MicroStrain 3DM-CV7-GNSS/INS Documentation Center

[Getting Started](#)

[MIP Protocol](#) / [MIP API](#)

[Specifications](#)

Product Description

The MicroStrain 3DM-CV7-GNSS/INS is a tactical-grade Inertial Navigation System (INS) in a miniature package. The flexible EKF can accept external aiding measurements to maintain a robust navigation solution during GNSS outages and touts a feature set designed specifically for the timing requirements of the robotics and autonomous vehicle industries.

Key features

GNSS

- High-rate GNSS input for improved low speed position accuracy
- Dual antenna heading for accurate heading measurements immune to magnetic disturbances
- RTK support for centimeter-level positioning accuracy

- NMEA input over GPIO simplifies external GNSS input using a single GPIO line

EKF

- INS with a flexible, adaptive EKF designed for demanding navigation applications
- 5 Reference Frame configurations for simplified external aiding sensor input
- External aiding input support for a wheel speed sensor, external GNSS, and more
- Multiple external aiding measurements accepted for improved GNSS-denied performance

Sensing

- User-adjustable sensor ranges (gyro and accelerometer)
- Fully temperature-compensated, tactical-grade IMU, mathematically-aligned to an orthogonal coordinate system

Features

- PPS input/output and external clock synchronization for precision time alignment with external devices
- Event triggering system and low latency data

Internal sensors

- 3-axis accelerometer
- 3-axis gyroscope
- 3-axis magnetometer
- Pressure altimeter
- Dual RTK-capable GNSS receivers

Supported part numbers

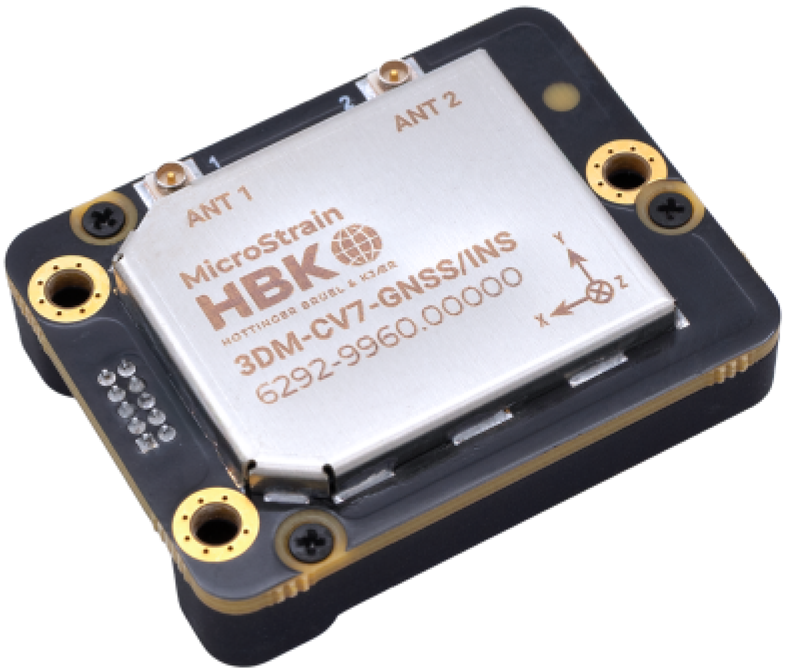
Name	Part Number	Description
3DM-CV7-GNSS/INS	6292-9960	GNSS-Aided Inertial Navigation System

support.microstrain.com

MicroStrainSupport@HBKWorld.com

+1-802-862-6629

8500-0099 Rev -



Getting Started

The 3DM-CV7 is a flexible, tactical-grade INS with an extensive feature set. The following list provides a simplified overview of the critical features that are required to properly set up and use the 3DM-CV7. Use this sequence as a guide to configure the 3DM-CV7. Not all steps may be necessary for your application.

Overview

(Each step is explained in more detail below.)

1. Install desired software
2. Install the 3DM-CV7
3. Capture the gyro bias
4. Select the aiding measurements

5. Configure the output data stream
6. Initialize the Navigation Filter
7. Monitor the Navigation Filter status

Install Desired software

There are multiple software offerings for interfacing with the 3DM-CV7, tailored to various applications and levels of experience with inertial products. These offerings are organized below by increasingly lower levels of device control.

SensorConnect™ is not required to use the 3DM-CV7. However, it provides an intuitive GUI that greatly simplifies the initial configuration and evaluation process. The remaining steps in this section can be easily performed in **SensorConnect™**. See **Software** for more details about **SensorConnect™**.

ROS drivers provide a standardized framework to easily interface with the 3DM-CV7 . ROS is recommended for applications where ease of integration is prioritized above optimum time synchronization. Examples include waypoint navigation, obstacle avoidance, sensor fusion for SLAM applications, simple embedded startup, etc. See **Software** for more details about ROS.

PX4 is an open source flight control software ecosystem designed to standardize and simplify the integration of systems into UAS and other unmanned vehicles. PX4 support for the 3DM-CV7 is currently in development. See **Software** for more details about PX4.

MIP SDK is a C and C++ SDK for interfacing with MicroStrain Inertial products at a low level and is the recommend software product for embedded, resource constrained, and time-critical applications. See **Software** for more details about MIP SDK.

Install the 3DM-CV7

Prior to installation, it is important that users read the instructions on **Powering** the 3DM-CV7. In addition, it is critical to correctly configure the mounting transform , navigation coordinate frame , and aiding sensor frames and measurements. **Installation** describes this process in more detail.

Capture the Gyro Bias

Capturing and removing gyro bias is one of the simplest ways to increase sensor performance. Gyro bias can shift during shipping and installation and should be captured and stored in non-volatile memory after sensor mounting, while the vehicle is stationary (including all motors and moving parts.) Additionally, it may need to be captured and stored periodically throughout the lifetime of the application. Once captured and stored it is automatically applied to the gyro data. See the **Capture Gyro Bias (0x0C,0x39)** command for details. To ensure this command is persistent through power cycling i.e. stored in non-volatile memory (NVM), use the **Device Settings (0x0C,0x30)** command to save to NVM.

Instructions for how to perform a gyro bias capture using **SensorConnect™** can be found here.

Select the Aiding Measurements

The 3DM-CV7's Navigation Filter can accept a variety of aiding measurements to improve its solution.

Example internal aiding measurements include GNSS position and velocity, dual antenna heading, barometer, and magnetometer.

Example external aiding measurements include GNSS position and velocity, true heading , wheel odometer, and x-axis vehicle velocity, body-frame velocity, barometer, and magnetometer.

See **Filter Aiding Measurements** for more details.

Configure the Output Data Stream

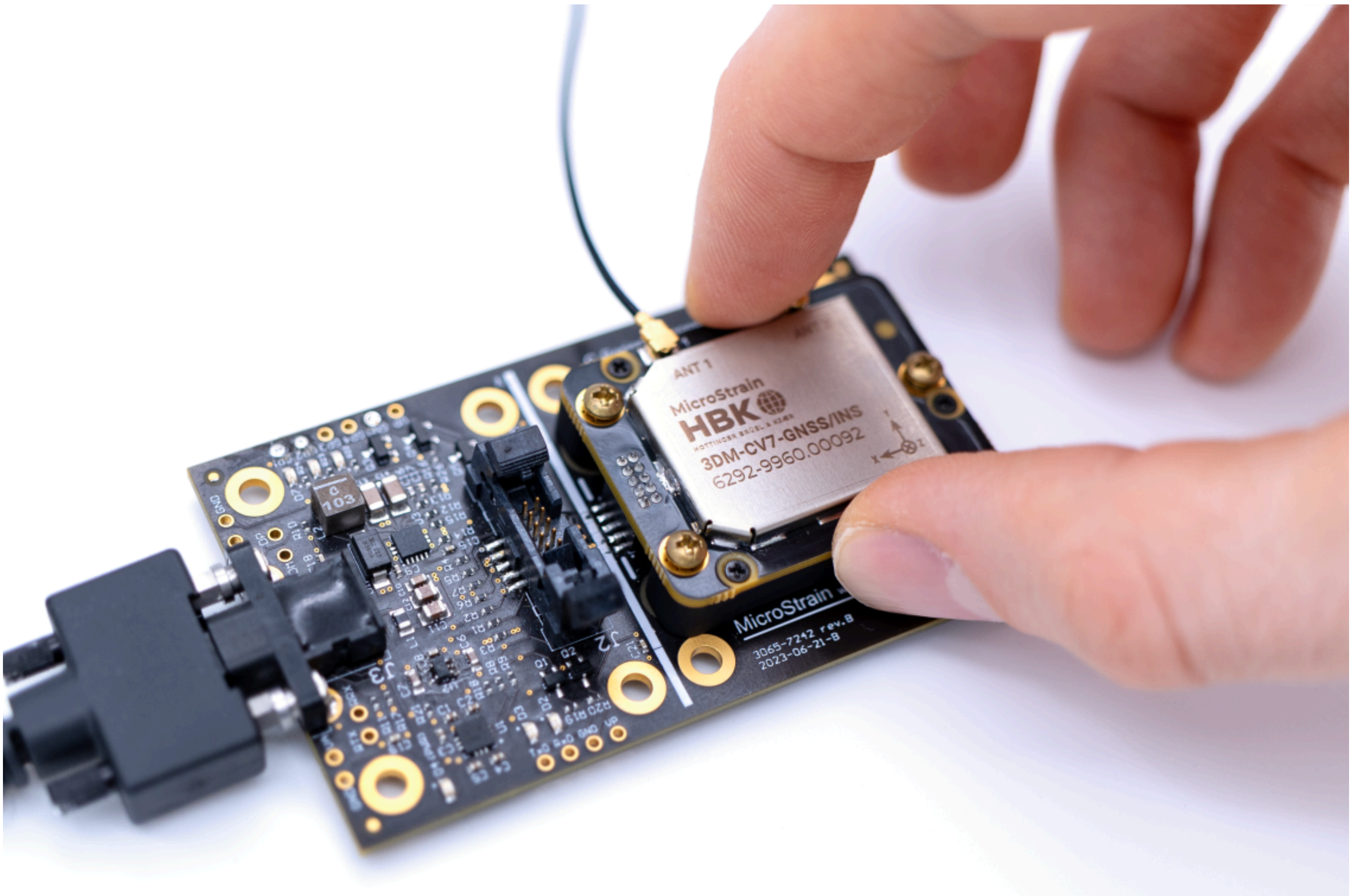
Configure the 3DM-CV7 to output the data quantities required for your application. See **Data** for a complete list of supported data quantities. This is easily completed through **SensorConnect** in the "Sampling" tile.

Initialize the Navigation Filter

To run in Full Navigation mode, the Navigation Filter requires an initial estimate of attitude, position and velocity. The Navigation Filter can be configured to automatically retrieve an initial attitude estimate from the 3DM-CV7's internal sensors, or they can be provided manually by the user. The initial position and velocity estimates can be provided by the 3DM-CV7-GNSS/INS's onboard GNSS receivers, through external aiding measurements, or manually configured by the user. The Navigation Filter can also be run in other modes that require fewer initial estimates, but these modes do not provide a full navigation solution. See **Filter Initialization** for more information.

Monitor the Navigation Filter Status

The status of the Navigation Filter can be monitored through the **Status (0x82,0x10)** data channel. For a full description of this message and the Navigation Filter modes, see **Filter Status**. Health checks can be performed by monitoring and comparing the Filter Uncertainty Messages with the application specific thresholds. For example, position can be monitored with **LLH Position Uncertainty (0x82,0x08)**, velocity with **NED Velocity Uncertainty (0x82,0x09)**, and attitude can be monitored with **Euler Angles Uncertainty (0x82,0x0A)**.



Installation

IMPORTANT: This equipment is not suitable for use in locations where children are likely to be present.

Mount the 3DM-CV7

Mechanical provides dimensions and tolerances as well as cabling and connector specifications and part numbers.

Properly Power the 3DM-CV7

WARNING: Only use power supplies within the operating range of the sensor, or permanent sensor damage or personal injury could result. Observe connection polarity.

Power must be supplied to the 3DM-CV7 via pin 3. See **Electrical** for pinout and input voltages.

Apply the Proper Mounting Transform

In its default state, the labeled axes of the 3DM-CV7 are assumed to be aligned with the vehicle frame (X axis in the direction of travel, Z axis down). If the device is mounted in an orientation where the sensor frame axes are not aligned to the vehicle axes, then a sensor-to-vehicle transform must be applied to correct for this misalignment. See the **Vehicle Frame** section for details on applying this transformation.

Install the Antenna(s)

The GNSS antenna(s) must be mounted and the offsets in relation to the 3DM-CV7 must be input into the Navigation Filter. See **Antenna(s)** for details on installing the antenna and accounting for offsets.

Install the Aiding Sensor(s)

The 3DM-CV7 is designed to accept external **Aiding Measurement** data to help improve its navigation solution in challenging GNSS environments. The effectiveness of these aiding measurements is directly affected by the accuracy of the aiding measurement and the care that is taken when installing the aiding sensors. The **External Aiding Sensors** page provides a brief overview of best practices when installing common aiding sensors and provides recommendations on field-testing approved models to simplify integration with the 3DM-CV7.

Configure the Aiding Frame(s)

The 3DM-CV7 provides a flexible, yet simplified method for accounting for the effects of translations and rotations of **External Aiding Sensors** from the 3DM-CV7 through the use of Aiding Frames. Correctly configuring these frames is critical to achieving a successful navigation solution. See the **Aiding Frames** page for instructions on properly configuring these frames for your application.

Perform a Magnetometer Calibration

If a magnetometer is being used as a heading aiding measurement, a magnetometer calibration with the 3DM-CV7 installed in the application and after all other components have been installed is crucial in ensuring accurate heading measurements. See **Magnetometer Calibration** for details on performing this calibration using our convenient desktop application, SensorConnect.

You are here: Installation > Antenna(s)

Antenna(s)

Antenna Type

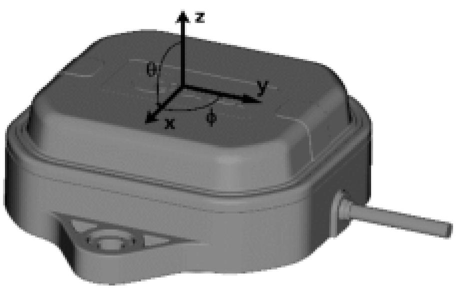
Both single frequency (L1) and multi-frequency (L1/L5) GNSS antennas can be used with the 3DM-CV7. However, a GNSS antenna with multi-frequency capability ***is required*** for RTK and/or dual antenna functionality.

Antenna Mounting Considerations

Proper installation of GNSS antennas is critical to GNSS/INS performance. To get the best GNSS performance possible, there are many RF considerations that must be taken into account. The following methods can be used to optimize performance:

1. Mitigate Multipath:

- Ensure the GNSS antennas have an unobstructed sky view. Even materials like plastic and glass can attenuate the GNSS signal and should not be placed between the antenna and the sky unless it is known they will not interfere with GNSS signal reception.
- Ensure the sky view from the GNSS antennas is not obstructed by trees or buildings. A "good" sky view is ≥45° in all directions from the vertical.
- Ensure the sensitive axis of the antenna is facing vertical towards the sky (z-axis aligned with vertical.)



- Install ground planes under each antenna: a ~10x10cm metallic ground plane under each antenna to reject multipath is recommended. This simple groundplane can be purchased from Sparkfun. (For more information on OEM ground plane design, see pg. 17 in u-blox GNSS Antennas App Note and pg. 7 of u-blox ANN-MB Datasheet)
- Ensure that there is a minimum of 3 centimeters of clearance around the circumference of the antenna.

2. Check antenna offsets:

- Ensure antenna offsets are correctly entered and are not swapped (a common problem.) See **How to Measure Antenna Offsets** section below.

3. Mitigate EMI:

WARNING: Even low levels of EMI can significantly degrade GNSS performance and must be considered when choosing an antenna mounting location.

1. Common EMI sources are Electric motors, computers, modems, power supplies, unshielded data lines, cameras, LIDAR, compute boxes, relays.
2. Mount the antennas as far away as possible from EMI Sources.
3. Route cables as far away as possible from EMI Sources.
4. Minimize cable length, ensuring to maintain cable integrity with strong solder joints and verify antenna cable is not shorted. For OEM applications, contact sales for custom cable lengths.
5. Store any excess coiled GNSS cable away from EMI sources as coiled cable can act as an antenna and introduce significant EMI.
6. For OEM designs, ensure antennas are mounted following manufacturers specification. If using the default U-Blox antennas, consult sections 5 and 6 of the antenna design guide.
7. If none of the above fixes the performance issues, contact MicroStrain by HBK customer support for a more detailed analysis of the application.

TIP: GNSS quality metrics such as position uncertainty can be used to identify EMI. Toggle power to the system suspected of generating EMI and monitor the effects on the position uncertainty. If the uncertainty increases when the system powers on, it could be impacting GNSS performance.

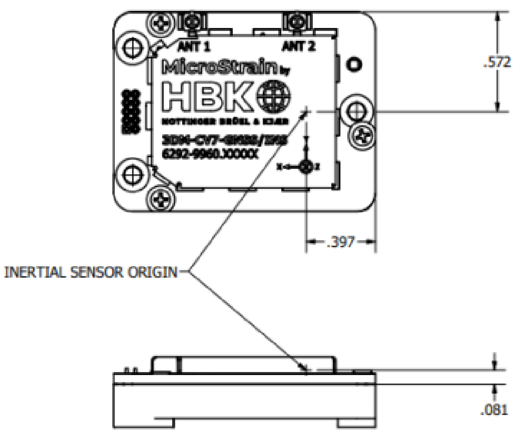
PPS Source

If using a single antenna, it is recommended to connect the primary antenna to antenna port 1 on the 3DM-CV7, because the PPS source selector (**PPS Source (0x0C,0x28)**)defaults to GNSS 1.

How to Measure GNSS Antenna Offsets

The GNSS antenna offsets must be measured and provided to the 3DM-CV7 through the **Frame Configuration (0x13,0x01)** command if using the **Aiding (0x13)** commands, or the **GNSS Offset Control (0x0D,0x13)** command if using the **NMEA Input** feature. The antenna offsets are measured with respect to the **Vehicle Frame** from the SENSOR ORIGIN of the 3DM-CV7 to the antenna phase center of the GNSS antenna. (See Figure: Mechanical Drawing: SENSOR ORIGIN and u-blox ANN-MB Mechanical Drawing: Antenna Reference Point). The antenna phase center of an example u-blox antenna is located in the x-y center of the antenna, 0.008m above the Antenna Reference Point (ARP) (+z). For best performance, measure the antenna offsets as accurately as possible.

CV7-GNSS/INS Mechanical Drawing: INERTIAL SENSOR ORIGIN



u-blox ANN-MB Mechanical Drawing: Antenna Reference Point

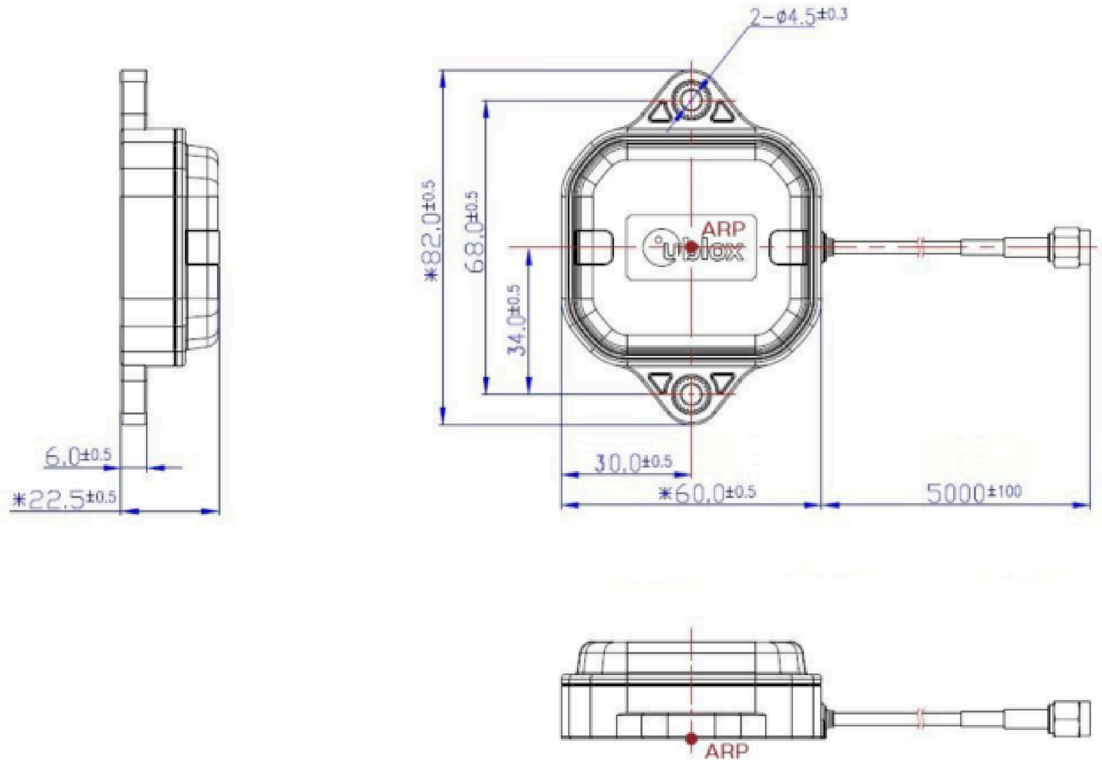


Figure 1: ANN-MB series mechanical drawing. Dimensions are given in mm. The antenna reference point (ARP) is marked with a red dot.

Example: Inputting the GNSS Antenna Offsets when using the MIP Aiding (0x13) command set

The following example illustrates how to measure and use the [Frame Configuration (0x13,0x01)] command when providing GNSS aiding measurements to the product via the MIP [Aiding (0x13)] command set.

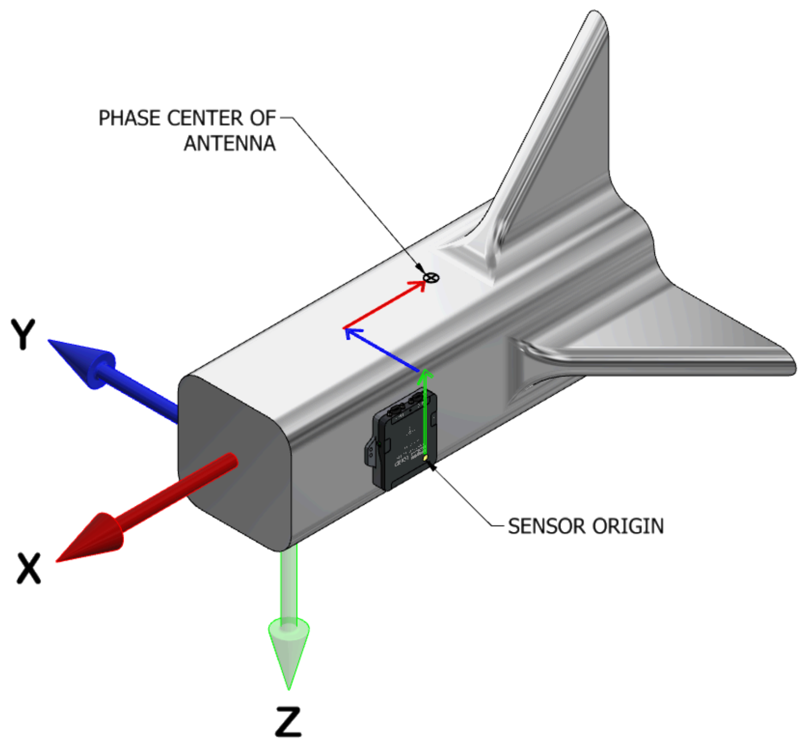
<u>Vehicle Frame Axes: CV7 to Antenna Offset Measurement Example</u>

Frame Configuration (0x13,0x01) Command Parameters	Values	Units
Frame ID	**1**	
Format	**1**	
Tracking Enabled	**1**	
Translation	**-0.32, 0.11, -0.64**	meters
Rotation	**0, 0, 0, 0**	

Example: Measuring the GNSS Antenna Offsets when using NMEA Input

The following example illustrates how to measure the antenna offsets and use the :

Vehicle Frame Axes: CV7 to Antenna Offset Measurement Example



In this example, the green arrow shows the offset along the z-axis of the vehicle frame. It starts at the INERTIAL SENSOR ORIGIN and points in the negative z direction. The blue arrow represents the offset along the y-axis the vehicle frame. It is in the positive y direction. The red arrow represents the offset along the x-axis of the vehicle frame. It is in the negative x direction.

