timely manner.

With the UART the opposite is true. The 3DM-GX3-35 operates without UART handshaking which means it streams data out at a very consistent rate without stopping. Since the host processor has no handshake method of pausing the stream, it must instead make sure that it can process the incoming packet stream non-stop without dropping packets.

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

Command Reference

Base Commands

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

Ping (0x01, 0x01)

<table>
<tr><td>Description</td><td colspan="9">Send a “Ping” command</td></tr>
<tr><td>Notes</td><td colspan="9">Device responds with ACK/NACK packet if present.</td></tr>
<tr><td>Field Format</td><td colspan="2">Field Length</td><td colspan="3">Field Descriptor</td><td colspan="4">Field Data</td></tr>
<tr><td>Command</td><td colspan="2">0x02</td><td colspan="3">0x01</td><td colspan="4">N/A</td></tr>
<tr><td>Reply
ACK/NACK</td><td colspan="2">0x04</td><td colspan="3">0xF1</td><td colspan="4">U8 – echo the command descriptor
U8 – error code (0:ACK, non-zero: NACK)</td></tr>
<tr><td rowspan="2">Example</td><td colspan="4">MIP Packet Header</td><td colspan="3">Command/Reply Fields</td><td colspan="2">Checksum</td></tr>
<tr><td>Sync1</td><td>Sync2</td><td>Desc Set</td><td>Payload Length</td><td>Field Length</td><td>Field Desc.</td><td>Field Data</td><td>MSB</td><td>LSB</td></tr>
<tr><td>Command
Ping</td><td>0x75</td><td>0x65</td><td>0x01</td><td>0x02</td><td>0x02</td><td>0x01</td><td></td><td>0xE0</td><td>0xC6</td></tr>
<tr><td>Reply
ACK/NACK</td><td>0x75</td><td>0x65</td><td>0x01</td><td>0x04</td><td>0x04</td><td>0xF1</td><td>Command echo: 0x01
Error code: 0x00</td><td>0xD5</td><td>0x6A</td></tr>
</table>

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	0x03		0x02			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 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 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 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="7">Get the device ID strings and firmware version</td></tr>
<tr><td>Notes</td><td colspan="7">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>Reply field 1
ACK/NACK</td><td>0x04</td><td>0xF1</td><td colspan="4">U8 – echo the command descriptor
U8 – error code (0: ACK, non-zero: NACK)</td></tr>
<tr><td rowspan="7">Reply field 2
Device Info Field</td><td rowspan="7">0x54</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>Reply Field 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: 0x044E
“ 3DM-GX3-35”
“ 6225-4220”
“ 6225-01319”
“ I042Y”
“ 5g, 300d/s”</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)

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

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

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

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

Description	Run the device Built-In Test (BIT). The Built-In Test command always returns a 32 bit value. A value of 0 means that all tests passed. A non-zero value indicates that not all tests passed. The failure flags are device dependent. The flags for the 3DM-GX3-35 are defined below.		
Notes	The BIT will take approximately 5 seconds to complete on the 3DM-GX3-35. The GPS power will be cycled during the test resulting in the temporary loss and subsequent recalculation of the position solution. **3DM-GX3-35 BIT Error Flags:** (see table below)		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x06	0x05	N/A
Reply field 1 ACK/NACK	0x04	0xF1	U8 – echo the command descriptor U8 – error code (0:ACK, non-zero: NACK)
Reply field 2 BIT Error Flags	0x06	0x83	U32 – BIT Error Flags

3DM-GX3-35 BIT Error Flags:

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

Example

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

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

Device Reset (0x01, 0x7E)

<table>
<tr><td>Description</td><td colspan="9">Resets the 3DM-GX3-35.</td></tr>
<tr><td>Notes</td><td colspan="9">This command has a single 32 bit security value parameter. Device responds with ACK if it recognizes the command and then immediately resets.</td></tr>
<tr><td>Field Format</td><td colspan="2">Field Length</td><td colspan="3">Field Descriptor</td><td colspan="4">Field Data</td></tr>
<tr><td>Command</td><td colspan="2">0x06</td><td colspan="3">0x7E</td><td colspan="4">N/A</td></tr>
<tr><td>Reply
ACK/NACK</td><td colspan="2">0x04</td><td colspan="3">0xF1</td><td colspan="4">U8 – echo the command descriptor
U8 – error code (0: ACK, non-zero: NACK)</td></tr>
<tr><td rowspan="2">Example</td><td colspan="4">MIP Packet Header</td><td colspan="3">Command/Reply Fields</td><td colspan="2">Checksum</td></tr>
<tr><td>Sync1</td><td>Sync2</td><td>Desc Set</td><td>Payload Length</td><td>Field Length</td><td>Field Desc.</td><td>Field Data</td><td>MSB</td><td>LSB</td></tr>
<tr><td>Command
Set Reset</td><td>0x75</td><td>0x65</td><td>0x01</td><td>0x02</td><td>0x02</td><td>0x7E</td><td>N/A</td><td>0x5D</td><td>0x43</td></tr>
<tr><td>Reply
ACK/NACK</td><td>0x75</td><td>0x65</td><td>0x01</td><td>0x04</td><td>0x04</td><td>0xF1</td><td>Command echo: 0x7E
Error code: 0x00</td><td>0x52</td><td>0x64</td></tr>
</table>

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

3DM Commands

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

Poll AHRS Data (0x0C, 0x01)

Description	Poll the 3DM-GX3-35 for an AHRS message with the specified format. The resulting message will maintain the order of descriptors sent in the command and any unrecognized descriptors are ignored. If the format is not provided, the device will attempt to use the stored format (set with the Set AHRS Message Format command.) If no format is provided and there is no stored format, the device will respond with a NACK. The reply packet contains an ACK/NACK field. The polled data packet is sent separately as an AHRS Data packet.		
Notes	*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 descriptor 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: **0x02** Reserved: **0x0000** 2nd Descriptor:**0x01** Reserved: **0x0000**	**0x00**	**0x0F**
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 0200 0001 0000 000F"

Poll GPS Data (0x0C, 0x02)

Description	Poll the 3DM-GX3-35 for a GPS message with the specified format. The resulting message will maintain the order of descriptors sent in the command and any unrecognized descriptors are ignored. If the format is not provided, the device will attempt to use the stored format (set with the Set GPS Message Format command.) If no format is provided and there is no stored format, the device will respond with a NACK. The reply packet contains an ACK/NACK field. The polled data packet is sent separately as an GPS Data packet.		
Notes	*Possible Option Selector Values:* 0x00 – Normal ACK/NACK Reply. 0x01 – Suppress the ACK/NACK reply. ***Special Note****: Due to the size of the Space Vehicle Information field, it will be sent in its own MIP packet independent of the other descriptors. The order will be maintained for all other descriptors in the command.*		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	4 + 3*N	0x02	U8 – Option Selector U8 – Number of Descriptors (N), N*(U8 – Descriptor, U16 Reserved)
Reply ACK/NACK	0x04	0xF1	U8 – echo the command descriptor U8 – error code (0:ACK, not 0:NACK)

Examples	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command Poll GPS data (use stored format)	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0x02**	Option: **0x00** Desc count: **0x00**	**0xF0**	**0xDD**
Command Poll GPS data (use specified format)	**0x75**	**0x65**	**0x0C**	**0x0A**	**0x0A**	**0x02**	Option: **0x00** Desc count: **0x02** 1st Descriptor: **0x03** Reserved: **0x0000** 2nd Descriptor:**0x04** Reserved: **0x0000**	**0x05**	**0x27**
Reply ACK/NACK (Data packet is sent separately if ACK)	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x02** *Error code:* **0x00**	**0xE1**	**0xAE**

Copy-Paste versions of the commands:

Stored format: "7565 0C04 0402 0000 F0DD"
Specified format: "7565 0C0A 0A02 0002 0300 0004 0000 0527"

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

Description	Get the decimation base for the AHRS Data rate calculations. Returns the value used for data rate calculations. See the AHRS Message Format command.		
Notes	*Most models of 3DM-GX3-35 have an AHRS Base Data Rate of 1000. This is used for all the examples in this document. For a given device, this value stays constant.*		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x02	0x06	*none*
Reply field 1 ACK/NACK Field	0x04	0xF1	U8 – echo the command descriptor U8 – error code (0:ACK, not 0:NACK)
Reply field 2 Communications Mode	0x06	0x83	U16-AHRS data rate decimation base

Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	MSB	LSB
Command Get Communications Mode	**0x75**	**0x65**	**0x0C**	**0x02**	**0x02**	**0x06**	*N.A.*	**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: **0x03E8**	**0x5B**	**0xF5**

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

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

Description	Get the decimation base for the GPS Data rate calculations. Returns the value used for data rate calculations. See the GPS Message Format command.		
Notes	*Most models of 3DM-GX3-35 have a GPS Base Data Rate of 4. This is used for all the examples in this document. For a given device, this value stays constant.*		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x02	0x07	*none*
Reply field 1 ACK/NACK Field	0x04	0xF1	U8 – echo the command descriptor U8 – error code (0:ACK, not 0:NACK)
Reply field 2 Communications Mode	0x06	0x84	U16 - GPS data rate decimation base

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

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

AHRS Message Format (0x0C, 0x08)

<table>
<tr><td>Description</td><td colspan="3">Set or read the format of the AHRS message packet. This command sets the format for the AHRS data packet when in standard mode. The resulting data messages will maintain the order of descriptors sent in the command. The command has a function selector and a descriptor array as parameters.</td></tr>
<tr><td>Notes</td><td colspan="3">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 rate decimation field is calculated as follows for the AHRS :

Data Rate = 1000Hz / Rate Decimation

The GX3-35 checks that all descriptors are valid prior to executing this command. If any of the descriptors are invalid for the AHRS descriptor set, a NACK will be returned and the message format will be unchanged. The descriptor array only needs to be provided if the function selector is = 1 (Use new settings). For all other functions it may be empty (Number of Descriptors = 0).

