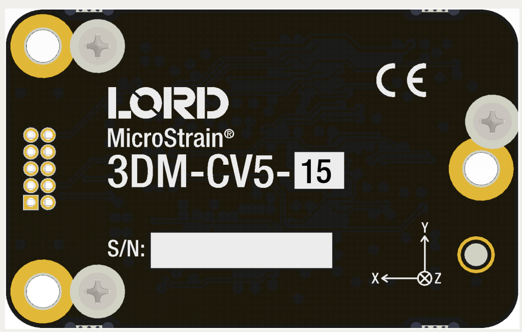
LORD **Data Communications Protocol Manual**

3DM®-CV5-15

Vertical Reference Unit



LORD SENSING
MicroStrain

MicroStrain® Sensing Systems
459 Hurricane Lane
Suite 102
Williston, VT 05495
United States of America

Phone: 802-862-6629

www.microstrain.com
sensing_support@LORD.com
sensing_sales@LORD.com

Document 8500-0072 Revision D

Subject to change without notice.

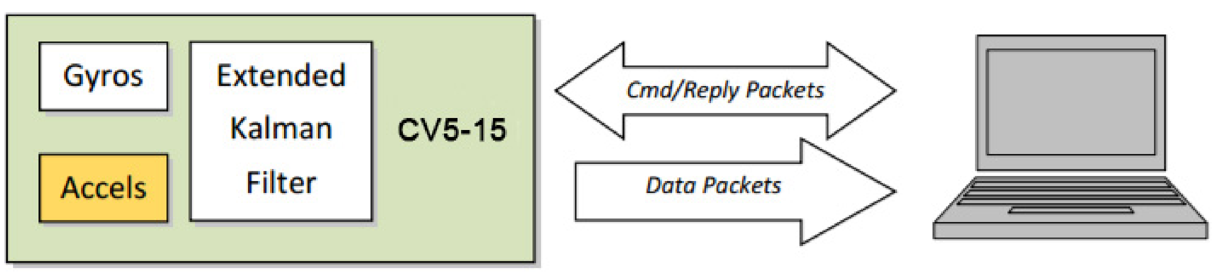
Table of Contents

1. API Introduction

The 3DM-CV5-15 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 four command sets and two data set corresponding to the internal architecture of the device. The 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 LORD Sensing inertial product line, a set of "Estimation Filter" commands that are specific to LORD Sensing navigation and advanced AHRS devices, and a set of "System" commands that are specific to sensor systems comprised of more than one internal sensor block. The data sets represent the two types of data that the 3DM-CV5-15 is capable of producing: "Estimation Filter" (Attitude data and "IMU" (Inertial Measurement Unit) data.

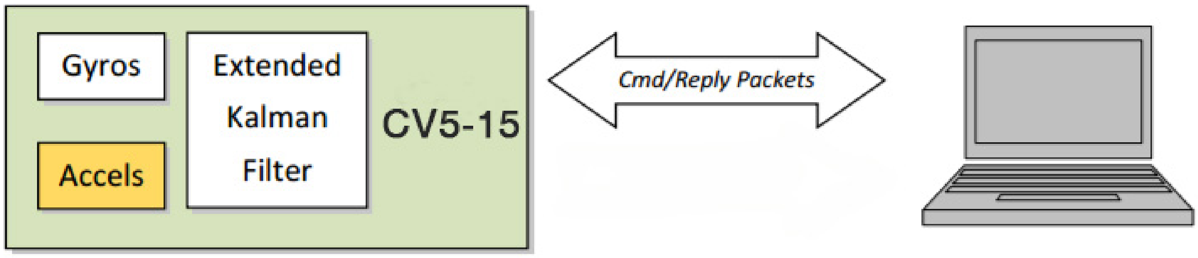
Base commands	Ping, Idle, Resume, Get ID Strings, etc.
3DM commands	Poll IMU Data, Estimation Filter Data, etc.
Estimation Filter commands	Reset Filter, Sensor to Vehicle Frame Transformation, etc.
System commands	Switch Communications Mode, etc.
IMU data	Acceleration Vector, Gyro Vector, etc.
Estimation Filter data	Attitude, Acceleration Estimates, etc.

The protocol is packet based. All commands, replies, and data are sent and received as fields in a message packet. Commands are all confirmed with an ack/nack (with a few exceptions). The packets have a descriptor type field based on their contents, so it is easy to identify if a packet contains IMU data, commands, or replies.



2. Basic Programming

The 3DM-CV5-15 is designed to stream Estimation Filter, and IMU 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-CV5-15 either through a COM utility or as a template for software development.

2.1 MIP Packet Overview

This is an overview of the 3DM-CV5-15 packet structure. The packet structure used is the LORD "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"*	*Descriptor Set byte*	*Payload Length byte*	*Field 1 Length*	*Field 1 Descriptor*	*Field 1 Data*	*Field 2 Length*	*Field 2 Descriptor*	*Field 2 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	0xE0	0xC6

2.2 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.

2.2.1 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 Byte Length*	*Field Descriptor Byte*	*Field Data*	*MSB*	*LSB*
0x75	0x65	0x01	0x02	0x02	0x01	N/A	0xE0	0xC6
Copy-Paste version of command: "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 payload 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).

2.2.2 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 Byte Length*	*Field Descriptor Byte*	*Field Data*	*MSB*	*LSB*
0x75	0x65	0x01	0x04	0x04	0xF1	Command Echo: 0x01 Error code: 0x00	0xD5	0x6A
Copy-Paste version of reply:… "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.

2.3 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.

2.3.1 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 Byte Length*	*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	0x84	0xEE
Copy-Paste version: "7565 800E 0E04 3E7A 63A0 BB8E 3B29 7FE5 BF7F 84EE"								

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

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

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

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

2.4 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-CV5-15 either programmatically or by using a COM utility. In most cases only minor alterations will be needed to adapt these examples for your application.

2.4.1 Continuous Data Example Command Sequence

Most applications will operate with the 3DM-CV5-15 sending a continuous data stream. In the following example, the IMU data format is set, followed by the Estimation Filter data format. To reduce the amount of streaming data, if present during the configuration, the device is placed into the idle state while performing the device initialization; when configuration is complete, the required data streams are enabled to bring the device out of idle mode. Finally, the configuration is saved so that it will be loaded on subsequent power-ups, eliminating the need to perform the configuration again.

1. Put the Device in Idle Mode

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

	MIP Packet Header				Command/Reply Fields			Checksum	
	SYNC1 "u	*SYNC2 "e"*	*Descriptor Set byte*	*Payload Length*	*Field Length*	*Cmd. Descriptor*	*Field Data*	*MSB*	*LSB*
Command: *Set to Idle*	0x75	0x65	0x01	0x02	0x02	0x02	N/A	0xE1	0xC7
Reply: *ACK/NACK*	0x75	0x65	0x01	0x04	0x04	0xF1	Cmd echo: **0x02** Error code: **0x00**	0xD6	0x6C
Copy-Paste version of the command: "7565 0102 0202 E1C7"									

2. **Configure the IMU Data-stream Format**

Send a "Set IMU Message Format" command (reply is ACK/NACK). This example requests GPS correlation timestamp, scaled gyro, and scaled accelerometer information at Hz (Hz base rate divided by a rate decimation of 10 on the 3DM-CV5-15 = Hz.) This will result in a single IMU data packet sent at Hz containing the IMU GPS correlation timestamp followed by the scaled gyro field and the scaled accelerometer field. This is a very typical configuration for a base level of inertial data. If different rates were requested, then each packet would only contain the data quantities that fall in the same decimation frame (see the Multiple Rate Data section). If the stream was not disabled in the previous step, the IMU data would begin stream immediately.

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

	MIP Packet Header				Command/Reply Fields			Checksum	
	SYNC1 "u	*SYNC2 "e"*	*Descriptor Set byte*	*Payload Length*	*Field Length*	*Cmd. Descriptor*	*Field Data*	*MSB*	*LSB*
Command: *New IMU Message Format*	0x75	0x65	0x0C	0x0D	0x0D	0x08	Function: **0x01** Desc. count: **0x03** GPS TS Desc.: **0x12** Rate Dec: **0x000A** Accel Desc.: **0x04** Rate Dec: **0x000A** Ang Rate Desc: **0x05** Rate Dec: **0x000A**	0x45	0xF2
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 1200 0A04 000A 0500 0A45 F2"									

3. Configure the Estimation Filter Data-stream Format

The following configuration command requests the GPS Timestamp followed by the Estimated , Estimated , and at Hz (Hz base rate divided by a rate decimation of 10 on the 3DM-CV5-15 = Hz.) This will result in a single IMU data packet sent at Hz containing the requested fields in the requested order. If different rates were requested, then each packet would only contain the data quantities that fall in the same data rate frame (see the Multiple Rate Data section). If the stream was not disabled in the previous step, the Estimation Filter data would begin stream immediately.

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

	MIP Packet Header				Command/Reply Fields			Checksum	
	SYNC1 “u	*SYNC2 “e”*	*Descriptor Set byte*	*Payload Length*	*Field Length*	*Cmd. Desc.*	*Field Data*	*MSB*	*LSB*
Command: *New Estimation Filter Message Format*	0x75	0x65	0x0C	0x10	0x10	0x0A	Function: **0x01** Desc. count: **0x04** GPS TS Desc.: **0x11** Rate Dec: **0x000A** EF Euler: **0x05** Rate Dec: **0x000A** EF Accel: **0x0D** Rate Dec: **0x000A** EF Ang Rate: **0x0E** Rate Dec: **0x000A**	0x6E	0xB0
Reply: *ACK/NACK*	0x75	0x65	0x0C	0x04	0x04	0xF1	**Cmd echo: 0x0A** **Error code: 0x00**	0xE9	0xBE
Copy-Paste version of the command: “7565 0C10 100A 0104 1100 0A05 000A 0D00 0A0E 000A 6EB0”									

4. Save the IMU and Estimation Filter MIP Message Format

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

	MIP Packet Header				Command/Reply Fields			Checksum	
	SYNC1 “u	*SYNC2 “e”*	*Descriptor Set byte*	*Payload Length*	*Field Length*	*Cmd. Desc.*	*Field Data*	*MSB*	*LSB*
Command Field 1: *Save Current IMU Message Format*	0x75	0x65	0x0C	0x08	0x04	0x08	Function: 0x03 Desc. count: 0x00		
Command Field 2: *Save Current Estimation Filter Message Format*					0x04	0x0A	Function: 0x03 Desc. count: 0x00	0x0E	0x31
Reply Field 1: *ACK/NACK*	0x75	0x65	0x0C	0x08	0x04	0xF1	Cmd echo: 0x08 Error code: 0x00		
Reply Field 2: *ACK/NACK*					0x04	0xF1	Cmd echo: 0x0A Error code: 0x00	0xEA	0x71
Copy-Paste version of the command: “7565 0C08 0408 0300 040A 0300 0E31”									

5. Enable the IMU and Estimation Filter Data-streams

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

	MIP Packet Header				Command/Reply Fields			Checksum	
	SYNC1 "u	*SYNC2 "e"*	*Descriptor Set byte*	*Payload Length*	*Field Length*	*Cmd. Desc.*	*Field Data*	*MSB*	*LSB*
Command Field 1: *Enable Continuous IMU Message*	0x75	0x65	0x0C	0x0A	0x05	0x11	Function: 0x01 IMU: 0x01 On: 0x01		
Command Field 2: *Enable Continuous Estimation Filter Message*					0x05	0x11	Function: 0x01 Estimation Filter: 0x03 On: 0x01	0x24	0xCC
Reply Field 1: *ACK/NACK*	0x75	0x65	0x0C	0x08	0x04	0xF1	Cmd echo: 0x11 Error code: 0x00		
Reply Field 2: *ACK/NACK*					0x04	0xF1	Cmd echo: 0x11 Error code: 0x00	0xFA	0xB5
Copy-Paste version of the command: "7565 0C0A 0511 0101 0105 1101 0301 24 CC"									

6. Resume the Device: (Optional)

Sending the "Resume" command is another method of re-enabling transmission of enabled data streams. If the "Resume" command is sent *before* the "Enable IMU Data Stream" command, the node will resume the state it was in when the "Idle" command was sent. If the "Resume" command is sent *after* enabling the IMU Data Stream, the node will continue streaming. (reply is ACK/NACK).

	MIP Packet Header				Command/Reply Fields			Checksum	
	SYNC1 "u	*SYNC2 "e"*	*Descriptor Set byte*	*Payload Length*	*Field Length*	*Cmd. Desc.*	*Field Data*	*MSB*	*LSB*
Command: *Resume*	0x75	0x65	0x01	0x02	0x02	0x06	N/A	0xE5	0xCB
Reply: *ACK/NACK*	0x75	0x65	0x01	0x04	0x04	0xF1	Cmd echo: 0x06 Error code: 0x00	0xDA	0x74
Copy-Paste version of the command: "7565 0102 0206 E5CB"									

7. Initialize the Filter

At this point in the set-up, the 3DM-CV5-15 is streaming data, but the Kalman Filter is not yet initialized. The orientation may be initialized in different ways: Setting all of the attitude elements manually, setting only the heading and allowing the device to determine pitch and roll, using the internal IMU solution to provide the initial orientation, or via auto-initialization, which uses the chosen heading update source to initialize. In this example, we will assume the magnetometers are available and use the IMU solution to initialize the Kalman Filter. Once the attitude is initialized and the GPS fix becomes valid, the Kalman Filter estimation will propagate. Note that this step is not necessary if you have the auto-initialize option enabled:

Poll for current Complementary Filter Euler Angle output:

	MIP Packet Header				Command/Reply Fields			Checksum	
	SYNC1 "u	*SYNC2 "e"*	*Desc. Set*	*Payload Length*	*Field Length*	*Cmd. Desc.*	*Field Data*	*MSB*	*LSB*
Command: *Poll for CF Euler*	0x75	0x65	0x0C	0x07	0x07	0x01	Function: 0x00 Field Count: 0x00 Euler Desc: 0x06 Reserved: 0x00	0x02	0xFC
Reply Field 1: *ACK/NACK*	0x75	0x65	0x0C	0x04	0x04	0xF1	Cmd echo: 0x01 Error code: 0x00	0xE0	0xAC
Reply Field 2: *Data Packet*	0x75	0x65	0x80	0x0E	0x0E	0x0C	Roll: 0xBAE3ED9B Pitch: 0x3C7D6DDF Yaw: 0xBF855CF5	0x41	0xBB
Copy-Paste version of the command: "7565 0C07 0701 0001 0C00 0002 FC"									

Initialize attitude:

	MIP Packet Header				Command/Reply Fields			Checksum	
	SYNC1 "u	*SYNC2 "e"*	*Desc. Set*	*Payload Length*	*Field Length*	*Cmd. Desc.*	*Field Data*	*MSB*	*LSB*
Command: *Initialize Attitude*	0x75	0x65	0x0D	0x06	0x06	0x02	Roll: 0xBAE3ED9B Pitch: 0x3C7D6DDF Yaw: 0xBF855CF5	0xC4	0x09
Reply : *ACK/NACK*	0x75	0x65	0x0D	0x04	0x04	0xF1	Cmd echo: 0x02 Error code: 0x00	0xE2	0xB4
Copy-Paste version of the command: "7565 0D0E 0E02 BAE3 ED9B 3C7D 6DDF BF85 5CF5 C409"									

2.4.2 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:

1. **Put the Device in Idle Mode (Disabling the data-streams)**
   Same as continuous streaming (*see Put the Device in Idle Mode on page 14*).