The table below uses some fictional magnitudes to detail the parameters that would or be entered into the or **GNSS Antenna Offset Control (0x0D,0x13)** command. The signs are correct for the above example.

or GNSS Antenna Offset Control (0x0D,0x13) Command Parameters	Values	Units
X offset	**-0.32**	meters
Y offset	**0.11**	
Z offset	**-0.64**	

Antenna Lever Arm Calibration

Antenna offset error can be introduced by inexact offset measurements and by the fact that the phase center of an antenna does not always correspond to its physical center. These errors can, in turn, produce positioning and heading errors. The **GNSS Antenna Cal Control (0x0D,0x64)** command can be used to enable active tracking of these errors in the Navigation Filter. For the 3DM-CV7, the maximum offset error defaults to 0.1 meters. The maximum offset that can be entered is 0.5 meters.

GNSS Antenna Cal Control (0x0D, 0x64)	
Maximum Offset	0.5 meters
Default Offset	0.1 meters

Board and Cable Kits

C-Series Development Board

The C-Series development board is a demonstration board that allows you to directly interface with the 3DM-CV7-GNSS/INS. MicroStrain by HBK offers various development kits that contain the C-Series development board.

A full schematic of the C-Series development board can be found here.

Magnetometer Calibration

The following provides instructions for how to calibrate the 3DM-CV7's magnetometer using **SensorConnect**. Some of the information will also apply to those looking to implement a magnetometer calibration with custom software.

Why Calibrate?

Calibrating a magnetometer compensates for the magnetic (hard iron) and ferrous material (soft iron) present on the device that a 3DM-CV7 is mounted to. The local hard and soft iron effects distort readings of the earth's magnetic field and must be calibrated out in order for the 3DM-CV7 to generate accurate heading estimates.

IMPORTANT: During calibration, the 3DM-CV7 must be mounted to the device for which it is generating heading data, i.e. the drone, the robot, the tractor, etc.

IMPORTANT: Do not perform a calibration near large ferrous objects or strong magnets.

Basics

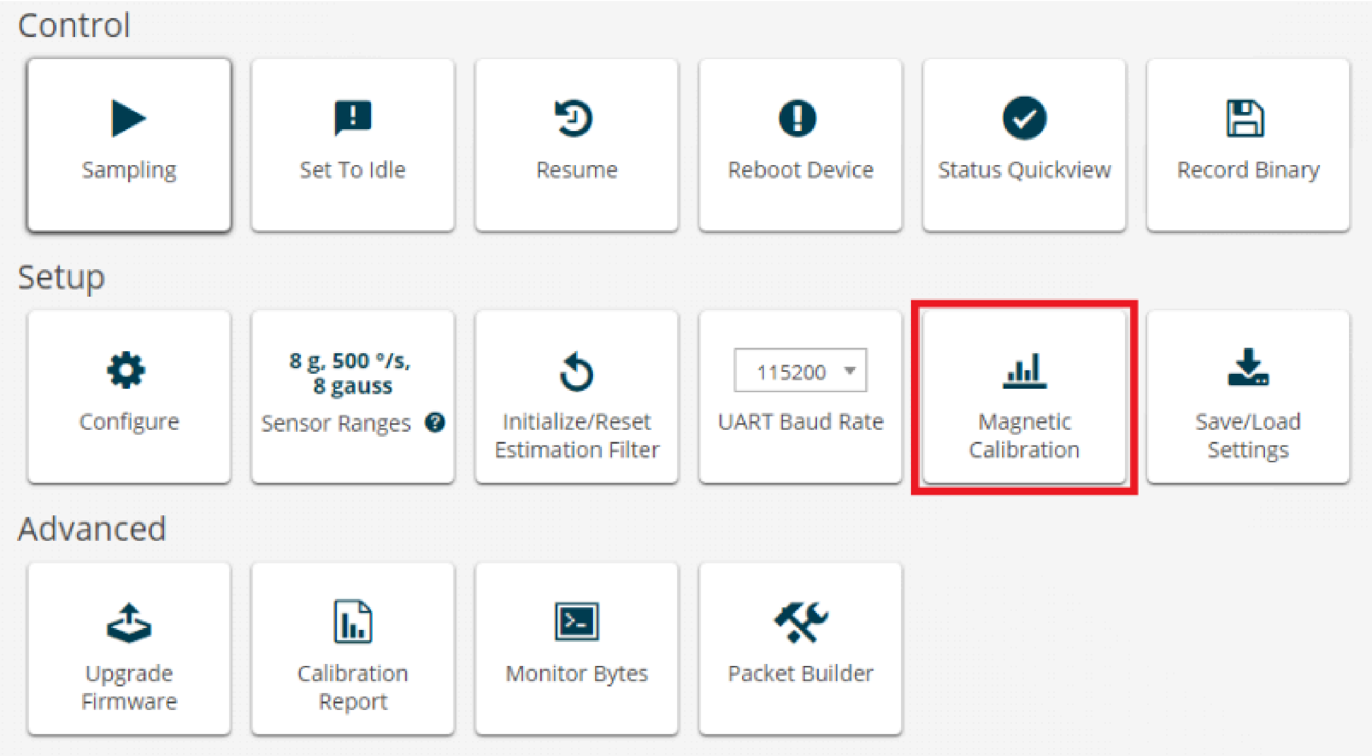
An ellipsoidal fit provides a better calibration than a spherical fit and should be used whenever possible. However, there are some applications where an ellipsoidal fit may not be possible because the device cannot be rotated enough during calibration to get good spatial coverage. An example would be calibrating a 3DM-CV7 mounted to a car as it drives on a flat parking lot. The quantity of spatial coverage can be determined with **SensorConnect**. More on this later.

When an ellipsoidal fit is not possible and a spherical fit must be used, it is highly recommended to specify the field strength of the earth's magnetic field in the application space. This can be determined through the World Magnetic Model (WMM). When an internet connection is available, **SensorConnect** can retrieve this data, or it can be entered manually.

Using SensorConnect

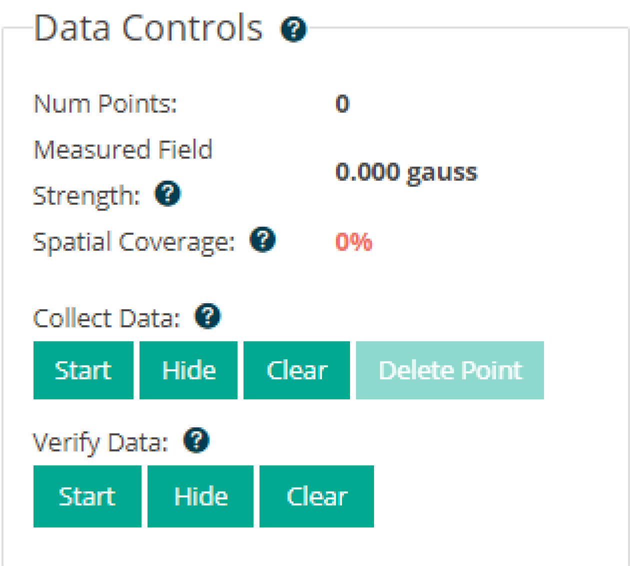
Step 1 - connect

Open SensorConnect and connect to your 3DM-CV7. Select the tile highlighted in red.



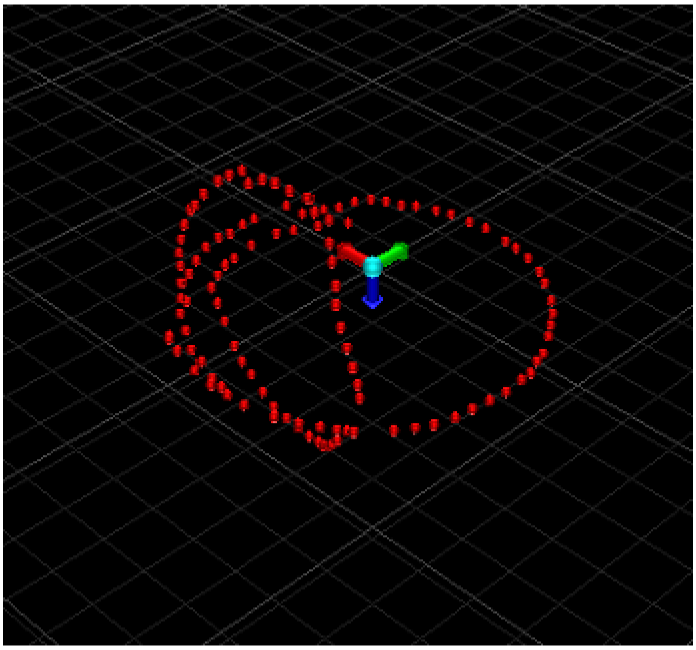
Step 2 - collect data points

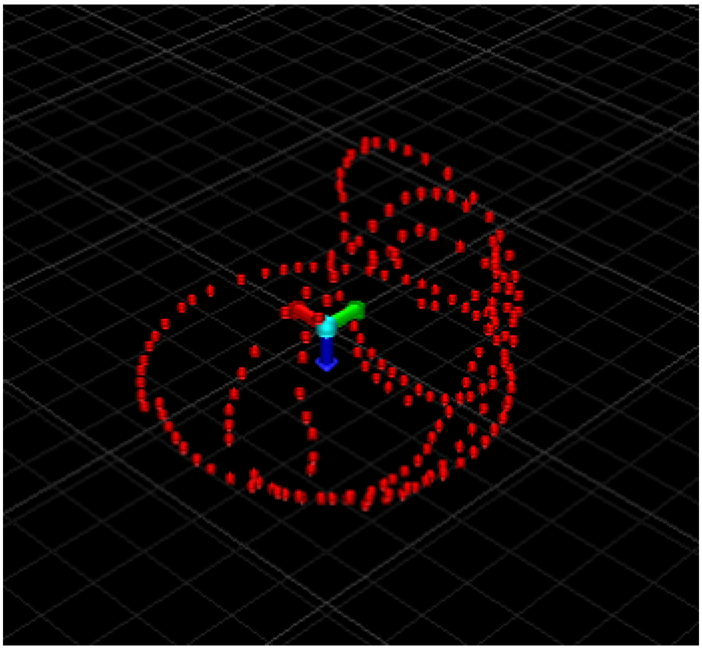
You will see the following panel in the Magnetic Calibration window.



Press **start** under "Collect Data" and rotate the device on which the 3DM-CV7 is mounted. For best results, rotate the device in all three dimensions. You should see two things happen:

2. The "Spatial Coverage" variable should increase. As it increases, the color will change from red to yellow, meaning enough coverage has been collected to perform a spherical fit, and eventually to green, when there is enough coverage to perform an ellipsoidal fit.
3. A point cloud will populate a graphical display (shown below). The point cloud can guide a user to the regions missing data points so a user can increase the spatial coverage. The graphic is interactive and can be rotated using a mouse.

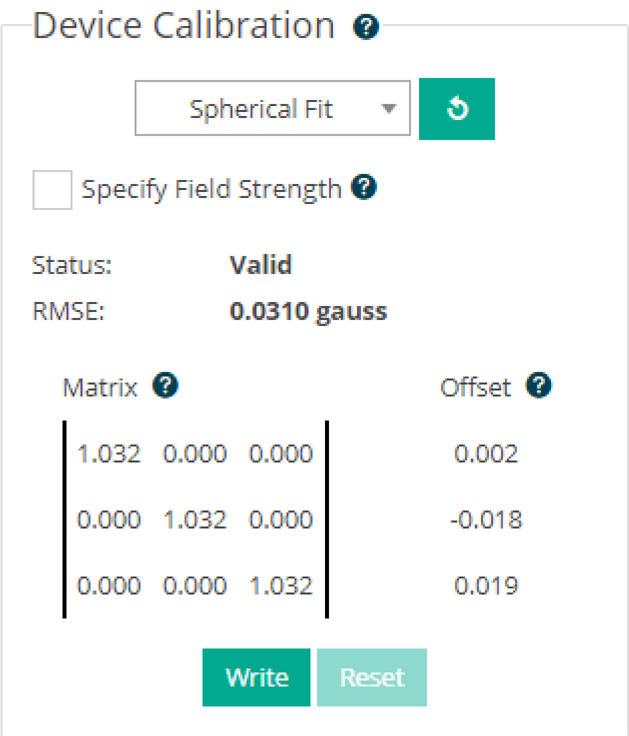




The image on the left shows an example of sufficient coverage for a spherical fit. The image on the right shows enough coverage for an ellipsoidal fit.

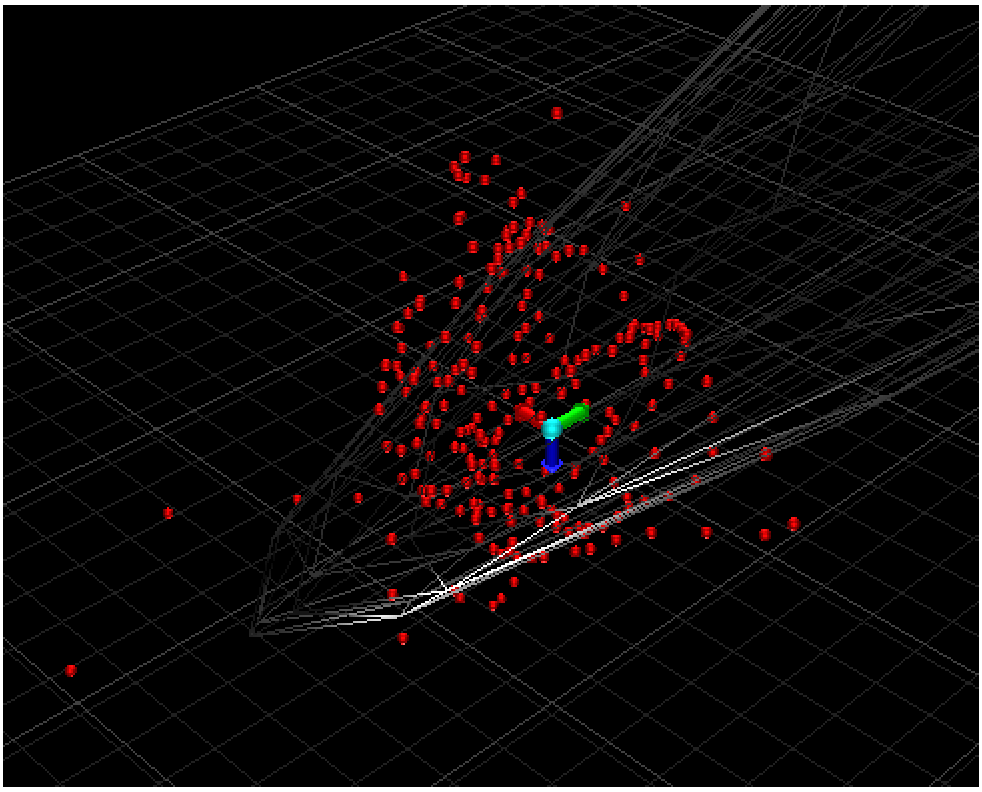
Once the coverage is sufficient for the application, press **stop** under "Collect Data".

Step 3 - choose fit



In the "Device Calibration" box, a user can select a spherical fit or an ellipsoidal fit from a drop-down menu. As mentioned previously, an ellipsoidal fit will produce the best calibration, but if you were unable to get enough spatial coverage for an ellipsoidal fit, use a spherical fit.

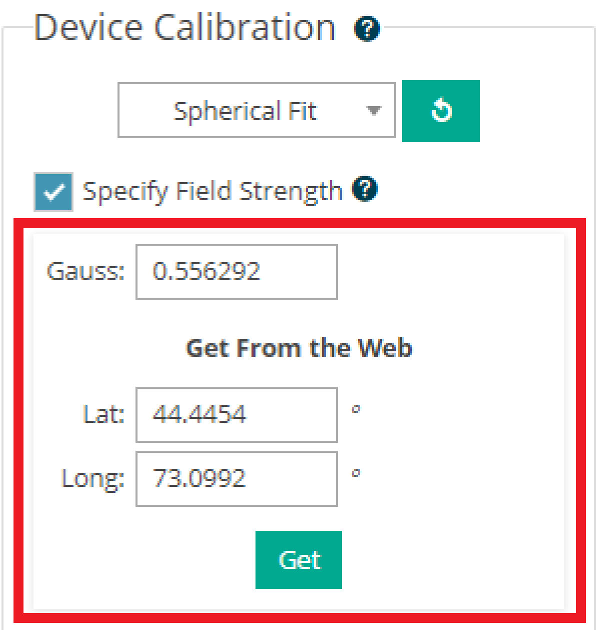
Too many outlier data points (outside of an imagined sphere or ellipsoid around the sensor) can result in a poor fit. Shown below. If this result occurs, redo step 2 being sure to constrain the motion of the sensor to rotation within a sphere.



To apply the calibration coefficients the user must press "Write". The calibration coefficients will be written to the non-volatile memory of the 3DM-CV7. These coefficients will not be erased between power cycles.

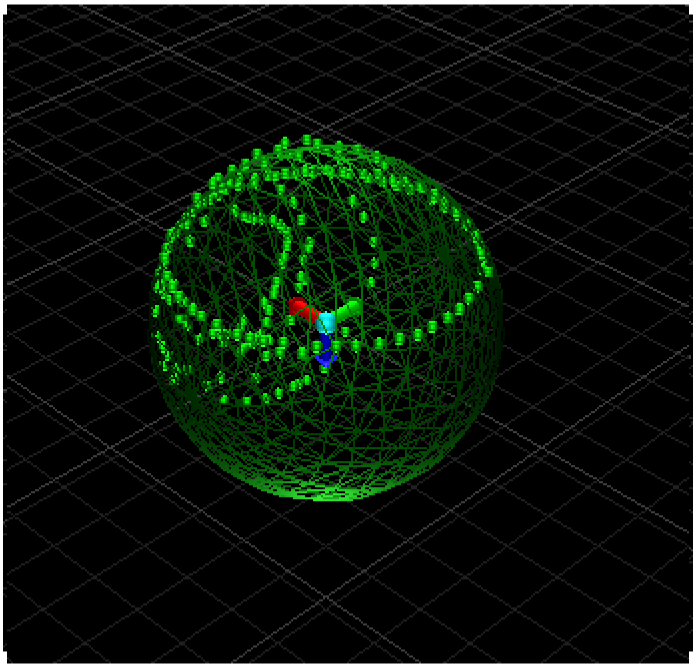
If using a spherical fit, it is highly recommended that, prior to writing the coefficients, the user specify the local magnetic field strength. This can be done by selecting the "Specify Field Strength" check box. The box highlighted in red will appear (see below).

A user can either manually enter the earth's magnetic field strength at their location in the box labeled "Gauss", or, if web access is available, the user can enter their latitude and longitude and press "Get". This will auto-populate the "Gauss" box with the field strength at their location as determined from the World's Magnetic Model (WMM).



Step 4 - verify data

This step is optional. When a user presses **start** the graphic displays the calibrated output as the user rotates the device the 3DM-CV7 is mounted to. A well-calibrated magnetometer will measure the same field strength for earth's magnetic field in every orientation; therefore, all the data points in the verification step should lie on the surface of a sphere. The radius of the sphere is the strength of the earth's magnetic field.



External Aiding Sensors

The Navigation Filter is designed to accept external **Aiding Measurement** data to help improve its navigation solution. The effectiveness of these aiding measurements is directly affected by the accuracy of the aiding measurement and the care that is taken when installing the aiding sensors. This page provides a brief overview of best practices when installing common aiding sensors and provides recommendations on field-testing approved models to simplify integration with the 3DM-CV7-GNSS/INS.

GNSS Receivers and Antennas

GNSS aiding measurements are provided to the 3DM-CV7-GNSS/INS from external GNSS receiver(s). Improper installation can lead to navigation solution errors in the 3DM-CV7-GNSS/INS that can be difficult to debug. Follow the recommendations on the **Antenna(s)** page to ensure performance is maximized.

u-blox ZED-F9P

The u-blox ZED-F9P has been tested extensively as a GNSS aiding sensor for the 3DM-CV7-GNSS/INS and provides great for performance for low SWAP-C, in a simple to use interface.

Single Antenna:

Example code is provided in our MIP SDK to simplify integration of this receiver with the 3DM-CV7-GNSS/INS for a single antenna position, velocity, and time GNSS solution. It is important to configure the 3DM-CV7-GNSS/INS for your application to ensure the device performs properly.

The ZED-F9P also can stream NMEA data on its UART ports which the 3DM-CV7-GNSS/INS can directly parse from any of its GPIO ports, greatly simplifying the GNSS integration process. The **NMEA Input** page provides a high level overview of the steps to configure each device for this simple integration. Use of the MIP **Aiding (0x13)** command set will enable better navigation performance than the **NMEA Input** due to the GNSS uncertainties being updated dynamically instead of set statically at initialization, but may require more configuration.

Body Frame Velocity Sensors

The following aiding sensors can provide useful aiding measurements to the 3DM-CV7-GNSS/INS via the **Body Frame Velocity (0x13,0x2A)** command.

Radar

MicroStrain's Robotics R&D group has performed extensive testing and found radar to be an effective and robust aiding sensor for ground vehicle applications. An example of this integration and the impressive 3DM-CV7-INS performance improvements during a GNSS outage has been published with example code in this white paper: Enhancing Inertial Navigation System Performance with Radar. This white paper provides example code using MicroStrain Inertial ROS driver, a custom ROS package for radar pre-processing, and configuration of a Ublox F9P.

When integrating a radar velocity sensor with the 3DM-CV7-GNSS/INS, pre-process the velocity into a 3 element vector float format and provide the aiding measurement via the **Body Frame Velocity (0x13,0x2A)** command. An example of this can be found on the **Filter Aiding Measurements** page.

Pitot Tube

While the 3DM-CV7-GNSS/INS does not natively accept air speed data, a pitot tube may be used in conjunction with other sensors to remove wind velocity to calculate the x-axis of the **Body Frame Velocity (0x13,0x2A)** command. Ensure to either provide Uncertainties of 0 m/s for the y and z-axes or set the Valid Flags to 0 for the same axes of the command.

Optical Flow

Optical Flow can be a useful aiding sensor for applications which are relatively close to the ground and wish to constrain their position error in the event that their GNSS signal becomes degraded or denied. The 3DM-CV7-GNSS/INS does not natively accept optical flow sensor data, but can accept pre-processed data through the **Body Frame Velocity (0x13,0x2A)** command.

External Magnetometer

The 3DM-CV7-GNSS/INS supports external magnetometer aiding to mitigate the magnetic interference errors ubiquitous on autonomous vehicle applications. When installing an external magnetometer, ensure to place it far away from potential magnetic interference sources to reduce or eliminate their effect. Examples of common magnetic interference sources are moving ferro-magnetic components like motors, robotic arms, fans etc. See the **Filter Aiding Measurements** for an example aiding command for external magnetometer input.