Note: The data rate of the Delta Theta and Delta Velocity vectors must be the same as the orientation calculation rate of the AHRS in order to be valid. The default orientation calculation rate is 100Hz. This rate can be changed using the AHRS Signal Conditioning Settings command. If the Delta Theta and Delta Velocity vectors are requested at any other rate than the orientation calculation rate, a NACK will be returned.</td></tr>
<tr><td>Field Format</td><td>Field Length</td><td>Field Descriptor</td><td>Field Data</td></tr>
<tr><td>Command</td><td>4 + 3*N</td><td>0x08</td><td>U8 - Function Selector
U8 – Number of Descriptors (N),
N*(U8 – Descriptor, U16 – Rate Decimation)</td></tr>
<tr><td>Reply
ACK/NACK</td><td>0x04</td><td>0xF1</td><td>U8 – echo the command descriptor
U8 – error code (0:ACK, not 0:NACK)</td></tr>
<tr><td>Reply field 2
(function = 2)</td><td>3 + 3*N</td><td>0x80</td><td>U8 – Number of Descriptors (N),
N*(U8 – Descriptor, U16 – Rate Decimation)</td></tr>
<tr><td>Examples</td><td>MIP Packet Header</td><td>Command/Reply Fields</td><td>Checksum</td></tr>
</table>

	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command AHRS Message Format (use new settings)	**0x75**	**0x65**	**0x0C**	**0x0A**	**0x0A**	**0x08**	Function: **0x01** Desc count: **0x02** 1st Descriptor: **0x04** Rate Dec: **0x000A** 2nd Descriptor:**0x05** Rate Dec: **0x000A**	**0x22**	**0xA0**
Reply ACK/NACK	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x08** *Error code:* **0x00**	**0xE7**	**0xBA**
Command AHRS Message Format (read back current settings)	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0x08**	Function: **0x02** Desc count: **0x00**	**0xF8**	**0xF3**
Reply field 1 ACK/NACK	**0x75**	**0x65**	**0x0C**	**0x0E**	**0x04**	**0xF1**	*Echo cmd:* **0x08** *Error code:* **0x00**		
Reply field 2 Current AHRS Message Format					**0x0A**	**0x80**	Desc count: **0x02** 1st Descriptor: **0x03** Rate Dec: **0x000A** 2nd Descriptor:**0x04** Rate Dec: **0x000A**	**0x98**	**0x1D**

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

GPS Message Format (0x0C, 0x09)

Description	Set, read, or save the format of the GPS message packet. This function sets the format for the GPS MIP data packet when in standard mode. The resulting message will maintain the order of descriptors sent in the command. The command has a function selector and a descriptor array as parameters.		
Notes	*Possible function selector values:* 0x01 – 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 rate decimation field is calculated as follows for the GPS: *Data Rate = 4Hz / Rate Decimation* The GX3-35 checks that all descriptors are valid prior to executing this command. If any of the descriptors are invalid for the GPS data descriptor set, a NACK will be returned and the message format will be unchanged. The descriptor array only needs to be provided if the function selector is = 1 (Use new settings). For all other functions it may be empty (Number of Descriptors = 0). ***Note**: Due to the size of the Space Vehicle Information field, it will be sent in its own MIP packet independent of the other descriptors. The order will be maintained for all other descriptors in the command.*		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	4 + 3*N	0x09	U8 - Function Selector U8 – Number of Descriptors (N), N*(U8 – Descriptor, U16 – Rate Decimation)
Reply field 1 ACK/NACK	0x04	0xF1	U8 – echo the command descriptor U8 – error code (0:ACK, not 0:NACK)
Reply field 2 (function = 2)	3 + 3*N	0x81	U8 – Number of Descriptors (N), N*(U8 – Descriptor, U16 – Rate Decimation)

Examples	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command	**0x75**	**0x65**	**0x0C**	**0x0A**	**0x0A**	**0x09**	Function: **0x01**	**0x16**	**0x85**

GPS Message Format (use new settings)							Desc count: **0x02** 1st Descriptor: **0x03** Data rate: **0x0004** 2nd Descriptor:**0x05** Data rate: **0x0004**		
Reply ACK/NACK	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x09** *Error code:* **0x00**	**0xE8**	**0xBC**
Command GPS Message Format (read back current settings)	**0x75**	**0x65**	**0x0C**	**0x03**	**0x03**	**0x09**	Function: **0x02**	**0xF7**	**0xF6**
Reply field 1 ACK/NACK	**0x75**	**0x65**	**0x0C**	**0x0E**	**0x04**	**0xF1**	*Echo cmd:* **0x09** *Error code:* **0x00**		
Reply field 2 Current GPS Message Format					**0x0A**	**0x81**	Desc count: **0x02** 1st Descriptor: **0x03** Data rate: **0x0004** 2nd Descriptor:**0x05** Datarate: **0x0004**	**0x8F**	**0x15**

Copy-Paste version of the commands:
Use New Settings: ”7565 0C0A 0A09 0102 0300 0405 0004 1685”
Read Current Settings: “7565 0C03 0309 02F7 F6“

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

Description	Control the streaming of AHRS and GPS 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 both streams enabled. For all functions except 0x01 (use new setting), the new enable flag value is ignored.		
Notes	Possible function selector values: 0x01 – Apply new settings 0x02 – Read back current settings. 0x03 – Save current settings as startup settings 0x04 – Load saved startup settings 0x05 – Load factory default settings *The device selector can be:* 0x01 – AHRS 0x02 – GPS *The enable flag can be either:* 0x00 – disable the selected stream(s). 0x01 – enable the selected stream(s). *(default)*		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x05	0x11	U8 – Function Selector U8 – Device Selector U8 – New Enable Flag
Reply field 1 ACK/NACK	0x04	0xF1	U8 – echo the command descriptor U8 – error code (0:ACK, not 0:NACK)
Reply field 2 (function = 2)	0x04	0x85	U8 – Device Selector U8 – Current Device Enable Flag

Examples	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command AHRS Stream ON	**0x75**	**0x65**	**0x0C**	**0x05**	**0x05**	**0x11**	Function(Apply):**0x01** Device (AHRS): **0x01** Stream (ON): **0x01**	**0x04**	**0x1A**
Command AHRS Stream OFF	**0x75**	**0x65**	**0x0C**	**0x05**	**0x05**	**0x11**	Function(Apply):**0x01** Device (AHRS): **0x01** Stream (OFF): **0x00**	**0x03**	**0x19**
Command	**0x75**	**0x65**	**0x0C**	**0x05**	**0x05**	**0x11**	Function(Apply):**0x01**	**0x05**	**0x1C**

GPS Stream ON							Device (GPS): **0x02** Stream (ON): **0x01**		
Command *GPS Stream OFF*	**0x75**	**0x65**	**0x0C**	**0x05**	**0x05**	**0x11**	Function(Apply):**0x01** Device (GPS): **0x02** Stream (OFF): **0x00**	**0x04**	**0x1B**
Reply *ACK/NACK*	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x11** *Error code:* **0x00**	**0xF0**	**0xCC**

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

Device Startup Settings (0x0C, 0x30)

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

AHRS Message Format
GPS Message Format
Enable/Disable Continuous Data Stream
GPS Dynamics Mode
AHRS Signal Conditioning Settings
UART BAUD Rate
Device Data Stream Format
Device Power States
Communications Mode</td></tr>
<tr><td>Notes</td><td colspan="9">Possible function selector values:

0x03 – Save current settings as startup settings
0x04 – Load saved startup settings
0x05 – Load factory default settings</td></tr>
<tr><td>Field Format</td><td colspan="2">Field Length</td><td colspan="2">Field Descriptor</td><td colspan="5">Field Data</td></tr>
<tr><td>Command</td><td colspan="2">0x03</td><td colspan="2">0x30</td><td colspan="5">U8 –Function Selector</td></tr>
<tr><td>Reply
ACK/NACK</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">Example</td><td colspan="4">MIP Packet Header</td><td colspan="3">Command/Reply Fields</td><td colspan="2">Checksum</td></tr>
<tr><td>Sync1</td><td>Sync2</td><td>Desc Set</td><td>Payload Length</td><td>Field Length</td><td>Field Desc.</td><td>Field Data</td><td>MSB</td><td>LSB</td></tr>
<tr><td>Command
Startup Settings
(Save All)</td><td>0x75</td><td>0x65</td><td>0x0C</td><td>0x03</td><td>0x03</td><td>0x30</td><td>Fctn(Save):0x03</td><td>0x1F</td><td>0x45</td></tr>
<tr><td>Reply
ACK/NACK</td><td>0x75</td><td>0x65</td><td>0x0C</td><td>0x04</td><td>0x04</td><td>0xF1</td><td>Echo cmd: 0x30
Error code: 0x00</td><td>0x0F</td><td>0x0A</td></tr>
</table>

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

GPS Dynamics Mode (0x0C, 0x34)

<table>
<tr><td>Description</td><td colspan="3">Set, read, or save the GPS dynamics mode. For all functions except 0x01 (use new settings), the new dynamics mode value is ignored.</td></tr>
<tr><td>Notes</td><td colspan="3">Possible function selector values:

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

Possible Modes:

<table>
<tr><th>Mode</th><th>Use</th><th>Altitude Limits</th><th>Velocity Limits</th></tr>
<tr><td>0x00 – Portable (default)</td><td>Applications with low acceleration</td><td>12,000 m</td><td>Horizontal - 310 m/s
Vertical - 50 m/s</td></tr>
<tr><td>0x02 – Stationary</td><td>Stationary Antenna/sensor</td><td>9000 m</td><td>Horizontal - 20 m/s
Vertical - 6 m/s</td></tr>
<tr><td>0x03 – Pedestrian</td><td>Low acceleration, low speed</td><td>9000 m</td><td>Horizontal - 30 m/s
Vertical - 20 m/s</td></tr>
<tr><td>0x04 – Automotive</td><td>Low vertical acceleration, wheeled-vehicle dynamics</td><td>6000 m</td><td>Horizontal - 84 m/s
Vertical - 15 m/s</td></tr>
<tr><td>0x05 – Sea</td><td>Zero vertical speed, applications at sea</td><td>500 m</td><td>Horizontal - 25 m/s
Vertical - 5 m/s</td></tr>
<tr><td>0x06 – Airborne <1G</td><td>Higher dynamics than automotive mode</td><td>50,000 m</td><td>Horizontal - 100 m/s
Vertical - 100 m/s</td></tr>
<tr><td>0x07 – Airborne <2G</td><td>Typical airborne application</td><td>50,000 m</td><td>Horizontal - 250 m/s
Vertical - 100 m/s</td></tr>
<tr><td>0x08 – Airborne <4G</td><td>Only for extreme dynamic environments</td><td>50,000 m</td><td>Horizontal - 500 m/s
Vertical - 100 m/s</td></tr>
</table></td></tr>
<tr><td>Field Format</td><td>Field Length</td><td>Field Descriptor</td><td>Field Data</td></tr>
<tr><td>Command</td><td>0x04</td><td>0x34</td><td>U8 – Function Selector
U8 – New Dynamics Mode</td></tr>
<tr><td>Reply
ACK/NACK</td><td>0x04</td><td>0xF1</td><td>U8 – echo the command descriptor
U8 – error code (0:ACK, not 0:NACK)</td></tr>
<tr><td>Reply field 2
(function = 2)</td><td>3</td><td>0x92</td><td>U8 –Current Dynamics Mode</td></tr>
</table>

Example	*MIP Packet Header*				*Fields*			*Checksum*	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command *GPS Settings*	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0x34**	*Fctn (Apply):* **0x01** *Mode (Portable):* **0x00**	**0x23**	**0x75**
Reply *ACK/NACK*	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x34** *Error code:* **0x00**	**0x13**	**0x12**

Copy-Paste version of the command: “7565 0C04 0434 0100 2375”