2. **Configure the IMU data-stream format**
   Same as continuous streaming (*see Configure the IMU data-stream format on page 15*).

3. **Configure the Estimation Filter data-stream format**
   Same as continuous streaming (*see Configure the Estimation Filter data-stream format on page 16*).

4. **Save the IMU and Estimation Filter MIP Message format**
   Same as continuous streaming (*see Save the IMU and Estimation Filter MIP Message Format on page 17*).

5. **Enable the IMU and Estimation Filter data-streams**
   Same as continuous streaming (*see Enable the IMU and Estimation Filter Data-streams on page 18*).

6. **Resume the Device**
   Returns to the state when Idle was called, except for when Enable Stream command is sent (*see Resume the Device (Optional) on page 19*).

7. **Initialize the Filter**
   Same as continuous streaming (*see Initialize the Filter on page 20*).

Send Individual Data Polling Commands

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

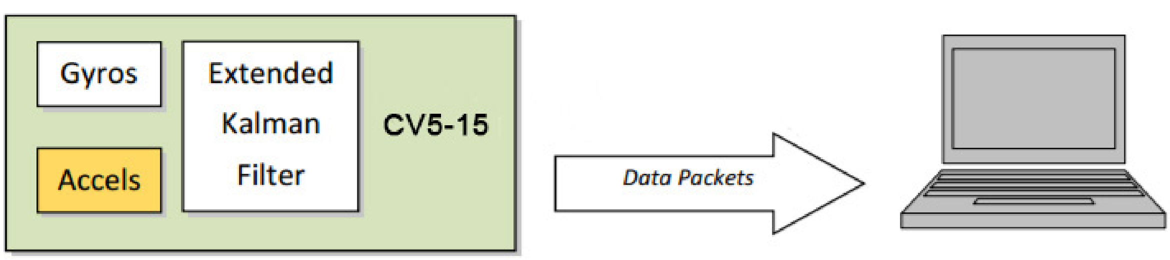
	MIP Packet Header				Command/Reply Fields			Checksum	
	SYNC1 "u	*SYNC2 "e"*	*Desc. Set*	*Payload Length*	*Field Length*	*Cmd. Desc.*	*Field Data*	*MSB*	*LSB*
Command: *Poll IMU Data*	0x75	0x65	0x0C	0x04	0x04	0x01	Option: 0x00 Desc Count: 0x00	0xEF	0xDA
Reply Field 1: *ACK/NACK*	0x75	0x65	0x0C	0x04	0x04	0xF1	Cmd echo: 0x01 Error code: 0x00	0xE0	0xAC
IMU Data Packet Field 1: *Gyro Vector*	0x75	0x65	0x80	0x1C	0x0E	0x04	0x3E 7A 63 A0 0xBB 8E 3B 29 0x7F E5 BF 7F	0x41	0xBB
IMU Data Packet Field 2: *Accel Vector*					0x0E	0x03	0x3E 7A 63 A0 0xBB 8E 3B 29 0x7F E5 BF 7F	0xAD	0xDC
Copy-Paste version of the command: "7565 0C04 0401 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 IMU Data and Poll Estimation Filter Data sections)

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

2.5 Parsing Incoming Packets

Setup is usually the easy part of programming the 3DM-CV5-15. Once you start continuous data streaming, parsing and processing the incoming data packet stream will become the primary focus. The stream of data from the IMU and Kalman Filter (Estimation Filter) are usually the dominant source of data since they come in the fastest. Polling for data may seem to be a logical solution to controlling the data flow, and this may be appropriate for some applications, but if your application requires the precise delivery of inertial data, it is often necessary to have the data stream drive the process rather than having the host try to control the data stream through polling.



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

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

Based on the value of the descriptor set field in the packet header, pass the packet to either a command handler (if it is a Base command or 3DM command descriptor set) or a data handler (if it is an IMU, or Estimation Filter data set). Since you know beforehand that the IMU and Estimation Filter data packets will be coming in fastest, you can tune your code to buffer or handle these packets at a high priority. Replies to commands generally happen sequentially after a command so the incidence of these is under program control.

For multi-threaded 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.

2.6 Multiple Rate Data

The message format commands (IMU Message Format and Estimation Filter Message Format) allow you to set different data rates for different data quantities. This is a very useful feature especially for IMU data because some data, such as accelerometer and gyroscope data, usually requires higher data rates (>100 Hz) than other IMU data such as Magnetometer (20 Hz 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 100 Hz and Delta Theta data at 50 Hz, the Delta Theta 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 100 Hz and every packet will have an accelerometer data field and EVERY OTHER packet will also include a Delta Theta data field:

Packet 1	*Packet 2*	*Packet 3*	*Packet 4*	*Packet 5*	*Packet 6*	*Packet 7*	*Packet 8*	*...*
Accel	Accel Delta Theta	Accel	Accel Delta Theta	Accel	Accel Delta Theta	Accel	Accel Delta Theta	Accel

If a timestamp is included at 100 Hz, 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 Delta Theta Timestamp	Accel Timestamp	Accel Delta Theta Timestamp	Accel Timestamp	Accel Delta Theta Timestamp	Accel

2.7 Data Synchronicity

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

IMU and Estimation Filter data are generated independently by two systems with different clocks. The importance of time is different in each system and the data they produce. The IMU data requires precise microsecond resolution and perfectly regular intervals in its timestamps. The Kalman Filter resides on a separate processor and must derive its timing information from the two data sources.

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

All data streams (IMU and Estimation Filter) on the 3DM-CV5-15 output a “GPS Time”-formatted timestamp. This allows a precise common time base for all data. Due to the differences in clocks on each device, the period between two consecutive timestamp values may not be constant; this occurs because periodic corrections are applied to the IMU and Estimation Filter timestamps when the GPS Time Update Command is applied.

2.8 Communications Bandwidth Management

Because of the large amount and variety of data that is available from the 3DM-CV5-15, it is quite easy to overdrive the bandwidth of the communications channel. This can result in dropped packets. The 3DM-CV5-15 does not do analysis of the bandwidth requirements for any given output data configuration, it will simply drop a packet if its internal serial buffer is being filled faster than it is being emptied. It is up to the programmer to analyze the size of the data packets requested and the available bandwidth of the communications channel. Often the best way to determine this is empirically by trying different settings and watching for dropped packets. Below are some guidelines on how to determine maximum bandwidth for your application.

2.8.1 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 of data rate in Hz

f_{mr} = maximum date rate in Hz

n = size of UART word = 10 bits

k = size of MIP wrapper = 6 bytes

which becomes:

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

Example:

For an IMU message format of Accelerometer Vector (14 byte data field) + Internal Timestamp (six 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.

2.8.2 USB vs. UART

The 3DM-CV5-15 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-CV5-15 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-CV5-15) 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-CV5-15 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.

3. 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, Estimation Filter Command, IMU data,or Estimation Filter data) and the second value denotes the unique command or data "descriptor" in that set. The pair of values constitutes a "full descriptor".

3.1 Commands

3.1.1 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)
Get Extended Device Descriptor Sets	(0x01, 0x07)
GPS Time Update	(0x01, 0x72)
Device Reset	(0x01, 0x7E)

3.1.2 3DM Command Set (0x0C)

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

3.1.3 Estimation Filter Command Set (0x0D)

Reset Filter	(0x0D, 0x01)

3.1.4 System Command Set (0x7F)

**Advanced commands*

3.2 Data

3.2.1 IMU Data Set (0x80)

3.2.2 Estimation Filter Data Set (0x82)

Filter Status	(0x82, 0x10)
GPS Timestamp	(0x82, 0x11)
Orientation, Quaternion	(0x82, 0x03)
Attitude Uncertainty, Quaternion Elements	(0x82, 0x12)
Orientation, Euler Angles	(0x82, 0x05)
Attitude Uncertainty, Euler Angles	(0x82, 0x0A)
Orientation, Matrix	(0x82, 0x04)
Compensated Angular Rate	(0x82, 0x0E)
Gyro Bias	(0x82, 0x06)
Gyro Bias Uncertainty	(0x82, 0x0B)
Compensated Linear Acceleration	(0x82, 0x1C)
Linear Acceleration	(0x82, 0x0D)
Pressure Altitude	(0x82, 0x21)
Gravity Vector	(0x82, 0x13)
WGS84 Local Gravity Magnitude	(0x82, 0x0F)
Heading Update Source State	(0x82, 0x14)

4. Command Reference

4.1 Base Commands

The Base command set is common to many LORD Sensing 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.

4.1.1 Ping (0x01, 0x01)

Description	Send "Ping" command Device responds with ACK if present.								
Field Format	*Field Length*		*Field Descriptor*			*Field Data*			
Command	0x02		0x01			N/A			
Reply: ACK/ NACK	0x04		0xF1			U8 - echo the command byte U8 - error code (0: ACK, non-zero: NACK)			
Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc. Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command: Ping	**0x75**	**0x65**	**0x01**	**0x02**	**0x02**	**0x01**		**0xE0**	**0xC6**
Reply: ACK/NACK	**0x75**	**0x65**	**0x01**	**0x04**	**0x04**	**0xF1**	**Command echo: 0x01 Error code: 0x00**	**0xD5**	**0x6A**
Copy-Paste version of the command: “7565 0102 0201 E0C6”									

4.1.2 Set To Idle (0x01, 0x02)

Description	Place device into idle mode Command has no parameters. Device responds with ACK if successfully placed in idle mode. This command will suspend streaming (if enabled) or wake the device from sleep (if sleeping) to allow it to respond to status and setup commands. You may restore the device mode by issuing the Resume command.									
Field Format	*Field Length*				*Field Descriptor*		*Field Data*			
Command	0x02				0x02		N/A			
Reply : ACK/ NACK	0x04				0xF1		U8 - echo the command byte U8 - error code (0: ACK, non-zero: NACK)			
Example	MIP Packet Header				Command/Reply Fields			Checksum		
	Sync1	*Sync2*	*Desc. Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*	
Command: Set to Idle	**0x75**	**0x65**	**0x01**	**0x02**	**0x02**	**0x02**		**0xE1**	**0xC7**	
Reply: ACK/NACK	**0x75**	**0x65**	**0x01**	**0x04**	**0x04**	**0xF1**	**Command echo: 0x02 Error code: 0x00**	**0xD6**	**0x6C**	
Copy-Paste version of the command: “7565 0102 0202 E1C7”										

4.1.3 Get Device Information (0x01, 0x03)

Description	Get the device ID strings and firmware version.					
Field Format	*Field Length*	*Field Descriptor*	*Field Data*			
Command	0x02	0x03	N/A			
Reply Field 1: ACK/ NACK	0x04	0xF1	U8 - echo the command byte U8 - error code (0: ACK, non-zero: NACK)			
Reply Field 2: Array of Descriptors	0x54	0x81	*Binary Offset*	*Description*	*Data Type*	*Units*
			0	Firmware version	U16	N/A
			2	Model Name	String(16)	N/A
			18	Model Number	String(16)	N/A
			34	Serial Number	String(16)	N/A
			50	Reserved	String (16)	N/A
			66	Options	String (16)	N/A

Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc. Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command: Get Device Info	**0x75**	**0x65**	**0x01**	**0x02**	**0x02**	**0x03**		**0xE2**	**0xC8**
Reply Field 1: ACK/NACK	**0x75**	**0x65**	**0x01**	**0x58**	**0x04**	**0xF1**	**Command echo: 0x03 Error code: 0x00**		
Reply Field 2: Device Info Field					**0x54**	**0x81**	**FW Version: 0x05FE** " 3DM-GX5-45" " 6232-4270" " 6232-00122" " " " 5g, 150d/s"	**0x##**	**0x##**

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

4.1.4 Get Device Descriptor Sets (0x01, 0x04)

<table>
<tr><td>Description</td><td colspan="5">Get the set of descriptors that this device supports

Reply has two fields: “ACK/NACK” and “Descriptors”. The “Descriptors” field is an array of 16 bit values. The MSB specifies the descriptor set and the LSB specifies the descriptor.</td></tr>
<tr><td>Field Format</td><td>Field Length</td><td>Field Descriptor</td><td colspan="3">Field Data</td></tr>
<tr><td>Command</td><td>0x02</td><td>0x04</td><td colspan="3">N/A</td></tr>
<tr><td>Reply Field 1: ACK/ NACK</td><td>0x04</td><td>0xF1</td><td colspan="3">U8 - echo the command byte
U8 - error code (0: ACK, non-zero: NACK)</td></tr>
<tr><td rowspan="6">Reply Field 2: Array of Descriptors</td><td rowspan="6"><(2 x n) + 2></td><td rowspan="6">0x82</td><td>Binary Offset</td><td>Description</td><td>Data Type</td></tr>
<tr><td rowspan="2">0</td><td>MSB: Descriptor Set</td><td rowspan="2">U16</td></tr>
<tr><td>LSB: Descriptor</td></tr>
<tr><td rowspan="2">2</td><td>MSB: Descriptor Set</td><td rowspan="2">U16</td></tr>
<tr><td>LSB: Descriptor</td></tr>
<tr><td>...</td><td>etc.</td><td>...</td></tr>
</table>

<table>
<tr><td rowspan="2">Example</td><td colspan="4">MIP Packet Header</td><td colspan="3">Command/Reply Fields</td><td colspan="2">Checksum</td></tr>
<tr><td>Sync1</td><td>Sync2</td><td>Desc. Set</td><td>Payload Length</td><td>Field Length</td><td>Field Desc.</td><td>Field Data</td><td>MSB</td><td>LSB</td></tr>
<tr><td>Command: Get Device Info</td><td>0x75</td><td>0x65</td><td>0x01</td><td>0x02</td><td>0x02</td><td>0x04</td><td></td><td>0xE3</td><td>0xC9</td></tr>
<tr><td>Reply Field 1: ACK/NACK</td><td>0x75</td><td>0x65</td><td>0x01</td><td>0x04</td><td>0x04</td><td>0xF1</td><td>Command echo: 0x01
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><(2 x n) + 2></td><td>0x82</td><td>0x0101
0x0102
0x0103
...
0x0C01
0x0C02
...
nth descriptor:</td><td>0x##</td><td>0x##</td></tr>
<tr><td colspan="10">Copy-Paste version of the command: “7565 0102 0204 E3C9”</td></tr>
</table>

4.1.5 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-CV5-15 are defined below.