External Pressure Sensor

The 3DM-CV7-GNSS/INS supports external pressure sensor aiding to enable the input of higher-accuracy static pressure sensors. This aiding measurement helps constrain altitude drift during GNSS denied environments. Ensure to install this sensor in a location on the vehicle where it is exposed to static atmospheric pressure, but avoid locations of high dynamic pressure ex: the leading edge of a high speed fixed wing drone, in the prop-wash of a quadcopter, etc.

NOTE: MicroStrain Support Engineers are always available to assist in sensor integration. See **FAQ: Support** for contact info.

You are here: Navigation Filter

Navigation Filter

The 3DM-CV7 is a highly-flexible INS built on a tactical-grade IMU. It is designed to accept many of the external **Aiding Measurements** available on autonomous vehicles to constrain drift and provide a useful navigation solution in challenging or contested GNSS environments.

Our Adaptive Kalman Filter is applied to all internal and external aiding measurements to reject anomalies in real time. See **Adaptive Filtering** for a better understanding of how the 3DM-CV7 can improve your application's navigation robustness.

The 3DM-CV7's Navigation Filter can be easily optimized by defining the kinematics for a given application. **Filter Kinematic Constraints** provides more detail. These can be considered assumptions of constant behavior and are synonymous with aiding measurements.

The Navigation Filter must be provided initial conditions before it will start running. The initial conditions can be generated internally through auto-initialization or provided externally by the user. A single command handles both cases. See Filter Initialization for more details.

Once the filter is set up and running, users should monitor the filter status through the Filter Status message **Status (0x82,0x10)**. The **Filter Status** section details how to interpret the Status message and explains the Filter Modes.

For a step-by-step guide to setting up 3DM-CV7 for your application, see **Getting Started**.

Adaptive Filtering

An Extended Kalman Filter (EKF) estimates the navigation solution by combining several different sources of information, including integrated acceleration and angular rates, navigation models, and external measurements such as GNSS signals. The weights given to these different components, all of which are imperfect, are determined by simple statistical models that quantify the expected accuracy of any measurement or constraint. In a standard EKF, these statistical models are determined prior to combining the measurements, and the weights are assigned without regard to what the actual measurement values are. For this reason, a standard EKF will suffer performance degradation in the face of measurement anomalies, or when the true system dynamics depart from the modeled system. An adaptive EKF overcomes these limitations by continuously monitoring the consistency of the actual measurements with the assumed statistical models, and modifying the statistical models to account for any discrepancy.

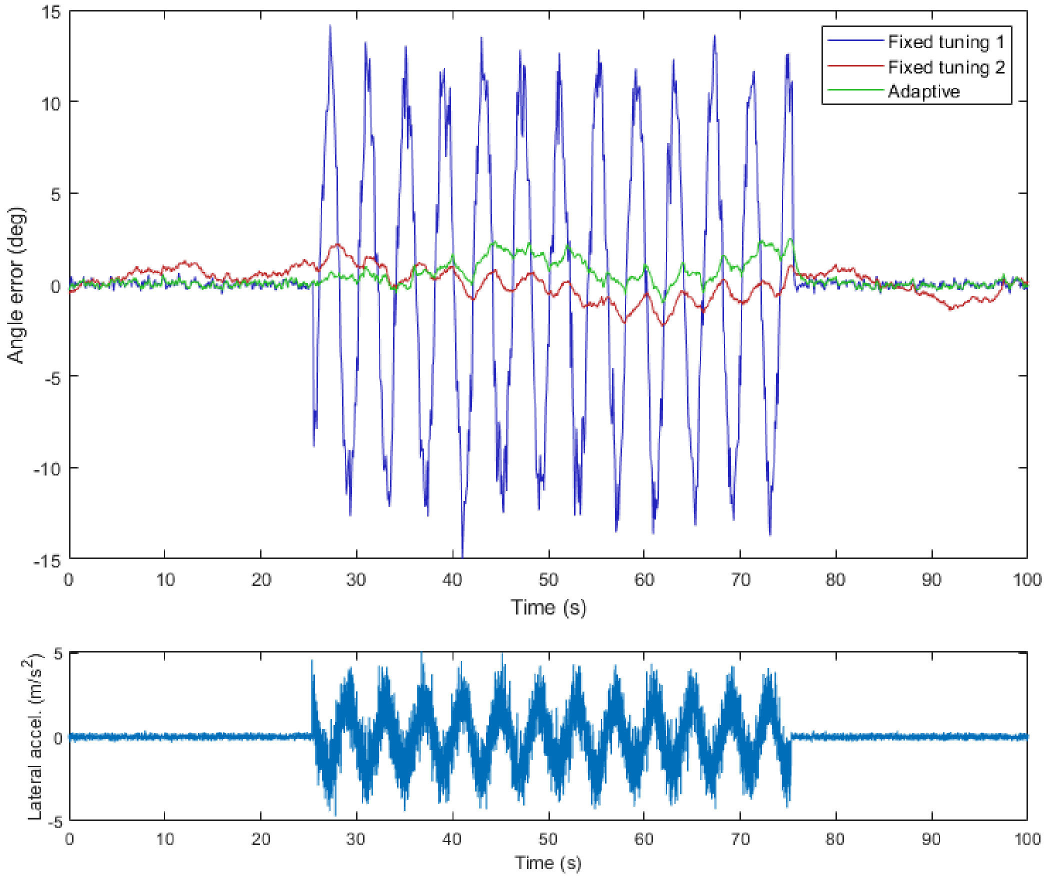
The most basic kind of adaptive filtering is simple outlier rejection, where measurements are discarded when they are deemed extremely unlikely according to the current state estimate and the measurement error model. The 3DM-CV7 goes further and uses additional feedback to modify the measurement and dynamic system error models, ensuring that they remain consistent with the actual measurements.

Adaptive Filter Examples

Two examples are provided here which illustrate the adaptive filtering concept as implemented on the 3DM-CV7. These simple examples are for a sensor operating in vertical gyro mode (tracking pitch and roll angles), but the concepts extend to all operating modes and supported aiding measurements.

Vibration Induced Error Rejection

Sensor performance can be sensitive to the tuning of the measurement models used by the Kalman filter, and a fixed filter tuning is only optimal for a specific level of sensor noise or background vibration level. Since it is not practical or convenient for a user to manually tune the filter for different applications, operating regimes, or dynamic conditions, the 3DM-CV7 uses adaptive Kalman filtering. Adaptive filtering continuously updates the measurement models in real time to maintain consistency between modeled and actual measurements. In this example, a 3DM-CV7 in vertical gyro mode is subjected to an intermittent period of increased vibration with a large lateral acceleration component (shown in the lower graph). The top graph compares the roll angle error with two different fixed tunings against the roll angle error experienced with adaptive filtering turned on.



Roll estimation error (top) of three differently tuned filters using simulated inertial data for a non-rotating sensor with a variable acceleration profile along the sensor y-axis (bottom).

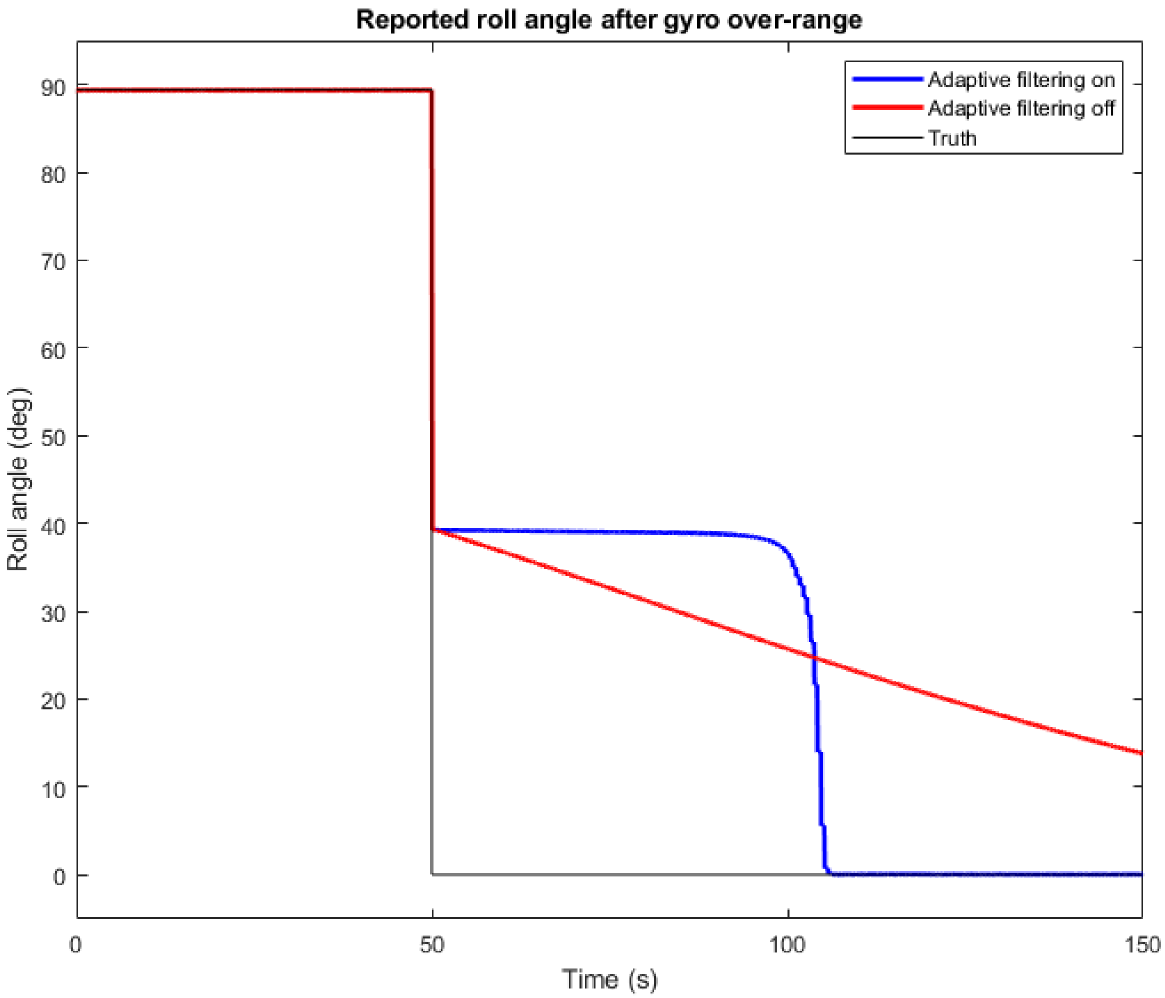
The first filter (Fixed tuning 1) has a fixed tuning which weights angle estimates from the accelerometers more. The second filter (Fixed tuning 2) has a fixed tuning that weights the accelerometers less. The third filter uses the 3DM-CV7 adaptive filter.

In this example, Fixed tuning 1 performs very well when there is very little vibration, but it does not give good results when the vibration level is high. Likewise, Fixed tuning 2 does well when the vibration level is high, but does not do as well when there is very little vibration. The adaptive tuning transitions smoothly between these two regimes, giving good performance over a wide range of operating scenarios.

Over-range Recovery

The adaptive Kalman filter hastens recovery from an unexpected dynamic error. In this example, a true rotation of 1400 deg/sec was briefly applied to the 3DM-CV7, exceeding the range of the on-board gyro by 400 deg/sec. The 3DM-CV7 was then returned to a 0 degree roll angle. The graph below compares the roll angle output with adaptive filtering turned on (blue line) to the output with the adaptive filtering turned off (red line). As the graph shows, the filter's roll angle output recovers much faster from the gyro over-range event when the adaptive filtering is turned on.

Without adaptive filtering, the time constant for recovery is constant. When adaptive filtering is turned on, the time constant starts high, then gets increasingly smaller until the filter has recovered.



Adaptive Filter Options

On the 3DM-CV7, users can control two parameters which influence the adaptive filtering. The first parameter is the adaptive filtering level, which controls how aggressively the filter will adjust its statistical models in response to inconsistent measurements. The second is the adaptive filtering time constant, which controls how quickly the filter will adapt. See for **Adaptive Options (0x0D,0x53)** more information on changing these parameters.

Filter Initialization

Upon start-up or after a filter reset command is issued, the Navigation Filter will require valid initial conditions to be provided before it will run. The running mode of the filter, and the available outputs, depend on the initialization conditions selected to initialize the filter. See **Filter States** for more information regarding initial conditions and filter mode.

Filter initialization is configured through a single command: **Initialization Configuration (0x0D,0x52)** command. The following discussion explains how to use each parameter in this command.

WARNING: In instances where the 3DM-CV7-GNSS/INS may experience large time jumps after the Navigation Filter has been initialized (i.e. acquiring a PPS fix), filter instabilities may occur. It is recommended that a **Reset Navigation Filter (0x0D,0x01)** command is performed if a large system time jump is experienced after filter initialization. To check the PPS status, monitor the **Time Sync Status (0xA0, 0x02)** and verify the Time Sync Boolean is true.

Wait for Run

The user may elect to have the Navigation Filter begin running as soon as the selected initial conditions are available or remain idle until a run command is issued. This option is set through the **Wait For Run Command** parameter.

Initial Condition Source

The initial conditions include an initial attitude (pitch, roll, and heading), initial velocity, and an initial position estimate. The user may elect to have some of these values provided automatically by the 3DM-CV7's on-board sensors, or they may choose to provide some or all of them manually or from external aiding measurements. The sources of these initial conditions – manual (user supplied) or automatic (from internal measurements) – are set through the **Initial Cond Src** parameter.

Sources for Auto-Initialized Conditions

The following describes the auto-initialization sources for the initial conditions:

Pitch/Roll

If pitch/roll is set to auto-initialize, the 3DM-CV7 will use its own accelerometer measurements for pitch/roll initialization.

Tip: For best performance, initialize pitch and roll while the device is stationary. Linear acceleration will introduce errors into this attitude measurement.

Auto-Heading Alignment Selector

If a user would like to auto-initialize heading, they have two sources. A user can opt to use one source, or two sources through the **Auto Heading Alignment Selector:**. The sources are: GNSS kinematic alignment, and magnetometer alignment. Each auto-heading source is described in more detail in the following section.

Heading

The sources for auto-initialization of heading are:

- **Dual antenna heading:** Heading is initialized from a GPS compass measurement from the dual antenna heading system. This method does not make any assumptions about vehicle dynamics, but does require proper setup and installation of the dual antenna system. See Installation for more information.
- **Kinematic alignment**: Heading is initialized from the GNSS velocity vector. Once the vehicle reaches a velocity threshold of 2 m/s, heading is initialized in the direction of travel. This method makes the assumption that the vehicle's direction of travel is in the positive X direction of the vehicle frame, and is **violated by moving in reverse**.Special care must be taken to accelerate in the X direction only until the velocity threshold is reached especially when using this alignment method on vehicles that have the ability to move perpendicular to the vehicle X axis.

- **Magnetometer:** Heading is initialized from a magnetometer measurement. Special care must be taken if this method is to be used. See **Magnetometer Warning** for details.
- **External Heading:** Heading is initialized from an external heading input provided once at least every 5 seconds (preferably more often).

Position/Velocity

If set to auto-initialize, the external GNSS position and velocity is used for initialization.

Manually Initialized Conditions

Manual estimates of the **Initial Heading**, **Initial Pitch**, **Initial Roll**, **Initial Position**, and **Initial Velocity** can be entered in their respective fields in the **Initialization Configuration (0x0D,0x52)** command. If a user is providing manual position/velocity estimates, they need to specify the reference frame in the **Reference Frame Selector**.

NMEA over GPIO Initialization

See **NMEA Input** page for instructions on initialization with NMEA.

You are here: Navigation Filter > Filter Kinematic Constraints

Filter Kinematic Constraints

A selection of kinematic constraints is available that can improve Navigation Filter performance under certain dynamic conditions. When enabling these kinematic constraints, it is important to consider the typical motion profile of the application and ensure that the assumption underlying each constraint is satisfied most of the time.

Zero velocity update (velocity ZUPT)

This constraint helps stabilize the navigation solution during prolonged stationary periods. This constraint is useful for vehicles which make frequent to occasional stops.

Assumption

The platform is stationary when the reported velocity falls below the user provided threshold.

Command

Body Frame Velocity (0x13,0x2A)

Implementation

To implement the velocity ZUPT on the 3DM-CV7, the **Body Frame Velocity (0x13,0x2A)** is employed to afford the user more flexibility to send velocity ZUPT's on any axis. This enables the user to tune for their application specific dynamics which may not be represented in a 2D planar ground vehicle ZUPT.

The user must send this command when the scalar magnitude of the GNSS reported velocity vector is equal-to or less than the threshold value set by the user. The External GNSS **NED Velocity (0x94,0x05)** vector can be found in the **External GNSS (0x94)** descriptor set.

Zero angular rate update (angular rate ZUPT)

This constraint helps reduce gyro bias errors on the vertical (yaw) axis. This constraint is useful for ground vehicles which make prolonged stops, especially when no heading aid is available.

Assumption

The platform is not rotating when the angular rate estimate falls below a user provided threshold.

Command

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

Wheeled vehicle constraint

This constraint helps maintain a more accurate platform heading, especially during periods of low acceleration and poor heading solution. It is not advisable to use this constraint in applications where the platform velocity and heading routinely diverge, such as with aircraft.

Assumption

Components of velocity perpendicular to the primary axis of the platform (as indicated by the mounting transformation) are zero.

Command

Body Frame Velocity (0x13,0x2A)

Implementation

To implement a wheeled-vehicle constraint on the 3DM-CV7, use the **Body Frame Velocity (0x13,0x2A)** command. Send the command at the desired rate with a Y-axis velocity of 0.0 m/s, and the X and Z-axes data marked as invalid. A minimum of 1 Hz is recommended.

Additionally, a user may implement a 2D mode by sending 0.0 m/s for the Z-axis.

You are here: Navigation Filter > Filter Status

Filter Status

The 3DM-CV7's Navigation Filter reports its status through the **Status (0x82,0x10)** message. The status message provides two important pieces of information for the user: the current **filter state**, or mode, and **status flags** that provide real-time warnings about the filter's condition and its estimates.

The filter status message should be monitored after the Navigation Filter has started running (see **Filter Initialization** for instructions on starting the filter). The **filter state** tells the user whether or not the Navigation Filter has initialized to the desired mode. The filter condition bits in the **status flags** tell the user whether the Navigation Filter is converging, stable or unstable.

Filter State

See **Filter States** for a complete discussion of filter modes and instructions on how to interpret the filter state field in the **Status (0x82,0x10)** message.

Status Flags

The status flags in the **Status (0x82,0x10)** message are explained on the **Filter Status Flags** page.

Filter States

The filter status field in the **Status (0x82,0x10)** message reports the Navigation Filter's current state. The Navigation Filter has four states, or modes. The state that the Navigation Filter operates in is determined by the initial conditions provided to the Navigation Filter through the **Initialization Configuration (0x0D,0x52)** command. See **Filter Initialization** for details.

Column one of the table below shows the four Navigation Filter states. The second column shows the corresponding value reported for each state in the filter status field in the **Status (0x82,0x10)** message. The third column shows the initial conditions that must be provided to the Navigation Filter to achieve each state.

Below the table, each state is described in greater detail.

Filter Mode	Filter State Value	Initial Conditions Required	Notes
Initialization	1	none	none
Vertical Gyro	2	none	heading relative to initial heading
AHRS	3	absolute heading measurement	heading relative to true or magnetic north (absolute heading)
Full Navigation	4	absolute heading measurement, initial position, initial velocity	heading relative to true or magnetic north (absolute heading), position, velocity

Initialization

In this mode, initial conditions are incomplete or invalid, or else the sensor is waiting for required input from the user.

Vertical Gyro

In this mode, the filter reports valid gyro stabilized pitch and roll estimates and heading estimates relative to the initial heading provided by the user. If an initial heading is not provided by the user, the filter will provide relative heading estimates referenced to its orientation at initialization time.

See **Filter Initialization** for details on entering an initial heading.

In this mode, only the 3DM-CV7's internal inertial measurements are incorporated by the Navigation Filter.

AHRS

This mode is similar to the **vertical gyro** mode, except a valid absolute heading reference has been provided from either the user or the magnetometer. See Filter Aiding Measurements for more details on heading measurement sources.

Full Navigation

Filter reports valid position, velocity, pitch, roll, and heading estimates. To enter this mode, the Navigation Filter requires valid initial position, velocity, and heading estimates. These estimates can be retrieved from external aiding measurements for position, velocity, and heading, or internal magnetometer heading, or provided by the user. See **Filter Initialization**.

Filter Status Flags

The filter status flags are available in the **Status (0x82,0x10)** message. The assignment of the filter status flags in the **Status (0x82,0x10)** is device specific. The following table shows an overview of the status flags for the 3DM-CV7. The sections below it provide a detailed description of each status flag.

Bit #	Description	Potential Cause	Recommended Action
0-1	Filter condition (Stable/Converging/Unstable)	Insufficient initial conditions supplied.	Verify required conditions are properly enabled. Review **Filter States**.
2	Roll/Pitch Warning	Sensor is not aligned with vehicle frame.	Review **Vehicle Frame** page.
3	Heading Warning	Heading source is not valid. For Magnetometer Aiding:time varying magnetic interference.	For External Heading:**FAQ:How to improve filter performance** For GNSS Kinematic Alignment: **FAQ:How to improve filter performance** For Magnetometer Aiding:perform **Magnetometer Calibration,** check for time varying magnetic interference via **Scaled Mag (0x80,0x06)**.
4	Position Warning	Multipath, EMI, incorrect antenna offsets, antennas obstructed, antenna cable shorted. Review **How to improve filter performance** above.	Review: **Antenna** offsets, FAQ: Antenna sections, **FAQ:How to improve filter performance.**
5	Velocity Warning		

Bit #	Description	Potential Cause	Recommended Action
6	IMU Bias Warning	Gyro bias is high.	Perform Capture Gyro Bias (0x0C,0x39)
8	Antenna Lever Arm Warning	Antenna Lever Arm offsets are likely incorrect.	Not critical concern if LLH Position Uncertainty (0x82,0x08) has low uncertainty and Fix Info (0x94,0x0B) has a sufficient fix. To improve performance, review Antenna(s) page.
9	Mounting Transform Warning	Transform is likely incorrect.	Review Vehicle Frame page.
10	Time Sync Warning	No PPS source detected, or external heading timestamp is too old.	Check PPS source is set correctly .
12-15	Solution Error	Filter computation warning flags. If any bits 12-15 are set, all filter outputs will be invalid.	Reset Navigation Filter (0x0D,0x01)

Filter condition (bits 0-1)

Regardless of the active filter mode (see Filter Status), the Navigation Filter will report one of two Navigation Filter conditions in the status message: **stable**, and **converging**.

Filter Condition Values	Description
1	Stable
2	Converging

Stable

The Navigation Filter reports that it is stable when the error state estimates are generally consistent with the available aiding measurements and state uncertainty estimates stabilize.

Converging

The Navigation Filter reports that it is **converging** when the error state estimates have not yet converged to stable values. The duration of the convergence condition is dependent on the filter mode (see Filter Status), the quality and frequency of aiding measurements, and the dynamics of the sensor platform. Typical convergence times can be several minutes after filter reset for the Navigation Filter to reach Full Navigation mode.

NOTE: After initialization, the filter will report a **converging** condition until state estimates have stabilized. Then a **stable** condition is reported. If the availability of aiding measurements changes, or the filter mode changes (see **Filter Status**), the Navigation Filter may revert from **stable** to **converging**.

Estimate warnings (bits 2-9)

An estimate warning means that the Navigation Filter's confidence in the accuracy of the flagged estimate is low. The following table details the outputs that are impacted by each estimate warning flag.