AHRS Signal Conditioning Settings (0x0C, 0x35)

Description	Set, read, or save the AHRS signal conditioning parameters. This function sets the AHRS signal conditioning parameters for all communications and streaming modes. For all functions except 0x01 (use new settings), the new parameter values are ignored.
Notes	*Possible function selector values:* 0x01 – Apply new settings 0x02 – Read back current settings. 0x03 – Save current settings as startup settings 0x04 – Load saved startup settings 0x05 – Load factory default settings *Possible Orientation Calculation Decimation values:* 0x0001 to 0x03E8 (1 to 1000): This value divided into 1000 will determine the rate at which coning & sculling integration, and orientation calculations are made (including Matrix, Euler, and Quaternion). For example, a value of 10 results in 1000/10 = 100Hz calculation rate. *Default is 0x000A (10)* ***Note****: Delta Theta and Delta Velocity require that the Orientation Calculation rate match the Data Rate setting in the AHRS Message Format*. *Possible Data Conditioning Flags:* 0x0001 – Enables Orientation Calculation (Matrix/Euler). *Default is "1"* 0x0002 – Enables Coning & Sculling. *Default is "1"* 0x0040 – Enables finite size correction. *Default is "0"* 0x0100 – Disables Magnetometer. *Default is "0"* 0x0400 – Disables "North" compensation. *Default is "0"* 0x0800 – Disables "Up" compensation. *Default is "0"* 0x1000 – Enables Quaternion calculation. *Default is "0"* ***NOTE: All undefined bits are reserved and must be set to "0"*** *Possible Gyro/Accel and Mag Filter Width values:* 0x01 to 0x20 (1 to 32): This value divided into 1000 determines the bandwidth of the adjustable filter. See the section on "AHRS Filtering" for more information. *Default is 15 for Accel/Gyro, 17 for Mag.* *Possible Up and North compensation values:* 0x0001 to 0x03E8 (1 to 1000): This value represents how quickly (in seconds) the gravitational /magnetometer vectors correct the inertial attitude/yaw orientation results. *Default is 10 (seconds) for both values.*

	Possible Mag Power/Bandwidth values: 0: High bandwidth, highest power consumption 1: Bandwidth is coupled to Data Rate; low power consumption. *Default is "1"*		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x10	0x35	U8 – Function Selector U16 – New Orientation Calc Decimation Value U16 – New Data Conditioning Flags U8 – New Accel/Gyro Filter Width U8 – New Mag Filter Width U16 – New Up Compensation U16 – New North Compensation U8 – New Mag Bandwidth/Power U16 - *Reserved*
Reply field 1 *ACK/NACK*	0x04	0xF1	U8 – echo the command descriptor U8 – error code (0:ACK, not 0:NACK)
Reply field 2 *(function = 2)*	0x0F	0x86	U16 – Current Orientation Decimation Value U16 – Current Data Conditioning Flags U8 – Current Accel/Gyro Filter Width U8 – Current Mag Filter Width U16 – Current Up Compensation U16 – Current North Compensation U8 – Current Mag Bandwidth/Power U16 - *Reserved*

Example	*MIP Packet Header*				*Fields*			*Checksum*	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command *GPS Settings*	**0x75**	**0x65**	**0x0C**	**0x10**	**0x10**	**0x35**	*Fctn (Apply):* **0x01** *Calc Decimation (100Hz):* **0x000A** *Flags(def):***0x0003** *Acc/GyroFilt:***0x0E** *Mag Filter:* **0x11** *Up Comp:* **0x000A** *N Comp:* **0x000A** *Mag BW:***0x01** *Reserved:***0x0000**	**0x7D**	**0xB7**
Reply *ACK/NACK*	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x35** *Error code:* **0x00**	**0x14**	**0x14**

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

AHRS Timestamp (0x0C, 0x36)

Description	Set the start value, or read the current value of the AHRS Timestamps. For all functions except 0x01 (apply new settings), the new time value is ignored. Use this command to reset the timestamp to zero or other predetermined starting value. The timestamp immediately starts using the new values.		
Notes	*Possible function selector values:* 0x01 – Apply new settings 0x02 – Read back current settings. *The time field selector can be:* 0x01 – AHRS internal tick counter (used in Internal Timestamp) 0x02 – Timestamp Seconds (used in Beaconed Timestamp)		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x08	0x36	U8 – Function Selector U8 – Time field selector U32 – New Time 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)	0x07	0x93	U8 – Time field selector U32 – Current Time Value

Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command Set Timestamp Command	**0x75**	**0x65**	**0x0C**	**0x08**	**0x08**	**0x36**	*Fctn(Apply)*:**0x01** *Field (AHRS Tick Counter)*: **0x01** *Value*: **0x00000000**	**0x2E**	**0x58**
Reply ACK/NACK	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x36** *Error code:* **0x00**	**0x15**	**0x16**

Copy-Paste version of the command: "7565 0C08 0836 0101 0000 0000 2E58"

IMU/AHRS Accel Bias (0x0C, 0x37)

Advanced

Description	Set the value, or read the current value of the IMU/AHRS 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"

IMU/AHRS Gyro Bias (0x0C, 0x38)

Advanced

Description	Set the value, or read the current value of the IMU/AHRS 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"

IMU/AHRS Capture Gyro Bias (0x0C, 0x39)

Description	This command will cause the 3DM-GX3-35 to sample its sensors for the specified number of milliseconds. The resulting data will be used to initialize its orientation, and to 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 IMU/AHRS Gyro Bias command.		
Notes	*Possible Sampling Time values:* Total sampling time in units of milliseconds. Range of values: 1000 to 30000. *Note:* The 3DM-GX3® 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 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 Capture Gyro Bias	**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"

AHRS Hard Iron Offset (0x0C, 0x3A)

Advanced

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] *Note*: **This command is not available on the 3DM-GX3-15**		
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"

AHRS Soft Iron Matrix (0x0C, 0x3B)

Advanced

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 "GX 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] *Note*: **This command is not available on the 3DM-GX3-15**		
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** **0x3F800000** **0x00000000** **0x00000000** **0x00000000** **0x3F800000**	**0xAD**	**0x59**
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"

IMU/AHRS Realign Up (0x0C, 0x3C)

Advanced

Description	This command will realign the gyro stabilized "Up" vector using the specified time constant. This temporarily changes the Up-comp gain to accelerate the realignment to the gravitational vector.		
Notes	*Possible function selector values:* 0x01 – Apply new settings 0x06 – Apply new settings with no ACK/NACK Reply *Possible Realign Time values:* 1 to 100 (in tenths of seconds)		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x04	0x3C	U8 – Function Selector U8 – Realign time (tenths of seconds)
Reply field 1 ACK/NACK	0x04	0xF1	U8 – echo the command descriptor 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 Realign Up	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0x3C**	*Fctn (Apply):* **0x01** *Realign Time:* **0x0A**	**0x35**	**0x97**
Reply field 1 ACK/NACK	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x3C** *Error code:* **0x00**	**0x1B**	**0x22**

Copy-Paste version of the command: "7565 0C04 043C 010A 3597"

AHRS Realign North (0x0C, 0x3D)

Advanced

<table>
<tr><td>Description</td><td colspan="10">This command will realign the gyro stabilized “North” vectors using the specified time constant. This temporarily changes the North comp gain to accelerate the realignment to the magnetic north vector.</td></tr>
<tr><td>Notes</td><td colspan="10">Possible function selector values:

0x01 – Apply new settings
0x06 – Apply new settings with no ACK/NACK Reply

Possible Realign Time values:

1 to 100 (in tenths of seconds)</td></tr>
<tr><td>Field Format</td><td colspan="3">Field Length</td><td colspan="3">Field Descriptor</td><td colspan="4">Field Data</td></tr>
<tr><td>Command</td><td colspan="3">0x04</td><td colspan="3">0x3D</td><td colspan="4">U8 – Function Selector
U8 – Realign time (tenths of seconds)</td></tr>
<tr><td>Reply field 1
ACK/NACK</td><td colspan="3">0x04</td><td colspan="3">0xF1</td><td colspan="4">U8 – echo the command descriptor
U8 – error code (0:ACK, not 0:NACK)</td></tr>
<tr><td rowspan="2">Example</td><td colspan="4">MIP Packet Header</td><td colspan="3">Command/Reply Fields</td><td colspan="2">Checksum</td></tr>
<tr><td>Sync1</td><td>Sync2</td><td>Desc Set</td><td>Payload Length</td><td>Field Length</td><td>Field Desc.</td><td>Field Data</td><td>MSB</td><td>LSB</td></tr>
<tr><td>Command
Realign North</td><td>0x75</td><td>0x65</td><td>0x0C</td><td>0x04</td><td>0x04</td><td>0x3D</td><td>Fctn (Apply): 0x01
Realign Time: 0x0A</td><td>0x36</td><td>0x9A</td></tr>
<tr><td>Reply field 1
ACK/NACK</td><td>0x75</td><td>0x65</td><td>0x0C</td><td>0x04</td><td>0x04</td><td>0xF1</td><td>Echo cmd: 0x3D
Error code: 0x00</td><td>0x1C</td><td>0x24</td></tr>
</table>

Copy-Paste version of the command: “7565 0C04 043D 010A 369A”

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 – 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 *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(Apply):***0x01** *BAUD (115200)*: **0x0001C200**	**0xF8**	**0xDA**
Reply *ACK/NACK*	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x40** *Error code:* **0x00**	**0x1F**	**0x2A**

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

Device Data Stream Format (0x0C, 0x60)

Advanced

Description	Set, read, or save the streaming format of the AHRS and/or GPS data. If set to "Standard MIP" format, the data packets are sent according to the GPS/AHRS message format settings. In "Wrapped Raw" format, the raw data from the internal sensors is sent in one of the native formats of the sensors themselves. The data is sent in a MIP packet in a single field with one of the following descriptors: AHRS Data: Wrapped Raw GX3-25 Single Byte Packet GPS Data: Wrapped Raw NMEA Packet GPS Data: Wrapped Raw UBX Packet "Wrapped Raw" format is useful when interfacing to an existing code-base that utilizes the legacy formats, or when the entire GPS message set is desired over the subset available in the standard mode. Particular attention should be paid to message formats and endianess when using Wrapped Raw mode. For all functions except 0x01 (use new settings), the new stream format value is ignored.		
Notes	*Possible function selector values:* 0x01 – Apply new settings 0x02 – Read back current settings. 0x03 – Save current settings as startup settings 0x04 – Load saved startup settings 0x05 – Load factory default settings *The device selector can be:* 0x01 – AHRS 0x02 – GPS *The stream format can be:* 0x01 – Standard MIP (*default*) 0x02 – Wrapped Raw (MIP wrapper around raw sensor data) *Advanced Users: When transitioning to wrapped-raw format to native MIP format, any GPS message settings made in GPS direct mode will be overwritten.*		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x05	0x60	U8 – Function Selector U8 – Device Selector U8 – New Stream Format
Reply	0x04	0xF1	U8 – echo the command descriptor