3DM-CV5-15 BIT Error Flags:

Byte	Byte 1 (LSB)	Byte 2	Byte 4 (MSB)
Device	Processor Board	Sensor Board	Kalman Filter
Bit 1 (LSB)	WDT Reset (Latching, Reset after first commanded BIT)	IMU Communication Fault	Solution Fault
Bit 2	Reserved	Magnetometer Fault (if applicable)	Reserved
Bit 3	Reserved	Pressure Sensor Fault (if applicable)	Reserved
Bit 4	Reserved	Reserved	Reserved
Bit 5	Reserved	Reserved	Reserved
Bit 6	Reserved	Reserved	Reserved
Bit 7	Reserved	Reserved	Reserved
Bit 8 (MSB)	Reserved	Reserved	Reserved

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

Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc. Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command *Built-In Test*	0x75	0x65	0x01	0x02	0x02	0x05	N/A	0xE4	0xCA

Reply Field 1: ACK/NACK	0x75	0x65	0x01	0x0A	0x04	0xF1	Echo cmd: 0x05 Error code: 0x00		
Reply Field 2: BIT Error Flags					0x06	0x83	BIT Error Flags: 0x00000000	0x68	0x7D
Copy-Paste version of the command: "7565 0102 0205 E4CA"									

4.1.6 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 mod-e.Command has no parameters. Device responds with ACK if stream successfully enabled.								
Field Format	*Field Length*			*Field Descriptor*		*Field Data*			
Command	0x02			0x06		N/A			
Reply: ACK/ NACK	0x04			0xF1		U8 - echo the command byte U8 - error code (0: ACK, non-zero: NACK)			
Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc. Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command: Resume	**0x75**	**0x65**	**0x01**	**0x02**	**0x02**	**0x06**		**0xE5**	**0xCB**
Reply: ACK/NACK	**0x75**	**0x65**	**0x01**	**0x04**	**0x04**	**0xF1**	**Command echo: 0x01 Error code: 0x00**	**0xDA**	**0x74**
Copy-Paste version of the command: "7565 0102 0206 E5CB"									

4.1.7 Get Extended Device Descriptor Sets (0x01, 0x07)

Description	Get the extended set of descriptors that this device supports (descriptors in addition to the set returned by the Get Device Descriptors command) 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.				
Notes	The Get Device Descriptor Sets command is present on all devices supporting the MIP protocol. Extended descriptors are only supported on some devices. You may check for extended descriptors by searching for the 0x0107 descriptor in the list returned by the Get Device Descriptor Sets command.				
Field Format	*Field Length*	*Field Descriptor*	*Field Data*		
Command	0x02	0x07	N/A		
Reply Field 1: ACK/ NACK	0x04	0xF1	U8 - echo the command byte U8 - error code (0: ACK, non-zero: NACK)		
Reply Field 2: Array of Descriptors	4 x <Number of descriptors> + 2	0x86	*Binary Offset*	*Description*	*Data Type*
			0	MSB: Descriptor Set LSB: Descriptor	U16
			1	MSB: Descriptor Set LSB: Descriptor	U16
			...	etc.	...

Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc. Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command: Get Device Info	**0x75**	**0x65**	**0x01**	**0x02**	**0x02**	**0x04**		**0xE6**	**0xCC**
Reply Field 1: ACK/NACK	**0x75**	**0x65**	**0x01**	**0x04**	**0x04**	**0xF1**	**Command echo: 0x07** **Error code: 0x00**		
Reply Field 2: Array of Descriptors					**<n*2>**	**0x86**	**0x0D27** **0x0D28** **...** **0x822B** **0x822C** **...** **first extended descriptor** **...**	**0x##**	**0x##**

							nth extended descriptor		
Copy-Paste version of the command: "7565 0102 0207 E6CC"									

4.1.8 GPS Time Update (0x01, 0x72)

Description	This message updates the internal GPS Time as reported in the Filter Timestamp. This command enables synchronization of IMU/AHRS Timestamps with an external GPS receiver. When combined with a PPS input applied to pin 7 of the I/O connector, the GPS Correlation Timestamp in the inertial data output is synchronized with the external GPS clock. It is recommended that this update command be sent once per second. See the GPS Correlation Timestamp command for more information. *Possible function selector values:* 0x01 - Apply new settings 0x02 - Read back current settings 0x06 - Apply new settings with no ACK/NACK reply *Possible field selector values:* 0x01 - GPS Week Number 0x02 - GPS Seconds		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x08	0x72	U8 - Function Selector U8 - GPS Time Field Selector U32 - New Time Value
Reply: ACK/NACK	0x04	0xF1	U8 - echo the command descriptor U8 - error code (0: ACK, non-zero: NACK)
Reply Field 2 (function = 2, selector = 1)	0x06	0x84	U32 - Current GPS Week Value
Reply Field 2 (function = 2, selector = 2)	0x06	0x85	U32 - Current GPS Seconds Value

Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc. Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command:	0x75	0x65	0x01	0x08	0x08	0x72	Fctn (Apply): 0x01	0xFD	0x32

GPS Time Update							Field (Week): 0x00 Val: 0x00000698		
Reply : ACK/NACK	0x75	0x65	0x01	0x04	0x04	0xF1	Cmd echo: 0x72 Error code: 0x00	0x46	0x4C
Copy-Paste version of the command: "7565 0108 0872 0101 0000 0698 FD32"									

4.1.9 Device Reset (0x01, 0x7E)

Description	Resets the device. Device responds with ACK if it recognizes the command and then immediately resets.								
Field Format	*Field Length*		*Field Descriptor*		*Field Data*				
Command	0x02		0x7E		N/A				
Reply Field 1: ACK/ NACK	0x04		0xF1		U8 - Echo the command byte U8 - Error code (0: ACK, non-zero: NACK)				
Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc. Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command: Ping	0x75	0x65	0x01	0x02	0x02	0x7E		0x5D	0x43
Reply Field 1: ACK/NACK	0x75	0x65	0x01	0x04	0x04	0xF1	Command echo: 0x7E Error code: 0x00	0x52	0x64
Copy-Paste version of the command: "7565 0102 027E 5D43"									

4.2 3DM Commands

The 3DM command set is common to the LORD Sensing 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.

4.2.1 Poll IMU Data (0x0C, 0x01)

Description	Poll the device for an IMU message with the specified format This function polls for an IMU message using the provided format. The resulting message will maintain the order of descriptors sent in the command and any unrecognized descriptors are ignored. If the format is not provided, the device will attempt to use the stored format (set with the Set IMU Message Format command.) If no format is provided and there is no stored format, the device will respond with a NACK. The reply packet contains an ACK/NACK field. The polled data packet is sent separately as an IMU Data packet. Possible Option Selector Values: 0x00 - Normal ACK/NACK Reply. 0x01 - Suppress the ACK/NACK reply.								
Field Format	*Field Length*		*Field Descriptor*		*Field Data*				
Command	4 + 3*N		0x01		U8 - Option Selector U8 - Number of Descriptors (N) N*(U8 - Descriptor, U16 Reserved)				
Reply: ACK/ NACK	0x04		0xF1		U8 - echo the command byte U8 - error code (0: ACK, 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: Poll IMU data (use stored format)	0x75	0x65	0x0C	0x04	0x04	0x01	Option: 0x00 Desc count: 0x00	0xEF	0xDA
Command: Poll IMU data (use specified format)	0x75	0x65	0x0C	0x0A	0x0A	0x01	Option: 0x00 Desc count: 0x02 1st Descriptor: 0x04 Reserved: 0x0000 2nd Descriptor: 0x05 Reserved: 0x0000	0x06	0x27

Reply: ACK/NACK	0x75	0x65	0x0C	0x04	0x04	0xF1	Command echo: 0x01 Error code: 0x00	0xE0	0xAC
Copy-Paste versions of the commands: *Stored format: "7565 0C04 0401 0000 EFDA"* *Specified format: "7565 0C0A 0A01 0002 0400 0005 0000 0627"*									

4.2.2 Poll Estimation Filter Data (0x0C, 0x03)

Description	Poll the device for an Estimation Filter message with the specified format This function polls for an Estimation Filter message using the provided format. The resulting message will maintain the order of descriptors sent in the command and any unrecognized descriptors are ignored. If the format is not provided, the device will attempt to use the stored format (set with the Set Estimation Filter Message Format command.) If no format is provided and there is no stored format, the device will respond with a NACK. The reply packet contains an ACK/NACK field. The polled data packet is sent separately as an Estimation Filter Data packet. Possible Option Selector Values: 0x00 - Normal ACK/NACK Reply. 0x01 - Suppress the ACK/NACK reply.		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	4 + 3*N	0x03	U8 - Option Selector U8 - Number of Descriptors (N) N*(U8 - Descriptor, U16 Reserved)
Reply: ACK/ NACK	0x04	0xF1	U8 - echo the command byte U8 - error code (0: ACK, 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: Poll IMU data (use stored format)	0x75	0x65	0x0C	0x04	0x04	0x03	Option: 0x00 Desc count: 0x00	0xF1	0xE0
Command: Poll IMU data (use specified format)	0x75	0x65	0x0C	0x0A	0x0A	0x03	Option: 0x00 Desc count: 0x02 1st Descriptor: 0x01 Reserved: 0x0000 2nd Descriptor: 0x02 Reserved: 0x0000	0x02	0x1E
Reply: ACK/NACK (Data packet is sent separately if ACK)	0x75	0x65	0x0C	0x04	0x04	0xF1	Command echo: 0x03 Error code: 0x00	0xE2	0xB0

Copy-Paste versions of the commands:
Stored format: "7565 0C04 0403 0000 F1E0"

Specified format: "7565 0C0A 0A03 0002 0100 0002 0000 021E"

4.2.3 Get IMU Data Base Rate (0x0C, 0x06)

Description	Get the base rate for the IMU data in Hz. Returns the value used for data rate calculations. See the IMU Message Format command.		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x02	0x06	None
Reply Field 1: ACK/ NACK	0x04	0xF1	U8 - echo the command byte U8 - error code (0: ACK, non-zero: NACK)
Reply Field 2: IMU Base Rate	0x04	0x83	U16 - IMU data base rate (Hz)

Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc. Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command: Get IMU Base Rate	0x75	0x65	0x0C	0x02	0x02	0x06		0xF0	0xF7
Reply Field 1: ACK/NACK	0x75	0x65	0x0C	0x08	0x04	0xF1	Command echo: 0x06 Error code: 0x00		
Reply Field 2: IMU Base Rate					0x04	0x83	Base rate (Hz): 0x0x0064	0xD4	0x6B
Copy-Paste version of the command: "7565 0C02 0206 F0F7"									

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

Description	Get the base rate for the Estimation Filter data in Hz. Returns the value used for data rate calculations. See the Estimation Filter Message Format command.									
Field Format	*Field Length*		*Field Descriptor*		*Field Data*					
Command	0x02		0x0B		None					
Reply Field 1: ACK/ NACK	0x04		0xF1		U8 - Echo the command byte U8 - Error code (0: ACK, non-zero: NACK)					
Reply Field 2: IMU Base Rate	0x04		0x8A		U16 - Estimation Filter data base rate (Hz)					
Example	MIP Packet Header				Command/Reply Fields			Checksum		
	Sync1	*Sync2*	*Desc. Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*	
Command: Get IMU Base Rate	0x75	0x65	0x0C	0x02	0x02	0x0B		0xF5	0xFC	
Reply Field 1: ACK/NACK	0x75	0x65	0x0C	0x08	0x04	0xF1	Command echo: 0x0B Error code: 0x00			
Reply Field 2: Estimation Filter Base Rate					0x04	0x8A	Base rate (Hz): 0x0x0064	0xE0	0x9E	
Copy-Paste version of the command: “7565 0C02 020B F5FC”										

LORD SENSING MicroStrain

4.2.5 IMU Message Format (0x0C, 0x08)

Description	Set, read, or save the format of the IMU message packet. This command sets the format for the IMU data packet when in standard mode. The resulting data messages will maintain the order of descriptors sent in the command. The command has a function selector and a descriptor array as parameters. Possible Function Selector Values: 0x01 - Use new settings 0x02 - Read back current settings. 0x03 - Save current settings as startup settings 0x04 - Load saved startup settings 0x05 - Reset to factory default settings The rate decimation field is calculated as follows for IMU messages: Rate Decimation = IMU Base Rate / Desired Data Rate You should always retrieve the Base Rate from the Get IMU Data Base Rate command for computing the desired rate decimation. Base rates vary from device to device. The IMU base rate for the 3DM-CV5 is 500. The device checks that all descriptors are valid prior to executing this command. If any of the descriptors are invalid for the IMU descriptor set, a NACK will be returned and the message format will be unchanged. The descriptor array only needs to be provided if the function selector is = 1 (Use new settings). For all other functions it may be empty (Number of Descriptors = 0). **Figure 1 -**									
Field Format	*Field Length*	*Field Descriptor*	*Field Data*						
Command	4 + 3*N	0x08	U8 - Function Selector U8 - Number of Descriptors (N) N*(U8 - Descriptor, U16 - Rate Decimation)						
Reply Field 1: ACK/ NACK	0x04	0xF1	U8 - echo the command byte U8 - error code (0: ACK, non-zero: NACK)						
Reply Field 2 : Function = 2	3 + 3*N	0x80	U8 - Number of Descriptors (N) N*(U8 - Descriptor, U16 - Rate Decimation)						
Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc. Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command: IMU Message	**0x75**	**0x65**	**0x0C**	**0x0A**	**0x0A**	**0x08**	**Function: 0x01 Desc count: 0x02**	**0x22**	**0xA0**

Format (use new settings)							1st Descriptor: 0x04 Rate Dec: 0x000A 2nd Descriptor: 0x05 Rate Dec: 0x000A		
Reply Field : ACK/NACK	0x75	0x65	0x0C	0x04	0x04	0xF1	Echo cmd: 0x01 Error code: 0x00	0xE7	0xBA
Command: IMU Message Format (read back cur-rent settings)	0x75	0x65	0x0C	0x04	0x04	0x08	Function: 0x02 Desc count: 0x00	0xF8	0xF3
Reply Field 1: ACK/NACK	0x75	0x65	0x0C	0x0D	0x04	0xF1	Echo cmd: 0x08 Error code: 0x00		
Reply Field 2 : Current IMU Message Format					0x09	0x80	Desc count: 0x02 1st Descriptor: 0x03 Rate Dec: 0x000A 2nd Descriptor: 0x04 Rate Dec: 0x000A	0x98	0x0F
Copy-Paste version of the commands: Use New Settings:"7565 0C0A 0A08 0102 0400 0A05 000A 22A0" Read Current Settings: "7565 0C04 0408 0200 F8F3"									

4.2.6 Estimation Filter Message Format (0x0C, 0x0A)

Description

Set, read, or save the format of the Estimation Filter message packet. This function sets the format for the Estimation Filter 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.

Possible function selector values:

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

The rate decimation field is calculated as follows for Estimation Filter messages:

Rate Decimation = EF Base Rate / Desired Data Rate

You should always retrieve the Base Rate from the Get Estimation Filter Data Base Rate command for computing the desired rate decimation. Base rates vary from device to device. The EF base rate for the 3DM-CV5 is 500.