Estimate Warning	Impacted Outputs
Roll/Pitch Warning	Attitude DCM (0x82,0x04), Attitude Quaternion (0x82,0x03), Euler Angles (0x82,0x05)
Heading Warning	Attitude DCM (0x82,0x04), Attitude Quaternion (0x82,0x03), Euler Angles (0x82,0x05)
Position Warning	LLH Position (0x82,0x01), ECEF Position (0x82,0x40), NED Relative Position (0x82,0x42)
Velocity Warning	NED Velocity (0x82,0x02), ECEF Velocity (0x82,0x41)
IMU Bias Warning	Accel Bias (0x82,0x07), Gyro Bias (0x82,0x06)
GNSS Clock Warning	Filter Timestamp (0x82,0x11), Clock Correction (0x82,0x32)
Antenna Lever Arm Warning	Multi Antenna Offset Correction (0x82,0x34)
Mounting Transform Warning	-

Solution error (bits 12-15)

An error flagged by bits 12-15 indicate that there is a non-numerical value (NaN) in the navigation solution. If bits 12-15 are non-zero, the user is advised to reset the Navigation Filter (**Reset Navigation Filter (0x0D,0x01)**).

Magnetometer Warning

The on-board magnetometer is provided as a convenient means to determine a heading estimate . To accurately infer sensor orientation based on a magnetic field measurement requires careful design with respect to sensor installation. Calibration is usually required post-installation, and may be difficult or impossible to achieve.

Before using the magnetometer as a heading source, it must be calibrated. Refer to the **Magnetometer Calibration** page for more details.

The 3DM-CV7 supports external magnetometer aiding in case an external magnetometer can be located on the vehicle in a more benign magnetic environment (e.g. on a mast above the vehicle.) Note that the data provided from the external magnetometer is assumed to be calibrated. Reference the **External Aiding Sensors** page for instructions on using an external magnetometer.

Filter Aiding Measurements

Internal aiding measurements are enabled and disabled by the **Aiding Measurement Control (0x0D,0x50)** command. External aiding measurements are input to the device via the **Aiding Descriptor Set (0x13)** commands.

Internal Aiding Sensors

Magnetometer Aiding

A measurement of the local magnetic field in conjunction with a location dependent model of Earth's magnetic field can be used for heading aiding. Special care must be taken when using this method. See **Magnetometer Warning** for details. This aiding measurement can only be sourced internally.

Altimeter Aiding

Altimeter aiding can help stabilize altitude in challenging GNSS environments. It can be sourced internally through the 3DM-CV7's pressure altimeter or through the external aiding command described below.

External Aiding Measurements

The 3DM-CV7-GNSS/INS has been designed to allow users to easily configure and provide external aiding measurements that improve state estimation in challenging GNSS conditions.

Note: Different aiding commands provide the same data in different reference frames (e.g. ECEF vs NED velocity.) Only one frame should be used at a time.

Tip: For best practices, installation, and recommendations on specific aiding sensors that have been field tested with the 3DM-CV7-GNSS/INS, refer to the **External Aiding Sensors** page.

Aiding Frame Configuration

The **Aiding Frame Configuration (0x13,0x01)** command enables the user to mount an aiding sensor at a specific location on the vehicle and to specify the parameters of that frame. This includes an offset vector and a rotation with-respect-to the vehicle frame. See the **Aiding Frames** page for more information.

- Enables user to configure relative translation and rotation for external aiding sensors
- These frames account for the effect of vehicle dynamics on external aiding measurements

Aiding Command Echo Control

The **Aiding Command Echo Control (0x13,0x1F)** enables the user to control whether aiding commands are echoed back out of the 3DM-CV7-GNSS/INS. This is useful for observing aiding measurements and their effects on the 3DM-CV7-GNSS/INS filter output.

This is required for post processing support in our desktop simulator and is useful for troubleshooting.

ECEF Position

Use the **ECEF Position (0x13,0x21)** aiding command to provide a position measurement with-respect-to the Earth-Centered, Earth-Fixed (ECEF) frame. See **Coordinate Frames** for definitions.

LLH Position

Use the **LLH Position (0x13,0x22)** aiding command to provide a position measurement with-respect-to the WGS84 ellipsoid using Geodetic coordinates (latitude, longitude, height.)
Note: Uncertainty must be in North/East/Down (NED) components.

This command can be used to input an absolute height measurement by setting the uncertainties or valid flags to 0 for Latitude and Longitude. As a reminder, this height is referenced to the

WGS84 ellipsoid, NOT height above ground.

ECEF Velocity

Use the **ECEF Velocity (0x13,0x28)** aiding command to provide a velocity measurement with-respect-to the ECEF frame. This aiding measurement should not be used to supply a velocity from a body-fixed sensor, e.g. radar-derived velocity. For this type of sensor, use the **Body Frame Velocity (0x13,0x2A)** aiding command instead.

NED Velocity

Use the **NED Velocity (0x13,0x29)** aiding command to provide a velocity measurement in the North/East/Down (NED) local-level frame. This aiding measurement should not be used to supply a velocity from a body-fixed sensor, e.g. radar-derived velocity. For this type of sensor, use the **Body Frame Velocity (0x13,0x2A)** aiding command instead.

Body Frame Velocity

Use the **Body Frame Velocity (0x13,0x2A)** aiding command to provide a velocity measurement in a frame that is fixed relative to the vehicle. This enables the 3DM-CV7-GNSS/INS to constrain its drift during periods of GNSS outage. The velocity must be provided as a pre-processed, 3-element vector. The 3DM-CV7-GNSS/INS allows for up to 4 velocity sensor inputs of this type.

Examples of aiding sensors used to populate this aiding command are pre-processed outputs from radar, LiDAR, wheel speed measurement, wind-corrected air speed, etc.

Note: If an axis is marked as valid, but uncertainty is set as 0 or negative, an invalid parameter response will be issued

Ex: Body Velocity with height unavailable

Description	**Ex: Velocity in the body frame, height unavailable**			
Notes				
Parameter Name	**Data Type**	**Description**	Value	Hex

Description	Ex: Velocity in the body frame, height unavailable			
Field Length	*u8*	*5*		
Descriptor	*u8*	*0x2A*		
Time	Time	Timestamp of the measurement.	158000	
Frame Id	u8	Frame ID for this measurement (must be > 0)	2	
Velocity	vector3f	[m/s]	3.745	
Uncertainty	vector3f	[m/s] 1-sigma uncertainty	0.120	
Valid Flags	u16 bitfield	Name / Bit / Description X / 0 / X-axis valid flag Y / 1 / Y-axis valid flag Z / 2 / Z-axis valid flag	1;1;0;	
Ack/Nack Reply	See standard MIP ack/nack reply format.			

True Heading

External heading aiding helps to maintain accurate vehicle heading estimates, especially during quasi-stationary periods with minimal linear acceleration. Use the **True Heading (0x13,0x3A)** aiding command to provide a heading measurements with-respect-to true north. This is commonly used to supply dual antenna / GNSS compass heading measurements.

External Magnetometer

Use the External **Magnetic Field (0x13,0x32)** aiding command to provide a magnetic field measurement from an external magnetometer. Note: this measurement is assumed to come from a calibrated source. A properly calibrated and thoughtfully placed magnetometer can be a good heading measurement for applications that experience GNSS denied environments. In the case where the 3DM-CV7-GNSS/INS cannot be mounted away from magnetic interference, this command provides a method for inputting magnetic field measurements from a better situated magnetometer.

Ex: External Magnetometer with all fields populated:

Description	Ex: External Magnetometer aiding measurement		
Notes			

Description	Ex: External Magnetometer aiding measurement			
Parameter Name	**Data Type**	**Description**	Value	Hex
Field Length	*u8*	*5*		
Descriptor	*u8*	*0x32*		
Time	Time	Timestamp of the measurement.	158000	
Frame Id	u8	Frame ID for this measurement (must be > 0)	3	
Magnetic Field	vector3f	[Gauss]	0.451 0.345 0.401	
Uncertainty	vector3f	[Gauss] 1-sigma uncertainty.	0.031	
Valid Flags	u16 bitfield	Name Bit(s) Description X 0 X-axis valid flag Y 1 Y-axis valid flag Z 2 Z-axis valid flag	1 1 1	
Ack/Nack Reply	See standard MIP ack/nack reply format.			

External Pressure

Use the External **Pressure (0x13,0x33)** aiding command to provide a static pressure measurement. In the case where the 3DM-CV7-GNSS/INS cannot be mounted where static pressure can be reliably measured, this command provides a method for inputting static pressure measurements from a better situated pressure sensor. Static pressure is used to constrain altitude drift during GNSS outages.

GNSS/RTK

The 3DM-CV7 currently supports GNSS from multiple satellite constellations. See **GNSS** for signal configuration options.

RTK corrections, or Real-Time Kinematic, enhances the accuracy of GNSS position estimates by providing real-time error corrections. For network based RTK solutions, sending RTCM3.X is easy with the SensorConnect NTRIP Client. If you prefer an L-band correction solution, the 3DM-CV7-GNSS/INS can accept SPARTN corrections natively via an L-band correction module.

GNSS

The 3DM-CV7 allows users to configure which GNSS constellations and frequency bands are used via the **MIP GNSS Signal Configuration (0x0E,0x02)** command.

Only certain combinations of constellation configurations are valid. When configuring the device with InertialConnect, configurations will be flagged if they are invalid.

When configuring with the **MIP GNSS Signal Configuration (0x0E,0x02)** command, the following conditions must be met:

1. The device must have a main constellation enabled: either GPS or Galileo.
2. The device must not have just GPS L1 and Beidou L1 enabled only.
3. The following combinations for each constellation are valid:
    1. GPS: none / L1-only / L1 + L5.
    2. Glonass: none / L1-only.
    3. Galileo: none / L1 + L5.
    4. Beidou: none / L1-only / L1 + L5.

SPARTN Corrections

The 3DM-CV7-GNSS/INS is able to accept SPARTN GNSS corrections natively via NTRIP or via L-band with the use of a L-band correction receiver mod like the Ublox D9S.

SPARTN corrections are configured with the SPARTN Configuration (0x0E,0x20) command.

For instructions on how to configure your 3DM-CV7-GNSS/INS to accept a SPARTN input via UART, and how to configure a D9S, see this example.

NOTE: Enabling SPARTN corrections will automatically block RTCM corrections.

RTCM3 Corrections

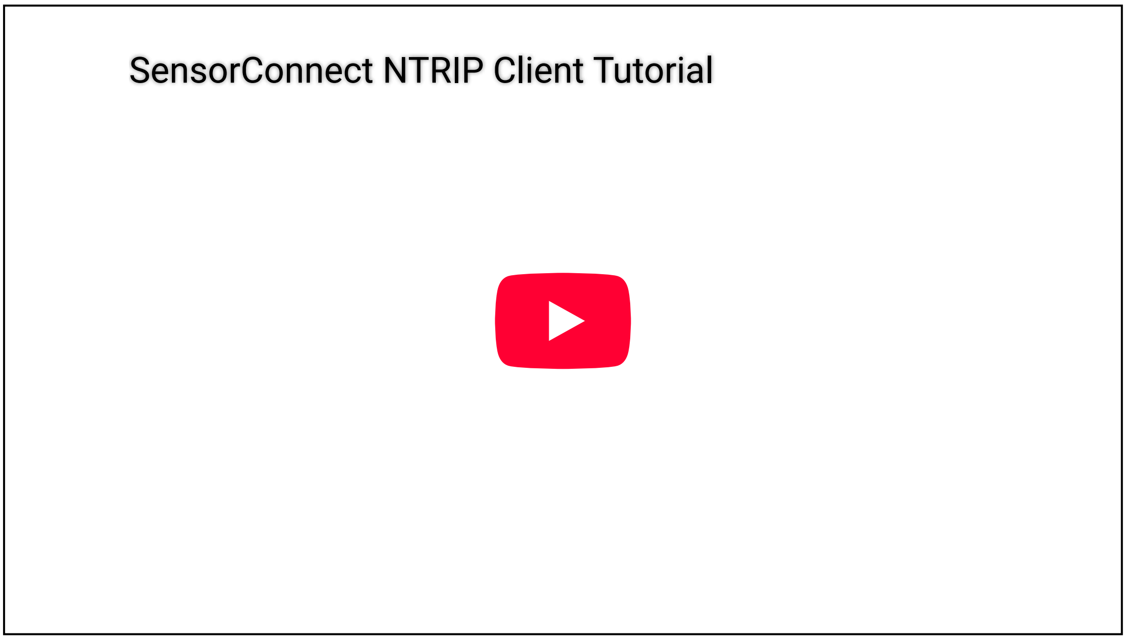
The 3DM-CV7-GNSS/INS is able to accept RTCM3.X corrections natively to any of it's ports, once configured using the Interface Control (0x7F, 0x02) command. Supported RTCM messages are listed in the table below.

3DM-CV7 Supported RTCM3 Messages		
Group Name	**Sub-Group Name**	**Message Type**
Observations	GPS MSMs	1074, 1075, 1077
	GLONASS MSMs	1084, 1085, 1087
	Galileo MSMs	1094, 1095, 1097
	Beidou MSMs	1124, 1025, 1127
Station Coordinates (Antenna)		1005, 1006
Antenna Description		1007
Antenna and Receiver Description		1033
GLONASS Bias Information		1230

A configuration example for sending RTCM data to the 3DM-CV7-GNSS/INS with the SensorConnect NTRIP client can be found here. For more information about the ports of the 3DM-CV7-GNSS/INS, please see the Ports page.

SensorConnect NTRIP Client

SensorConnect has an integrated NTRIP client to make streaming RTCM correction data easy. The following tutorial video demonstrates the steps required to get started with the NTRIP client.



Additional Features

Complementary Filter

In addition to attitude estimates provided by the primary **Navigation Filter**, the 3DM-CV7 can provide attitude estimates produced by a separate Complementary Filter which uses scaled accelerometer and gyroscope data to estimate pitch and roll and allows the option of using the magnetometer data to correct the heading estimate.

The Complementary Filter is completely independent and runs in parallel with the Navigation Filter. Attitude estimates are computed by blending the integrated gyro output with pitch and roll estimates from the accelerometer data, using the assumption that linear acceleration is on average equal to zero.

The user can independently enable or disable the pitch/roll correction and the magnetometer heading correction, as well as independently set the time constant for each correction using the **Complementary Filter Settings (0x0C,0x51)** command. If the magnetometer is enabled, the magnetic field measurement is used to correct heading as well.

The primary strength of the Complementary Filter is its robustness: the estimates will never diverge, even after severe sensor over-ranging. However, it is more sensitive to transient errors caused by linear acceleration and magnetic disturbances. Furthermore, the Complementary Filter does not correct for accelerometer and gyro biases which both impart an angular offset in the attitude estimate. The Complementary Filter is best used as a secondary, partially independent source of attitude estimates that can be used when the **Navigation Filter** is in **Unstable mode**.

LED States

The LED on the 3DM-CV7 uses both color and blink pattern to convey information about the operating state of the system. Users can get information on the GNSS system, RTK corrections,filter mode, data streaming mode and more through the LED interface.

To better understand the Navigation Filter modes described by the LED, see **Filter States**.

LED Color	System State
Blue	Filter is in Full Navigation mode and processing differential corrections
Green	Filter is in Full Navigation mode
Yellow	Filter is not in Full Navigation mode (Initialization, Vertical Gyro, or AHRS mode)
Red	Continuous BIT error.* See Built-in Tests for full list of errors.
Solid White	System is initializing
Violet	Non-standard operating mode

*Only the first 32-bits of the continuous BIT cause a red LED. The most common error conditions are buffer overruns due to streaming too much data for the selected baud rate.

LED Blink Pattern	System State
Slow Pulse	Idle
Fast Blink	Transmitting data or command responses
White Flash	Blinks in sync with the PPS when it's available
White/Blue Flash	PPS and receiving RTCM or SPARTN correction data

You are here: Additional Features > PPS

PPS

WARNING: The filter needs a valid PPS to initialize and to process measurements. In almost all cases, the DISABLED and GENERATED options in the **Source** field of the PPS Source (0x0C,0x28) command should not be used. Additionally, GNSS antennas, if used, need to have a clear view of the sky and be receiving GNSS signals for a valid PPS. Temporary GNSS outages can be tolerated.

The pulse-per-second (PPS) is a timing signal that can be used to synchronize the 3DM-CV7 with other systems.

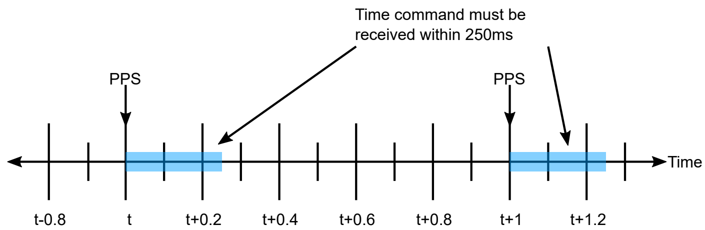
The 3DM-CV7 can accept an external PPS or generate one as an output derived from the internal clock. Both input and output signals use a configurable digital control line available through the GPIO pins .

IMPORTANT: the PPS source and the assigned GPIO pin are separate configurations.

The PPS time sync status can be monitored by using the **Time Sync Status (0xA0,0x02)** data descriptor. This message tells the user when a PPS lock has been achieved.

Once the GPIO and PPS source have been configured, the user can set the whole number of seconds using the **Time Broadcast Command (0x01,0x72)**. The user must send a separate instance of this command to set the week number. See shared **GPS Timestamp (0xFF,0xD3)** for more information.

As shown in the figure below, this must be done within 250 milliseconds after the PPS line goes high. This command can be sent every second, or just once at initialization.



NOTE: If the time command is sent only once, the time will be kept synchronized by the PPS input. If the PPS signal stops, the 3DM-CV7 will continue tracking time using its internal clock. In this case, accuracy will be degraded.

IMPORTANT: If a glitch occurs on the PPS line, the time command should be sent again to ensure seconds were not double-counted or lost.

To receive data timestamps referenced to the PPS (whether input or output), use the following messages from the **Shared Data (0xFF)** set: **GPS Timestamp (0xFF,0xD3)**, **Delta Time (0xFF,0xD4)** , **External Timestamp (0xFF,0xD7)**, **External Time (0xFF,0xD8)**.

TIP: Time synchronization via command or data packets is not very accurate due to transmission and processing delays. Typical delays might be on the order of a few to tens of milliseconds, or even longer. The 3DM-CV7's PPS offers a very precise, very low-latency alternative.

Outputting a PPS

When configured to provide a PPS output, the 3DM-CV7 will generate a square wave pulse with a 50% duty cycle and a period that defines one second. The rising edge of the pulse signals the beginning of a new second.

Use the **PPS Source (0x0C,0x28)** command to set the PPS source. To generate a PPS from the 3DM-CV7's internal clock, set **source** to **generated**.

The 3DM-CV7's GPIOs must be configured to provide a PPS through the **GPIO Configuration (0x0C,0x41)**. See **User GPIO** for a list of GPIOs that support PPS output and instructions on GPIO configuration.

Inputting a PPS

When configured to receive a PPS input, the user should provide a square wave signal with a period that defines one second and whose rising edge signals the start of a new second.

Set the PPS **source** to **GPIO** using the **PPS Source (0x0C,0x28)**. Configure the appropriate GPIO pin as a PPS input using the **GPIO Configuration (0x0C,0x41)** command. See **User GPIO** for more details.

NMEA Output

The 3DM-CV7 supports transmission of selected NMEA-0183 sentences via all communication ports once enabled. See the Ports page for more information.

Supported Sentences and Talker IDs

Sentence	Description	Supported Talker IDs	Applicable Sources
GGA	GNSS Fix Information	GN, GP, GA, GL	GNSS, Filter
GLL	Lat/lon position	GN, GP, GA, GL	GNSS, Filter
GSV	Satellites in View	Constellation-specific	GNSS
RMC	Recommended Minimum GNSS data	GN, GP, GA, GL	GNSS, Filter
VTG	Course over Ground	GN, GP, GA, GL	GNSS, Filter
HDT	True Heading	GN, GP, GA, GL	GNSS, Filter
ZDA	Time and Date	GN, GP, GA, GL	GNSS
GST	GNSS Error Statistics	GN, GP, GA, GL	GNSS, Filter
PMSRA	Euler Angles (Proprietary)	Proprietary	Filter
PMSRR	Angular Rate and Accel (Proprietary)	Proprietary	IMU

Proprietary Messages

PMSRA - Euler Angles

Information	Format	Examples
UTC Time - Hour (24-hour format, 00 = midnight)	2 digits, HH	00, 01, 13, 23
UTC Time - Minute, 00-59	2 digits, MM	00, 15, 30, 59
UTC Time - Second, 00-60	2 digits, SS	00, 05, 10, 55, 60
UTC Time - Millisecond, .00 to .99	2 digits after a decimal, .mm	.00, .10, .67, .99
Euler Angles - Roll, Pitch, Yaw [degrees]	Float with 3 decimal places	0.523, -23.887, 149.447

Example: `$PMSRA,221336.93,-0.172,-0.681,3.072*5D`

PMSRR - Angular Rate and Acceleration

Information	Format	Examples

UTC Time - Hour (24-hour format, 00 = midnight)	2 digits, HH	00, 01, 13, 23
UTC Time - Minute, 00-59	2 digits, MM	00, 15, 30, 59
UTC Time - Second, 00-60	2 digits, SS	00, 05, 10, 55, 60
UTC Time - Millisecond, .00 to .99	2 digits after a decimal, .mm	.00, .10, .67, .99
Acceleration - X, Y, Z [g]	Float with 6 decimal places	0.123456, 1.102300, -5.52223
Gyro - Roll, Pitch, Yaw [deg/s]	Float with 6 decimal places	0.123456, 1.102300, -5.52223

Example:

```
$PMSRR,221336.93,-0.012996,0.004631,-1.000471,0.280331,0.130576,0.207637*4D
```

Sources and Talker IDs

The NMEA sentences can be produced using data from the IMU, Filter, GNSS 1, or GNSS 2. The source is selected on a per-sentence basis using the corresponding MIP descriptor set. The talker IDs are also configurable for each sentence except for the GSV and proprietary messages.

Configuration

The NMEA system is configured using the **NMEA Message Format (0x0C,0x0C)** MIP command. It accepts a list of sentence types which includes the talker IDs, source descriptor sets, and decimations for each sentence. A sentence may be specified more than once using different talker IDs or sources. The decimation is relative to the base rate of the source descriptor set.

Important: due to processing limitations, a maximum of 10 Hz applies to each produced NMEA sentence from the IMU and filter descriptor sources and 2 Hz from a GNSS descriptor source. Attempts to set a higher rate than this will result in the command being NACK'd.

The device will transmit NMEA sentences if all of the following conditions are met:

- At least one sentence is specified in the NMEA message format.
- The device has not been set to idle with the **Set to idle (0x01,0x02)** command (issue the **Resume (0x01,0x06)** command if this is the case).
- The data required for the configured sentence(s) is available. In some cases, the required data may not be valid, such as when a GNSS fix has not been obtained. A sentence will not be transmitted unless the source data is valid.

If the NMEA sentence list is not empty on startup, transmission will begin as soon as the data becomes available (e.g. once satellite signals have been acquired).

NMEA sentences can also be polled manually using the **Poll NMEA Message (0x0C,0x04)** MIP command. This command behaves like the NMEA message format command except that messages will only be transmitted once, if the data is available.

Parsing mixed MIP and NMEA data streams

Because each port has the ability to stream both MIP and NMEA data, MIP command replies and data will be mixed into the data stream if any MIP communication is enabled or initiated. Applications requiring mixed NMEA and MIP should deal with this by running parsers in parallel; that is, by feeding the data stream to both parsers simultaneously. The two parsers must be robust and reject unexpected data for this to work correctly. Each type of packet will be transmitted whole; therefore, the user should not expect a NMEA packet to fall in the middle of a MIP packet and vice-versa.