ACK/NACK			U8 – error code (0:ACK, not 0:NACK)							
Reply field 2 (if function = 2)	4	0x88	U8 – Device Selector U8 – Current Stream Format							
Examples	MIP Packet Header				Command/Reply Fields			Checksum		
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*	
Command Stream Format	**0x75**	**0x65**	**0x0C**	**0x05**	**0x05**	**0x60**	*Fctn(Apply):***0x01** *Device (AHRS):* **0x01** *Format (Wrapped Raw):* **0x02**	**0x54**	**0x57**	
Reply ACK/NACK	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x60** *Error code:* **0x00**	**0x3F**	**0x6A**	

Copy-Paste version of the command: ”7565 0C05 0560 0101 0254 57”

Device Power States (0x0C, 0x61)

Advanced

<table>
<tr><td>Description</td><td colspan="3">Set, read, or save the power settings of the 3DM-GX3-35 and its internal GPS and AHRS sensors. For all functions except 0x01 (use new settings), the new power state value is ignored.</td></tr>
<tr><td>Notes</td><td colspan="3">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 mask can be:

0x01 – AHRS
0x02 – GPS

Supported Power States are:

0x01 – On (Full Performance),
0x02 – On (Low Power/ECO mode),
0x03 – Sleep (Very Low power, fast startup),
0x04 – Off/Deep Sleep (Lowest Power)

<table><tr><th>State</th><th>GPS</th><th>AHRS</th></tr><tr><td>1</td><td>Yes</td><td>Yes</td></tr><tr><td>2</td><td>Yes</td><td>No</td></tr><tr><td>3</td><td>No</td><td>Yes</td></tr><tr><td>4</td><td>Yes</td><td>Yes</td></tr></table>
If a device does not support a specified state, a NACK will be returned.

Upon power-up, the devices will be placed in the state stored in the user settings. Please see the UBLOX-5 documentation for descriptions of the different “On” states.</td></tr>
<tr><td>Field Format</td><td>Field Length</td><td>Field Descriptor</td><td>Field Data</td></tr>
<tr><td>Command</td><td>0x05</td><td>0x61</td><td>U8 – Function Selector
U8 – Device Mask
U8 – New Power State</td></tr>
<tr><td>Reply field 1
ACK/NACK</td><td>0x04</td><td>0xF1</td><td>U8 – echo the command descriptor
U8 – error code (0:ACK, not 0:NACK)</td></tr>
</table>

Reply field 2 (function = 2)	0x04	0x89		U8 – Device Mask U8 – Current Power State					
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**	**0x05**	**0x05**	**0x61**	*Fctn(Apply):***0x01** Device (AHRS): **0x01** State (Off): **0x04**	**0x57**	**0x5D**
Reply ACK/NACK	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x61** *Error code:* **0x00**	**0x40**	**0x6C**

Copy-Paste version of the command: “7565 0C05 0561 0101 0457 5D”

Save/Restore Advanced GPS Startup Settings (0x0C, 0x62)

Advanced

Description	Save/Load/Reset the current GPS startup settings for advanced GPS "wrapped raw" stream format.								
Notes	*Possible function selector values:* 0x03 – Save current settings as startup settings 0x04 – Load saved startup settings 0x05 – Load factory default settings This command is used only if you are using the device in the "wrapped raw" GPS data stream format. In this streaming format, you switch the GX3-35 to "GPS Direct" communications mode and then set up your own UBX or NMEA message settings using the u-blox "u-Center" application (or other host application) . You then switch back to "Standard" communications mode and use this command to remember the advanced settings in flash memory. You must set the device GPS streaming mode to "wrapped raw" prior to switching to "GPS Direct" mode or else your custom GPS message settings will be overridden by the standard GPS Message Format settings.								
Field Format	*Field Length*			*Field Descriptor*		*Field Data*			
Command	0x03			0x62		U8 –Function Selector			
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 GPS Settings	**0x75**	**0x65**	**0x0C**	**0x03**	**0x03**	**0x62**	*Selector (Save current settings):* **0x03**	**0x51**	**0xA9**
Reply ACK/NACK	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x62** *Error code:* **0x00**	**0x41**	**0x6E**

Copy-Paste version of the command: "7565 0C03 0362 0351 A9"

Device Status (0x0C, 0x64)

Advanced

Description	Get device specific status for the 3DM-GX3-35		
Notes	Reply has two fields: “ACK/NACK” and “Device Status Field”. The device status field may be one of two selectable formats – basic and diagnostic. The reply data for this command is device specific. The reply is specified by two parameters in the command. The first parameter is the model number (which for the 3DM-GX3-35 is always = 6225 (0x1851)). That is followed by a status selector byte which determines the type of data structure returned. In the case of the 3DM-GX3-35, there are two selector values – one to return a basic status structure and a second to return an extensive diagnostics status structure. A list of available values for the selector values and specific fields in the data structure are as follows: *Possible Status Selector Values:* 0x01 – Basic Status Structure 0x02 – Diagnostic Status Structure *Possible Communication Mode Values:* 0x01 – Standard MIP Mode 0x02 – Advanced AHRS Direct Mode 0x03 – Advanced GPS Direct Mode *Possible Communication Device Values:* 0x01 - Com1 (Serial) 0x02 - USB *Possible Settings Flags:* 0x00000001 – AHRS Continuous Stream Enabled 0x00000002 – AHRS Raw Format Enabled 0x00000100 – GPS Continuous Stream Enabled 0x00000200 – GPS Raw Format Enabled *Possible Com1 State, USB State, GPS Driver State, GPS Port State, AHRS Driver State, AHRS Port State Values:* 0x00 – Not Initialized 0x01 – Initialized		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*

Command	0x02	0x64	U16-Device Model Number: set = 6225 (0x1851) U8-Status Selector			
Reply field 1 ACK/NACK Field	0x04	0xF1	U8 – echo the command descriptor U8 – error code (0:ACK, not 0:NACK)			
Reply field 2 Basic Device Status (if selector byte = 1)	0x11	0x90	*Binary Offset*	*Description*	*Data Type*	*Units*
			0	Echo of the Device Model Number	U16	N/A
			2	Echo of the selector byte	U8	N/A
			3	Communication Mode	U8	See Notes
			4	Communication Device	U8	See Notes
			5	Settings Flags	U32	See Notes
			9	Com 1 State	U16	See Notes
			11	Com1 BAUD rate	U32	BAUD
Reply field 2 Diagnostic Device Status (if selector byte = 2)	0x6B	0x90	*Binary Offset*	*Description*	*Data Type*	*Units*
			0	Echo of the Device Model Number	U16	N/A
			2	Echo of the selector byte	U8	N/A
			3	Communication Mode	U8	See Notes
			4	Communication Device	U8	See Notes
			5	Settings Flags	U32	See Notes
			9	Com 1 State	U16	See Notes
			11	Com1 BAUD rate	U32	BAUD
			15	Com1 TX Bytes	U32	# Bytes
			19	Com1 RX Bytes	U32	# Bytes
			23	Com1 TX Overruns	U32	# Bytes
			27	Com1 RX Overruns	U32	# Bytes
			31	USB State	U16	See Notes
			34	USB TX Bytes	U32	# Bytes
			37	USB RX Bytes	U32	# Bytes
			41	USB TX Overruns	U32	# Bytes
			45	USB RX Overruns	U32	# Bytes
			49	GPS Driver State	U16	See Notes
			51	GPS Port State	U16	See Notes
			53	GPS TX Bytes	U32	# Bytes
			57	GPS RX Bytes	U32	# Bytes
			61	GPS TX Overruns	U32	# Bytes
			65	GPS RX Overruns	U32	# Bytes

			69	GPS Messages Processed	U32	# Messages
			73	GPS Messages Delayed	U32	# Messages
			77	AHRS Driver State	U16	See Notes
			79	AHRS Port State	U16	See Notes
			81	AHRS TX Bytes	U32	# Bytes
			85	AHRS RX Bytes	U32	# Bytes
			89	AHRS TX Overruns	U32	# Bytes
			93	AHRS RX Overruns	U32	# Bytes
			97	AHRS Messages Processed	U32	# Messages
			101	AHRS Messages Delayed	U32	# Messages

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 # (6225): **0x1851** Status Selector (basic status): **0x01**	**0xBE**	**0x4B**
Reply field 1 ACK/NACK	**0x75**	**0x65**	**0x0C**	**0x19**	**0x04**	**0xF1**	*Echo cmd:* **0x64** *Error code:* **0x00**		
Reply field 2 Device Status (Basic Status structure)					**0x11**	**0x90**	Echo Model#: **0x1851** Echo Selector: **0x01** U8: U8: U32: U16: U32:	**0x##**	**0x##**

Copy-Paste version of the command: "7565 0C05 0564 1851 01BE 4B"

System Commands

Advanced

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

IMPORTANT NOTE: The Communications Mode command is unique in that *it is always active* regardless of the communications mode. This allows you to switch the system between protocols at any time.

Communication Mode (0x7F, 0x10)

Advanced

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

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

Possible Communications Modes:

<table>
<tr><th>Value</th><th>Mode</th><th>Protocol(s)</th></tr>
<tr><td>0x01</td><td>Standard Mode</td><td>3DM-GX3-35 MIP Packet (default)</td></tr>
<tr><td>0x02</td><td>AHRS Direct</td><td>3DM-GX3-25 Single Byte, ASPP</td></tr>
<tr><td>0x03</td><td>GPS Direct</td><td>NMEA, UBX</td></tr>
</table>

IMPORTANT: GPS message settings are automatically reloaded (see <u>Advanced GPS Startup Settings</u>) when switching from direct modes back in to standard mode <u>unless</u> the GPS message stream format has been set to “Wrapped Raw” PRIOR to switching to direct mode (see <u>Device Data Stream Format</u>).