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

Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	4 + 3*N	0x0A	U8 - Function Selector U8 - Number of Descriptors (N) N*(U8 - Descriptor, U16 - Rate Decimation)
Reply Field 1: ACK/ NACK	0x04	0xF1	U8 - echo the command descriptor U8 - error code (0: ACK, non-zero: NACK)
Reply Field 2: Function = 2	3 + 3*N	0x82	U8 - Number of Descriptors (N) N*(U8 - Descriptor, U16 - Rate Decimation)

Examples	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc. Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command:	0x75	0x65	0x0C	0x0A	0x0A	0x0A	Function: 0x01	0x0C	0x6A

Estimation Filter Message Format (use new settings)							Desc count: 0x02 1st Descriptor: 0x01 Data Rate: 0x0001 2nd Descriptor: 0x02 Data Rate: 0x0001		
Reply Field : ACK/NACK	0x75	0x65	0x0C	0x04	0x04	0xF1	Echo cmd: 0x0A Error code: 0x00	0xE9	0xBE
Command: Estimation Filter Message Format (read back current settings)	0x75	0x65	0x0C	0x04	0x04	0x0A	Function: 0x02 Desc count: 0x00	0xFA	0xF9
Reply Field 1: ACK/NACK	0x75	0x65	0x0C	0x0D	0x04	0xF1	Echo cmd: 0x0A Error code: 0x00		
Reply Field 2 : Current Message Format					0x09	0x82	Desc count: 0x02 1st Descriptor: 0x01 Data Rate: 0x0001 2nd Descriptor: 0x02 Data Rate: 0x0001	0x84	0xED
Copy-Paste version of the commands: Use New Settings: "7565 0C0A 0A09 0102 0300 0405 0004 1685" Read Current Settings: "7565 0C04 0409 0200 F9F6"									

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

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

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

OFF							Stream (OFF): 0x00		
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"									

4.2.8 Device Startup Settings (0x0C, 0x30)

Description	Read, Save, Load, or Reset to Default the values for all device settings. Possible function selector values: 0x03 - Save current settings as startup settings** 0x04 - Load saved startup settings 0x05 - Reset to factory default settings							
Notes	***When a save current settings command is issued a brief data disturbance may occur as all settings are written to non-volatile memory.*							
Field Format	*Field Length*		*Field Descriptor*		*Field Data*			
Command	0x03		0x30		U8 - Function selector			
Reply: ACK/ NACK	0x04		0xF1		U8 - echo the command byte U8 - error code (0: ACK, non-zero: NACK)			
Examples	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc. Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command: Save All	**0x75**	**0x65**	**0x0C**	**0x03**	**0x03**	**0x30**	Fctn (Save): **0x03**	**0x1F**	**0x45**
Reply: ACK/NACK	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0xF1**	Echo cmd: **0x30** Error code: **0x00**	**0x0F**	**0x0A**
Copy-Paste version of the command: "7565 0C03 0330 031F 45"									

4.2.9 Accel Bias (0x0C, 0x37) *Advanced*

Description	Set the value, or read the current value of the IMU7 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. 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 byte U8 - error code (0: ACK, non-zero: 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

Examples	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 Field : 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"

4.2.10 Gyro Bias (0x0C, 0x38)

Advanced

Description	Set the value, or read the current value of the IMU7 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. 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 byte U8 - error code (0: ACK, non-zero: 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			
Examples	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 Field : 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"									

4.2.11 Capture Gyro Bias (0x0C, 0x39)

Description	This command will cause the 3DM-CV5-15 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 Gyro Bias command. Possible sampling time values: Total sampling time in units of milliseconds. Range of values: 1000 to 3000.		
Notes	Note: The 3DM-CV5-15 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 byte U8 - error code (0: ACK, non-zero: 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

Examples	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**	**0x04**	**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"

LORD SENSING MicroStrain

4.2.12 Coning and Sculling Enable (0x0C, 0x3E)

<table>
<tr><td>Description</td><td colspan="9">Set, read, or save the Coning and Sculling Compensation Enable. This function sets the Coning and Sculling Compensation Enable. For all functions except 0x01 (use new setting), the new parameter values are ignored.

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 enable flag can be either:
0x00 - Disable the Coning and Sculling compensation
0x01 - Enable the Coning and Sculling compensation (default)</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">0x10</td><td colspan="2">0x3E</td><td colspan="5">U8 - Function selector
U8 - New Coning and Sculling enable setting</td></tr>
<tr><td>Reply Field 1: ACK/ NACK</td><td colspan="2">0x04</td><td colspan="2">0xF1</td><td colspan="5">U8 - echo the command descriptor
U8 - error code (0: ACK, non-zero: NACK)</td></tr>
<tr><td>Reply Field 2: Function = 2</td><td colspan="2">0x03</td><td colspan="2">0x9E</td><td colspan="5">U8 - Current Coning and Sculling enable setting</td></tr>
<tr><td rowspan="2">Examples</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: Enable Settings</td><td>0x75</td><td>0x65</td><td>0x0C</td><td>0x04</td><td>0x04</td><td>0x3E</td><td>Fctn (Apply): 0x01
Enable: 0x01</td><td>0x2E</td><td>0x94</td></tr>
<tr><td>Reply Field : ACK/NACK</td><td>0x75</td><td>0x65</td><td>0x0C</td><td>0x04</td><td>0x04</td><td>0xF1</td><td>Echo cmd: 0x38
Error code: 0x00</td><td>0x1D</td><td>0x26</td></tr>
<tr><td colspan="10">Copy-Paste version of the command: “7565 0C04 043E 0101 2E94”</td></tr>
</table>

4.2.13 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. 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 - Reset to factory default settings Supported baud rates are: 9600, 19200, 115200 *(default)*, 230400, 460800, 921600		
Notes	*The ACK/NACK packet is sent at the current baud rate and then there is a 0.25 second delay before the device will respond to commands at the new BAUD rate.*		
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, non-zero: NACK)
Reply Field 2: Function = 2	0x06	0x87	U32 - Current baud rate

Examples	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	**0x75**	**0x65**	**0x0C**	**0x07**	**0x07**	**0x40**	Fctn (USE): **0x01** Baud (115200): **0x0001C200**	**0xF8**	**0xDA**
Reply Field : 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"

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

<table>
<tr><td rowspan="1">Description</td><td colspan="3">Advanced configuration for low-pass filter settings.

The scaled data quantities are by default filtered through a single-pole IIR low-pass filter which is configured with a -3dB cutoff frequency of half the reporting frequency (set by decimation factor in the IMU Message Format command) to prevent aliasing on a per data quantity basis. This advanced configuration command allows for the cutoff frequency to be configured independently of the data reporting frequency as well as allowing for a complete bypass of the digital low-pass filter.

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 - Reset to factory default settings

Possible data descriptors:
0x04 - Scaled accel data
0x05 - Scaled gyro data
0x06 - Scaled mag data (if applicable)
0x17 - Scaled pressure data

Possible filter enable values:
0x01 - Apply low-pass filter
0x00 - Do not apply low-pass filter

Manual filter bandwidth configuration:
0x01 - Use user specified -3 dB cutoff frequency
0x00 - Automatically configure -3 dB cutoff frequency to half reporting rate

-3 dB Cutoff Frequency:
Cutoff Frequency value specified must be no greater than 250 Hz.
**This value in a write command is ignored if Automatic Bandwidth is selected.

Reserved Byte:
This byte is reserved for internal use and should be left in the 0x00 state</td></tr>
<tr><td>Field Format</td><td>Field Length</td><td>Field Descriptor</td><td>Field Data</td></tr>
</table>

Command	0x09	0x50	U8 - Function selector U8 - Data Descriptor U8 - Low-Pass Filter Enable/Disable U8 - Manual/Auto -3 dB Cutoff Frequency Configuration U16 - -3 dB Cutoff Frequency U8 - Reserved Byte
Reply Field 1: ACK/ NACK	0x04	0xF1	U8 - echo the command descriptor U8 - error code (0: ACK, non-zero: NACK)
Reply Field 2: Function = 2	0x08	0x8B	U8 - Data Descriptor U8 - Filter (0x01: Enabled, 0x00: Disabled) U8 - Cutoff Frequency (0x00: Auto, 0x01: Manual) U16 - -3 dB Cutoff Frequency Hz U8 - Reserved

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

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

4.2.15 Complementary Filter Settings (0x0C, 0x51)

Description	Configuration for the AHRS complementary filter. The Complementary Filter data outputs are supported in the IMU/AHRS Data set (0x80) to provide compatibility with the 3DM-GX3. Possible function selector values: 0x01 - Use new settings 0x02 - Read back current settings 0x03 - Save current settings as startup settings 0x04 - Load saved startup settings 0x05 - Reset to factory default settings Possible up/north compensation enable values: 0x00 - Disable 0x01 - Enable (default) Range of up/north compensation time constants: 1-1000 seconds, default = 10 seconds Values outside of the specified range for up/north compensation will be NACK'd.		
Notes	*The Complementary Filter provides attitude outputs (Matrix, Euler, Quaternion, Up, and North) that are independent of the Estimation Filter outputs. The CF outputs are calculated using the same algorithm as the 3DM-CV5 series of Inertial Devices. This provides drop-in compatibility that duplicates the performance of the 3DM-CV5. It is highly recommended that you transition to the EF outputs as they will provide better performance as well as compatibility with higher grade devices such as the 3DM-RQ1.*		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x0D	0x51	U8 - Function selector U8 - Up compensation enable U8 - North compensation enable float - Up compensation time constant (sec) float - North compensation time constant (sec) U8 - echo the command descriptor U8 - error code (0:ACK, not 0:NACK)
Reply Field 1: ACK/ NACK	0x04	0xF1	U8 - echo the command descriptor U8 - error code (0: ACK, non-zero: NACK)
Reply Field 2: Function = 2	0x0C	0x97	U8 - Up compensation enable U8 - North compensation enable

			float - Up compensation time constant (sec) float - North compensation time constant (sec)

Examples	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc. Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command	**0x75**	**0x65**	**0x0C**	**0x0D**	**0x0D**	**0x51**	Fctn Selector (Write): **0x01** Up Compensation Enable: **0x01** North Compensation Enable: **0x01** Up Compensation Time Constant: **5.0 (sec)** North Compensation Time Constant: **5.0 (sec)**	**0xXX**	**0xXX**
Reply Field : ACK/NACK	**0x75**	**0x65**	**0x0C**	**0x04**	**0x04**	**0xF1**	Echo cmd: **0x51** Error code: **0x00**	**0x**	**0x**
Copy-Paste version of the command: “7565 0C09 0951 0104 0100 0000 00"									

4.2.16 Device Status (0x0C, 0x64)

Description

Get the device-specific status for the 3DM-CV5-15.

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-CV5-15 is always = 6258 (0x 1872). That is followed by a status selector byte which determines the type of data structure returned. In the case of the 3DM-CV5-15, 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

Notes

The reply field for this command is tightly tied to the model number. Make sure you check the model number in the reply and match it to the correct structure for the data field for the specific device model number. This reply data descriptor 0x0C,0X90 is an exception to the rule for MIP descriptors that the structure of descriptor data is the same for all devices. In this case, it is the same for all devices with the same model number but not necessarily the same for devices with different model numbers.

<table>
<tr><th>Field Format</th><th>Field Length</th><th>Field Descriptor</th><th colspan="4">Field Data</th></tr>
<tr><td>Command</td><td>0x02</td><td>0x64</td><td colspan="4">U16-Device Model Number: set = 6258 (0x 1872))
U8-Status Selector</td></tr>
<tr><td>Reply Field 1: ACK/ NACK</td><td>0x04</td><td>0xF1</td><td colspan="4">U8 - echo the command byte
U8 - error code (0: ACK, non-zero: NACK)</td></tr>
<tr><td rowspan="5">Reply Field 2: Basic Device Status Field</td><td rowspan="5">0x0F</td><td rowspan="5">0x90</td><td>Binary Offset</td><td>Description</td><td>Data Type</td><td>Units</td></tr>
<tr><td>0</td><td>Echo of the Device Model Num-ber</td><td>U16</td><td>N/A</td></tr>
<tr><td>2</td><td>Echo of the selector byte</td><td>U8</td><td>N/A</td></tr>
<tr><td>3</td><td>Status Flags (Reserved)</td><td>U32</td><td>N/A</td></tr>
<tr><td>7</td><td>System State</td><td>U16</td><td>N/A</td></tr>
</table>

			9	System Timer (since start-up)	U32	millisecond
Reply Field 2: Diagnostic Device Status Field	0x35	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	Status Flags (Reserved)	U32	N/A
			7	System State	U16	N/A
			9	System Timer (since start-up)	U32	millisecond
			13	IMU Stream Enabled	U8	1 - on 0 - off
			14	Estimation Filter Stream Enabled	U8	1 - on 0 - off
			15	Outgoing IMU Stream Dropped Packet Count	U32	count
			19	Outgoing Estimation Filter Stream Dropped Packet Count	U32	count
			23	Number of bytes written to com port	U32	count
			27	Number of bytes read from com port	U32	count
			31	Number of overruns when writing to com port	U32	count
			35	Number of overruns when reading com port	U32	count
			39	Number of IMU message parsing errors	U32	count
			43	Total IMU messages read	U32	count
			47	Last IMU message read (System Timer)	U32	millisecond

Examples	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc. Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command: Get Device	**0x75**	**0x65**	**0x0C**	**0x05**	**0x05**	**0x64**	Status selector (basic status): **0x01**		

Status (return Basic Status structure: selector = 1)									
Reply Field 1: ACK/NACK	0x75	0x65	0x0C	0x15	0x04	0xF1	Echo cmd: **0x64** Error code: **0x00**		
Reply Field 2: Device Status (Basic Status structure)					0x0F	0x90	Echo selector: **0x01** Additonal data: **...**	0x##	0x##

4.3 Estimation Filter Commands

The 3DM command set is common to the LORD Sensing 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.

4.3.1 Reset Filter (0x0D, 0x01)

Description	Reset the filter to the initialize state.								
Notes	If the auto-initialization feature is disabled, the initial attitude or heading must be set in order to enter the run state after a reset.								
Field Format	*Field Length*		*Field Descriptor*		*Field Data*				
Command	0x02		0x01		N/A				
Reply Field: ACK/ NACK	0x04		0xF1		U8 - Echo the command byte U8 - Error code (0: ACK, non-zero: NACK)				
Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc. Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command	**0x75**	**0x65**	**0x0D**	**0x02**	**0x02**	**0x01**		**0xEC**	**0xF6**
Reply Field: ACK/NACK	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	**Command echo: 0x01 Error code: 0x00**	**0xE1**	**0xB2**
Copy-Paste version of the command: "7565 0D02 0201 ECF6"									

4.3.2 Set Initial Attitude (0x0D, 0x02)

Description

Set the initial attitude.

This command can only be issued in the “INIT” state and should be used with a good estimate of the vehicle attitude. The Euler Angles are the sensor body frame with respect to the local NED frame.

The valid input ranges are as follows:

Roll: [-п, п]
Pitch: [-п/2, п/2]
Yaw: [-п, п]

Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x0E	0x02	Float - Roll (radians) Float - Pitch (radians) Float - Heading (radians)
Reply Field : ACK/ NACK	0x04	0xF1	U8 - echo the command byte U8 - error code (0: ACK, non-zero: NACK)

Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc. Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command	**0x75**	**0x65**	**0x0D**	**0x0E**	**0x0E**	**0x02**	Roll: **0x00000000 (0.0f)** Pitch: **0x00000000 (0.0f)** Heading: **0x00000000 (0x0f)**	**0x05**	**0x6F**
Reply Field: ACK/NACK	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	**Command echo: 0x02** **Error code: 0x00**	**0xE2**	**0xB4**

Copy-Paste version of the command: "7565 0D0E0E02 0000 0000 0000 0000 0000 0000 0000 0000 056F”

4.3.3 Set Initial Heading (0x0D, 0x03)

Description	Set the initial heading angle. This command can only be issued in the "INIT" state and should be used with a good estimation of Heading. The device will use this value in conjunction with the output of the accelerometers to determine the initial attitude estimate. The Euler Angles are the sensor body frame with respect to the local NED frame. The valid input range for heading is [-п, п].								
Field Format	*Field Length*		*Field Descriptor*		*Field Data*				
Command	0x06		0x03		Float - Heading (radians)				
Reply Field : ACK/ NACK	0x04		0xF1		U8 - Echo the command byte U8 - Error code (0: ACK, non-zero: NACK)				
Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc. Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command	**0x75**	**0x65**	**0x0D**	**0x06**	**0x06**	**0x03**	Heading: **0x00000000 (0x0f)**	**0xF6**	**0xE4**
Reply Field: ACK/NACK	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	**Command echo: 0x03 Error code: 0x00**	**0xE3**	**0xB6**
Copy-Paste version of the command: "7565 0D06 0603 0000 0000 F6E4"									

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

<table>
<tr><td>Description</td><td colspan="3">Set the sensor to vehicle frame transformation matrix using Roll, Pitch, and Yaw Euler angles.

These angles define the rotation from the sensor body frame to the fixed vehicle frame. Please reference the device Theory of Operation for more information.