NMEA Input

To simplify integration of many different GNSS receivers with the 3DM-CV7-GNSS/INS, in addition to the MIP external **Aiding (0x13)** commands, the 3DM-CV7-GNSS/INS accepts standard NMEA-0183 protocol sentences directly via the Main port, USB2 port, or any GPIO configured Aux port. All options are described below.

Supported NMEA Sentences

The following sentences are supported. The talker ID is ignored.

- RMC - Recommended Minimum Specific GNSS Data
- ZDA - Time and Date
- HDT - Heading, True
- GGA - Global Positioning System Fix Data
- VTG - Course Over Ground and Ground Speed
- GLL - Geographic Position Latitude / Longitude
- GSV - GNSS Satellites in View

External Receiver Data

NMEA data from an external GNSS receiver can be streamed out from the 3DM-CV7-GNSS/INS in the MIP **GNSS External (0x94)** descriptor set. This data will mirror the contents of the NMEA data as closely as possible, given the differences in available information. Some MIP fields require data from more than one NMEA sentence to be valid, and some contain data not available in NMEA. If the associated NMEA data is not available, the MIP parameter will have the corresponding valid flag set to 0.

Timing

Latency

For NMEA data to be processed, it must be received with a timestamp that is different from the previous sentence. This means there is a delay of *T* seconds between message reception and corresponding MIP data output, where *T* is the period between NMEA updates, plus some processing time. At least two NMEA updates must have been received for the corresponding MIP data to be transmitted on the **GNSS External (0x94)** descriptor set.

Timestamps

Timestamps emitted from the device via the MIP **GNSS External (0x94)** descriptor set will match those from the corresponding NMEA sentences.

Port Configuration

NMEA Input via the Main Port

Without any configuration, NMEA data can be input via the Main port (either USB or serial, but not both simultaneously) by interleaving NMEA sentences between any MIP command packets. The device will separate the two protocols as long as the packets are not fragmented. It is recommended that a single application be responsible for communicating with the 3DM-CV7-GNSS/INS over the Main port, rather than two programs which could interfere with each other while trying to send data at the same time (e.g. MIP commands and NMEA).

MIP and NMEA packets may be interleaved, but must not be fragmented: `[7565...] [$GNGGA,...*XX][7565...][$GNGLL,...*XX][$GNZDA,...*XX] ...`

NMEA Input via the Aux Port

In some applications, it may be advantageous to connect an external GNSS receiver or other equipment directly to the 3DM-CV7-GNSS/INS. To do this, one of the **User GPIO** pins must be configured as a UART port using the **GPIO Configuration (0x0C,0x41)** command. This is easily configured in **SensorConnect's** Configure tile, or via MIP API commands. Instructions for both interfaces are provided below:

Configure the 3DM-CV7-GNSS/INS GPIO pins:

SensorConnect:

1. Plug the 3DM-CV7-GNSS/INS into a computer running SensorConnect

2. Enter the Configure tile
    1. Configure the GNSS Antenna Offset:
        1. In the Mounting Setup section, enter the GNSS antenna offset in the NMEA Frame Offset field. Be sure to measure the antenna offset in the **Vehicle Frame**, even if a Sensor to Vehicle Frame Transformation is applied to the 3DM-CV7-GNSS/INS.
    2. Specify the GPIO pin for PPS input:
        1. Scroll down to the PPS Configuration section
            1. Click the Source drop-down and select GPIO
            2. In the GPIO Configuration section, select the desired GPIO pin to receive PPS input
                1. Click the Feature drop-down and select PPS
                2. Click the Behavior drop-down and select Input
    3. Specify the GPIO pin for NMEA Input:
        1. In the same GPIO section, select the desired GPIO pin to receive NMEA input data
            1. Click the Feature drop-down and select NMEA (UART)
            2. Click the Behavior drop-down and select Receive
                1. It is not required to select Pullup, Pulldown, or Open drain for options

MIP API:

1. Configure the GNSS Antenna Offset:
    1. Be sure to measure the antenna offset in the **Vehicle Frame**, even if a Sensor to Vehicle Frame Transformation is applied to the 3DM-CV7-GNSS/INS.
    2. Input the GNSS Antenna Offset using the **GNSS Offset Control (0x0D,0x13)** command.

        1. **IMPORTANT**: This command should only be used for GNSS data via NMEA input, not MIP API input.

2. Configure the desired GPIO pin for UART input:
    1. Using the **GPIO Configuration (0x0C,0x41)** command:
        1. Specify the GPIO pin desired
        2. Select UART (5) as the feature
        3. Select and UART_PORT2_RX (0x22) or UART_PORT3_RX (0x32) as the behavior.
            1. More information on Port definitions and supported functionality can be found on the **Ports** page.
        4. Select Pin Mode as open drain (0)
    2. The baud rate of the 3DM-CV7-GNSS/INS can be set using the **Comm Port Speed (0x01,0x09)** command with port=2.
        1. NOTE: The baud rate of the selected GPIO pin on the 3DM-CV7-GNSS/INS must match the baud rate of the UART output of the external GNSS receiver.
    3. NOTE: Once an Aux port pin is configured, NMEA data will **ONLY** be accepted on the Aux port instead of the Main port.

Configure Filter Initialization Conditions

WARNING: A known bug exists for NMEA input. See **Errata Sheet**

SensorConnect:

1. Specify **Filter Initialization** Conditions:
    1. Enter the Configure tile
        1. If the user wishes to initialize position and velocity from the external GNSS receiver solution, they should enable Auto-Initialize Filter and send GGA and optionally VTG messages from the receiver.

2. If the user wishes to have the device Auto-Initialize Heading, enable that mode and select one of the following options:
    1. **GNSS Kinematic:** The VTG message is used as the GNSS Velocity Vector to initialize heading.
        1. If a VTG sentence is not provided, the 3DM-CV7-GNSS/INS will use successive position measurements to approximate an initial velocity.
        2. NOTE: Special care must be taken with this initialization mode as described on the the **Filter Initialization** page.
    2. **Magnetometer Heading:** The on-board magnetometer is used to initialize heading
    3. **External Heading:** the HDT sentence is used to initialize heading **External Heading**

2. Configure the Filter Aiding Measurement sources to include at a minimum for NMEA Input:
    1. GNSS Position and Velocity Aiding: GNSS_POS_VEL
    2. If using the HDT message, configure External Heading: EXTERNAL_HEADING
    3. NOTE: Additional **Filter Aiding Measurements** can be used when providing NMEA Input but are not required.

MIP API:

1. Specify **Filter Initialization** Conditions:
    1. If the user wishes to initialize position and velocity from the external GNSS receiver solution, they should enable an Auto-Initialization option for position and velocity with **Navigation Filter Initialization (0x0D,0x52)** command and then provide (GGA or GLL) and optionally a VTG sentence.
    2. If a VTG sentence is not provided, the 3DM-CV7-GNSS/INS will use successive position measurements to approximate an initial velocity.
    3. If the user wishes to initialize the heading using **Kinematic Alignment**, this option should be enabled using the **Navigation Filter Initialization (0x0D,0x52)** command.

4. If the user wishes to initialize heading automatically from the HDT sentence, they should enable the **External Heading** initialization option using the **Navigation Filter Initialization (0x0D,0x52)** command then provide the HDT sentence.

2. Configure the Filter Aiding Measurement sources using the **Aiding Measurement Control (0x0D,0x50)** command to include at a minimum for NMEA Input:

   1. GNSS Position and Velocity Aiding: GNSS_POS_VEL

   2. If using the HDT message, configure External Heading: EXTERNAL_HEADING

   3. NOTE: Additional **Filter Aiding Measurements** can be used when providing NMEA Input but are not required.

Save Settings to Non-Volatile Memory:

If you want this configuration to be persistent through power cycles, or if the device will be power-cycled before connecting to the external GNSS receiver, it is recommended to "save as startup settings" before removing power. This can be performed using the **Device Settings (0x0C,0x30)** command with the SAVE function selector. For instructions using other interface methods, refer to the **FAQ: Do settings get erased from the** 3DM-CV7-GNSS/INS **if I unplug it?**

Configure the External GNSS Receiver

The **NMEA Input with u-blox ZED-F9P** application note provides detailed instructions on how to easily configure the popular GNSS receiver, to provide these messages.

Configure the external GNSS receiver to output the desired NMEA messages.

The following NMEA messages are recommended to provide sufficient GNSS data to the 3DM-CV7-GNSS/INS:

1. ZDA - Sets the time on the 3DM-CV7-GNSS/INS to UTC time in the shared **GPS Timestamp (0xFF,0xD3)**. If GPS time is desired, a manual update of the TOW and Week Number can be provided via the **Time Broadcast Command (0x01,0x72)** as described on the **PPS** page.

2. GGA - Provides Lat, Lon, Height and GNSS Fix data with timestamp

    3. GSV - Provides GNSS sky view information

    4. VTG - Provides GNSS velocity

    5. HDT - (optional) Provides True Heading in case External Heading is used for Navigation Filter Heading Alignment Method (see [Filter Initialization](#))

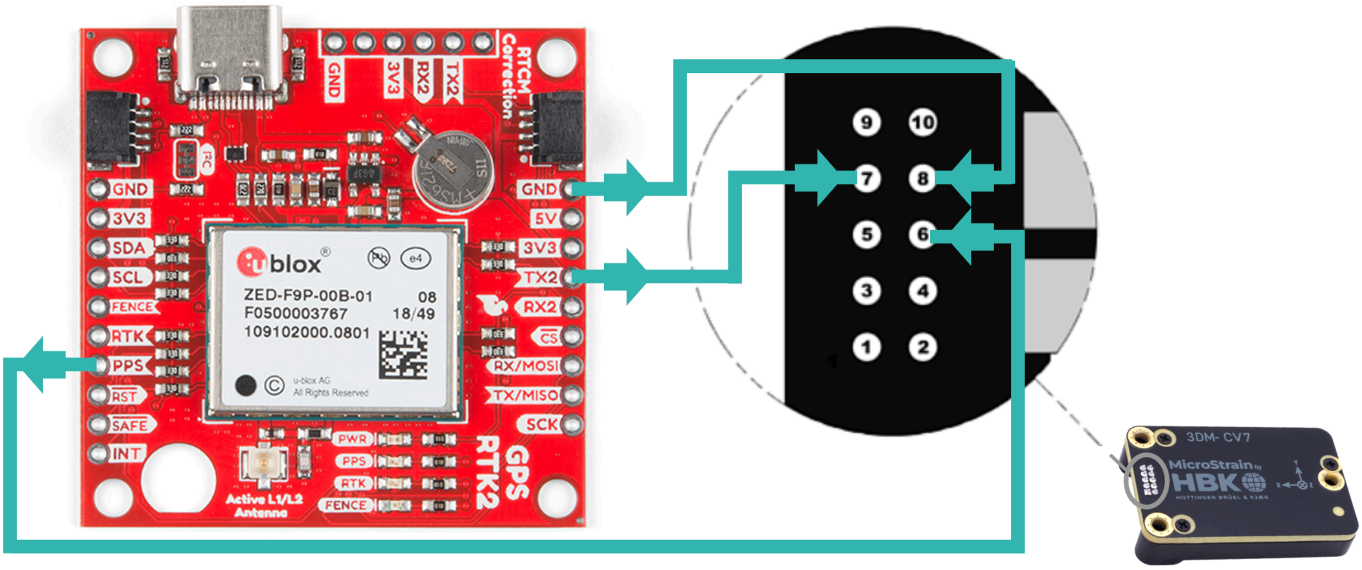
        1. Note: Dual Antenna is required to compute an HDT message

2. Save the settings of the receiver to its non-volatile memory.

Connect External GNSS Receiver to 3DM-CV7-GNSS/INS

1. See [User GPIO](#) and [Electrical](#) pages for relevant specifications.

2. Follow the wiring schematic below:



Monitor the Navigation Filter Output

1. Monitor the Navigation Filter health check data channels:

2. Reference: [FAQ: How do I know when I can trust the Navigation Filter solution?](#) and ensure the Navigation Filter has achieved: Full Navigation [Filter State](#), Converged [Filter Status Flags](#), and the [LLH Position Uncertainty (0x82,0x08)](#) desired.

    1. Additionally, ensure [Aiding Measurement Summary](#) Indicates some of the following measurements are enabled, used, and ideally not reporting any errors (2-5):

        - GNSS

- POS_ECEF / POS_LLH
- VEL_ECEF / VEL_NED
- HEADING_TRUE (for HDT)

1. When the correct Initialization Conditions are provided, the Navigation Filter will typically initialize in less than 5 seconds. If the 3DM-CV7-GNSS/INS hasn't converged in 1 minute, review the **FAQ: Why is the filter stuck in Vertical Gyro Mode?**.

Note: GPIO pins use TTL level signals. Please see the **Electrical** specifications for details. To use it with an RS232 signal, an external level shifter must be used.

If a GPIO port pin is configured for NMEA, NMEA will only be accepted on the GPIO port; **NMEA input to the Main port will be ignored while the GPIO port is configured.**

You are here: Additional Features > User GPIO

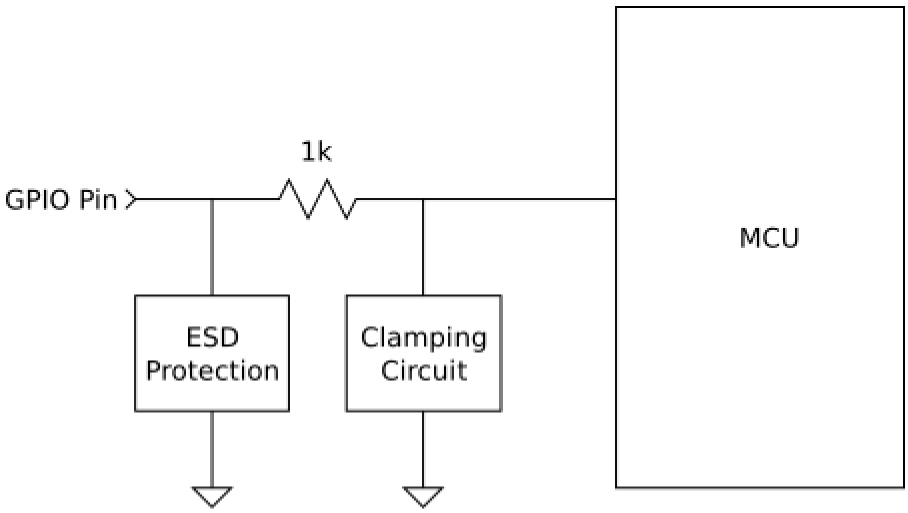
User GPIO

The 3DM-CV7-GNSS/INS supports user-configurable GPIO pins which can be used for a variety of purposes as outlined below. The following table lists the features supported by each pin:

Logical Pin Number	Physical Connector Pin	Supported Features (X=supported)				Default Setting
		General Purpose I/O	PPS	Events	UART	
1	Pin 7	X	X	X	X	UNUSED
2	Pin 9	X	*	X	X	UNUSED
3	Pin 6	X	*	X	X	UNUSED
4	Pin 10	X	*	X	X	UNUSED

See **Electrical** for details on GPIO pin specifications.

The GPIO operate in the 0 - 3 volt range but will accept input voltages up to 5.5 volts without damage. The input impedance of the GPIO may decrease as the voltage exceeds 3 volts due to an internal clamping circuit.



GPIO Features

Unused

The unused mode is recommended when a GPIO pin will not be connected. This helps reduce EMI issues and power consumption.

General Input or Output ("GPIO" feature)

A pin can be used as a sampled input or as a general-purpose control output. The state of the pin can be controlled via the **GPIO Configuration (0x0C,0x41** command. The state of a pin can be queried with the **GPIO State (0x0C,0x42)** command.

Pulse-per-Second (PPS)

The pulse-per-second (PPS) input or output can be used to synchronize time between the 3DM-CV7 and another system to achieve sub-microsecond synchronization. For full instructions on inputting or outputting a PPS, see **PPS**.

Event System

The GPIO can be used with the **Event System** for various purposes, such as time synchronization. See **Action: GPIO** and **Trigger: GPIO** for more details. When using the GPIO Trigger in edge mode, the GPIO feature should be set to the TIMESTAMP feature. Otherwise, set it to the GPIO feature for use with the event system.

UART

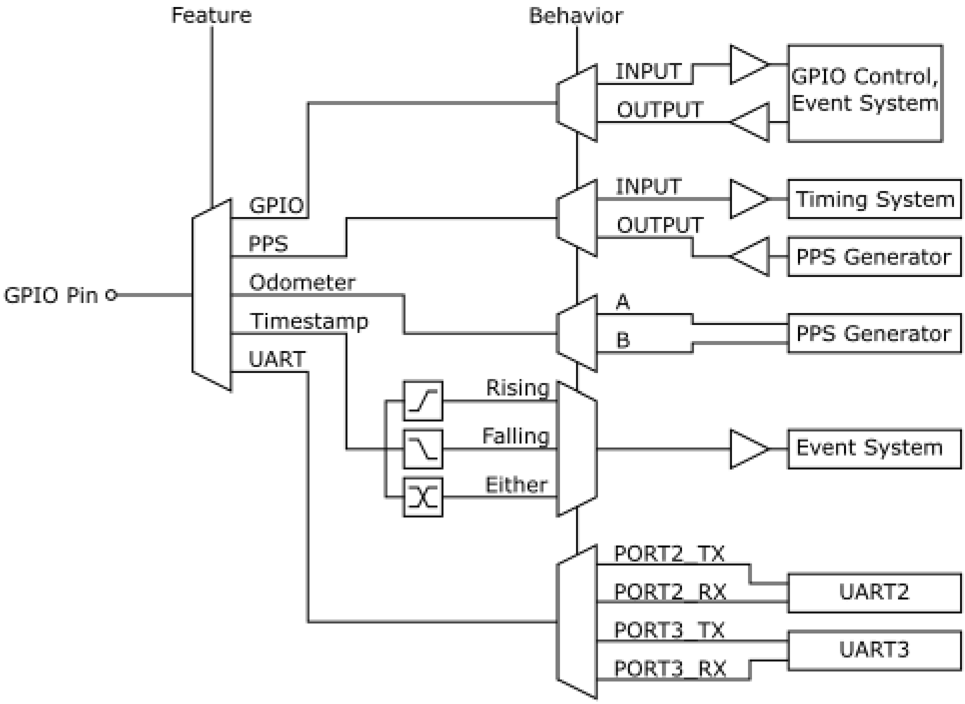
The 3DM-CV7 supports additional UART connections which may be used for aiding data (e.g. NMEA sentences, aiding data via the (0x13) command set, or RTK corrections) or as an additional data output channel. Each pin maps to a specific UART. Any UART-enabled pin may operate as either RX or TX mode. No two pins can be assigned the same RX/TX function for the same UART. For protocol configuration, see **Ports**.

The 3DM-CV7-GNSS/INS can be configured to accept NMEA sentences via UART on one of the GPIO pins. See NMEA Input. The behavior must be set to the RX mode. Only one pin should be configured for UART RX at a time, otherwise the behavior is unspecified. To change the NMEA input pin, first deconfigure the original pin before selecting the new one.

Using the GPIO Configuration Command

The following details the use of the **GPIO Configuration (0x0C,0x41)** command. The above table details the possible values for the **Pin** parameter ("GPIO Pin") as well as the **Feature(s)** supported by each pin.

The **feature**, in combination with the **behavior**, determines what functionality will be enabled:



Pin

This parameter selects the pin a user is going to configure. This is a positive number starting at 1. When saving, loading, or restoring the default behavior, 0 may be specified to mean all pins.

Feature

This parameter determines the feature that will be enabled on the selected GPIO pin. Users should keep the pin in the unused state if it is not connected externally.

Behavior

This parameter defines the role a GPIO pin will play in implementing the selected **Feature**. For instance, if a user is defining pin 1 as supporting the PPS, a 2 passed to this parameter would configure the pin as a PPS_OUTPUT. However, if a user is defining pin 1 as supporting the GPIO, a 2 passed to this parameter would configure the initial state of this pin as a GPIO_OUTPUT_LOW. The interpretation of the value entered in this parameter depends on the **Feature** that is enabled.

Pin Mode

This parameter allows a user to alter the hardware configuration of the GPIO pins. This can be useful for interfacing with external equipment. The supported options are listed below by **Feature**. The GPIO Configuration (0x0C,0x41) command documentation provides more detail on the behavior of the pins when configured for open drain, pullup, and pulldown. The table below shows the supported configurations:

Feature	Behavior	Open Drain	Pullup	Pulldown
Unused	-			
GPIO	GPIO_INPUT		X	X
	GPIO_OUTPUT	X	only if also open drain	
PPS	PPS_INPUT			
	PPS_OUTPUT*			
Event Timestamp	TIMESTAMP_x		X	X
UART	UART_PORTx_RX		X	X
	UART_PORTx_TX	X	only if also open drain	

* No special pin modes are supported for PPS output behavior when PPS source is set to an internal GNSS receiver. PPS output from internal receivers is always push/pull.

Built-in Tests

The 3DM-CV7 provides users two types of built-in tests (BIT) for verifying its proper operation: the **continuous BIT** and the **commanded BIT**. The Commanded BIT runs a self-test of the internal sensors to verify they respond properly to physical stimuli. The sensor must temporarily pause normal operation during the test. In contrast, the Continuous BIT reports the status of many different continuously-monitored systems.

Commanded Built-in Test

Description

The commanded BIT is a disruptive process which pauses normal operation and performs a series of tests on various system components. The primary purpose of this command is to verify that the IMU sensors are operating normally; however, other tests are also conducted. See the table below for a complete description.

It may take several seconds for this command to complete. During this time, 3DM-CV7 data will be unavailable.

IMPORTANT: The IMU sensors can report false failures if the system is subjected to significant acceleration, rotation, or vibration during this test. Therefore, it is advised that users only run this test when the 3DM-CV7 is at rest.

NOTE: For some errors, the continuous BIT provides more detail than the commanded BIT. Therefore, a continuous BIT executed after a commanded BIT may offer additional insight into a commanded BIT failure.

Command

Built in Test (0x01,0x05)

Message Description

The commanded BIT is formatted as a single uint32 bitfield.

Bit(s)	Error	Notes
0	General Hardware Fault	A critical hardware component, such as the internal oscillator, has failed or is nc operating properly.

Bit(s)	Error	Notes
1	General Firmware Fault	System firmware has detected a serious, unexpected, or abnormal condition.
2	Timing Overload	The system is unable to keep up with the processing demand. Usually this is a res of turning on too many features at once, such as streaming too many descriptors This bit resets after this command is processed.
3	Buffer Overrun	A data buffer in the system has reached its capacity. Usually this means the connection bandwidth has been exceeded by streaming too much data.
4-7	Reserved (System)	
8-11	Reserved (Comms)	
12	Accelerometer Fault	The accelerometer has been damaged or is inoperative.
13	Gyroscope Fault	The gyroscope has been damaged or is inoperative.
14	Magnetometer Fault	The magnetometer has been damaged or is inoperative.
15	Pressure Sensor Fault	The pressure sensor has been damaged or is inoperative.
16-17	Reserved (IMU)	
18	Calibration Error	One or more IMU calibration tables are invalid.
19	General IMU Fault	A problem has been detected in the IMU subsystem. This bit is only for miscellaneous error conditions and will not necessarily be flagged along with oth faults.
20-21	Reserved (filter)	
22	Filter Solution Fault	The filter solution is invalid. This bit will be set if a NaN appears in the filter outpu The filter should be reset if this occurs.
23	General Filter Fault	A problem has been detected in the filter subsystem. This bit is only for miscellaneous error conditions and will not necessarily be flagged along with oth faults.
24-32	Reserved	

Continuous Built-in Test

The continuous BIT reports the status of various internal flags which are continuously updated. The Continuous BIT command is not disruptive and may be run at any time. It is available in command form or in streaming form (see below).