Note: Switching to and from GPS Direct Mode takes longer than most commands to complete due to the amount of GPS setup data that needs to be stored/retrieved.</td></tr>
</table>

Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x04	0x10	U8 –Function Selector U8 –New Communications Mode
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	0x90	U8 –Current Communications Mode

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

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

Error Codes

```
////////////////////////////////////////////////////////////////////////////////
// GLOBAL ACK/NACK ERROR CODES
////////////////////////////////////////////////////////////////////////////////

#define MIP_ACK_NACK_ERROR_NONE              0x00

#define MIP_ACK_NACK_ERROR_UNKNOWN_COMMAND    0x01
#define MIP_ACK_NACK_ERROR_CHECKSUM_INVALID   0x02
#define MIP_ACK_NACK_ERROR_PARAMETER_INVALID  0x03
#define MIP_ACK_NACK_ERROR_COMMAND_FAILED     0x04
#define MIP_ACK_NACK_ERROR_COMMAND_TIMEOUT    0x05
```

Data Reference

AHRS Data

Raw Accelerometer Vector (0x80, 0x01)

Description	Raw Accelerometer Vector					
Notes	This vector represents the raw binary values of the accelerometers before normalization, scaling and temperature compensation.					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
	14 (0x0E)	0x01	*Binary Offset*	*Description*	*Data Type*	*Units*
			0	Accel 1	float	bits
			4	Accel 2	float	bits
			8	Accel 3	float	bits

Raw Gyro Vector (0x80, 0x02)

Description	Raw Gyro Vector					
Notes	This vector represents the raw binary values of the angular rate before normalization, scaling and temperature compensation.					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
	14 (0x0E)	0x02	*Binary Offset*	*Description*	*Data Type*	*Units*
			0	Gyro 1	float	bits
			4	Gyro 2	float	bits
			8	Gyro 3	float	bits

Raw Magnetometer Vector (0x80, 0x03)

Description	Raw Magnetometer Vector					
Notes	This vector represents the raw binary values of the magnetometer before normalization, scaling and temperature compensation.					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
	14 (0x0E)	0x03	*Binary Offset*	*Description*	*Data Type*	*Units*
			0	Mag 1	float	bits
			4	Mag 2	float	bits
			8	Mag 3	float	bits

Scaled Accelerometer Vector (0x80, 0x04)

Description	Scaled Accelerometer Vector					
Notes	This is a vector quantifying the direction and magnitude of the acceleration that the 3DMGX3 ® is exposed to. This quantity is derived from Raw Accelerometer, but is fully temperature compensated and scaled into physical units of *g* (1 *g* = 9.80665 m/sec^2). It is expressed in terms of the 3DM-GX3®'s local coordinate system.					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
			Binary Offset	*Description*	*Data Type*	*Units*
	14 (0x0E)	0x04	0	X Accel	float	g
			4	Y Accel	float	g
			8	Z Accel	float	g

Scaled Gyro Vector (0x80, 0x05)

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

Scaled Magnetometer Vector (0x80, 0x06)

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

Delta Theta Vector (0x80, 0x07)

<table>
<tr><td>Description</td><td colspan="6">Time integral of angular rate.</td></tr>
<tr><td>Notes</td><td colspan="6">This is a vector which gives the time integral of Angular Rate where the limits of integration are the beginning and end of the most recent data rate period (eg., 0.01 seconds for a data rate of 100Hz). It is expressed in terms of the 3DM-GX3®'s local coordinate system in units of radians.

Note: Delta Theta and Delta Velocity require that the Orientation Calculation rate in the AHRS Signal Conditioning Settings match the Data Rate setting in the AHRS Message Format.</td></tr>
<tr><td rowspan="5">Field Format</td><td>Field Length</td><td>Data Descriptor</td><td colspan="4">Message Data</td></tr>
<tr><td rowspan="4">14 (0x0E)</td><td rowspan="4">0x07</td><td>Binary Offset</td><td>Description</td><td>Data Type</td><td>Units</td></tr>
<tr><td>0</td><td>X Delta Theta</td><td>float</td><td>radians</td></tr>
<tr><td>4</td><td>Y Delta Theta</td><td>float</td><td>radians</td></tr>
<tr><td>8</td><td>Z Delta Theta</td><td>float</td><td>radians</td></tr>
</table>

Delta Velocity Vector (0x80, 0x08)

<table>
<tr><td>Description</td><td colspan="6">Time integral of velocity.</td></tr>
<tr><td>Notes</td><td colspan="6">This is a vector which gives the time integral of Accel where the limits of integration are the beginning and end of the most recent data rate period (eg., 0.01 seconds for a data rate of 100Hz). It is expressed in terms of the 3DM-GX3®'s local coordinate system in units of g*second where g is the standard gravitational constant. To convert Delta Velocity into the more conventional units of m/sec, simply multiply by the standard gravitational constant, 9.80665 m/sec^2

Note: Delta Theta and Delta Velocity require that the Orientation Calculation rate in the AHRS Signal Conditioning Settings match the Data Rate setting in the AHRS Message Format.</td></tr>
<tr><td rowspan="5">Field Format</td><td>Field Length</td><td>Data Descriptor</td><td colspan="4">Message Data</td></tr>
<tr><td rowspan="4">14 (0x0E)</td><td rowspan="4">0x08</td><td>Binary Offset</td><td>Description</td><td>Data Type</td><td>Units</td></tr>
<tr><td>0</td><td>X Delta Velocity</td><td>float</td><td>g*seconds</td></tr>
<tr><td>4</td><td>Y Delta Velocity</td><td>float</td><td>g*seconds</td></tr>
<tr><td>8</td><td>Z Delta Velocity</td><td>float</td><td>g*seconds</td></tr>
</table>

Orientation Matrix (0x80, 0x09)

<table>
<tr><td>Description</td><td colspan="6">3 x 3 Orientation Matrix M</td></tr>
<tr><td>Notes</td><td colspan="6">This is a 9 component coordinate transformation matrix which describes the orientation of the 3DM-GX3® with respect to the fixed earth coordinate system.
$$M = \begin{bmatrix} M_{1,1} & M_{1,2} & M_{1,3} \\ M_{2,1} & M_{2,2} & M_{2,3} \\ M_{3,1} & M_{3,2} & M_{3,3} \end{bmatrix}$$
M satisfies the following equation:
$V\_IL_i = M_{ij} \cdot V\_E_j$
Where: V_IL is a vector expressed in the 3DM-GX3®'s local coordinate system.
V_E is the same vector expressed in the stationary, earth-fixed coordinate system</td></tr>
<tr><td rowspan="11">Field Format</td><td>Field Length</td><td>Data Descriptor</td><td colspan="4">Message Data</td></tr>
<tr><td rowspan="10">38 (0x26)</td><td rowspan="10">0x09</td><td>Binary Offset</td><td>Description</td><td>Data Type</td><td>Units</td></tr>
<tr><td>0</td><td>M_{11}</td><td>float</td><td>n/a</td></tr>
<tr><td>4</td><td>M_{12}</td><td>float</td><td>n/a</td></tr>
<tr><td>8</td><td>M_{13}</td><td>float</td><td>n/a</td></tr>
<tr><td>12</td><td>M_{21}</td><td>float</td><td>n/a</td></tr>
<tr><td>16</td><td>M_{22}</td><td>float</td><td>n/a</td></tr>
<tr><td>20</td><td>M_{23}</td><td>float</td><td>n/a</td></tr>
<tr><td>24</td><td>M_{31}</td><td>float</td><td>n/a</td></tr>
<tr><td>28</td><td>M_{32}</td><td>float</td><td>n/a</td></tr>
<tr><td>32</td><td>M_{33}</td><td>float</td><td>n/a</td></tr>
</table>

Quaternion (0x80, 0x0A)

<table>
<tr><td>Description</td><td colspan="6">4 x 1 quaternion Q.</td></tr>
<tr><td>Notes</td><td colspan="6">This is a 4 component quaternion which describes the orientation of the 3DM-GX3 with respect to the fixed earth coordinate quaternion.
$$Q = \begin{bmatrix} q0 \\ q1 \\ q2 \\ q3 \end{bmatrix}$$
Q satisfies the following equation:
V_ILi = Q^{-1} • V_E • Q
Where: V_IL is a vector expressed in the 3DM-GX3®'s local coordinate system.
V_E is the same vector expressed in the stationary, earth-fixed coordinate system</td></tr>
<tr><td rowspan="6">Field Format</td><td>Field Length</td><td>Data Descriptor</td><td colspan="4">Message Data</td></tr>
<tr><td rowspan="5">18 (0x12)</td><td rowspan="5">0x0A</td><td>Binary Offset</td><td>Description</td><td>Data Type</td><td>Units</td></tr>
<tr><td>0</td><td>q_0</td><td>float</td><td>n/a</td></tr>
<tr><td>4</td><td>q_1</td><td>float</td><td>n/a</td></tr>
<tr><td>8</td><td>q_2</td><td>float</td><td>n/a</td></tr>
<tr><td>12</td><td>q_3</td><td>float</td><td>n/a</td></tr>
</table>

Orientation Update Matrix (0x80, 0x0B)

<table>
<tr><td>Description</td><td colspan="6">3 x 3 Orientation Update Matrix C</td></tr>
<tr><td>Notes</td><td colspan="6">This is a 9 component coordinate transformation matrix which describes the change in orientation of the 3DM-GX3® during the period of the most recent calculation cycle.

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

$M2_i = C_{ij} \cdot M1_{ij}$

Where: M1 is the orientation matrix at the beginning of the calculation cycle.
M2 is the orientation matrix at the end of the calculation cycle.</td></tr>
<tr><td rowspan="11">Field Format</td><td>Field Length</td><td>Data Descriptor</td><td colspan="4">Message Data</td></tr>
<tr><td rowspan="10">38 (0x26)</td><td rowspan="10">0x0B</td><td>Binary Offset</td><td>Description</td><td>Data Type</td><td>Units</td></tr>
<tr><td>0</td><td>C_{11}</td><td>float</td><td>n/a</td></tr>
<tr><td>4</td><td>C_{12}</td><td>float</td><td>n/a</td></tr>
<tr><td>8</td><td>C_{13}</td><td>float</td><td>n/a</td></tr>
<tr><td>12</td><td>C_{21}</td><td>float</td><td>n/a</td></tr>
<tr><td>16</td><td>C_{22}</td><td>float</td><td>n/a</td></tr>
<tr><td>20</td><td>C_{23}</td><td>float</td><td>n/a</td></tr>
<tr><td>24</td><td>C_{31}</td><td>float</td><td>n/a</td></tr>
<tr><td>28</td><td>C_{32}</td><td>float</td><td>n/a</td></tr>
<tr><td>32</td><td>C_{33}</td><td>float</td><td>n/a</td></tr>
</table>

Euler Angles (0x80, 0x0C)

<table>
<tr><td>Description</td><td colspan="6">Pitch, Roll, and Yaw (aircraft) values</td></tr>
<tr><td>Notes</td><td colspan="6">This is a 3 component vector containing the Roll, Pitch and Yaw angles in radians. It is computed by the AHRS from the orientation matrix M.

$Euler = \begin{bmatrix} Roll \\ Pitch \\ Yaw \end{bmatrix}$ (radians)</td></tr>
<tr><td rowspan="5">Field Format</td><td>Field Length</td><td>Data Descriptor</td><td colspan="4">Message Data</td></tr>
<tr><td rowspan="4">14 (0x0E)</td><td rowspan="4">0x0C</td><td>Binary Offset</td><td>Description</td><td>Data Type</td><td>Units</td></tr>
<tr><td>0</td><td>Roll</td><td>float</td><td>radians</td></tr>
<tr><td>4</td><td>Pitch</td><td>float</td><td>radians</td></tr>
<tr><td>8</td><td>Yaw</td><td>float</td><td>radians</td></tr>
</table>

Internal Timestamp (0x80, 0x0E)

<table>
<tr><td>Description</td><td colspan="6">32 bit free running internal AHRS timer value (tick)</td></tr>
<tr><td>Notes</td><td colspan="6">This is a timer value which measures the time since system power-up or timer set command. The timer interval on the 3DM-GX3-35 AHRS is 16 microseconds (µs). To convert the timer value to time in seconds, divide by 62,500. The system clock has an accuracy of +/- 0.01%. The timer value rolls over from its maximum value to 0 approximately every 68719 seconds (~1145 minutes or ~19 hours).