Possible function selector values:
0x01 - Use new settings
0x02 - Read back current settings.
0x03 - Save current settings as startup settings
0x04 - Load saved startup settings
0x05 - Reset to factory default settings

This transformation affects the following output quantities:
IMU:
Scaled Acceleration
Scaled Gyro
Scaled Magnetometer
Delta Theta
Delta Velocity

Estimation Filter:
Estimated Orientation, Quaternion
Estimated Orientation, Matrix
Estimated Orientation, Euler Angles
Estimated Linear Acceleration
Estimated Angular Rate
Estimated Gravity Vector</td></tr>
<tr><td>Field Format</td><td>Field Length</td><td>Field Descriptor</td><td>Field Data</td></tr>
<tr><td>Command</td><td>0x0F</td><td>0x11</td><td>U8 - Function Selector
Float - Roll Angle (radians)
Float - Pitch Angle (radians)
Float - Yaw Angle (radians)</td></tr>
<tr><td>Reply Field 1: ACK/ NACK</td><td>0x04</td><td>0xF1</td><td>U8 - echo the command descriptor
U8 - error code (0: ACK, non-zero: NACK)</td></tr>
</table>

Reply Field 2: Function = 2	0x0E			0x81		Float - Roll Angle (radians) Float - Pitch Angle (radians) Float - Yaw Angle (radians)		
Example	MIP Packet Header				Command/Reply Fields		Checksum	
	Sync1	*Sync2*	*Desc. Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command	**0x75**	**0x65**	**0x0D**	**0x0F**	**0x0F**	**0x11**	Fctn (Apply): **0x01** Roll: **0x00000000 (0.0f)** Pitch: **0x00000000 (0.0f)** Yaw: **0x00000000 (0x0f)**	**0x17**	**0x72**
Reply Field : ACK/NACK	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	**Command echo: 0x11 Error code: 0x00**	**0xF1**	**0xD2**
Copy-Paste version of the command: "7565 0D0F 0F11 0100 0000 0000 0000 0000 0000 0017 72"									

4.3.5 Estimation Control Flags (0x0D, 0x14)

Description	Controls which parameters are estimated by the Kalman Filter. Possible function selector values: 0x01 - Use new settings 0x02 - Read back current settings. 0x03 - Save current settings as startup settings 0x04 - Load saved startup settings 0x05 - Reset to factory default settings Available Flags : 0x0001 - Enable Gyro Bias Estimation (Recommended) Examples : 0x0001 - Enable Gyro Bias Estimation		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x05	0x14	U8 - Function Selector U16 - Estimation Control Flags
Reply Field 1: ACK/ NACK	0x04	0xF1	U8 - Echo the command descriptor U8 - Error code (0: ACK, non-zero: NACK)
Reply Field 2: Function = 2	0x04	0x84	U16 - Estimation Control Flags

Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc. Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command:	**0x75**	**0x65**	**0x0D**	**0x05**	**0x05**	**0x14**	Fctn (Apply): **0x01** Flags: **0xFFFF (enable all states)**	**0x04**	**0x27**
Reply Field: ACK/NACK	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	Echo cmd: **0x14** Error code: **0x00**	**0xF4**	**0xD8**

Copy-Paste version of the command: "7565 0D05 0514 01FF FF04 27"

4.3.6 Heading Update Control (0x0D, 0x18)

<table>
<tr><td>Description</td><td colspan="9">Select the source for aiding heading updates to the Kalman Filter.
Possible function selector values:
0x01 - Use new settings
0x02 - Read back current settings.
0x03 - Save current settings as startup settings
0x04 - Load saved startup settings
0x05 - Reset to factory default settings
Possible Enable Option values:
0x00 - No heading aids
0x03 - Use external heading messages for heading updates</td></tr>
<tr><td>Notes</td><td colspan="9"></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">0x04</td><td colspan="2">0x18</td><td colspan="5">U8 - Function Selector
U8 - Enable Flag</td></tr>
<tr><td>Reply Field 1: ACK/ NACK</td><td colspan="2">0x04</td><td colspan="2">0xF1</td><td colspan="5">U8 - Echo the command descriptor
U8 - Error code (0: ACK, non-zero: NACK)</td></tr>
<tr><td>Reply Field 2: Function = 2</td><td colspan="2">0x03</td><td colspan="2">0x87</td><td colspan="5">U8 - Enable Flag</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</td><td>0x75</td><td>0x65</td><td>0x0D</td><td>0x04</td><td>0x04</td><td>0x18</td><td>Apply: 0x01
Enable: 0x01</td><td>0x09</td><td>0x28</td></tr>
<tr><td>Reply Field 1: ACK/NACK</td><td>0x75</td><td>0x65</td><td>0x0D</td><td>0x04</td><td>0x04</td><td>0xF1</td><td>Echo cmd: 0x18
Error code: 0x00</td><td>0xF8</td><td>0xE0</td></tr>
<tr><td colspan="10">Copy-Paste version of the command: "7565 0D04 0418 0101 0928"</td></tr>
</table>

4.3.7 External Heading Update (0x0D, 0x17)

Description	Trigger a filter update step using external heading information. **The heading must be the sensor frame with respect to the NED frame.** The heading update control must be set to external for this command to update the filter; it will be ignored/NACK'd otherwise. The maximum rate for this message is 20 Hz. Angle uncertainties of 0.0 will be NACK'd. Possible Heading Type Commands: 0x01 - True Heading* 0x02 - Magnetic Heading**								
Notes	• On the -25 model, if the declination source (0x0D, 0x43) is not valid, **true heading updates** will be NACK'd. • On the -45 model, if the declination source is invalid, **magnetic heading updates** will be NACK'd.								
Field Format	*Field Length*	*Field Descriptor*	*Field Data*						
Command	0x0B	0x17	Float - Heading Angle (radians, true north, +- PI) Float - Heading Angle Uncertainty (radians, 1-sigma) U8 - Heading type (1 - true, 2 - magnetic)						
Reply Field : ACK/ NACK	0x04	0xF1	U8 - Echo the command byte U8 - Error code (0: ACK, non-zero: NACK)						
Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc. Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command	0x75	0x65	0x0D	0x0B	0x0B	0x17	Angle: **0.1f** Angle Sigma: **0.1f** Heading Type: **0x01 (True)**	0xXX	0xXX
Reply Field: ACK/NACK	0x75	0x65	0x0D	0x04	0x04	0xF1	Echo cmd: **0x17** Error code: **0x00**	0xF7	0xDE
Copy-Paste version of the command: N/A									

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

Description	Trigger a filter update step using external heading information that is time-tagged with a specific GPS Time. This is more accurate than the External Heading Update (0x0D, 0x17) and should be used in applications where the vehicle heading experiences high angular rate, which may cause significant error in the applied measurement due to the sampling, transmission, and processing time required for the command. Accurate time-stamping of the heading information is important. The maximum rate for this message is 20 Hz. Angle uncertainties of 0.0 will be NACK'd. Possible Heading Type Commands: 0x01 - True Heading* 0x02 - Magnetic Heading* **The heading must be the sensor frame with respect to the NED frame.**		
Notes	• On the -25 model, if the declination source (0x0D, 0x43) is not valid, **true heading updates** will be NACK'd. • On the -45 model, if the declination source is invalid, **magnetic heading updates** will be NACK'd.		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x15	0x1F	Double - TOW (time-of-week, seconds) U16 - week number Float - Heading Angle (radians, true north, +- PI) Float - Heading Angle Uncertainty (radians, 1-sigma) U8 - Heading type (1 - true, 2 - magnetic)
Reply Field : 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	**0x75**	**0x65**	**0x0D**	**0x15**	**0x15**	**0x1F**	TOW: **30,000.0** Week Number: **1700** Angle: **(0.01f)** Angle Sigma: **(0.01f)** Heading **0x01**	**0xXX**	**0xXX**

							Type: **(True)**		
Reply Field: ACK/NACK	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	Echo Cmd: **0x01** Error Code: **0x00**	**0xFF**	**0xEE**
Copy-Paste version of the command: N/A									

4.3.9 Pitch/Roll Aiding Control (0x0D, 0x4B)

Description	Select pitch/roll aiding input. Aiding inputs are used to improve that solution during periods of low dynamics . *Possible function selector values:* 0x01 - Use new settings 0x02 - Read back current settings 0x03 - Save current settings as startup settings 0x04 - Load saved startup settings 0x05 - Reset to factory default settings *Possible altitude aiding selector values:* 0x00 - No pitch/roll aiding (disable) 0x01 - Enable gravity vector aiding								
Field Format	*Field Length*		*Field Descriptor*		*Field Data*				
Command	0x05		0x4B		U8 - Function Selector U8 - Aiding (0 - Disable, 1 - Enable)				
Reply Field: ACK/ NACK	0x04		0xF1		U8 - Echo the command descriptor U8 - Error code (0: ACK, non-zero: NACK)				
Reply Field : Function = 2	0x03		0xBB		U8 - Aiding Selector Value				
Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc. Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0x4B**	Fctn (Apply): **0x01** Enable: **0x01**	**0x3C**	**0xC1**
Reply Field: ACK/NACK	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	Echo cmd: **0x47** Error code: **0x00**	**0xB9**	**0xF0**
Copy-Paste version of the command: "7565 0D04 044B 0101 3CC1 "									

4.3.10 Auto-Initialization Control (0x0D, 0x19)

Description

Enable/Disable automatic initialization upon device startup.

Possible function selector values:

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

Possible enable values:

0x00 - Disable auto-initialization
0x01 - Enable auto-initialization (requires valid heading source)

Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x04	0x19	U8 - Function Selector U8 - Enable Value
Reply Field 1: ACK/ NACK	0x04	0xF1	U8 - Echo the command descriptor U8 - Error code (0: ACK, non-zero: NACK)
Reply Field 2: Function = 2	0x03	0x88	U8 - Enable Value

Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc. Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command:	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0x19**	Fctn (Apply): **0x01** Enable: **0x01 (Enable auto-initialization)**	**0x0A**	**0x2B**
Reply Field 1: ACK/NACK	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	Echo cmd: **0x19** Error code: **0x00**	**0xF9**	**0xE2**

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

4.3.11 Gravity Noise Standard Deviation (0x0D, 0x28)

Description	Set the expected gravity noise 1-sigma values. This function can be used to tune the filter performance in the target application. Each of the noise values must be greater than 0.0 The noise value represents process noise in the EKF. Changing this value modifies how the filter responds to dynamic input and can be used to tune the performance of the filter. Default values provide good performance for most laboratory conditions. Possible function selector values: 0x01 - Use new settings 0x02 - Read back current settings 0x03 - Save current settings as startup settings 0x04 - Load saved startup settings 0x05 - Reset to factory default settings		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x05	0x28	U8 - Function Selector Float - X Gravity Noise 1-sigma (g) Float - Y Gravity Noise 1-sigma (g) Float - Z Gravity Noise 1-sigma (g)
Reply Field 1: ACK/ NACK	0x04	0xF1	U8 - Echo the command descriptor U8 - Error code (0: ACK, non-zero: NACK)
Reply Field 2: Function = 2	0x04	0x93	Float - X Gravity Noise 1-sigma (g) Float - Y Gravity Noise 1-sigma (g) Float - Z Gravity Noise 1-sigma (g)

Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc. Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command	**0x75**	**0x65**	**0x0D**	**0x05**	**0x05**		*Fctn (Apply):* **0x01** *X:* **(0.01f)** *Y:* **(0.01f)** *Z:* **(0.01f)**	**0x**	**0x**
Reply Field 1: ACK/NACK	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	*Echo cmd:* **0x28** *Error code:* **0x00**	**0x**	**0x**

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

Description	Set the expected accelerometer noise 1-sigma values. This function can be used to tune the filter performance in the target application. Possible function selector values: 0x01 - Use new settings 0x02 - Read back current settings 0x03 - Save current settings as startup settings 0x04 - Load saved startup settings 0x05 - Reset to factory default settings Each of the noise values must be greater than 0.0 The noise value represents process noise in the 3DM-CV5-15 NAV Estimation Filter. Changing this value modifies how the filter responds to dynamic input and can be used to tune the performance of the filter. Default values provide good performance for most laboratory conditions.		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x0F	0x1A	U8 - Function Selector Float - X Accel Noise 1-sigma (meters/second^2) Float - Y Accel Noise 1-sigma (meters/second^2) Float - Z Accel Noise 1-sigma (meters/second^2) U8 - echo the command descriptor U8 - error code (0:ACK, not 0:NACK)
Reply Field 1: ACK/ NACK	0x04	0xF1	U8 - Echo the command descriptor U8 - Error code (0: ACK, non-zero: NACK)
Reply Field 2: Function = 2	0x0E	0x89	Float - X Accel Noise 1-sigma (meters/second^2) Float - Y Accel Noise 1-sigma (meters/second^2) Float - Z Accel Noise 1-sigma (meters/second^2)

Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc. Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command	**0x75**	**0x65**	**0x0D**	**0x0F**	**0x0F**	**0x1A**	Fctn (Apply): **0x01** X: **(0.02f)** Y: **(0.02f)** Z: **(0.02f)**	**0x60**	**0xA3**
Reply Field 1: ACK/NACK	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	Echo cmd: **0x1A** Error code: **0x00**	**0xFA**	**0xE4**

Copy-Paste version of the command: "7565 0D0F 0F01 1A013CA3D70A3CA3D70A3CA3D760A3"

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

Description	Set the expected gyroscope noise 1-sigma values. This function can be used to tune the filter performance in the target application. Possible function selector values: 0x01 - Use new settings 0x02 - Read back current settings 0x03 - Save current settings as startup settings 0x04 - Load saved startup settings 0x05 - Reset to factory default settings Each of the noise values must be greater than 0.0 The noise value represents process noise in the 3DM-CV5 NAV Estimation Filter. Changing this value modifies how the filter responds to dynamic input and can be used to tune the performance of the filter. Default values provide good performance for most laboratory conditions.		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x0F	0x1B	U8 - Function Selector Float - X Gyro Noise 1-sigma (rad/second) Float - Y Gyro Noise 1-sigma (rad/second) Float - Z Gyro Noise 1-sigma (rad/second)
Reply Field 1: ACK/ NACK	0x04	0xF1	U8 - Echo the command descriptor U8 - Error code (0: ACK, non-zero: NACK)
Reply Field 2: Function = 2	0x0E	0x8A	Float - X Gyro Noise 1-sigma (rad/second) Float - Y Gyro Noise 1-sigma (rad/second) Float - Z Gyro Noise 1-sigma (rad/second)

Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc. Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command	**0x75**	**0x65**	**0x0D**	**0x0F**	**0x0F**	**0x1B**	Fctn (Apply): **0x01** X: **(0.0000539f)** Y: **(0.0000539f)** Z: **(0.0000539f)**	**0xDE**	**0xE8**

Reply Field 1: ACK/NACK	0x75	0x65	0x0D	0x04	0x04	0xF1	Echo cmd: **0x1B** Error code: **0x00**	0xFB	0xE6
Copy-Paste version of the command: "7565 0D0F 0F1B 013A 0D4B AD3A 0D4B AD3A 0D4B ADDE E8"									