These bits are latching to allow reporting of intermittent faults. Once set, they will not be cleared until transmitted to the user's application. Each message contains all of the faults that have been detected since the last transmission of the same message in the same context. This applies for the command and streaming versions separately. Similar to how integrated data works, each event message action has its own independent copy.

Note: The "sensor self-test fault" flags (e.g. bit 42) are only updated when the commanded BIT is executed. These flags are added to the continuous BIT message to provide greater detail than the commanded BIT command response provides.

Note: The sensor over-range flags are also available in a smaller message which provides results for individual axes and is better suited for high rate streaming. See the **Overrange Status**

(0x80,0x18) message for details.

Command ("Continuous BIT")

This is the command form of the continuous BIT.

Continuous Built-In Test (0x01,0x08)

Data Message ("Streaming BIT")

This is the streaming form of the continuous BIT. Due to its size, it is recommended to stream this message at low rates (this will not result in missed flags, per the latching mechanism).

Streaming Built-In Test (0xA0,0x01)

Message Description

The continuous BIT is formatted as a bitfield composed of 16 bytes. Each consecutive set of 4 bytes represents a subsystem of the device:

- **Bits 0..31** System bits - these represent the device as a whole.
- **Bits 32..63** IMU bits - these correspond to the IMU subsystem.
- **Bits 64..95** Nav bits - these correspond to the EKF and Navigation subsystem.
- **Bits 96..127** GNSS bits - these represent anything related to GNSS or aiding data (for applicable devices).

Bit(s)	Relative Index (Byte.Bit)	Error	Notes
0	0.0	System Clock Failure	The system clock is not operating properly. Timing may n be accurate and the system may be unreliable.
1	0.1	Power Fault	Power supply voltage is outside of the recommended ranც or an internal power fault exists.
2-3	0.2-0.3	Reserved	
4	0.4	Firmware Fault	System firmware has detected an unexpected or abnorm condition.
5	0.5	Timing Overload	The system is unable to keep up with the processing demand. Usually this is a result of turning on too many features at once, such as streaming too many descriptor
6	0.6	Buffer Overrun	A data buffer in the system has reached its capacity anc data may have been lost. Usually this means the connecti bandwidth has been exceeded by streaming too much da
7-15	0.7-1.7	Reserved	
16	2.0	IMU Process Fault	The IMU subsystem is not functioning or did not start properly. Contact support.
17	2.1	IMU Data Process Time Exceeded	IMU data processing has taken longer than the maximur allowed period. Try reducing the number of streaming descriptors.
18	2.2	IMU Data Rate Error	IMU data is not being produced at the expected base rat This may happen if a timing overload occurs.
19	2.3	IMU Stall	The IMU subsystem has stalled and is not producing dat

Bit(s)	Relative Index (Byte.Bit)	Error	Notes
20-23	2.4-2.7	Filter Process Fault	The Filter subsystem is unresponsive or not operating normally.
20	2.4	Filter Major Cycle Fault	A problem with the navigation filter major cycle has bee identified. Use in combination with bits 22 and 23.
21	2.5	Filter Minor Cycle Fault	A problem with the navigation filter minor cycle has bee identified. Use in combination with bits 22 and 23.
22	2.6	Filter Data Rate Error	One of the navigation filter processes is not running at th proper base rate. Use in combination with bits 20 and 2
23	2.7	Filter Process Stall	One of the navigation filter processes is stalled and is no producing data. Use in combination with bits 20 and 21
24-27	3.0-3.3	GNSS Subsystem Fault	The GNSS subsystem is unresponsive or not operating normally.
24	3.0	GNSS Main Process Fault	The master GNSS process has not run within the timeou period.
25	3.1	GNSS Receiver 1 Stall	Data has not been processed from internal receiver 1 wit the timeout period.
26	3.2	GNSS Receiver 2 Stall	Data has not been processed from internal receiver 2 wit the timeout period.
27	3.3	Reserved	
28-31	3.4-3.7	System Reserved	
32	4.0	IMU Reserved	
33	4.1	IMU Communication Fault	No communication with IMU.
34	4.2	IMU Timing Overrun	IMU core processes are taking longer than their allotted time.
35	4.3	IMU Reserved	
36	4.4	IMU Calibration Error - Accelerometer	A problem is detected with the accelerometer calibratior
37	4.5	IMU Calibration Error - Gyroscope	A problem is detected with the gyroscope calibration.
38	4.6	IMU Calibration Error - Magnetometer	A problem is detected with the magnetometer calibratior
39	4.7	IMU Calibration Error - Reserved	
40	5.0	Accelerometer General Fault	Accelerometer will not initialize.
41	5.1	Accelerometer Over-range	One or more axes subjected to accelerations near or outs the selected range.
42	5.2	Accelerometer Self-test Fail	The internal self-test of the accelerometer failed.
43	5.3	Accelerometer Reserved	
44	5.4	Gyroscope General Fault	Gyroscope will not initialize.
45	5.5	Gyroscope Over-range	One or more axes subjected to rotational rates near or outside the selected range.
46	5.6	Gyroscope Self-test Fail	The internal self-test of the gyroscope failed.
47	5.7	Gyroscope Reserved	
48	6.0	Magnetometer General Fault	Magnetometer will not initialize.
49	6.1	Magnetometer Over-range	One or more axes subjected to magnetic fields strength near or outside the supported range.
50	6.2	Magnetometer Self-test Fail	The internal self-test of the magnetometer failed.
51	6.3	Magnetometer Reserved	
52	6.4	Pressure Sensor General Fault	Pressure sensor won't initialize.
53	6.5	Pressure Sensor Over-range	Pressure sensor subjected to pressures outside of the supported range.

Bit(s)	Relative Index (Byte.Bit)	Error	Notes
54	6.6	Pressure Sensor Self-test Fail	The internal self-test of the pressure sensor failed.
55	6.7	Pressure Sensor Reserved	
56	7.0	Factory Bits Invalid	Device has lost configuration. Contact support.
57-63	7.1-7.7	Reserved	
64	8.0	Reserved	
65	8.1	Reserved	
66	8.2	Filter Timing Overrun	Filter processes are taking longer than their allotted time
67	8.3	Filter Timing Underrun	Filter processes are being skipped.
68-104	8.4-13.0	GNSS Reserved	
105	13.1	Receiver 1 Fault	Could not communicate with internal GNSS receiver #1. Contact support.
106-107	13.2-13.3	GNSS Reserved	
108	13.4	Receiver 1 No Solution	Internal GNSS receiver #1 does not have a valid solution This is usually not a device problem. Check the antenna connected and that it has a clear view of the sky.
109	13.5	GNSS Reserved	
110	13.6	Receiver 2 Fault	Could not communicate with internal GNSS receiver #2. Contact support.
111-112	13.7-14.0	GNSS Reserved	
113	14.1	Receiver 2 No Solution	Internal GNSS receiver #2 does not have a valid solution This is usually not a device problem. Check the antenna connected and that it has a clear view of the sky.
114-127	14.1-15.7	GNSS Reserved	

Adjustable Range & Over-range Detection

Adjustable Range

The 3DM-CV7 supports multiple accelerometer and gyro ranges. The range can be set using the **Sensor Range (0x0C,0x52)** command. The table be each supported range for use with the **Sensor Range (0x0C,0x52)** command . The supported ranges and selectors can also be determined through th **Ranges (0x0C,0x53)** command. SensorConnect™ fully supports the sensor range feature.

Device Name	Part Number	Selector & Supported Ranges - Accelerometer	Selector & S
3DM-CV7-INS	6291-9960	2 = 4g, 3 = 8g, 4 = 16g	2 = 250dp
3DM-CV7-AHRS	6286-9960		
3DM-CV7-AR	6287-9960		

Over-range Detection

The 3DM-CV7 provides over-range detection. This feature is always on. The **Overrange Status (0x80,0x18)** message notifies of the user of over-rang also be used to monitor for over-ranges. See **Built-in Tests** for details.

The LED also indicates over-ranges. See **LED States** for a complete description of the LED indicator.

TIP: Over-range events degrade filter performance. It is recommended to increase the sensor range until these events are not encountered.

Timing

This page provides timing specifications and describes each specification in detail.

TIMING SPECIFICATIONS			
SUBSYSTEM	**Specification**	**Typical Value**	**Type**
IMU	Motion to Message Latency	1.9 ms	latency
IMU	Timestamp Repeatability	< 1us	error
GPIO	GPIO Timestamp Accuracy	<< 1 us	error

Explanation of Specifications

Motion to Message Latency

This specification describes the amount of time (latency) it takes from the moment a motion (either an acceleration or a rotation) is imparted to the 3DM-CV7 to when a message is generated that captures the magnitude of that motion.

This specification does not include the time it then takes to send the message over the communications interface.

Timestamp Repeatability

This specification accounts for variations in timestamp accuracy introduced by internal hardware processing delays. Repeatability will also be impacted by any additional clock jitter introduced by an external clock.

See **PPS** and **Synchronizing Data Output with an External System** for options the 3DM-CV7 provides for applying an external clock.

GPIO Timestamp Accuracy

The accuracy of the timestamp generated from a GPIO Event Trigger with the pin set to the hardware timestamping feature.

Additional Sources of Latency

- Transmitting data can take a significant amount of time. The device is not blocked from processing during transmission time, but it does factor into the latency between the originating event and the final application. If the UART is used, the baud rate will determine how long it takes to transfer a given packet. Therefore, it is recommended to use the highest possible baud rate if latency is a concern. USB offers much higher transfer speeds, but it comes at the expense of non-deterministic delays and stalls imparted by the host, which can be significant.
- Latency specifications can be subject to device configuration. Enabling more features or MIP messages can increase processing time and latency.
- These specifications are not valid when a timing overrun condition exists (i.e. if the timing overrun bit is set in the **Built-in Tests** result). This can occur if too many features are enabled simultaneously and must be avoided for proper operation.

Time Synchronization

The 3DM-CV7 offers a host of features which can be used to synchronize the 3DM-CV7 and an external system. This application note provides a high-level overview of the options at a user's disposal.

The 3DM-CV7 provides two types of synchronization: **time domain synchronization** and **sampling synchronization**. Some applications may need one or both of these options. Each method is addressed below, but first it is important to understand the two types of timestamp that the 3DM-CV7 provides.

Timestamps

The 3DM-CV7 maintains two sets of timestamps - Internal or Reference Time, and External Time.

Reference time is a monotonic timer that starts at power-on. It can only increase, will never be adjusted, will not roll over, and does not exhibit any discontinuities. This reference time is used for all internal timekeeping and most computations. Since it is derived from the 3DM-CV7's internal crystal oscillator, it is subject to drift relative to other systems. The main benefit to using reference time is its simplicity and reliability.

The following data fields represent internal reference time:
Reference Timestamp (0xFF,0xD5)
Reference Time Delta (0xFF,0xD6)

External time is meant to be synchronized with other systems. Through the use of the **GPS Time Update Command (0x01,0x72)** and a pulse-per-second (**PPS**) signal, external time can track another time source. A dedicated PPS signal allows very precise alignment and elimination of drift. The time update command allows control of the absolute time reference or epoch. The most common use case is for external time to track GPS time, but it can also be used with other systems such as Unix time.

The following data fields represent external time:
External Timestamp (0xFF,0xD7)
External Time Delta (0xFF,0xD8)

[GPS Timestamp (0xFF,0xD3)](#)

[Delta Time (0xFF,0xD4)](#)

To avoid confusion, we recommend only using the GPS Timestamp (0xFF,0xD3) field if you're actually using GPS time in your application.

External Time Domain Synchronization

As described in the External Time section above, the 3DM-CV7 can synchronize with another time source. An example would be configuring the 3DM-CV7 to timestamp its IMU data in reference to an external time domain such as GPS time. In this case, the sampling of IMU data would be driven by the 3DM-CV7's internal reference clock, but the (external) timestamps generated for that data would be expressed in the GPS time domain.

Method of Time Domain Synchronization

External time synchronization is controlled by the **PPS Source Control (0x0C,0x28)** . The PPS signal may be generated by the 3DM-CV7 or by another device. The differences are described below according to the PPS source.

To achieve time domain synchronization, a digital PPS line must be connected between the 3DM-CV7 and the other system. This provides a very precise time alignment that cannot be achieved with a serial packet alone.

The external time must be initialized by sending the **GPS Time Update Command (0x01,0x72)** . This command may be sent once during initialization or periodically (e.g. every second). It must be received within a half-second of the PPS signal's rising edge.

Configuration of the PPS and GPIO is covered on the **PPS** page.

GPIO Source

A PPS signal from another device can be applied to a GPIO pin. Drift between the external time and the other device's clock will be removed and timing will be very accurate.

GENERATED Source

The 3DM-CV7 can generate its own PPS signal from the internal reference clock. There will be zero drift between the internal and external times. It can be useful to use this mode even without a PPS hardware line as it will ensure the 3DM-CV7 has valid time in all situations. External time can be in whatever reference frame is required by the application.

RECEIVER Source

The 3DM-CV7 has built-in GNSS receivers which can provide a PPS signal aligned to the global GPS time. In this case, the external time will be synchronized to GPS time. It does not need to be initialized using the Time Update command in most cases. Note that the time will only be valid while the corresponding receiver has a GNSS fix. This mode can also be used without an external PPS line if tight (sub-second) time synchronization is not required.

Sampling/Event Synchronization

Sampling synchronization refers to synchronization methods that align data samples with a digital signal provided by or to an external system.

Methods of Sampling Synchronization

Sampling synchronization is achieved through the event system. A full explanation of the event system can be found **here**. There are three cases relevant to synchronization with other systems.

Trigger Input

The event system can be configured to generate data synchronized with a clock signal on a GPIO. The 3DM-CV7 will generate interpolated data that corresponds to the instant the GPIO receives a rising or falling edge. This data can be timestamped in the 3DM-CV7's internal reference time or in an external time domain.

The basic concept is to configure one or more event messages linked to a GPIO edge trigger, so that when a pulse arrives data is transmitted using the timestamp of the pulse. **Synchronizing Data Output with an External System** provides an example configuration.

Trigger Output

The event system can be configured to control a GPIO pin based on a time interval. The interval may be derived from either the reference or external absolute timestamps. Data sampling is unaffected and will be output per the normal streaming behavior. This mode is primarily useful to control another device, such as a camera shutter, which needs to be roughly aligned with data frames. In the camera example, the camera would take pictures at a rate that is locked to the CV7's sample rate.

Note that each pulse has a resolution limitation of 1 ms and thus is subject to much more phase noise than a PPS signal. Despite the low resolution, this method is useful to ensure data rates are consistent between devices and to output signals other than 1 Hz. Data output is not phase-aligned when using regular streaming, but by using event message actions instead (similar to the Trigger Input section), phase alignment is possible.

For this mode, configure a Data Trigger over the desired interval (say 60 Hz) and a corresponding GPIO action which sets the output to the state of the trigger. This is similar to running the following pseudocode every millisecond:

```
if reference_time.nanoseconds mod 16666667 < 8333334
  then gpio_state = HIGH
  else gpio_state = LOW
```

See **Synchronized Square Wave Output** for an example.

Combining Methods

It's important to understand that a PPS signal can't be used to synchronize sampling. The purpose of a PPS signal is to establish a common understanding of time between two or more devices, like two people tuning their watches so they always read the same time.

Likewise, the event system can't establish a common time base. It can only control when things happen, according to the two available clocks in the 3DM-CV7.

Therefore, it may be necessary to use both of these systems in some applications. If your application requires data output alignment and timestamping according to another clock source (for example, generating GPS-timestamped data according to an input pulse), then both systems are required.

Introduction

The 3DM-CV7 supports user-configurable "events" which can trigger transmission of data or change the state of a GPIO pin. They can be triggered from a variety of sources including data quantities and GPIO inputs. These events are highly flexible and can reduce the bandwidth, processing, and power requirements of the system by only transmitting data when it's needed.

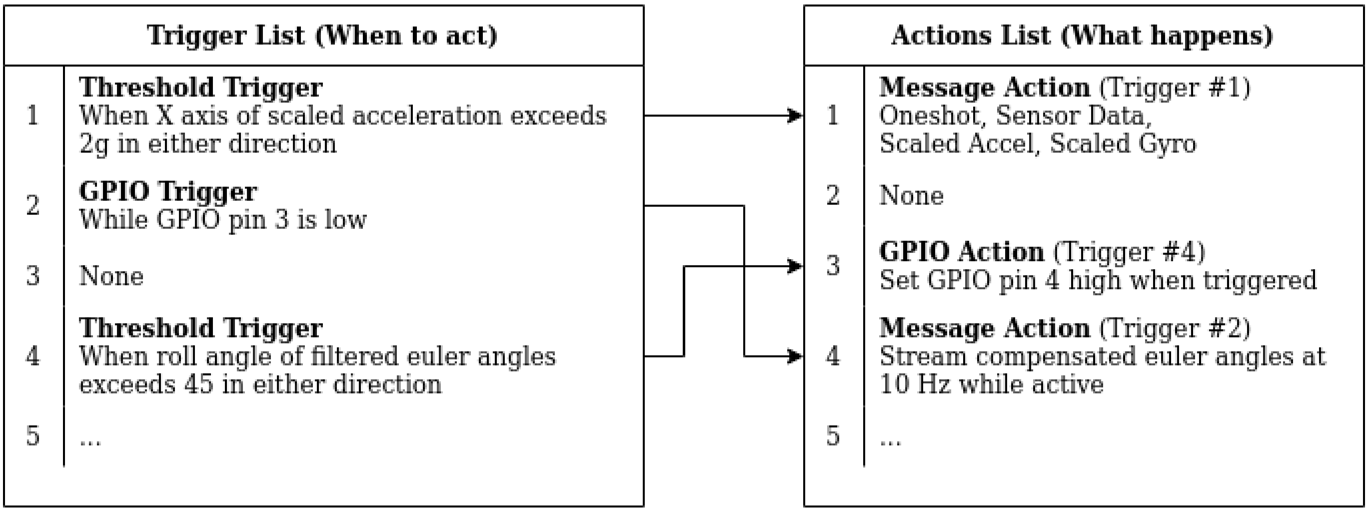
The event system can also be leveraged to provide real-time synchronization between systems. For example, it could be used to generate attitude and heading information that is synchronized to a camera's shutter.

Details about timing accuracy and latency of event system components can be found in **Timing**.

Event Components: Triggers & Actions

An event consists of two components: a trigger and an action. Actions define what should happen, while triggers define when it should happen. The device maintains a list of triggers and a list of actions. Each trigger or action is referenced by a number called the instance **id**. The **id** represents its position in its corresponding list (see figure below). Actions are linked to triggers by a trigger **id** field which is set to match the instance **id** of the trigger.

For example, an action might be configured to emit a MIP data packet, and a trigger might be configured to detect acceleration exceeding a threshold. The configuration is outlined in the figure below. The threshold trigger (trigger **id** = 1) sends a MIP data packet (action **id** = 1) when the acceleration exceeds 2g.



An action can be activated by any trigger, and more than one action can be activated by the same trigger. The actions must simply reference the **id** of the correct trigger.

Types of triggers

- **None** - Does nothing. A trigger set to None will never activate and cannot be enabled.
- [Trigger: GPIO](#) - Watches the state of a GPIO input pin. It can activate while the state is high or low, or on a transition. The edge detection mode can be used for precise timestamping and synchronization.
- [Trigger: Threshold](#) - Watches a specified data quantity. It can check if the value is within or outside of a user-defined window. It can also trigger on time intervals (e.g. every second).
- [Trigger: Combination](#) - A logical combination of up to four other triggers.

Types of actions

- **None** - Does nothing.
- [Action: GPIO](#) - Controls the state of a GPIO pin set to output mode. Useful for triggering external devices.
- [Action: Message](#) - Sends a MIP message containing user-specified data quantities. It uses a message format similar to regular data streaming.

Activating Events

Triggers are "active" when the specified condition is met. This depends on the type of trigger and its configuration parameters. An action linked to an active trigger will be executed. In the above example, if the z-axis acceleration is 3 g (i.e. more than 2 g), the trigger will be active and a MIP message will be emitted.

A trigger remains active as long as its conditions are satisfied. If the event in the example had been configured to send packets at 100 Hz instead of once, it would continue to do so as long as the acceleration stayed above 2 g.

Triggers must be enabled in order to become active. See the configuration section below for more details.

Oneshot vs. Continuous Components

Some components can be configured for either oneshot mode or continuous mode.

- Most triggers operate continuously. Actions are executed as long as the trigger remains active.
- The GPIO trigger, if set to "edge" mode, will only activate for one event cycle each time the pin changes state. Associated actions will execute once for each occurrence, but not more often than once per event cycle. Each pin state transition counts as a new activation, as if the trigger was deactivated in between.
- The Message Action has a decimation parameter which controls the rate at which packets are transmitted. When used with a continuous trigger, data is streamed while the trigger is active. If decimation is 0, only one packet will be emitted per activation; the trigger must reactivate before another packet will be sent.

Event Cycle

During an “event cycle” or “event tick” all of the configured triggers and actions are updated. Triggers are updated first, before any actions, because actions depend on the trigger states. Actions can be executed at most once per cycle.

The event cycle occurs at the base rate for the System Data descriptor set (0xA0). See **Protocol Support and Specifications**. With the exception of a GPIO trigger on a timestamped pin, trigger data is sampled at this rate.

Configuration

The event system is configured through MIP commands. The following provides a description of the relevant commands.

Trigger & Action Configuration

By default, all trigger and action instances are assigned the None type, which does nothing. The user can change this using the **Event Trigger Configuration (0x0C,0x2E)** and **Event Action Configuration (0x0C,0x2F)** commands. These commands require the user to specify the instance id, the type of trigger or action, and some parameters which depend on the trigger or action type. Actions additionally require the trigger id with which to be associated. The values for the 'type' parameters are listed in the documentation for the respective setup command.

The device reports what types of events are supported via the **Get Supported Events (0x0C,0x2A)** command. Device resources are limited, so this command also identifies the maximum number of instances of each component. This information is also provided in **Protocol Support and Specifications**.

Enabling & Disabling Triggers

Each trigger can be enabled, disabled, or placed into test mode using the **Event Control (0x0C,0x2B)** command:

- **Disabled** - The trigger will never activate.
- **Enabled** - The trigger performs its specified function.
- **Test** - Like disabled, but the trigger is forced to the active state. This is useful to test if an action is properly configured and linked to the right trigger.
- **Oneshot** - This setting puts the trigger into test mode for exactly one event cycle, after which it returns to its original setting (either enabled or disabled).

By default, triggers are disabled after configuration. Once configuration setup is complete, use the event control command to enable them. Trigger instances set to None cannot be enabled.

Event Status

Both triggers and actions have a command to get the status of one or more instances. This is useful to produce a summary of configurations and to verify the configurations.

Action status is obtained with the **Get Action Status (0x0C,0x2D)** command. The response identifies the type of each action and its associated trigger.

Trigger status is obtained with the **Get Trigger Status (0x0C,0x2C)** command. The response identifies the type of each trigger, whether it's enabled, disabled, or in test mode, and if it's currently active.

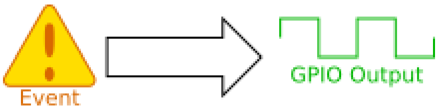
Both commands take an optional list of instance IDs to query. If the list is empty (the count field is 0), then all supported instances are returned. The response is a list with entries in order corresponding to the requested instance IDs.

Examples

All of the examples assume the device has been reset to the factory default settings.