When the timestamp is included in the data message format with other AHRS data quantities, the timestamp represents the time at which the data quantities were sampled. See the Data Synchronicity section of this manual for more details.</td></tr>
<tr><td rowspan="3">Field Format</td><td>Field Length</td><td>Data Descriptor</td><td colspan="4">Message Data</td></tr>
<tr><td rowspan="2">6 (0x06)</td><td rowspan="2">0x0E</td><td>Binary Offset</td><td>Description</td><td>Data Type</td><td>Units</td></tr>
<tr><td>0</td><td>Timestamp</td><td>U32</td><td>16µs ticks</td></tr>
</table>

Beaconed Timestamp (0x80, 0x0F)

<table>
<tr><td>Description</td><td colspan="6">Beaconed system synchronization timestamp</td></tr>
<tr><td>Notes</td><td colspan="6">This timestamp has three fields:

U8 Timestamp Status
U32 Seconds counter
U32 Nanoseconds counter

Timestamp Status Flags:
Bit0 – PPS Beacon Good If set, GPS PPS signal is present

The Seconds and Nanoseconds time values are relative to the system one pulse per second (1PPS) system beacon signal produced by the GPS. The seconds counter increments with each PPS, and the nanoseconds counter resets to zero on each PPS. In the event of a lost GPS PPS beacon, the internal system clock is used to generate the PPS. When the GPS PPS is reestablished, the timestamp is resynchronized immediately resulting in a single timestamp offset jump proportional to the outage time interval and the drift of the internal clock. The “PPS Beacon Good” flag in the Timestamp Status byte indicates if the PPS beacon coming from the GPS is good. If this flag is not asserted, it means that the internal clock is being used for the PPS.

See the Data Synchronicity section of this manual for more information on timestamps.</td></tr>
<tr><td rowspan="4">Field Format</td><td>Field Length</td><td>Data Descriptor</td><td colspan="4">Message Data</td></tr>
<tr><td rowspan="4">11 (0x0B)</td><td rowspan="4">0x0F</td><td>Binary Offset</td><td>Description</td><td>Data Type</td><td>Units</td></tr>
<tr><td>0</td><td>Timestamp Status</td><td>U8</td><td></td></tr>
<tr><td>1</td><td>1PPS counter</td><td>U32</td><td>Seconds</td></tr>
<tr><td></td><td>5</td><td>Nanosecond counter</td><td>U32</td><td>Nanoseconds</td></tr>
</table>

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

<table>
<tr><td>Description</td><td colspan="6">Gyro stabilized estimated vector for geomagnetic vector.</td></tr>
<tr><td>Notes</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 Magnetometer. When transient magnetic interference is present, Magnetometer will be subject to transient (possibly large) errors. The AHRS complementary filter computes Stabilized North 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">Field Format</td><td>Field Length</td><td>Data Descriptor</td><td colspan="4">Message Data</td></tr>
<tr><td rowspan="4">14 (0x0E)</td><td rowspan="4">0x10</td><td>Binary Offset</td><td>Description</td><td>Data Type</td><td>Units</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>

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

<table>
<tr><td>Description</td><td colspan="6">Gyro stabilized estimated vector for the gravity vector.</td></tr>
<tr><td>Notes</td><td colspan="6">This is a vector which represents the 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.</td></tr>
<tr><td rowspan="5">Field Format</td><td>Field Length</td><td>Data Descriptor</td><td colspan="4">Message Data</td></tr>
<tr><td rowspan="4">14 (0x0E)</td><td rowspan="4">0x11</td><td>Binary Offset</td><td>Description</td><td>Data Type</td><td>Units</td></tr>
<tr><td>0</td><td>X Stab Accel</td><td>Float</td><td>g</td></tr>
<tr><td>4</td><td>Y Stab Accel</td><td>Float</td><td>g</td></tr>
<tr><td>8</td><td>Z Stab Accel</td><td>Float</td><td>g</td></tr>
</table>

GPS Correlation Timestamp (0x80, 0x12)

<table>
<tr><td>Description</td><td colspan="6">GPS correlation timestamp.</td></tr>
<tr><td>Notes</td><td colspan="6">This timestamp has three fields:

Double GPS TOW
U16 GPS Week number
U16 Timestamp flags

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

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

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

If the GPS loses signal, the GPS and AHRS timestamps become free running and will slowly drift away from each other. If the timestamp clocks have drifted apart, then there will be a jump in the timestamp when the PPS Beacon Good reasserts, reflecting the amount of drift of the clocks.

See the Data Synchronicity section of this manual for more information on timestamps.</td></tr>
<tr><td rowspan="4">Field Format</td><td rowspan="2">Field Length</td><td rowspan="2">Data Descriptor</td><td colspan="4">Message Data</td></tr>
<tr><td>Binary Offset</td><td>Description</td><td>Data Type</td><td>Units</td></tr>
<tr><td rowspan="3">14 (0x0E)</td><td rowspan="3">0x12</td><td>0</td><td>GPS Time of Week</td><td>Double</td><td>Seconds</td></tr>
<tr><td>8</td><td>GPS Week Number</td><td>U16</td><td></td></tr>
<tr><td></td><td>10</td><td>Timestamp Flags</td><td>U16</td><td>See Notes</td></tr>
</table>

Wrapped Raw GX3-25 Single Byte Packet (0x80, 0x82)

<table>
<tr><td>Description</td><td colspan="9">A legacy single byte command wrapped in a MIP format packet</td></tr>
<tr><td>Notes</td><td colspan="9">This takes an “old style” data packet from the internal 3DM-GX3-25 AHRS sensor and places it in a single field with the field descriptor of 0x82. Please see the “3DM-GX3-25 Data Communications Protocol” document for information on legacy single byte commands. See the Device Data Stream Format command for more information on using this data descriptor.</td></tr>
<tr><td>Field Format</td><td colspan="2">Field Length</td><td colspan="3">Field Descriptor</td><td colspan="4">Field Data</td></tr>
<tr><td>Command</td><td colspan="2">2 + N</td><td colspan="3">0x01</td><td colspan="4">Data (3DM-GX3-25 Format)</td></tr>
<tr><td rowspan="2">Example</td><td colspan="4">MIP Packet Header</td><td colspan="3">Command/Reply Fields</td><td colspan="2">Checksum</td></tr>
<tr><td>Sync1</td><td>Sync2</td><td>Desc Set</td><td>Payload Length</td><td>Field Length</td><td>Field Desc.</td><td>Field Data</td><td>MSB</td><td>LSB</td></tr>
<tr><td>3DM-GX3-25 “0xB0” message</td><td>0x75</td><td>0x65</td><td>0x80</td><td>0x1C</td><td>0x1C</td><td>0x82</td><td>0xB0,0x18,0x0E, 0x01,0x47,0x7C, 0x90,0x00,0x47, 0x77,0x85,0x00, 0x47,0x7B,0x81, 0x00,0x0A,0x0C, 0x07,0x0F,0x0B, 0xF4,0x0B,0xE3, 0x0C,0x4A</td><td>0xMM</td><td>0xLL</td></tr>
</table>

GPS Data

LLH Position (0x81, 0x03)

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

ECEF Position (0x81, 0x04)

Description	Position Data in the Earth-Centered, Earth-Fixed Frame					
Notes	Valid Flag Mapping: 0x0001 – ECEF Position Valid 0x0002 – Position Accuracy Valid					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
			Binary Offset	*Description*	*Data Type*	*Units*
	32 (0x20)	0x04	0	X Position	Double	Meters
			8	Y Position	Double	Meters
			16	Z Position	Double	Meters
			24	Position Accuracy	Float	Meters
			28	Valid Flags	U16	See Notes

NED Velocity (0x81, 0x05)

<table>
<tr><td>Description</td><td colspan="6">Velocity Data in the North-East-Down Frame</td></tr>
<tr><td>Notes</td><td colspan="6">Valid Flag Mapping:

0x0001 – NED Velocity Valid
0x0002 – Speed Valid
0x0004 – Ground Speed Valid
0x0008 – Heading Valid
0x0010 – Speed Accuracy Valid
0x0020 – Heading Accuracy Valid</td></tr>
<tr><td rowspan="11">Field Format</td><td>Field Length</td><td>Data Descriptor</td><td colspan="4">Message Data</td></tr>
<tr><td rowspan="10">36(0x24)</td><td rowspan="10">0x05</td><td>Binary Offset</td><td>Description</td><td>Data Type</td><td>Units</td></tr>
<tr><td>0</td><td>North</td><td>Float</td><td>Meters / Sec</td></tr>
<tr><td>4</td><td>East</td><td>Float</td><td>Meters / Sec</td></tr>
<tr><td>8</td><td>Down</td><td>Float</td><td>Meters / Sec</td></tr>
<tr><td>12</td><td>Speed</td><td>Float</td><td>Meters / Sec</td></tr>
<tr><td>16</td><td>Ground Speed</td><td>Float</td><td>Meters / Sec</td></tr>
<tr><td>20</td><td>Heading</td><td>Float</td><td>Decimal Degrees</td></tr>
<tr><td>24</td><td>Speed Accuracy</td><td>Float</td><td>Meters / Sec</td></tr>
<tr><td>28</td><td>Heading Accuracy</td><td>Float</td><td>Decimal Degrees</td></tr>
<tr><td>32</td><td>Valid Flags</td><td>U16</td><td>See Notes</td></tr>
</table>

ECEF Velocity (0x81, 0x06)

Description	Velocity Data in the Earth-Centered, Earth-Fixed Frame					
Notes	Valid Flag Mapping: 0x0001 – ECEF Velocity Valid 0x0002 – Velocity Accuracy Valid					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
			Binary Offset	*Description*	*Data Type*	*Units*
	20 (0x14)	0x06	0	X Velocity	Float	Meters / Sec
			4	Y Velocity	Float	Meters / Sec
			8	Z Velocity	Float	Meters / Sec
			12	Velocity Accuracy	Float	Meters / Sec
			16	Valid Flags	U16	See Notes

DOP Data (0x81, 0x07)