4.3.14 Gyroscope Bias Model Parameters (0x0D, 0x1D)

Description	Set the gyroscope bias model parameters. Possible function selector values: 0x01 - Use new settings 0x02 - Read back current settings 0x03 - Save current settings as startup settings 0x04 - Load saved startup settings 0x05 - Reset to factory default settings Each of the noise values must be greater than 0.0		
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x1B	0x1D	U8 - Function Selector Float - X Gyro Bias Beta (1/second) Float - Y Gyro Bias Beta (1/second) Float - Z Gyro Bias Beta (1/second) Float - X Gyro Bias Noise 1-sigma (rad /second) Float - Y Gyro Bias Noise 1-sigma (rad /second) Float - Z Gyro Bias Noise 1-sigma (rad /second)
Reply Field 1: ACK/ NACK	0x04	0xF1	U8 - Echo the command descriptor U8 - Error code (0: ACK, non-zero: NACK)
Reply Field 2: Function = 2	0x1A	0x8C	Float - X Gyro Bias Beta (1/second) Float - Y Gyro Bias Beta (1/second) Float - Z Gyro Bias Beta (1/second) Float - X Gyro Bias Noise 1-sigma (rad /second) Float - Y Gyro Bias Noise 1-sigma (rad /second) Float - Z Gyro Bias Noise 1-sigma (rad /second)

Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc. Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command	**0x75**	**0x65**	**0x0D**	**0x0F**	**0x1B**	**0x1D**	Fctn (Apply): **0x01** X Beta: **(0.01f)** Y Beta: **(0.01f)** Z Beta: **(0.01f)** X Noise: **(0.00016f)** Y Noise: **(0.00016f)** Z Noise: **(0.00016f)**	**0xXX**	**0xXX**
Reply Field 1: ACK/NACK	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	Echo cmd: **0x1D** Error code: **0x00**	**0xFD**	**0xEA**
Copy-Paste version of the command: N/A									

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

Description	Control the use of zero angular rate updates. Possible function selector values: 0x01 - Use new settings 0x02 - Read back current settings 0x03 - Save current settings as startup settings 0x04 - Load saved startup settings 0x05 - Reset to factory default settings The zero angular rate update is triggered when the scalar magnitude of the angular rate vector is equal-to or less than the threshold value. The device will NACK threshold values that are less than zero (i.e. negative.)								
Field Format	*Field Length*		*Field Descriptor*		*Field Data*				
Command	0x08		0x20		U8 - Function Selector U8 - Enable Value (0 - disable, 1 - enable) Float -Threshold (rad/s)				
Reply Field 1: ACK/ NACK	0x04		0xF1		U8 - Echo the command descriptor U8 - Error code (0: ACK, non-zero: NACK)				
Reply Field 2: Function = 2	0x07		0x8E		U8 - Enable Value Float - ZUPT threshold (rad/s)				
Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc. Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command	**0x75**	**0x65**	**0x0D**	**0x08**	**0x08**	**0x20**	Fctn (Apply): **0x01** Enable: **0x01 (Enable)** Threshold: **0x00000000 (0.0f)**	**0x19**	**0xC8**
Reply Field 1: ACK/NACK	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	Echo cmd: **0x20** Error code: **0x00**	**0x00**	**0xF0**
Copy-Paste version of the command: "7565 0D08 0820 0101 00000000 19C8"									

4.3.16 Tare Orientation (0x0D, 0x21)

<table>
<tr><td>Description</td><td colspan="10">This function uses the current device orientation relative to the NED frame as the current sensor to vehicle transformation.

This command is provided as a convenient way to set the sensor to vehicle frame transformation.

Possible function selector values:
0x01 - Use new settings
0x03 - Save current settings as startup settings
0x04 - Load saved startup settings
0x05 - Reset to factory default settings

Possible axis bitfield values:
0x00 - Reset all axis
0x01 - Tare the roll axis
0x02 - Tare the pitch axis
0x04 - Tare the yaw axis

Example Combinations:
0x03 - Tare the roll and pitch axis
0x07 - Tare all 3 axis

Note: The filter must be initialized and have a valid attitude output. If the attitude is not valid, an error will be returned.</td></tr>
<tr><td>Notes</td><td colspan="10">The filter must be initialized and have a valid attitude output. If the attitude is not valid, an error will be returned.</td></tr>
<tr><td>Field Format</td><td colspan="2">Field Length</td><td colspan="2">Field Descriptor</td><td colspan="6">Field Data</td></tr>
<tr><td>Command</td><td colspan="2">0x04</td><td colspan="2">0x21</td><td colspan="6">U8 - Function Selector
U8 - Tare Axis Bitfield</td></tr>
<tr><td>Reply Field: ACK/ NACK</td><td colspan="2">0x04</td><td colspan="2">0xF1</td><td colspan="6">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="4">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 colspan="2">Field Data</td><td>MSB</td><td>LSB</td></tr>
<tr><td>Command</td><td>0x75</td><td>0x65</td><td>0x0D</td><td>0x04</td><td>0x04</td><td>0x21</td><td colspan="2">Fctn (Apply): 0x01</td><td>0x18</td><td>0x49</td></tr>
</table>

							X:Beta: **0x07 (All axis)**		
Reply Field: ACK/NACK	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	Echo cmd: **0x21** Error code: **0x00**	**0x**	**0x**
Copy-Paste version of the command: "7565 0D04 0421 0107 1849"									

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

Description	Perform a commanded zero-angular rate update.							
Notes	The maximum rate for this message is 10 Hz.							
Field Format	*Field Length*		*Field Descriptor*		*Field Data*			
Command	0x02		0x23		N/A			
Reply Field : ACK/ NACK	0x04		0xF1		U8 - Echo the command byte U8 - Error code (0: ACK, non-zero: NACK)			
Example	MIP Packet Header				Command/Reply Fields			Checksum
	Sync1	*Sync2*	*Desc. Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB* / *LSB*
Command	**0x75**	**0x65**	**0x0D**	**0x02**	**0x02**	**0x23**		**0x0E** / **0x18**
Reply Field: ACK/NACK	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	Echo cmd: **0x23** Error code: **0x00**	**0x03** / **0xF6**
Copy-Paste version of the command: "7565 0D02 0223 0E18"								

4.3.18 Enable/Disable Measurements (0x0D, 0x41)

Description	Allows users to control accelerometer and magnetometer measurement updates.
Notes	Possible function selector values: 0x01 - Use new settings 0x02 - Read back current settings 0x03 - Save current settings as startup settings 0x04 - Load saved startup settings 0x05 - Reset to factory default settings Possible control bitfield values: Bit 0 (0x00000001) - Accelerometer Measurements (1 - enable, 0 - disable)

Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x05	0x41	U8 - Function Selector U16 - Control Bitfield
Reply Field: ACK/ NACK	0x04	0xF1	U8 - Echo the command descriptor U8 - Error code (0: ACK, non-zero: NACK)
Reply Field 2: (function = 2)	0x04	0xB0	U16 - Control Bitfield

Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc. Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command	**0x75**	**0x65**	**0x0D**	**0x05**	**0x05**	**0x41**	**Fctn (Apply): 0x01 X:0x0003 (Enable Accel/Mag measurements)**	**0x36**	**0xE1**
Reply Field: ACK/NACK	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	**Command echo: 0x41 Error code: 0x00**	**0x21**	**0xB2**

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

4.3.19 Gravity Magnitude Error Adaptive Measurement (0x0D, 0x44)

Description	Enable or disable the gravity[1] magnitude error adaptive measurement feature. This function can be used to tune the filter performance in the target application. Possible function selector values: 0x01 - Use new settings 0x02 - Read back current settings. 0x03 - Save current settings as startup settings 0x04 - Load saved startup settings 0x05 - Reset to factory default settings Possible adaptive measurement selector values: 0x00 - No adaptive measurement (disable) 0x01 - Enable fixed adaptive measurement (use specified limits) 0x02 - Enable auto adaptive measurement[2] Filter and limit parameters (only used for enable option 1): Pick values that give you the least occurrence of invalid EF attitude output. The default values are good for standard low vibration applications. Increase values for higher vibration conditions, lower values for lower vibration. Too low a value will result in excessive heading errors. Higher values increase pitch and roll errors when undergoing linear accelerations. Adaptive measurements can be enabled/disabled without the need for providing the additional parameters. In this case, only the function selector and enable value are required; all other parameters will remain at their previous values. When "auto-adaptive" is selected, the filter and limit parameters are ignored. Instead, aiding measurements which rely on the gravity vector will be automatically reweighted by the Kalman filter according to the perceived measurement quality.			
Notes	1. This command is also referred to as "Accelerometer Magnitude Error Adaptive Measurement." 2. Enable option 2 (auto-adaptive) is only available on 3DM-CV5 and later.			
Field Format	*Field Length*	*Field Descriptor*	*Field Data*
Command	0x1C	0x44	U8 - Function Selector U8 - Disable/Fixed/Auto Float - Low-pass filter cutoff frequency (Hz) Float - Low Limit (meters/second^2) Float - High Limit (meters/second^2) Float - Low Limit Uncertainty, 1-Sigma (meters/second^2)

LORD SENSING MicroStrain

			Float - High Limit Uncertainty, 1-Sigma (meters/second^2) Float - Minimum Uncertainty, 1-Sigma (meters/second^2)
Reply Field 1: ACK/ NACK	0x04	0xF1	U8 - Echo the command descriptor U8 - Error code (0: ACK, non-zero: NACK)
Reply Field 2: Function = 2	0x1B	0xB3	U8 - Enable (0 - Disable, 1 - Enable) Float - Low-pass filter cutoff frequency (Hz) Float - Low Limit (meters/second^2) Float - High Limit (meters/second^2) Float - Low Limit Uncertainty, 1-Sigma (meters/second^2) Float - High Limit Uncertainty, 1-Sigma (meters/second^2) Float - Minimum Uncertainty, 1-Sigma (meters/second^2)

Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc. Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command	**0x75**	**0x65**	**0x0D**	**0x1C**	**0x1C**	**0x44**	Fctn (Apply): **0x01** Enable: **0x01** Freq (Hz): **(1.0f)** Low Limit: **(-0.2f)** High Limit: **(0.2f)** Low Limit 1-sigma: **(0.2f)** High Limit 1-sigma: **(0.2f)** Min 1-sigma: **(0.004f)**	**-**	**-**
Reply Field 1: ACK/NACK	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	Echo cmd: **0x44** Error code: **0x00**	**0xB2**	**0xE2**

4.3.20 Set Reference Position (0x0D, 0x26)

Description	Set the Lat/Long/Alt reference position for the sensor. Possible function selector values: 0x01 - Use new settings 0x02 - Read back current settings 0x03 - Save current settings as startup settings 0x04 - Load saved startup settings 0x05 - Reset to factory default settings This position is used by the sensor to calculate the WGS84 gravity and WMM2015 magnetic field parameters.		
Field Format	*Field Length*	*Field*	*Field Data*

		Descriptor	
Command	0x01C (28)	0x26	U8 - Function Selector U8 - Enable (0 - disable, 1 - enable) Double - Latitude (decimal degrees) Double - Longitude (decimal degrees) Double - Altitude (meters)
Reply Field: ACK/ NACK	0x04	0xF1	U8 - Echo the command descriptor U8 - Error code (0: ACK, non-zero: NACK)
Reply Field 2: *(function = 2)*	0x1B (27)	0x90	U8 - Enable (0 - disable, 1 - enable) Double - Latitude (decimal degrees) Double - Longitude (decimal degrees) Double - Altitude (meters)

Example	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc. Set*	*Payload Length*	*Field Length*	*Field Desc.*	*Field Data*	*MSB*	*LSB*
Command	**0x75**	**0x65**	**0x0D**	**0x1C**	**0x1C**	**0x26**	**Fctn (Apply): 0x01** **Enable: 0x01 Latitude (deg): (44.437f)** **Longitude (deg): (-73.106) Altitude (m): (155.0f)**	**0xXX**	**0xXX**
Reply Field: ACK/NACK	**0x75**	**0x65**	**0x0D**	**0x04**	**0x04**	**0xF1**	**Command echo: 0x26 Error code: 0x00**	**0x06**	**0xFC**

4.4 System Commands

The System Command set provides a set of advanced commands that are specific to devices such as the 3DM-CV5-15 that have multiple intelligent internal sensor blocks. These commands allow special modes such as talking directly to the native protocols of the embedded sensor blocks. For example, with the 3DM-CV5-15, you may switch into a mode that talks directly to another LORD Sensing Inertial Sensor with an internal IMU.

4.4.1 Communication Mode (0x7F, 0x10)

Advanced

Description

Advanced specialized communication modes.

This will change the communications protocol to and from “Estimation Filter”mode to “Sensor Direct” (MIP IMU protocol for the 3DM-CV5-15). 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.

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 - Reset to factory default settings

Possible Communications Modes:

Value	*Mode*	*Protocol(s)*
0x01	Standard	3DM-CV5-15 MIP Packet *(default)*
0x02	Sensor Direct	MIP IMU
0x03	GNSS Direct	NMEA, UBX (GNSS Models only)

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, non-zero: 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	**0x75**	**0x65**	**0x7F**	**0x04**	**0x04**	**0x10**	Fctn (USE): **0x01** New mode (IMU direct): **0x02**	**0x74**	**0xBD**
Reply Field 1: 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"									

4.5 Error Codes

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

5. Data Reference

5.1 IMU Data

5.1.1 Scaled Accelerometer Vector (0x80, 0x04)

Description	Scaled Accelerometer Vector					
Notes	This is a vector quantifying the direction and magnitude of the acceleration that the 3DM-CV5-15 is exposed to. This quantity 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-CV5-15's local coordinate system.					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
	14 (0x0E)	0x04	*Binary Off-set*	*Description*	*Data Type*	*Units*
			0	X Accel	float	g
			4	Y Accel	float	g
			8	Z Accel	float	g

LORD SENSING MicroStrain

5.1.2 Scaled Gyro Vector (0x80, 0x05)

Description	Scaled Gyro Vector					
Notes	This is a vector quantifying the rate of rotation (angular rate) of the 3DM-CV5-15. This quantity is fully temperature compensated and scaled into units of radians/second. It is expressed in terms of the 3DM-CV5-15's local coordinate system.					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
	14 (0x0E)	0x05	*Binary Offset*	*Description*	*Data Type*	*Units*
			0	X Gyro	float	Radians/second
			4	Y Gyro	float	Radians/second
			8	Z Gyro	float	Radians/second

5.1.3 Scaled Ambient Pressure (0x80, 0x17)

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

5.1.4 Delta Theta Vector (0x80, 0x07)

Description	Time integral of angular rate.					
Notes	This is a vector which gives the time integral of angular rate over the interval set by the IMU message format command. It is expressed in terms of the 3DM-CV5-15's local coordinate system in units of radians.					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
	14 (0x0E)	0x07	*Binary Offset*	*Description*	*Data Type*	*Units*
			0	X Delta Theta	float	radians
			4	Y Delta Theta	float	radians
			8	Z Delta Theta	float	radians

5.1.5 Delta Velocity Vector (0x80, 0x08)

Description	Time integral of acceleration.					
Notes	This is a vector which gives the time integral of specific acceleration over the interval set by the IMU message format command. It is expressed in terms of the 3DM-CV5-15's local coordinate system in units of g*second where g is the standard gravitational constant. To convert Delta Velocity into the more conventional units of m/sec, simply multiply by the standard gravitational constant, 9.80665 m/sec^2.					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
			Binary Offset	*Description*	*Data Type*	*Units*
	14 (0x0E)	0x08	0	X Delta Velocity	float	g*seconds
			4	Y Delta Velocity	float	g*seconds
			8	Z Delta Velocity	float	g*seconds

5.1.6 CF Orientation Matrix (0x80, 0x09)

Description

3 x 3 Orientation Matrix *M*.

This value is produced by the Complementary Filter fusion algorithm.