- **Synchronizing Data Output with an External System**
- **Gated Data Streaming using a GPIO**
- **Synchronized Square Wave Output**
- **Stopping a Robot if It Tips Over**
  - Advanced: **Using Two GPIO Actions in Oneshot mode** (Hysteresis)
  - Advanced: **Using a Latching Combination Trigger for Hysteresis**
  - Advanced: **Using Two Thresholds with a Combination Trigger**
- **Troubleshooting Events**

Action: GPIO



The GPIO action controls the state of a GPIO pin based on it's activation state.

The pin must be configured to the GPIO feature and OUTPUT behavior using the **GPIO Configuration (0x0C,0x41)** command.

Normal mode

Mode is ACTIVE_HIGH or ACTIVE_LOW

In normal mode, the pin state is set during every event cycle. When mode is ACTIVE_HIGH, the pin directly reflects the state of the trigger; it is high when the trigger is active and low otherwise. When mode is ACTIVE_LOW, the pin reflects the opposite state.

Example: Square wave output

Oneshot mode

Mode is ONESHOT_HIGH or ONESHOT_LOW

In oneshot mode, the state of the pin is changed once and only in one direction. For ONESHOT_HIGH, the pin is set high when the trigger activates, and for ONESHOT_LOW, the pin is set low. No change is applied when the trigger deactivates. The change will be applied again the next time the trigger activates (the trigger must first deactivate).

This is useful when the pin should be manually reset using the **GPIO State (0x0C,0x42)** command. Alternatively, the pin can be reset via a separate action which is configured for the opposite oneshot mode.

Example: Emergency stop

Toggle mode

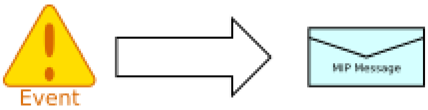
Mode is TOGGLE

Toggle mode changes the state of the pin every time the trigger re-activates.

Examples

- **Gated Data Streaming using a GPIO**

Action: Message



Message actions cause a MIP packet containing user-selectable data to be transmitted. They can be configured for oneshot mode, where one packet is emitted immediately when activated, or for streaming mode, where packets are emitted at a fixed rate while active.

Configuration

A message action has several configuration options: the decimation, descriptor set, and a list of descriptors. The decimation controls how often packets are emitted. If set to 0, only a single packet will be emitted (this is oneshot mode). The descriptor set selects the MIP data descriptor set. It must be supported by the device. The list of descriptors selects the data quantities from that descriptor set.

Because a packet can only contain data from one descriptor set, multiple message actions are needed to send data from more than one descriptor set. Configure each one to the same associated trigger so that they all transmit together. Similarly, if multiple-rate data is required, use one message action per data rate.

Oneshot mode

decimation = 0

In this mode, one packet is output per trigger activation. The associated trigger must deactivate and reactivate before another packet will be transmitted.

The primary use case for oneshot mode is to synchronize sampling to an external device (see timing information below). It can also be used to receive notifications from the device when something happens.

Example

To output scaled acceleration, use the following configuration.

- Descriptor set 0x80 (Sensor Data)
- Decimation 0 (oneshot mode)
- Descriptors 0x04

See Synchronizing sampling to an external device for a more detailed example.

Streaming mode

decimation > 0

This mode behaves similar to the scheduled data streams (i.e. those set up via the 3DM Message Format (0x0C,0x0F) command). One packet is transmitted immediately when activated, just like with oneshot mode. Another packet is transmitted every decimation ticks of the base rate for the selected descriptor set.

Example

To stream scaled accel (0x80,0x04) and scaled gyro (0x80,0x05) data at 100 Hz, use the following configuration.

- Descriptor set 0x80 (Sensor Data)
- Decimation 10 (Sensor Data base rate, typically 1000, divided by the desired rate, 100 Hz)
- Descriptors 0x04, 0x05

Event Source

Because of the number of different streaming and event channels, it can be confusing and difficult to identify why a packet was transmitted. Therefore, it is **strongly** recommended that the **Event Source (0xFF,0xD0)** field be included as the first field in all event packets. It contains the action ID responsible for its transmission, which can be used to reconfigure the event or disable it entirely. Using this field also allows event data packets to be differentiated from regular streaming packets.

It's also possible to specify the event source field in scheduled streaming data. In this case the action ID will be 0, which is not a valid instance ID (recall that instance IDs start at 1).

Because this is a **Shared Data Descriptors**, the field will be present even if the number of fields "rolls over" into another MIP packet.

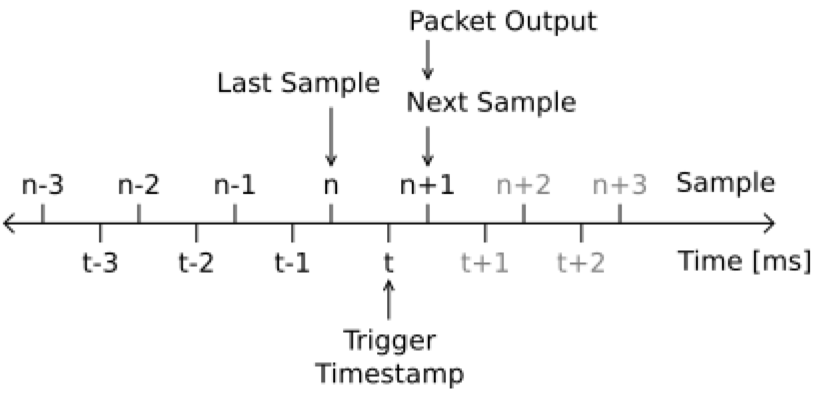
Timing Information

Timestamps and Synchronization

To facilitate time synchronization features, all event triggers record and send to actions the timestamp of the last activation. The message action produces a packet that appears to have been created at this time. If any of the supported timestamp fields[1] are included, they will reflect the trigger time. Additionally, data quantities are interpolated using the previous two samples where such interpolation makes sense.

In the case of the GPIO trigger configured for edge mode, the trigger timestamp is a precise hardware timestamp from the time the gpio pin changed state. Because the packets appear to be sampled at time of activation, this provides a mechanism to synchronize sampling to an external device.

GPIO-triggered MIP packets are output at the next event sample tick.



Delta time

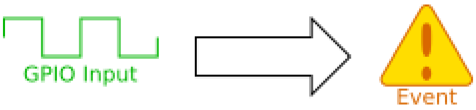
The delta time quantities[2] behave similar to scheduled streaming; they report the time since the last packet containing the delta time field. When used with message actions in oneshot mode, they effectively report the time since the last trigger.

The delta time of each message action is unique, as is the delta time of each descriptor set for scheduled streaming. In other words, two message actions containing delta time will not interfere with each other.

[1] Timestamp fields are **GPS Timestamp (0xFF,0xD3)**, **Reference Timestamp (0xFF,0xD5)**, and **External Timestamp (0xFF,0xD7)**.

[2] Delta time fields are **Delta Time (0xFF,0xD4)**, **Reference Time Delta (0xFF,0xD6)**, and **External Time Delta (0xFF,0xD8)**.

Trigger: GPIO



The GPIO trigger monitors one GPIO pin and can activate while the pin is low or high (state mode), or can activate briefly for one event tick when the pin changes state (edge mode). Among other things, state mode can be used to enable streaming by hardware, without having to send a MIP command. Edge mode can be used for obtaining precisely-timestamped data.

Configuration

State Mode

Mode = WHILE_HIGH or WHILE_LOW

State mode causes the trigger to be active as long as the pin remains in the specified state. The pin must be configured for the GPIO feature using the **GPIO Configuration (0x0C,0x41)** command. The pin behavior may be either input or output.

The pin state is sampled every event cycle.

For an example using this mode, see **Gated Data Streaming using a GPIO**.

Edge Mode

Mode = EDGE

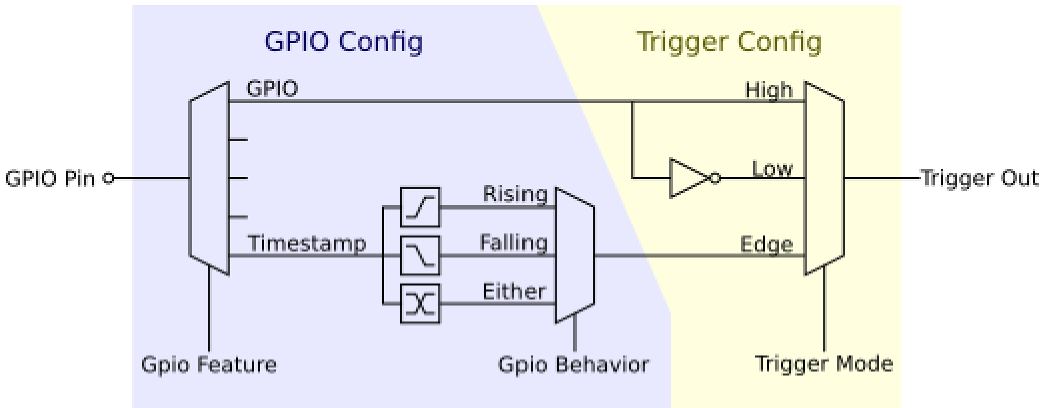
Edge mode causes the trigger to activate for a single event cycle when the pin changes state. To use this mode, the pin must be configured for the TIMESTAMP feature with the **GPIO Configuration (0x0C,0x41)** command. The behavior setting determines the edge direction; it can be rising, falling, or both.

Because the pin is configured with the timestamp feature, the exact time of the edge is captured directly by the hardware. When combined with a Message action, the data packet will be emitted with data as if it were sampled at that time. If included, any timestamp field (such as the **Reference Timestamp (0xFF,0xD5)**) will contain the time of the pin's transition. Other data quantities will be interpolated using the nearest available samples. See the Message action for more details.

For an example using this mode, see **Synchronizing Data Output with an External System**.

GPIO Configuration

Since the event trigger relies on GPIO functionality, the pin must always be configured to the proper mode with the **GPIO Configuration (0x0C,0x41)** command. The following diagram illustrates how the signal is routed from the GPIO pin to the trigger state.



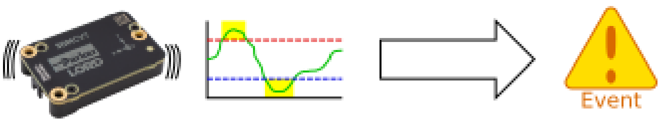
Notice that the signal is routed through different paths for state mode and edge mode. The GPIO feature setting must select the same path as the trigger or it will not work. Also notice that the edge direction is selected by the GPIO configuration and not the trigger mode. This is because the edge detector is a hardware feature of the GPIO controller.

Timing

Because the event system operates on a fixed timing cycle, only one transition per event tick can be registered. If two transitions are detected, only one trigger pulse and associated timestamp will occur. The timestamp feature allows very precise and accurate timing but does not allow data sampling or processing above the event tick rate.

See the Event Cycle rate in **Protocol Support and Specifications**.

Trigger: Threshold



The threshold trigger allows actions to be performed in response to a user-specified data quantity crossing a threshold. There are several modes which make it very flexible.

Configuration

The data to be checked is identified by its corresponding MIP descriptor set, field descriptor, and parameter index. The parameter index is the 1-based index of the parameter within the MIP data field. E.g. Parameter 2 of Scaled Accel (0x80,0x04), a 3-vector, would represent the Y axis.

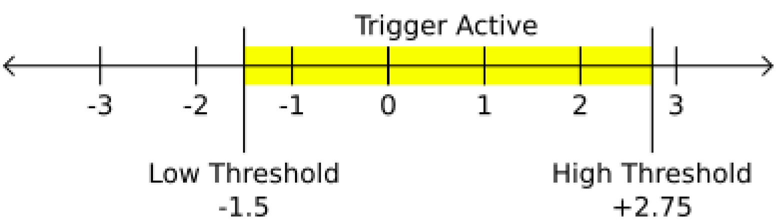
The interpretation of the thresholds depends on the mode parameter.

Window Comparator

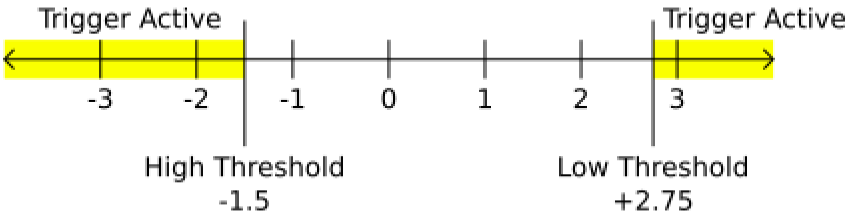
Mode = WINDOW

In window mode, the last two parameters are double-precision lower and upper thresholds. When the data quantity is in the window (i.e. between the thresholds) the trigger is activated. The thresholds can be reversed to invert the logic and make the trigger active when the data is outside the window.

The data value is first converted to a double-precision float for the purposes of the comparison.



The trigger can also be reversed by swapping the low and high thresholds, i.e. if the high threshold is below the low threshold:

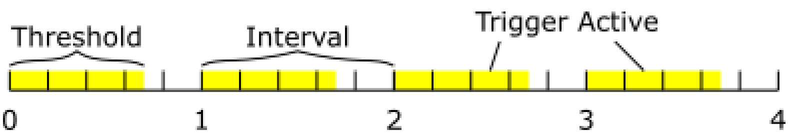


Example: **Stopping a Robot if It Tips Over**

Interval Trigger

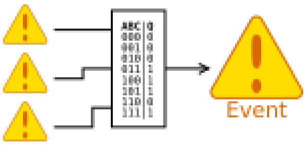
Mode = INTERVAL

In interval mode, the last two parameters are threshold and interval as doubles. The trigger is active when the data value, modulo the interval, is less than the threshold. This forms a repeating pattern which is useful mostly for time quantities.



Example: **Synchronized Square Wave Output**.

Trigger: Combination



The combination trigger is a very powerful feature which allows the logical combination of up to 4 other triggers. The combination trigger activates when the input triggers are active according to a user-defined logic table. Any possible logical expression with up to 4 inputs can be provided.

Combination triggers are updated every event cycle.

Configuration

There are two configuration parameters: an unsigned 16-bit logic value and a list of 4 input triggers.

Input Triggers

The input triggers are specified by the trigger instance ID. Any valid trigger ID other than the one being configured can be specified, including other combination triggers. In this documentation and in software, these 4 inputs are referred to as inputs A through D, with A being first in the list. Unused inputs can be set to trigger ID 0 which means always inactive.

Note: It's possible to specify the trigger's own ID as an input, in which case the value will be the trigger's previous state. Generally this should be avoided as care must be taken to avoid it getting stuck in one state or rapidly toggling between active and inactive.

Logical Truth Table

A logic table lists every possible combination of input states in a specific order, along with the desired output state. For the combination trigger, it looks like the table below. The input triggers are labeled A through D, with A representing the first trigger in the list. '0' represents an inactive input while '1' represents an active input. Each row is one possible situation given 4 input triggers.

D	C	B	A	Output	Comment
0	0	0	0		No inputs active
0	0	0	1		Only A is active
0	0	1	0		Only B is active
0	0	1	1		Both A and B are active
0	1	0	0		Only C is active
0	1	0	1		Both A and C
0	1	1	0		Both B and C
0	1	1	1		A, B, and C
1	0	0	0		Only D
1	0	0	1		A and D
1	0	1	0		B and D
1	0	1	1		A, B, and D
1	1	0	0		C and D
1	1	0	1		A, C, and D
1	1	1	0		B, C, and D
1	1	1	1		All 4 active

To complete the table, write a ‘`1`' in the output column for any row which should activate the combination trigger. Write ‘`0`’ in the other rows. Here’s an example for the expression “`C and (A or B)`”.

A	B	C	D	Output	Comment
0	0	0	0	0	
0	0	0	1	0	C is inactive, therefore all these rows are 0.
0	0	1	0	0	
0	0	1	1	0	
0	1	0	0	0	C is active, but not A or B.
0	1	0	1	1	
0	1	1	0	1	Expression is satisfied.
0	1	1	1	1	
1	0	0	0	0	
1	0	0	1	0	
1	0	1	0	0	The only difference between these 8 rows and the previous 8 is that input D would be active. The previous 8 rows are copied so the result is the same regardless of input D. Technically, as long as trigger id 0 is used for input D, it can never be active and any value can be written in these rows without affecting the result.
1	0	1	1	0	
1	1	0	0	0	
1	1	0	1	1	
1	1	1	0	1	
1	1	1	1	1	

To convert this table to the 16-bit integer required in the MIP command, take the output column and rotate it 90 degrees to the right; the top bit in the first row is the least significant bit and the bottom bit is the most significant. The value for the above example is 1110000011100000 in binary, or E0E0 in hexadecimal. This is the value that would be used to configure the combination trigger.

Unused inputs

The trigger assumes there are 4 inputs, but many times only 2 or 3 are needed. The unused trigger inputs should be set to trigger 0, which is not a valid trigger instance (trigger instance IDs start with 1). For simplicity, use inputs in order and leave the unused ones at the end. Though it's not required, this makes building the logic table easier.

The logic table has repeating patterns. Each additional input doubles the length of the table. Thus, every unused input halves the length of the table. For two inputs, the table needs to be only 4 rows. To be independent of the unused inputs, technically the bit pattern should be repeated until all 16 bits are filled. As noted in the comment above however, since the unused inputs with trigger ID 0 are always inactive, unused rows can be any value. It's generally safe to just pad the remaining bits with 0. For the example above, that would result in the value 0x00E0 instead of 0xE0E0.

It's possible to select the same trigger twice in two different inputs, though there is generally no advantage to doing so outside of testing.

Common Logic Values

Most use cases require only basic logic such as ALL or NONE. A few logic values have been defined below for convenience.

Value	Name	Notes
0x0000	Never	Never activates. Only useful for testing.
0x0001	None / Nor	Only active if no inputs are active.
0x0002	A only	Activates if A is the only active input.
0x0004	B only	Activates if B is the only active input.
0x0008	A and B only	Activates if A and B but not C and D are active. Same as 0x8888 if C and D are unused.
0x8888	A and B	C, D are irrelevant.
0x8080	A, B, and C	D is irrelevant.
0x8000	All	All 4 inputs must be active (requires all 4 inputs to be used)
0x6666	A xor B	A or B but not both; C and D irrelevant.
0xFFFE	Any / Or	Any input or multiple inputs; opposite of None (0x0001). Can be used with fewer than 4 inputs if unused trigger IDs are 0.
0xFFFF	Always	Always active. Only useful for testing.

Memory and Latching Behavior (Advanced)

As mentioned in the note above, it's possible to specify the trigger's own ID as an input. The input state in this case is the result from the previous event cycle. It's also possible to link combination triggers in a circular fashion. This feedback mechanism can be used to implement a set/reset latch for things like threshold hysteresis.

A set/reset latch needs 3 inputs: set, reset, and the previous state. The latch is active if:

- The set input is active (A), or
- The output was previously active (C) and the reset input is inactive (not B).

As an example, assume the combination trigger is instance 4, the set input will come from trigger 1, and the reset input from trigger 2. The input IDs would then be (1,2,4,0), and the logic expression would be (A or (C and not B). By filling out the first half of the table (because D is unused), the binary value 00100000 (20 in hexadecimal) is obtained. Padding with 0 for the second half gives the hexadecimal value 0020.

The final configuration would be logic value 0x0020 and triggers (1,2,4,0).

Examples

Using Two Thresholds with a Combination Trigger

Using a Latching Combination Trigger for Hysteresis

Troubleshooting Events

The 3DM-CV7 provides a few ways to help determine why an event configuration might not be working as intended. This page offers some ways to gain insight into why your configuration may not be working properly.

General Procedure

Check the Action Trigger ID

Actions have a trigger ID field which must be set to the instance ID of the trigger which should cause it to activate. If this is not set correctly, the action may never activate, or may activate at inappropriate times if the wrong trigger is selected.

To check the action ID, use either the **Get Action Status (0x0C,0x2D)** command or the **Event Action Configuration (0x0C,0x2F)** command. The Action Status command will return the action type and associated trigger. The Action Configuration command will return all of the action's parameters including the trigger ID.

Verify the Associated Trigger is Configured

Use the **Get Trigger Status (0x0C,0x2C)** command or the **Event Trigger Configuration (0x0C,0x2E)** command to check the trigger referenced by the action is configured. Pass the action's trigger ID as the instance ID for either command. The Trigger Status command will return the trigger type and its current state. The Trigger Configuration command will return all of the trigger's parameters.

Check if the Trigger is Enabled

The **Get Trigger Status (0x0C,0x2C)** command will return the trigger type and its current status. The enabled bit must be set for it to work properly. Use the **Event Control (0x0C,0x2B)** command to change it to the enabled state if necessary.

Test the Action using the Trigger Test Mode

If your action is not activating at all, you can put the trigger into test mode. Use the **Event Control (0x0C,0x2B)** command with the trigger ID and the TEST mode. This will force the trigger into the active state and trigger the action continuously.

- If the action works when testing the trigger, there is a problem with your trigger's configuration.
- If the action still does not work, the problem is in your action's configuration.

An additional test mode, TEST_PULSE, is provided to test what happens when the trigger activates for a single event cycle. After the test cycle, the trigger will return to the previous enabled/disabled state.

Don't forget to set the trigger back to the ENABLED mode after you're done testing.

GPIO Events

If a GPIO trigger or action is not functioning as desired, ensure that the pin has been configured properly using the **GPIO Configuration (0x0C,0x41)** command as described here:

- For **GPIO Actions**, the pin must be set to the General Input / Output (GPIO) feature. The behavior must be either OUTPUT_LOW or OUTPUT_HIGH. If you've selected the open drain option, ensure the pin has either the internal pullup enabled or is pulled up externally.
- For **GPIO Triggers in state mode**, the pin must be configured for the GPIO feature with the INPUT behavior.
- For **GPIO Triggers in edge mode**, the pin must be configured for the TIMESTAMP feature with the behavior set to the desired edge direction.

See **Trigger: GPIO** and **Action: GPIO** for details.

If a GPIO event component doesn't work after a device reset, make sure the GPIO configuration is saved along with the event configuration.

Test the GPIO

In some cases, it may be useful to test the GPIO pin by itself using the **GPIO State (0x0C,0x42)** command. The pin must be configured for the GPIO feature to perform this test (it will not work when set to TIMESTAMP).

- For inputs, use the READ function selector. Verify that changing the state of the pin externally affects the value read back by the command.

- For outputs, use the WRITE function selector. Verify that sending the command with high or low states changes the pin's voltage. Note that if there is a GPIO action configured for ACTIVE_HIGH or ACTIVE_LOW on this pin, it will override your changes with the GPIO State command, making this test difficult. If you use an oscilloscope, you may observe the changes briefly between event cycles. It is recommended to disable the action by temporarily changing the pin, mode, or action type (disabling the trigger is not enough).

Message Actions

Message actions will usually work if the trigger is configured properly. If you're not seeing any MIP packets even when the trigger is in test mode, there are a few things to check.

- Is the device set to idle? Try sending the **Resume (0x01,0x06)** command to enable packet output.
- If you've selected oneshot mode, try continuous mode (decimation > 0). This will cause packets to be output continuously instead of just once, which increases the likelihood of noticing it.

Testing Triggers

The **Get Trigger Status (0x0C,0x2C)** can be used to get the current status of a trigger. This returns the trigger type, along with a bitfield of status information. Contained in the bitfield are flags for the following:

- Is the trigger currently activated? This is an instantaneous reading of the trigger's current state as seen by any associated actions.
- Is the trigger enabled? Triggers must be enabled to be active. Use the **Event Control (0x0C,0x2B)** command to enable it if necessary.
- Is the trigger in test mode? In test mode, the trigger is continuously active. Use the Event Control command to change it back to the enabled state if this is not intended.