Description	Dilution of Precision Data					
Notes	Valid Flag Mapping: 0x0001 – GDOP Valid 0x0002 – PDOP Valid 0x0004 – HDOP Valid 0x0008 – VDOP Valid 0x0010 – TDOP Valid 0x0020 – NDOP Valid 0x0040 – EDOP Valid					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
			Binary Offset	*Description*	*Data Type*	*Units*
	32 (0x20)	0x07	0	Geometric DOP	Float	N/A
			4	Position DOP	Float	N/A
			8	Horizontal DOP	Float	N/A
			12	Vertical DOP	Float	N/A
			16	Time DOP	Float	N/A
			20	Northing DOP	Float	N/A
			24	Easting DOP	Float	N/A
			28	Valid Flags	U16	See Notes

UTC Time (0x81, 0x08)

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

GPS Time (0x81, 0x09)

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

Clock Information (0x81, 0x0A)

Description	Detailed information about the GPS Clock					
Notes	Valid Flag Mapping: 0x0001 – Bias Valid 0x0002 – Drift Valid 0x0004 – Accuracy Estimate Valid					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
			Binary Offset	*Description*	*Data Type*	*Units*
	28(0x1C)	0x0A	0	Clock Bias	Double	Seconds
			8	Clock Drift	Double	Seconds/Second
			16	Accuracy Estimate	Double	Seconds
			24	Valid Flags	U16	See Notes

GPS Fix Information (0x81, 0x0B)

<table>
<tr><td>Description</td><td colspan="6">Current GPS Fix Status Information</td></tr>
<tr><td>Notes</td><td colspan="6">Valid Flag Mapping:

0x0001 – Fix Type Valid
0x0002 – Number of SVs Valid
0x0004 – Fix Flags Valid

Possible Fix Types values are:

0x00 – 3D Fix
0x01 – 2D Fix
0x02 – Time Only
0x03 – None
0x04 – Invalid</td></tr>
<tr><td rowspan="6">Field Format</td><td>Field Length</td><td>Data Descriptor</td><td colspan="4">Message Data</td></tr>
<tr><td rowspan="5">8(0x08)</td><td rowspan="5">0x0B</td><td>Binary Offset</td><td>Description</td><td>Data Type</td><td>Units</td></tr>
<tr><td>0</td><td>Fix Type</td><td>U8</td><td>See Notes</td></tr>
<tr><td>1</td><td>Number of SVs used for solution</td><td>U8</td><td>Count</td></tr>
<tr><td>2</td><td>Fix Flags (Reserved)</td><td>U16</td><td>N/A</td></tr>
<tr><td>4</td><td>Valid Flags</td><td>U16</td><td>See Notes</td></tr>
</table>

Space Vehicle Information (0x81, 0x0C)

<table>
<tr><td>Description</td><td colspan="6">Individual Space Vehicle Information Entry</td></tr>
<tr><td>Notes</td><td colspan="6">When enabled, these fields will arrive in a separate MIP packet.

Valid Flag Mapping:

0x0001 – Channel Valid
0x0002 – SV ID Valid
0x0008 – Carrier to Noise Ratio Valid
0x0010 – Azimuth Valid
0x0020 – Elevation Valid
0x0040 – SV Flags Valid

SV Flag Mapping:

0x0001 – SV Used for Navigation
0x0002 – SV Healthy</td></tr>
<tr><td rowspan="9">Field Format</td><td>Field Length</td><td>Data Descriptor</td><td colspan="4">Message Data</td></tr>
<tr><td rowspan="8">14(0x0E)</td><td rowspan="8">0x0C</td><td>Binary Offset</td><td>Description</td><td>Data Type</td><td>Units</td></tr>
<tr><td>0</td><td>Channel</td><td>U8</td><td>Channel Number</td></tr>
<tr><td>1</td><td>Space Vehicle ID</td><td>U8</td><td>SV ID Number</td></tr>
<tr><td>2</td><td>Carrier to Noise Ratio</td><td>U16</td><td>dBHz</td></tr>
<tr><td>4</td><td>Azimuth</td><td>S16</td><td>Integer Degrees</td></tr>
<tr><td>6</td><td>Elevation</td><td>S16</td><td>Integer Degrees</td></tr>
<tr><td>8</td><td>Space Vehicle Flags</td><td>U16</td><td>See Notes</td></tr>
<tr><td>10</td><td>Valid Flags</td><td>U16</td><td>See Notes</td></tr>
</table>

Hardware Status (0x81, 0x0D)

<table>
<tr><td>Description</td><td colspan="6">GPS Hardware Status Information</td></tr>
<tr><td>Notes</td><td colspan="6">Valid Flag Mapping:

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

Possible Sensor State values:

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

Possible Antenna State values:

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

Possible Antenna Power values:

0x00 – Antenna Off
0x01 – Antenna On
0x02 – Antenna Power Unknown

Note: The hardware status message is only available at the 1Hz data rate</td></tr>
<tr><td rowspan="6">Field Format</td><td>Field Length</td><td>Data Descriptor</td><td colspan="4">Message Data</td></tr>
<tr><td rowspan="5">7(0x07)</td><td rowspan="5">0x0D</td><td>Binary Offset</td><td>Description</td><td>Data Type</td><td>Units</td></tr>
<tr><td>0</td><td>Sensor State</td><td>U8</td><td>See Notes</td></tr>
<tr><td>1</td><td>Antenna State</td><td>U8</td><td>See Notes</td></tr>
<tr><td>2</td><td>Antenna Power</td><td>U8</td><td>See Notes</td></tr>
<tr><td>3</td><td>Valid Flags</td><td>U16</td><td>See Notes</td></tr>
</table>

Wrapped Raw NMEA Packet (0x81, 0x01)

Description	A raw NMEA packet wrapped in MIP format		
Notes	Please see the UBLOX-5 Protocol Specification for possible NMEA data packets and formats.		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	2 + N	0x01	U8*N – NMEA text data of length N

Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
NMEA GGA Message	**0x75**	**0x65**	**0x81**	**0x4D**	**0x4D**	**0x01**	`"$GPGGA,092725.00,4717.11399,N,00833.91590,E,1,8,1.01,499.6,M,48.0,M,,0*5B<cr><lf>"`	0xMM	0xLL

Wrapped Raw UBX Packet (0x81, 0x02)

Description	A raw UBX packet wrapped in MIP format		
Notes	Please see the UBLOX-5 Protocol Specification for possible UBX data packets and formats.		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	2 + N	0x02	U8*N – UBX binary data of length N

Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
UBX NAV-CLOCK Message	**0x75**	**0x65**	**0x81**	**0x1D**	**0x1D**	**0x02**	`0xB5,0x62, 0x01,0x22, 0x14, 0x0000012F, 0x00050030, 0x00100400, 0x00050020, 0x00000507, 0x35,0xA8`	0xMM	0xLL

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>	$k_1+k_2+...k_n$	MIP Field 1 length = k_1	...	MIP Field *n* length = k_n	0x*MM*	0x*LL*

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

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

Payload Length Range

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

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

MicroStrain provides a “MIP Packet Builder” 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	Descrip tor Set	Payload Length	MIP Data Fields	MSB (byte1)	LSB (byte2)
<-- *Checksum Range* -->						

16-bit Fletcher Checksum Algorithm (C language)

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

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

Advanced Programming

Multiple Commands in a Single Packet

MIP packets may contain one or more individual commands. In the case that multiple commands are transmitted in a single MIP packet, the GX3-35 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):

<table>
<tr><th rowspan="2">Step 2 and 3</th><th colspan="4">MIP Packet Header</th><th colspan="3">Command/Reply Fields</th><th colspan="2">Checksum</th></tr>
<tr><th>Sync1</th><th>Sync2</th><th>Desc Set</th><th>Payload Length</th><th>Field Length</th><th>Cmd Desc.</th><th>Field Data</th><th>MSB</th><th>LSB</th></tr>
<tr><td>Command field 1
Set AHRS Message Format</td><td>0x75</td><td>0x65</td><td>0x0C</td><td>0x14</td><td>0x0A</td><td>0x08</td><td>Function: 0x00
Desc count: 0x02
1st Descriptor: 0x03
Rate Dec: 0x000A
2nd Descriptor:0x04
Rate Dec: 0x000A</td><td></td><td></td></tr>
<tr><td>Command field 2
Set GPS Message Format</td><td></td><td></td><td></td><td></td><td>0x0A</td><td>0x09</td><td>Function: 0x00
Desc Count: 0x02
ECEF pos desc: 0x04
Rate dec: 0x0004
ECEF vel desc: 0x06
Rate dec: 0x0004</td><td>0x50</td><td>0x98</td></tr>
<tr><td>Reply field 1
ACK/NACK</td><td>0x75</td><td>0x65</td><td>0x0C</td><td>0x08</td><td>0x04</td><td>0xF1</td><td>Cmd echo: 0x08
Error code: 0x00</td><td></td><td></td></tr>
<tr><td>Reply field 2
ACK/NACK</td><td></td><td></td><td></td><td></td><td>0x04</td><td>0xF1</td><td>Cmd echo: 0x09
Error code: 0x00</td><td>0xE9</td><td>0x6F</td></tr>
</table>

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 Communications Modes

The GX3-35 has special “direct” communication modes that switch the device into a “GX3-25” AHRS or a “u-blox” GPS device. The Communications Mode command is used to switch between modes. When in these modes, the GX3-35 acts just like a GX3-25 AHRS or a u-blox GPS sensor respectively. Any code or tools developed for these devices may be used in these modes. For example, when in the “u-blox” direct mode, the u-blox “u-center” application works perfectly with the GPS chip embedded in the GX3-35.

You cannot communicate to both sensors at the same time while in direct mode. In order to interact with both sensors and receive interleaved AHRS and GPS data, you must be in “Standard” communications mode.

In “Standard” communications mode the GX3-35 processor automatically configures the AHRS and GPS sensors to match the settings made in standard mode. Any changes you made while in direct mode will normally be overridden by the standard mode configuration. However, these standard mode settings can be bypassed by using the “Wrapped Raw” streaming format for AHRS, GPS, or both. For example, if you wish to configure and receive NMEA messages from the GPS instead of standard MIP GPS messages, you would first set the Device Data Stream Format for GPS to “Wrapped Raw”, then switch to “GPS Direct” communications mode. There you can use u-blox or NMEA message commands to enable specific NMEA messages you wish to receive. When you switch back to “Standard MIP” communications mode, the NMEA messages will be passed through as a single field inside a MIP packet instead of the standard mode packets. Any AHRS message formats that you made in standard mode will be preserved and interleaved with the wrapped raw GPS packets.

This same example can be used to enable native AHRS message formats (such as the original GX series single byte data commands). To utilize this mode, you would set the AHRS message stream mode to “Wrapped Raw” and then switch to “AHRS Direct” communications mode, setup your message format and switch back to Standard Communication mode.

You can save the custom GPS message formats by issuing the Save/Restore Advanced GPS Startup Settings command. Those settings will be used on startup if the startup GPS data stream format is also set to “wrapped Raw”. The AHRS custom message may be saved using the commands available in the AHRS direct mode (see the “3DM-GX3-25 Data Communications Protocol” document).