Notes

This is a nine component coordinate transformation matrix which describes the orientation of the 3DM-CV5 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-CV5's local coordinate system.

5.1.6 CF Orientation Matrix (0x80, 0x09)

	V_E is the same vector expressed in the stationary, earth-fixed coordinate system					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
	38 (0x26)	0x09	*Binary Offset*	*Description*	*Data Type*	*Units*
			0	$M_{1,1}$	Float	N/A
			4	$M_{1,2}$	Float	N/A
			8	$M_{1,3}$	Float	N/A
			12	$M_{2,1}$	Float	N/A
			16	$M_{2,2}$	Float	N/A
			20	$M_{2,3}$	Float	N/A
			24	$M_{3,1}$	Float	N/A
			28	$M_{3,2}$	Float	N/A
			32	$M_{3,3}$	Float	N/A

5.1.7 CF Quaternion (0x80, 0x0A)

Description

4 x 1 quaternion Q.

This value is produced by the Complementary Filter fusion algorithm.

Notes

This is a four component quaternion which describes the orientation of the 3DM-CV5 with respect to the fixed earth coordinate system.

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

Q satisfies the following equation:

$$V\_IL_i = Q^{-1} \cdot V\_E \cdot Q$$

Where:

V_IL is a vector expressed in the 3DM-CV5's local coordinate system.

V_E is the same vector expressed in the stationary, earth-fixed coordinate system

Field Format

Field Length	*Data Descriptor*	*Message Data*			
		Binary Offset	*Description*	*Data Type*	*Units*
18 (0x12)	0x0A	0	q_0	Float	N/A
		4	q_1	Float	N/A
		8	q_2	Float	N/A
		12	q_3	Float	N/A

5.1.8 CF Euler Angles (0x80, 0x0C)

<table>
<tr><td>Description</td><td colspan="6">Pitch, Roll, and Yaw (aircraft) values.
This value is produced by the Complementary Filter fusion algorithm.</td></tr>
<tr><td>Notes</td><td colspan="6">This is a three component vector containing the Roll, Pitch and Yaw angles in radians. It is computed by the IMU/AHRS from the orientation matrix M.
$$Euler = \begin{bmatrix} Roll \\ Pitch \\ Yaw \end{bmatrix}$$</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>

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

Description	Gyro stabilized estimated vector for geomagnetic vector. *This value is produced by the Complementary Filter fusion algorithm.*					
Notes	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 IMU/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.					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
			Binary Offset	*Description*	*Data Type*	*Units*
	14 (0x0E)	0x10	0	X Stab Mag	Float	Gauss
			4	Y Stab Mag	Float	Gauss
			8	Z Stab Mag	Float	Gauss

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

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

5.1.10 CF Stabilized Up Vector (0x80, 0x11)						
			4	Y Stab Accel	Float	G
			8	Z Stab Accel	Float	G

5.1.11 GPS Correlation Timestamp (0x80, 0x12)

Description	GPS correlation timestamp.					
Notes	This timestamp has three fields: Double GPS TOW U16 GPS Week number U16 Timestamp flags Timestamp Status Flags: Bit0 - PPS Beacon Good If set, PPS signal is present Bit1 - GPS Time Refresh (toggles with each refresh) Bit2 - GPS Time Initialized (set with the first GPS Time Refresh) (*See GPS Time Update (0x01, 0x72) on page 39*) This timestamp correlates the IMU packets with the GPS packets. It is identical to the GPS Time record except the flags are defined specifically for the IMU. When the GPS Time Initialized flag is asserted, the GPS Time and IMU GPS Timestamp are correlated. This flag is only set once upon the first valid GPS Time record. After that, each time the GPS Time becomes invalid (from a lack of signal) and then valid again (regains signal) the GPS Time Refresh flag will toggle. The GPS Time Initialized will remain set. The "PPS Beacon Good" flag in the Timestamp flags byte indicates if the PPS beacon coming from the GPS is present. If this flag is not asserted, it means that the IMU internal clock is being used for the PPS. The fractional portion of the GPS TOW represents the amount of time that has elapsed from the last PPS.					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
			Binary Offset	*Description*	*Data Type*	*Units*
	14 (0x0E)	0x12	0	GPS Time of Week	Double	Seconds
			8	GPS Week Number	U16	N/A
			10	Timestamp Flags	U16	See Notes

5.2 Estimation Filter Data

5.2.1 Filter Status (0x82, 0x10)	
Description	Estimation Filter Status
Notes	Possible Filter States: 0x00 - Startup 0x01 - Initialization (see status flags) 0x02 - Running, Solution Valid 0x03 - Running, Solution Error (see status flags) Possible Dynamics Modes: 0x01 - Portable 0x02 - Automotive 0x03 - Airborne Possible Status Flags: Filter State = Initialization: 0x1000 - Attitude not initialized 0x2000 - Position & Velocity not initialized Filter State = Running: 0x0001 - IMU unavailable 0x0002 - GNSS (GNSS versions only) 0x0008 - Matrix singularity in calculation 0x0010 - Position covariance high warning* 0x0020 - Velocity covariance high warning* 0x0040 - Attitude covariance high warning* 0x0080 - NAN in solution 0x0100 - Gyro bias estimate high warning 0x0200 - Accel bias estimate high warning 0x0400 - Gyro scale factor estimate high warning 0x0800 - Accel scale factor estimate high warning 0x1000 - Mag bias estimate high warning 0x4000 - Hard Iron offset estimate high warning 0x8000 - Soft iron correction estimate high warning **Note: The covariance high warnings are triggered when any axis of the covariance vector exceeds normal operating limits. If more information is required, please*

5.2.1 Filter Status (0x82, 0x10)

	inspect the relevant uncertainty packet to determine which axis is in error.					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
	08 (0x08)	0x10	*Binary Offset*	*Description*	*Data Type*	*Units*
			0	Filter State	U16	See Notes
			2	Dynamics Mode	U16	See Notes
			4	Status Flags	U16	See Notes

5.2.2 GPS Timestamp (0x82, 0x11)

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

5.2.3 Orientation, Quaternion (0x82, 0x03)

Description	Estimated Orientation in quaternion form.					
Notes	This is a four component quaternion which describes the orientation of the 3DM-CV5 with respect to the fixed earth coordinate system. $$Q = \begin{bmatrix} q0 \\ q1 \\ q2 \\ q3 \end{bmatrix}$$ Q satisfies the following equation: $$V\_IL_i = Q \cdot V\_E \cdot Q^{-1}$$ Where: **V_IL** *is a vector expressed in the 3DM-CV5's local coordinate system.* **V_E** *is the same vector expressed in the stationary, earth-fixed coordinate system* Valid Flag Mapping: 0x0000 - Quaternion is Invalid 0x0001 - Quaternion Valid					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
	20 (0x14)	0x03	*Binary Offset*	*Description*	*Data Type*	*Units*
			0	q_0	Float	N/A
			4	q_1*i	Float	N/A
			8	q_2*j	Float	N/A
			12	q_3*k	Float	N/A
			16	Valid Flags	U16	See Notes

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

Description	Estimated attitude 1-sigma uncertainty expressed in quaternion components.					
Notes	This is a three component vector containing the attitude uncertainty expressed in quaternion elements. Valid Flag Mapping: 0x0000 - Attitude uncertainties are Invalid 0x0001 - Attitude uncertainties are Valid					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
	20 (0x14)	0x12	*Binary Offset*	*Description*	*Data Type*	*Units*
			0	1-Sigma Attitude Uncertainty (q_0)	Float	
			4	1-Sigma Attitude Uncertainty (q_1)	Float	
			8	1-Sigma Attitude Uncertainty (q_2)	Float	
			12	1-Sigma Attitude Uncertainty (q_3)	Float	
			16	Valid Flags	U16	See Notes

<table>
<tr><th colspan="7">5.2.5 Orientation, Euler Angles (0x82, 0x05)</th></tr>
<tr><td>Description</td><td colspan="6">Estimated Pitch, Roll, and Yaw (aircraft) values.</td></tr>
<tr><td>Notes</td><td colspan="6">This is a three component vector containing the Roll, Pitch and Yaw angles in radians. It is computed by the INS from the orientation quaternion Q.
$$Euler = \begin{bmatrix} Roll \\ Pitch \\ Yaw \end{bmatrix}$$
Valid Flag Mapping:
0x0000 - Euler Angles are Invalid
0x0001 - Euler Angles Valid</td></tr>
<tr><td rowspan="6">Field Format</td><td>Field Length</td><td>Data Descriptor</td><td colspan="4">Message Data</td></tr>
<tr><td rowspan="5">16 (0x10)</td><td rowspan="5">0x05</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>
<tr><td>12</td><td>Valid Flags</td><td>U16</td><td>See Notes</td></tr>
</table>

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

<table>
<tr><td>Description</td><td colspan="6">Estimated attitude 1-sigma uncertainty expressed in Pitch, Roll, and Yaw (aircraft) elements.</td></tr>
<tr><td>Notes</td><td colspan="6">This is a three component vector containing the Roll, Pitch and Yaw angle uncertainties in radians.

IMPORTANT: These values are derived from the quaternion elements and become increasingly inaccurate as the pitch angle approaches +-90 degrees. To compensate for this limitation, these values will be marked as invalid when the pitch angle exceeds +-70 degrees.

Valid Flag Mapping:

0x0000 - Attitude Uncertainties are Invalid
0x0001 - Attitude Uncertainties Valid</td></tr>
<tr><td rowspan="6">Field Format</td><td>Field Length</td><td>Data Descriptor</td><td colspan="4">Message Data</td></tr>
<tr><td rowspan="5">16 (0x10)</td><td rowspan="5">0x0A</td><td>Binary Offset</td><td>Description</td><td>Data Type</td><td>Units</td></tr>
<tr><td>0</td><td>1-Sigma Attitude Uncertainty (Roll)</td><td>Float</td><td>Radians</td></tr>
<tr><td>4</td><td>1-Sigma Attitude Uncertainty (Pitch)</td><td>Float</td><td>Radians</td></tr>
<tr><td>8</td><td>1-Sigma Attitude Uncertainty (Yaw)</td><td>Float</td><td>Radians</td></tr>
<tr><td>12</td><td>Valid Flags</td><td>U16</td><td>See Notes</td></tr>
</table>

5.2.7 Orientation, Matrix (0x82, 0x04)

Description	Estimated orientation in matrix form.					
Notes	This is a nine component coordinate transformation matrix which describes the orientation of the 3DM-CV5 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-CV5's local coordinate system.* **V_E** *is the same vector expressed in the stationary, earth-fixed coordinate system* Valid Flag Mapping: 0x0000 - Orientation Matrix is Invalid 0x0001 - Orientation Matrix Valid					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
			Binary Offset	*Description*	*Data Type*	*Units*
	40 (0x28)	0x04	0	$M_{1,1}$	Float	N/A
			4	$M_{1,2}$	Float	N/A
			8	$M_{1,3}$	Float	N/A
			12	$M_{2,1}$	Float	N/A
			16	$M_{2,2}$	Float	N/A
			20	$M_{2,3}$	Float	N/A
			24	$M_{3,1}$	Float	N/A

5.2.7 Orientation, Matrix (0x82, 0x04)

			28	$M_{3,2}$	Float	N/A
			32	$M_{3,3}$	Float	N/A
			36	Valid Flags	U16	See Notes

5.2.8 Compensated Angular Rate (0x82, 0x0E)

Description	Filter-Compensated Angular Rate Data expressed in: 1. The Sensor Frame, if no sensor to body rotation has been defined. 2. The Vehicle Frame, if a sensor to body rotation has been defined.					
Notes	The estimated gyro bias has been removed from these angular rate values. Valid Flag Mapping: 0x0000 - Angular Rates are not Valid 0x0001 - Angular Rates are Valid					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
	16 (0x10)	0x0E	*Binary Offset*	*Description*	*Data Type*	*Units*
			0	X	Float	Radians/Sec
			4	Y	Float	Radians/Sec
			8	Z	Float	Radians/Sec
			12	Valid Flags	U16	See Notes

LORD SENSING MicroStrain

5.2.9 Gyro Bias (0x82, 0x06)

Description	Estimated Gyro Biases expressed in the Sensor Body Frame.					
Notes	Valid Flag Mapping: 0x0000 - Gyro Bias are Invalid 0x0001 - Gyro Bias Valid					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
	16 (0x10)	0x06	*Binary Offset*	*Description*	*Data Type*	*Units*
			0	X Gyro Bias	Float	Radians/Sec
			4	Y Gyro Bias	Float	Radians/Sec
			8	Z Gyro Bias	Float	Radians/Sec
			12	Valid Flags	U16	See Notes

5.2.10 Gyro Bias Uncertainty (0x82, 0x0B)

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

5.2.11 Compensated Acceleration (0x82, 0x1C)

Description	Filter-Compensated Acceleration Data expressed in: 1. The Sensor Frame, if no sensor to body rotation has been defined. 2. The Vehicle Frame, if a sensor to body rotation has been defined.					
Notes	Valid Flag Mapping: 0x0000 - Compensated Accelerations are Invalid 0x0001 - Compensated Accelerations are Valid					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
	16 (0x10)	0x1C	*Binary Offset*	*Description*	*Data Type*	*Units*
			0	X	Float	Meters / Sec^2
			4	Y	Float	Meters / Sec^2
			8	Z	Float	Meters / Sec^2
			12	Valid Flags	U16	See Notes

5.2.12 Linear Acceleration (0x82, 0x0D)

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

5.2.13 Pressure Altitude (0x82, 0x21)

Description	Estimated Pressure Altitude.					
Notes	The US 1976 Standard Atmosphere Model is used to calculate the pressure altitude in meters. A valid pressure sensor reading is required for the pressure altitude to be valid. The minimum pressure reading supported by the model is 0.0037 mBar, corresponding to an altitude of 84,852 meters. Valid Flag Mapping: 0x0000 - Pressure Altitude is Invalid 0x0001 - Pressure Altitude is Valid					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
	8 (0x08)	0x21	*Binary Offset*	*Description*	*Data Type*	*Units*
			0	Pressure Altitude	Float	Meters
			4	Valid Flags	U16	See Notes

5.2.14 Gravity Vector (0x82, 0x13)

Description	Estimated Gravity Vector expressed in: 1. The Sensor Frame, if no sensor to body rotation has been defined. 2. The Vehicle Frame, if a sensor to body rotation has been defined.					
Notes	Valid Flag Mapping: 0x0000 - Gravity vector is Invalid 0x0001 - Gravity vector is Valid					
Field Format	*Field Length*	*Data Descriptor*	*Message Data*			
	16 (0x10)	0x13	*Binary Offset*	*Description*	*Data Type*	*Units*
			0	X	Float	Meters / Sec^2
			4	Y	Float	Meters / Sec^2

5.2.14 Gravity Vector (0x82, 0x13)						
			8	Z	Float	Meters / Sec^2
			12	Valid Flags	U16	See Notes

<table>
<tr><th colspan="7">5.2.15 WGS84 Local Gravity Magnitude (0x82, 0x0F)</th></tr>
<tr><td>Description</td><td colspan="6">Local Magnitude of Earth's gravity using the WGS84 gravity model.</td></tr>
<tr><td>Notes</td><td colspan="6">The -CV5-15 implements the WGS84 gravity model, valid for altitudes of 20 km or less.

Valid Flag Mapping:

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

5.2.16 Heading Update Source State (0x82, 0x14)

Description	Heading Update Source information expressed in the sensor frame.
Notes	Heading updates can be applied from a number of sources (listed below. Also see Heading Update Control.) The heading value is always relative to true north. Possible Source Flags (may be combined): 0x0000 - No source, heading updates disabled 0x0004 - External Heading Update or External Heading Update with Timestamp Message Valid Flag Mapping: 0x0000 - No heading update received in 2 seconds. 0x0001 - The heading update source has provided data within 2 seconds.

Field Format

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

6. MIP Packet Reference

6.1 Structure

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

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.

6.2 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.

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

6.3 MIP Checksum Range

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

6.4 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;
```

7. Advanced Programming

7.1 Multiple Commands in a Single Packet

MIP packets may contain one or more individual commands. In the case that multiple commands are transmitted in a single MIP packet, the 3DM-CV5-15 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):

Examples	MIP Packet Header				Command/Reply Fields			Checksum	
	Sync1	*Sync2*	*Desc. Set*	*Payload Length*	*Field Length*	*Cmd Desc.*	*Field Data*	*MSB*	*LSB*
Command Field 1: Set IMU Message Format	0x75	0x65	0x0C	0x20	0x0D	0x08	Function: **0x01** Desc. count: **0x03** GPS TS Descriptor: **0x12** Rate Dec: **0x000A** Accel Descriptor: **0x04** Rate Dec: **0x000A** Ang Rate Descriptor: **0x05** Rate Dec: **0x000A**		
Command Field 2: Set EF Message Format					0x13	0x0A	Function: **0x01** Desc. count: **0x05** GPS TS Desc.: **0x11** Rate Dec: **0x000A** Filter Status Desc: **0x10** Rate Dec: **0x000A** Est. Pos. Desc.: **0x01** Rate Dec: **0x000A** Est. Vel. Desc.: **0x02** Rate Dec: **0x000A** Est. Quat. Desc: **0x03** Rate Dec: **0x000A**	0xD4	0x3D
Reply Field 1: ACK/NACK	0x75	0x65	0x0C	0x08	0x04	0xF1	Echo cmd: **0x08** Error code: **0x00**		
Reply Field 2: ACK/NACK					0x04	0xF1	Echo cmd: **0x0A** Error code: **0x00**	0xEA	0x71
Copy-paste version of the command: "7565 0C20 0D08 0103 1200 0A04 000A 0500 0A13 0A01 0511 000A 1000 0A01 000A 0200 0A03 000A D43D"									

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.

7.2 Direct Modes

The 3DM-CV5-15 has special "direct" modes that switch the device into a Sensor direct device. The Device Communications Mode command is used to switch between modes. When in these modes, the 3DM-CV5-15 acts like an "IMU only" sensor. Any code or tools developed for these devices may be used in these modes.

These modes can be used to access advanced (native) data of the individual sensors, data that isn't represented in the 3DM command sets of the 3DM-CV5-15. These modes are primarily advanced modes for programmers to allow the 3DM-CV5-15 to be used in unusual situations where the normal functions of the 3DM-CV5-15 are bypassed.

IMPORTANT: *When you switch modes, you are switching to a new device protocol EXCEPT for two commands: the Device Communications Mode and commands. Those commands are always available regardless of which mode you are in. For example, if you switch to direct mode, then the protocol recognized by the device is protocol, however the 3DM-CV5-15 is still "listening" for mode switch or device status commands and will respond to them. It will not respond to any other 3DM-CV5-15 Base or 3DM commands until switched back to the "Standard Mode".*

7.3 Internal Diagnostic Functions

The 3DM-CV5-15 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 LORD Sensing devices will have their own implementations of these functions depending on the internal hardware of the devices.

7.3.1 3DM-CV5-15 Internal Diagnostic Commands

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

7.4 Handling High Rate Data

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

7.4.1 Runaway Latency

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

7.4.2 Dropped Packets

Many applications do not use an operating system but are written from scratch or on top of proprietary application frameworks. These are most often embedded MCUs or small single board microcontrollers. On these systems, port handling is usually done in code at the hardware level. Collecting data from a port requires the use one of three techniques: register polling, hardware interrupts, or direct memory access (DMA). Register polling is very easy to do and is adequate for simple communications where data comes in very small chunks and at reasonable data rates. The problem with register polling is that you either waste time looping while waiting for a byte to come in at the port or you get too busy doing other tasks so that by the time you poll the port, the byte is lost because the next one overwrites it. This causes dropped packets. On these systems, it is imperative to utilize either a hardware interrupt or hardware DMA on the UART receiving data from the 3DM-CV5-15. The DMA or UART interrupt service routine only takes processor time when a byte is ready and as long as the interrupts are preemptive, the processor will fetch every byte received. Using the interrupt routine to fill a ring buffer makes the most efficient use of an MCU and makes it easier to write your application main line code. This is essentially what drivers in operating systems do.

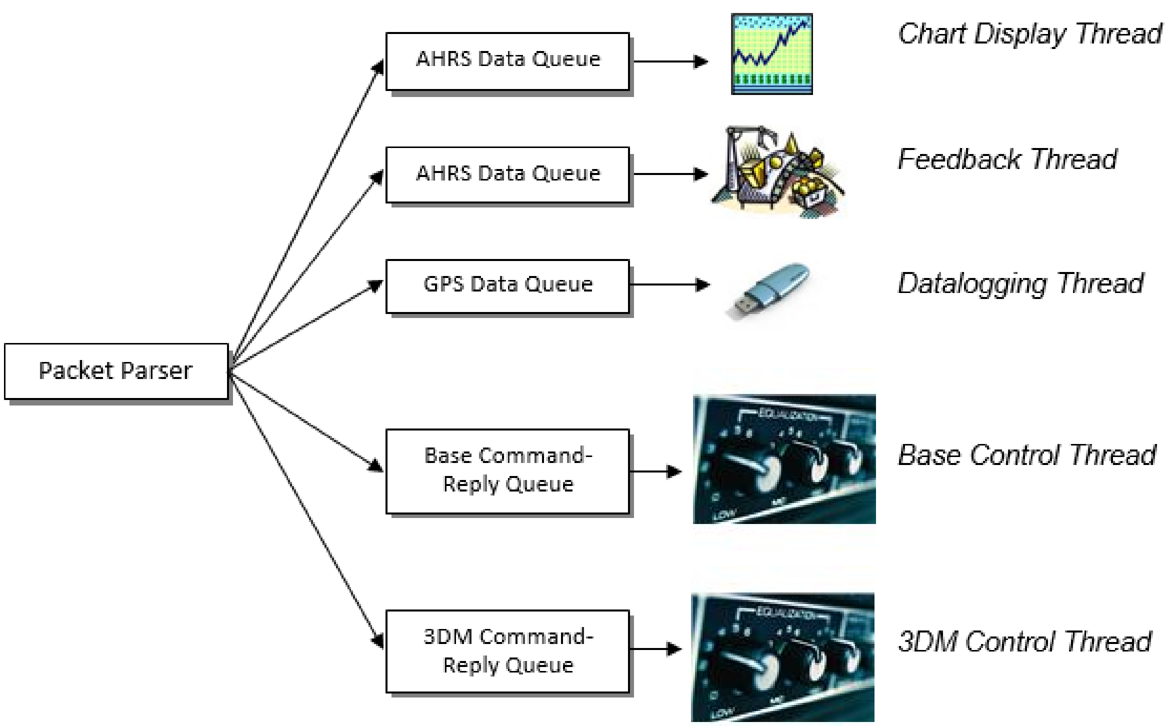
7.5 Creating Fixed Data Packet Format

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

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

7.6 Advanced Programming Models

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



Multithreaded application with multiple incoming packet queues

8. Glossary

A

A/D Value
The digital representation of analog voltages in an analog-to-digital (A/D) conversion. The accuracy of the conversion is dependent on the resolution of the system electronics. Higher resolution produces a more accurate conversion.

Acceleration
In physics,acceleration is the change in the rate of speed (velocity) of an object over time.

Accelerometer
A sensor used to detect and measure magnitute and direction of an acceleration force (g-force) in reference to its sensing frame. For example, at rest perpendicular to the Earth's surface an accelerometer will measure 9.8 meters/second squared as a result of gravity. If the device is tilted the acceleration force will change slightly, indicating tilt of the device. When the accelerometer is moving it will measure the dynamic force (including gravity).

Adaptive Kalman Filter (AKF)
A type of Extended Kalman Filter (EKF) that contains an optimization algorithm that adapts to dynamic conditions with a high dependency on adaptive technology. Adaptive technology refers to the ability of a filter to selectively trust a given measurement more or less based on a trust threshold when compared to another measurement that is used as a reference. Sensors that have estimation filters that rely on adaptive control elements to improve their estimations are refered to as an AKF.

AHRS (Attitude and Heading Reference System)
A navigation device consisting of sensors on the three primary axes used to measure vehicle direction and orientation in space. The sensor measurements are typically processed by an onboard algorthim, such as an Estimation Filter, to produce a standardized output of attitude and heading.

Algorithm
In math and science, an algorithm is a step-by-step process used for calculations.

Altitude
the distance an object is above the sea level

Angular rate
The rate of speed of which an object is rotating. Also know as angular frequency, angular speed, or radial frequency. It is typically measured in radians/second.

API (Applications Programming Interface)
A library and/or template for a computer program that specifies how components will work together to form a user application: for example, how hardware will be accessed and what data structures and variables will be used.

ASTM (Association of Standards and Testing)
a nationally accepted organization for the testing and calibration of technological devices

Attitude
the orientaion of an object in space with reference to a defined frame, such as the North-East-Down (NED) frame

Azimuth
A horizontal arc measured between a fixed point (such as true north) and the vertical circle passing through the center of an object

B

Bias
A non-zero output signal of a sensor when no load is applied to it, typically due to sensor imperfections. It is also called offset.

C

Calibration
to standardize a measurement by determining the deviation standard and applying a correction, or calibration, factor

Complementary Filter (CF)
A term commonly used for an algorithm that combines the readings from multiple sensors to produce a solution. These filters typically contain simple filtering elements to smooth out the effects of sensor over-ranging or anomalies in the magnetic field.

Configuration
A general term applied to the sensor indicating how it is set up for data acquisition. It includes settings such as sampling rate, active measurements, measurement settings, offsets, biases, and calibration values

Convergance
when mathematical computations approach a limit or a solution that is stable and optimal.

D

Data Acquisition
the process of collecting data from sensors and other devices

Data Logging
the process of saving acquired data to the system memory, either locally on the device, or remotely on the host computer

Data rate
the rate at which sampled data is transmitted to the host

Delta-Theta
the time integral of angular rate expressed with refernce to the device local coordinate system, in units of radians

Delta-velocity
the time integral of velocity expressed with refernce to the device local coordinate system, in units of g*second where g is the standard gravitational constant

E

ECEF (Earth Centered Earth Fixed)
a reference frame that is fixed to the earth at the center of the earth and turning about earth's axis in the same way as the earth

Estimation Filter
A mathematical algorithm that produces a statistically optimum solution using measurements and references from multiple sources. Best known estimation filters are the Kalman Filter, Adaptive Kalman Filter, and Extended Kalman Filter.

Euler angles
Euler angles are three angles use to describe the orientation of an object in space such as the x, y and z or pitch; roll; and yaw. Euler angles can also represent a sequence of three elemental rotations around the axes of a coordinate system.

Extended Kalman Filter (EKF)
Used generically to describe any estimation filter based on the Kalman Filter model that can handle non-linear elements. Almost all inertial estimation filters are fundamentally EKFs.

G

GNSS (Global Navigation Statellite System)
a global network of space based statellites (GPS, GLONASS, BeiDou, Galileo, and others) used to triangulate position co-ordinates and provide time information for navigational purposes

GPS (Global Positioning System)
a U.S. based network of space based statellites used to triangulate position co-ordinates and provide time information for navigational purposes

Gyroscope
a device used to sense angular movements such as rotation

H

Heading
an object's direction of travel with reference to a co-ordinate frame, such as lattitude and longitude

Host (computer)
The host computer is the computer that orchestrates command and control of attached devices or networks.

I

IMU
Inertial Measurement System

Inclinometer
device used to measure tilt, or tilt and roll

Inertial
pertaining to systems that have inertia or are used to measure changes in inertia as in angular or linear accelerations

INS (Inertial Navigation System)
systems that use inertial measurements exclusively to determine position, velocity, and attitude, given an initial reference

K

Kalman Filter
a linear quadratic estimation algorithm that processes sensor data or other input data over time, factoring in underlying noise profiles by linearizing the current mean and covariance to produces an estimate of a system’s current state that is statistically more precise than what a single measurement could produce

L

LOS (Line of Sight)
Describes the ideal condition between transmitting and receiving devices in a wireless network. As stated, it means they are in view of each other with no obstructions.

M

Magnetometer
A type of sensor that measures the strength and direction of the local magnetic field with refernce to the sensor frame. The magnetic field measured will be a combination of the earth's magnetic field and any magnetic field created by nearby objects.

MEMS (Micro-Electro-Mechanical System)
The technology of miniaturized devices typically made using micro fabrication techniques such as nanotechnology. The devices range in size from one micron to several millimeters and may include very complex electromechanical parts.

N

NED (North-East-Down)
A geographic reference system

O

OEM
acronym for Original Equipment Manufacturer

Offset
A non-zero output signal of a sensor when no load is applied to it, typically due to sensor imperfections. Also called bias.

Orientation
The orientaion of an object in space with reference to a defined frame. Also called attitude.

P

Pitch
In navigation pitch is what occurs when vertical force is applied at a distance forward or aft from the center of gravity of the platform, causing it to move up or down with respect to the sensor or platform frame origin.

Position
The spatial location of an object

PVA
acronym for Position, Velocity, Attitude

Q

Quaternion
Mathematical notation for representing orientation and rotation of objects in three dimensions with respect to the fixed earth coordinate quaternion. Quaternions convert the axis-angle representation of the object into four numbers and to apply the corresponding rotation to a position vector representing a point relative to the origin.

R

Resolution
In digital systems, the resolution is the number of bits or values available to represent analog voltages or information. For example, a 12-bit system has 4096 bits of resolution and a 16-bit system has 65536 bits.

RMS
acronym for Root Mean Squared

Roll
In navigation roll is what occurs when a horizontal force is applied at a distance right or left from the center of gravity of the platform, causing it to move side to side with respect to the sensor or platform frame origin.

RPY
acronym for Roll, Pitch, Yaw

RS232
a serial data communications protocol

RS422
a serial data communications protocol

S

Sampling
the process of taking measurements from a sensor or device

Sampling rate
rate at which the sensors are sampled

Sampling Rate
the frequency of sampling

Sensor
a device that physically or chemically reacts to environmental forces and conditions and produces a predictable electrical signal as a result

Sigma
In statistics, sigma is the standard deviation from the mean of a data set.

Space Vehicle Information
refers to GPS satellites

Streaming
typically when a device is sending data at a specified data rate continuously without requiring a prompt from the host

U

USB (Universal Serial Bus)
A serial data communications protocol

UTC (Coordinated Universal Time)
The primary time standard for world clocks and time. It is similar to Greenwich Mean Time (GMT).

V

Vector
a measurement with direction and magnitude with refernce from one point in space to another

Velocity
The rate of change of position with respect to time. Also called speed.

W

WAAS (Wide Area Augmentation System)
An air navigation aid developed to allow aircraft to rely on GPS for all phases of flight, including precision approaches to any airport.

WGS (World Geodetic System)
a protocol for geo-referencing such as WGS-84

Y

Yaw
In navigation yaw is what occurs when rotational force is applied at a distance forward or aft from the center of gravity of the platform, causing it to move around the center axis of a sensor or platform frame origin.