IMPORTANT: When you switch modes, you are switching to a new device protocol EXCEPT for two commands: the Device Communications Mode and *Device Status commands. Those commands are always available regardless of which mode you are in. For example, if you switch to GPS direct mode, then the protocol recognized by the device is NMEA and UBX protocol, however the GX3-35 is still “listening” for mode switch or device status commands and will respond to them. It will not respond to any other 3DM-GX3-35 Basic or 3DM commands until switched back to the ”Standard Mode”.*

IMPORTANT: GPS message settings are automatically reloaded (see Advanced GPS Startup Settings) when switching from direct modes back in to standard mode unless the GPS message stream has been set to “Wrapped Raw” messages PRIOR to switching to direct mode (see Device Data Stream Format).

Internal Diagnostic Functions

The 3DM-GX3-35 supports two device specific internal functions used for diagnostics and system status. These are Device Built In Test and Device Status. These commands are defined generically but the implementation is very specific to the hardware implemented on this device. Other MicroStrain devices will have their own implementations of these functions depending on the internal hardware of the devices.

3DM-GX3-35 INTERNAL DIAGNOSTIC COMMANDS

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

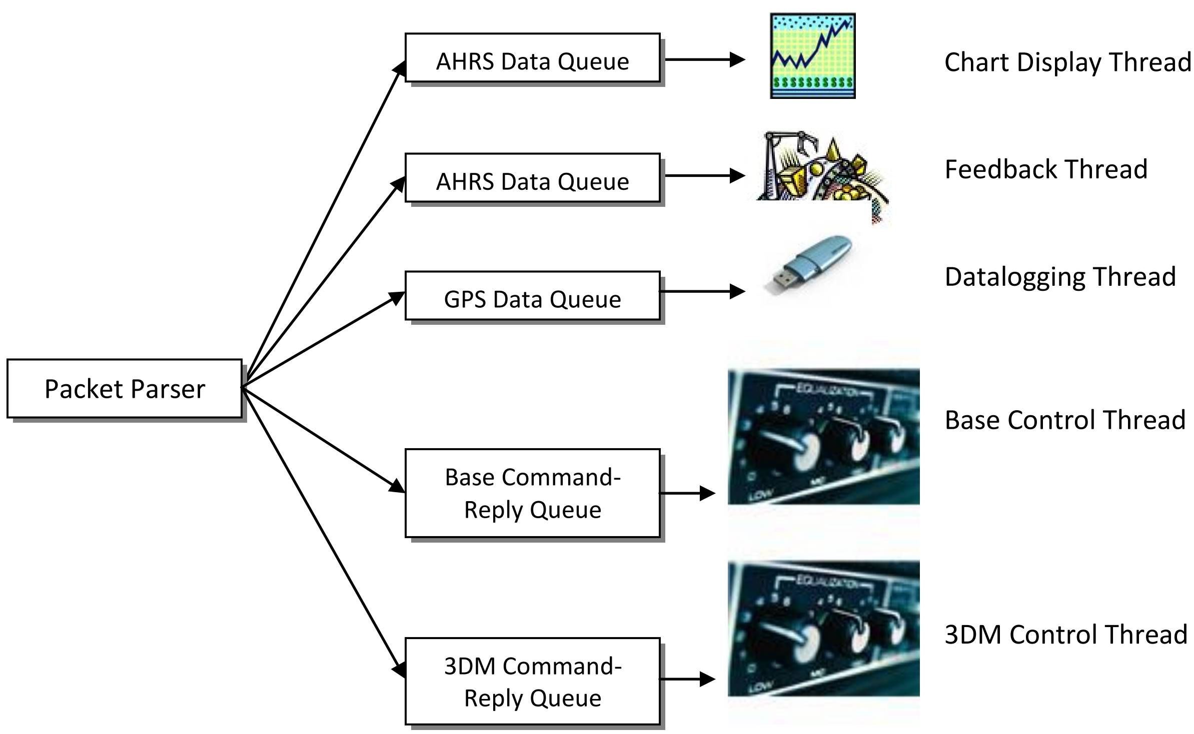
Legacy Protocols

An advanced feature allows you to “tunnel” native GX3-25 data, NMEA messages, or UBX messages as fields in standard AHRS data or GPS data packets. This is helpful in applications that already have code that utilizes those formats. NMEA code in particular is very prevalent. For new applications, we recommend using the 3DM-GX3-35 “unified” AHRS and GPS data formats. This will allow applications to easily migrate to future MicroStrain Inertial products that may not support the legacy protocols.

To utilize the legacy protocols, refer to the Direct Communications Modes section.

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 an 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, especially at high data rates.



Multithreaded application with multiple incoming packet queues

AHRS Filtering

Noise filtering of the MEMS inertial devices on the 3DM-GX3® is accomplished using analog anti-aliasing filters followed by a two stage digital moving average filter. The analog filters are fixed and have bandwidths characterized in Table 1.

Analog Anti Alias Filter Bandwidths		Min	Nom	Max	
Accelerometer (1.7g/5g/16g)	RC	164	226	335	Hz
Accelerometer (50g)	2 pole Bessel	360	400	440	Hz
Gyroscope (all rates)	RC	500		600	Hz
Magnetometer	*no analog filter*	-	-	-	

TABLE 1

Two Stage Digital Filter

The digital filter has two stages. The first stage is a fixed 30kHz 30 tap moving average filter. The second stage is a 1kHz variable width moving average filter. The second stage is adjustable by means of the filter window size (aka filter width; filter taps, filter points). The transfer function of the digital filter is as follows:

$$H[f] = \left| \frac{\sin(M\pi f/1000)}{M\sin(\pi f/1000)} \times \frac{\sin(30\pi f/30000)}{30\sin(\pi f/30000)} \right|$$

EQUATION 1

M is the width of the second stage filter and *f* is the input frequency in *Hz*. For example, for an input frequency of 75*Hz*, and a filter width of 10, the attenuation is:

$$H[f] = \left| \frac{\sin(10\pi 75/1000)}{10\sin(\pi 75/1000)} \times \frac{\sin(30\pi 75/30000)}{30\sin(\pi 75/30000)} \right|$$

$$H[f] = 0.300$$

EXAMPLE 1

The first stage 30kHz filter removes high frequency spectral noise produced by the MEMs sensors and is a smaller factor in attenuating signals in the hundred hertz range.

Magnetometer Digital Filter (High Resolution)

The magnetometer has special sampling criteria that result in an oversample rate of ¼ of the fixed oversample rate of 30000 which is used for the other sensors. This means the oversample rate of the magnetometer is 7500Hz. A fixed 2 point averaging filter is applied to the signal followed by a 7 point averaging filter. The result of this filter is fed at 1kHz to an adjustable filter with a window size from 1 to 32.

Note: The Magnetometer does not have anti-aliasing filters. Magnetic noise above 3750Hz will be aliased.

The transfer function for the magnetometer becomes:

$$H[f] = \left| \frac{\sin(2\pi f / 7500)}{2\sin(\pi f / 7500)} \times \frac{\sin(7\pi f / 3750)}{7\sin(\pi f / 3750)} \times \frac{\sin(M_m \pi f / 1000)}{M_m \sin(\pi f / 1000)} \right|$$

EQUATION 2

Where f is the input frequency in Hz and M_m is the magnetometer filter width. Note that the first stage of the filter changes the sampling frequency of the second stage to 3750Hz. This also results in a first null at 3750Hz.

As an example, an input frequency of 60Hz and filter window width, M_m, of 16 is attenuated as follows:

$$M_m = 16$$

$$H[f] = \left| \frac{\sin(2\pi 60 / 7500)}{2\sin(\pi 60 / 7500)} \times \frac{\sin(7\pi 60 / 3750)}{7\sin(\pi 60 / 3750)} \times \frac{\sin(16\pi 60 / 1000)}{16\sin(\pi 60 / 1000)} \right|$$

$$= 0.9997 \times 0.9799 \times 0.0418$$

$$H[f] = 0.041$$

EXAMPLE 2

As can be seen, the first two filter terms are close to 1 so a simplified transfer function can be used for the purpose of calculating the attenuation of the adjustable filter:

$$H[f] \sim \left| \frac{\sin(M_m \pi f / 1000)}{M_m \sin(\pi f / 1000)} \right|$$

EQUATION 3

Magnetometer Digital Filter (Low Power)

The magnetometer may be put into a lower power mode which results in lower resolution and slightly increased noise. The filtering is affected in two ways: (1) it removes the second stage filtering and (2) it changes the adjustable filter sample rate factor from 1000 to [datarate] where the [datarate] is in Hz (see Sampling Settings command). The result is that the simplified filtering transfer function becomes:

$$H[f] \sim \left| \frac{\sin(M_m \pi f / [datarate])}{M_m \sin(\pi f / [datarate])} \right|$$

EQUATION 4

Digital Filter Characteristics

One of the most important aspects of the moving average FIR filter is that it is the best filter to use with respect to step response in the time domain. This is important for systems that want to avoid overshoot and ringing from stepped input signals. The attenuation is moderate in the frequency domain but reasonable for applications that require a low cutoff frequency and have moderate noise in the near-band spectrum. The analog anti-alias filters (on the accelerometers and gyros) cascade with the digital filter to improve attenuation of above-band signals and noise. The first stages of the magnetometer filter attenuate high frequency noise up to 3750Hz. The adjustable stage of the magnetometer can be adjusted to filter out the highly prevalent 50/60Hz power line noise.

Figure 1 shows frequency response curves for a 3 point, 11 point, and 31 point single stage moving average filter.

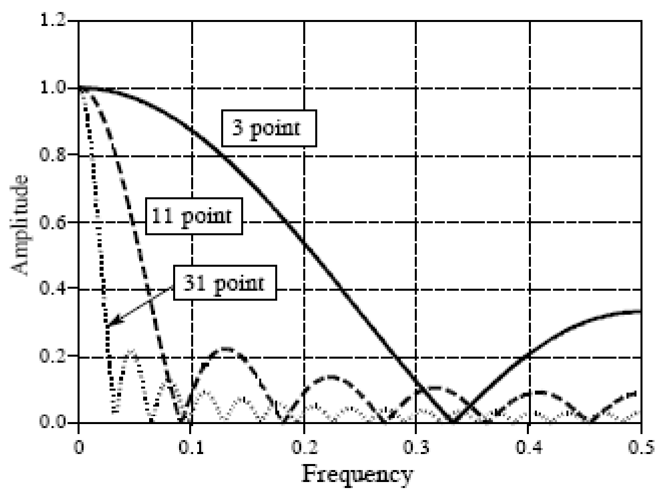


FIGURE 1

Best Performance

The best performance of the 3DM-GX3® occurs after all the sensors have warmed up and the operating temperature gradients of the unit have stabilized. These are the conditions that the 3DM-GX3® is calibrated under and where the best performance will be realized.