Threshold Triggers

Threshold triggers can be tricky to test because they depend on particular data quantities. A few things to check in the configuration are:

- Are the descriptor set and field descriptor correct?

- Is the parameter ID correct? This value is the parameter number of the quantity within the MIP field. E.g. for a 3-vector like **Scaled Accel (0x80,0x04)**, 1 = X, 2 = Y, 3 = Z.
- If using WINDOW mode, are the thresholds correct? Check if the trigger activates when the thresholds are swapped. Swapping them reverses the window. If the low threshold is lower than the high threshold, the trigger will be activated when the data value is between them. Otherwise, the trigger will activate when the data value is from negative infinity to the high threshold (which is lower), or when the value is from the low threshold (which is higher) to positive infinity.
- If using INTERVAL mode, are the interval and threshold correct? The threshold represents the activation point which repeats periodically at every interval. If the value is lower than the threshold, the trigger is active, and otherwise it is inactive. Often the threshold is set to half the interval.

If the configuration is correct, check if the data is actually meeting the condition specified. Sometimes this can be difficult to reproduce, but a good method of testing it is to set up scheduled streaming and monitor the value continuously. If using a Message Action, it should be easy to see when the trigger is active by comparing the scheduled stream with the event messages. Select the **Event Source (0xFF,0xD0)** and a timestamp field (e.g. **Reference Timestamp (0xFF,0xD5)**) in both messages. If you're not using a Message Action, you can add one using a free action slot and the same trigger ID as your other action. Both will be activated from the same trigger.

Combination Triggers

There are two ways a combination trigger might not work as expected. The first is that the input triggers may not be working properly. Try testing them individually first. Make sure the inputs are set correctly. Unused inputs should be set to trigger 0. The second possibility is that the logic is not configured as intended. The logic can be verified by using the **Event Control (0x0C,0x2B)** command on each of the input triggers. By setting them to DISABLED or TEST, work through all of the possible disabled / enabled cases. Follow the logic table on the **Trigger: Combination** page and identify any bits which are incorrect.

Ports

The 3DM-CV7-GNSS/INS is equipped with multiple communication ports to facilitate various data transmission needs. Depending on your application and communication preferences, commands and data may be sent to the 3DM-CV7 via TTL level UART or USB. In total, the device has 5 ports: three UART ports, and 2 USB ports. Ports, and their specified protocols, are enabled and disabled with the Interface Control (0x7F, 0x02) command.

The 3DM-CV7-GNSS/INS does not have two physical USB ports. Instead, the computer that the device is connected to via USB will recognize the single connection as two separate communication ports: USB1 and USB2 (i.e. COM5, and COM6).

Port ID	Type	Name	Description
0x01	Special	Main	Alias for USB1 (Port ID 0x21) if port is open (DTR set), otherwise serial (Port ID 0x11)
0x11	UART	Uart1	“Main” serial port
0x12	UART	Uart2	Serial port via GPIO 1/2
0x13	UART	Uart3	Serial port via GPIO 3/4
0x21	USB	Usb1	“Main” usb port
0x22	USB	Usb2	"Auxiliary" usb port

The Main Port

The "Main" port refers to the main communication port that is used to send and receive data from the 3DM-CV7. When the device is connected to a computer via USB, the Main port will default to USB1. Otherwise, the Main port will be UART1. All other ports can be thought of as "Auxiliary" ports.

Auxiliary Ports

Auxiliary ports refer to any port that is not actively the Main port. Once enabled, Auxiliary ports contain all of the same functionality as the Main port. All ports, once configured with the Interface Control (0x7F, 0x02) command, have support for the protocols listed in the following table:

Port ID	MIP	RTCM	SPARTN	NMEA
0x01	Input/Output	Input	Input	Input/Output
0x11	Input/Output	Input	Input	Input/Output
0x12	Input/Output	Input	Input	Input/Output
0x13	Input/Output	Input	Input	Input/Output
0x21	Input/Output	Input	Input	Input/Output
0x22	Input/Output	Input	Input	Input/Output

- MIP - MicroStrain Inertial Protocol, the main communication protocol used for all MicroStrain by HBK inertial devices.
- RTCM - A data format used to communicate GNSS corrections from an RTK service provider to a user, commonly used for network-based RTK solutions.
- SPARTN - A data format used to communicate GNSS corrections from an RTK service provider to a user, commonly used for PPP-RTK solutions.
- NMEA-0183 - A standard GNSS protocol.

Input Limits

1. MIP is always enabled on Main port (Port ID 0x01).
2. MIP input may be enabled on up to 2 additional ports.
3. NMEA input may be enabled on up to 1 port.
4. RTCM input may be enabled on up to 1 port.
5. SPARTN input may be enabled on up to 1 port.

There is no limit on output protocol quantities, but all streamed data will have the same format and decimation across ports.

Polling data always responds directly on the port that the Poll Data (0x0C, 0x0D) or Poll NMEA Data (0x0C, 0x04) command was issued, even if that output protocol isn't configured.

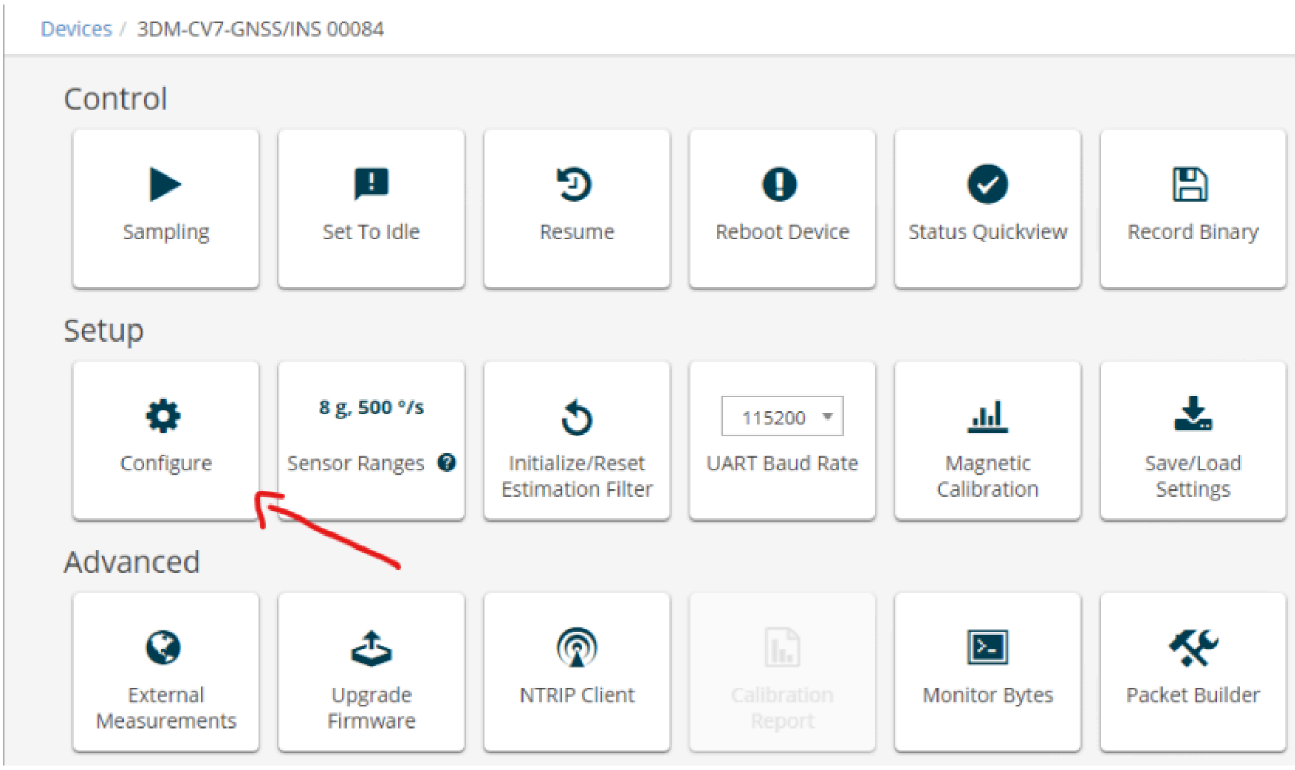
Sending RTCM data to USB2 with SensorConnect

SensorConnect has an integrated NTRIP client to make streaming RTCM correction data easy. For more information on the RTCM messages that the 3DM-CV7-GNSS/INS is able to accept, please see the RTCM section of the manual.

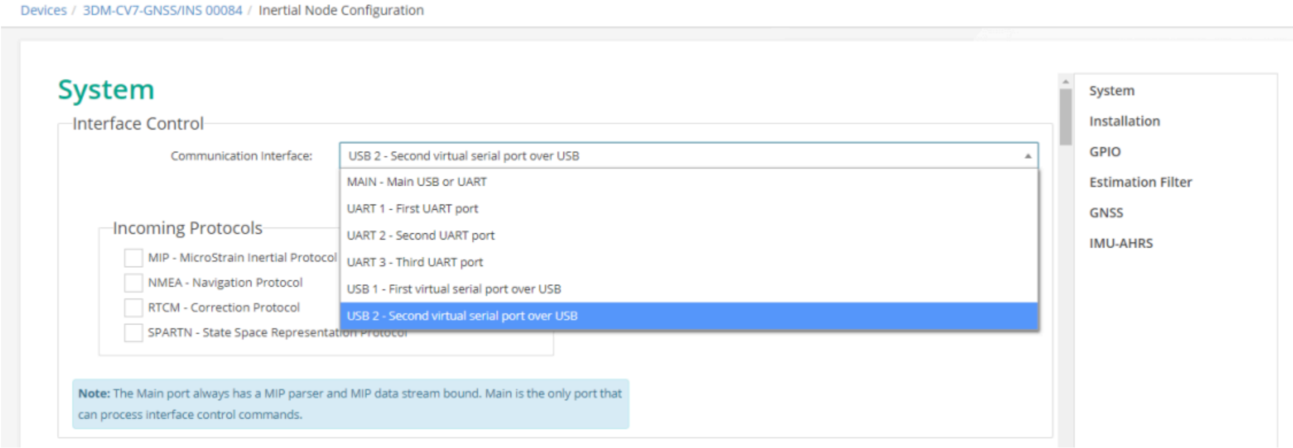
First, we must enable the USB2 port and configure it to both receive an RTCM input, and output a NMEA GGA message. A NMEA GGA message is required for RTK networks like SensorCloud RTK.

1. Connect the 3DM-CV7-GNSS/INS to a computer with SensorConnect installed via USB.

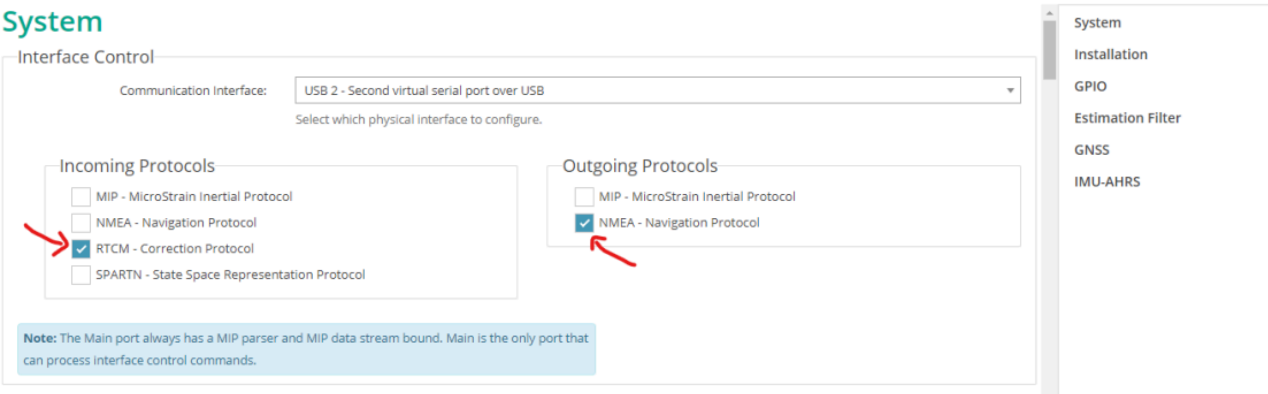
2. Open SensorConnect, and navigate to the Configure window.



3. Select the USB 2 port from the "Communication Interface" dropdown.

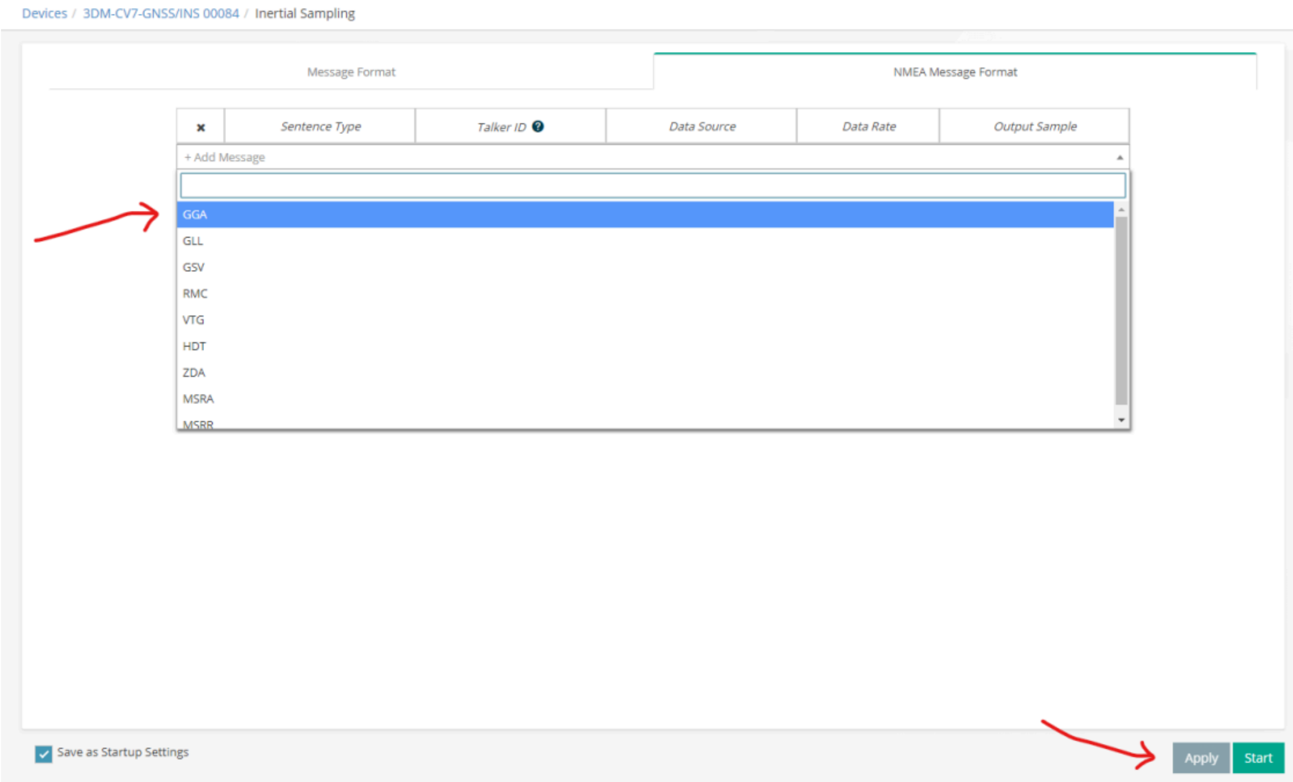


4. For "Incoming Protocols", select RTCM. For "Outgoing Protocols", select NMEA. Once selected, apply the configuration.



5. Go back to the device homescreen and under the "Control" section, select the "Sampling" window.

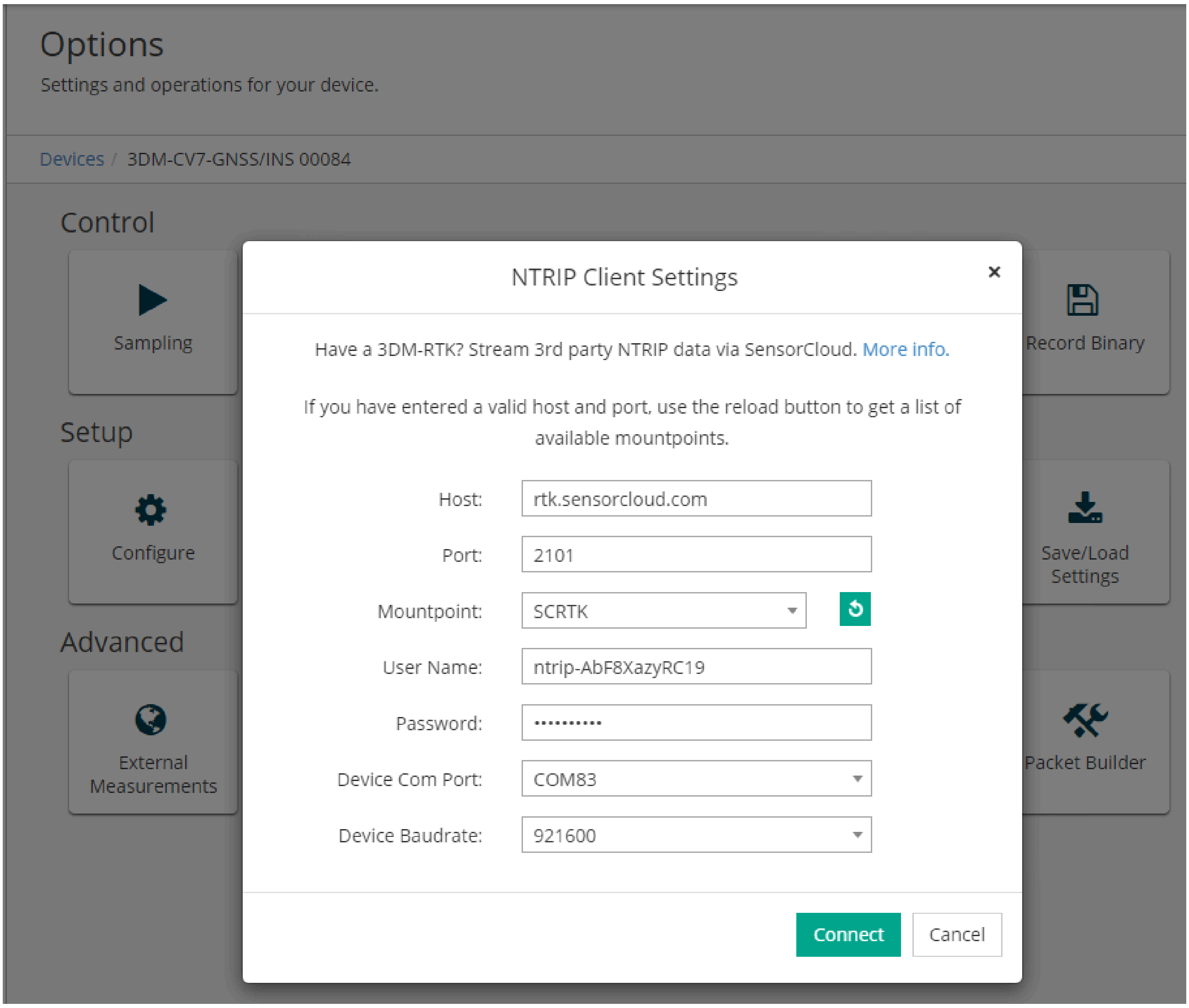
6. Under the "NMEA Message Format" tab, select the GGA message, and then click "Apply" at the bottom of the screen. Now, the device is ready to receive RTCM corrections!



7. Navigate back to the device home screen, and select the "NTRIP Client" window under the "Advanced" section.

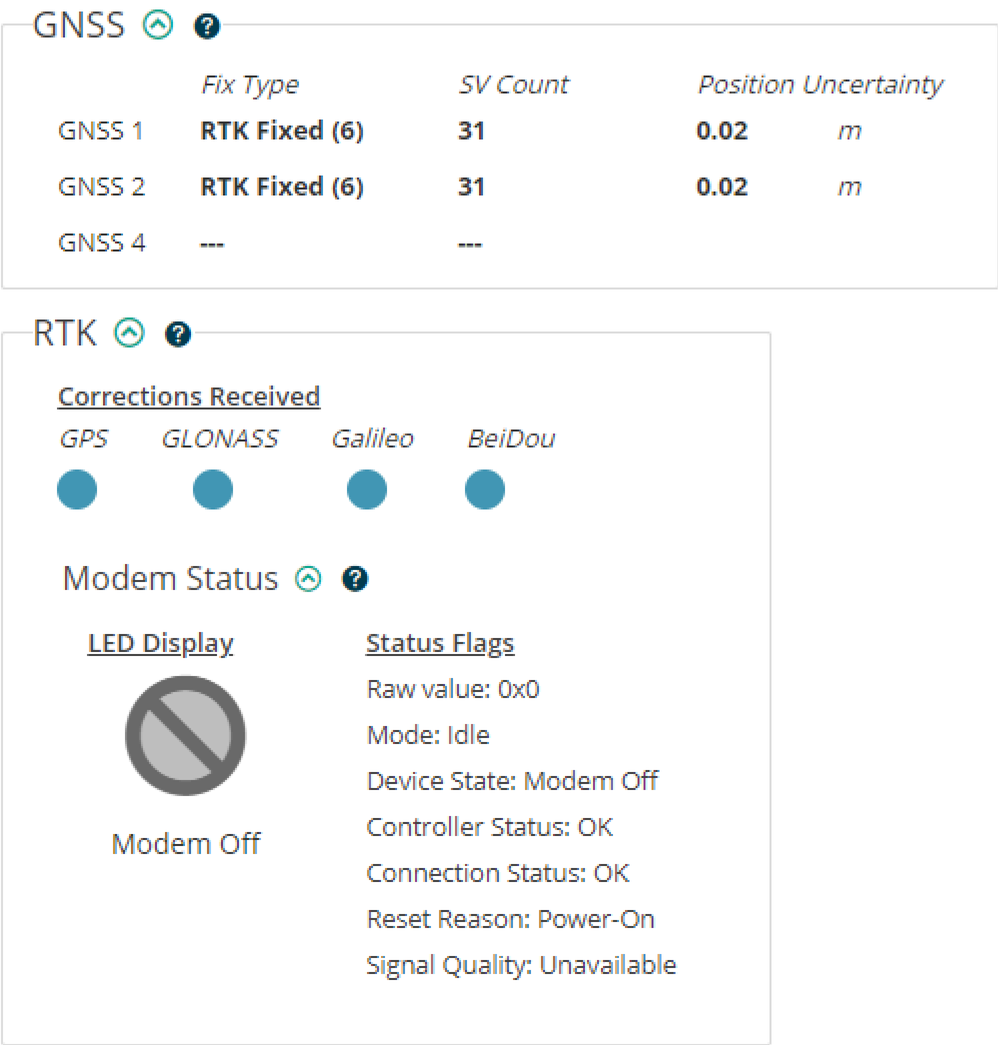
8. A window will pop up, enter your NTRIP Host, Port, Mountpoint, and credentials. You can use any NTRIP provider here, but we recommend SensorCloud RTK. For more information about SensorCloud RTK, check out rtkapp.sensorcloud.com/overview. For the "Device Com Port" box, select the Com port that corresponds to USB 2. The baud rate can be

whatever you want.



9. Click the green "Connect" button to start streaming RTCM corrections to the device.

To make sure the device is receiving corrections, you can navigate to the "Status Quickview" window of SensorConnect to make sure you are receiving and processing corrections. In the "GNSS" section, you should see "Fix Type" as "RTK Fixed (6)", and a position uncertainty of about 2 centimeters. You will also see the device's LED turn blue, please see LED States for more information.



SPARTN Corrections via UART

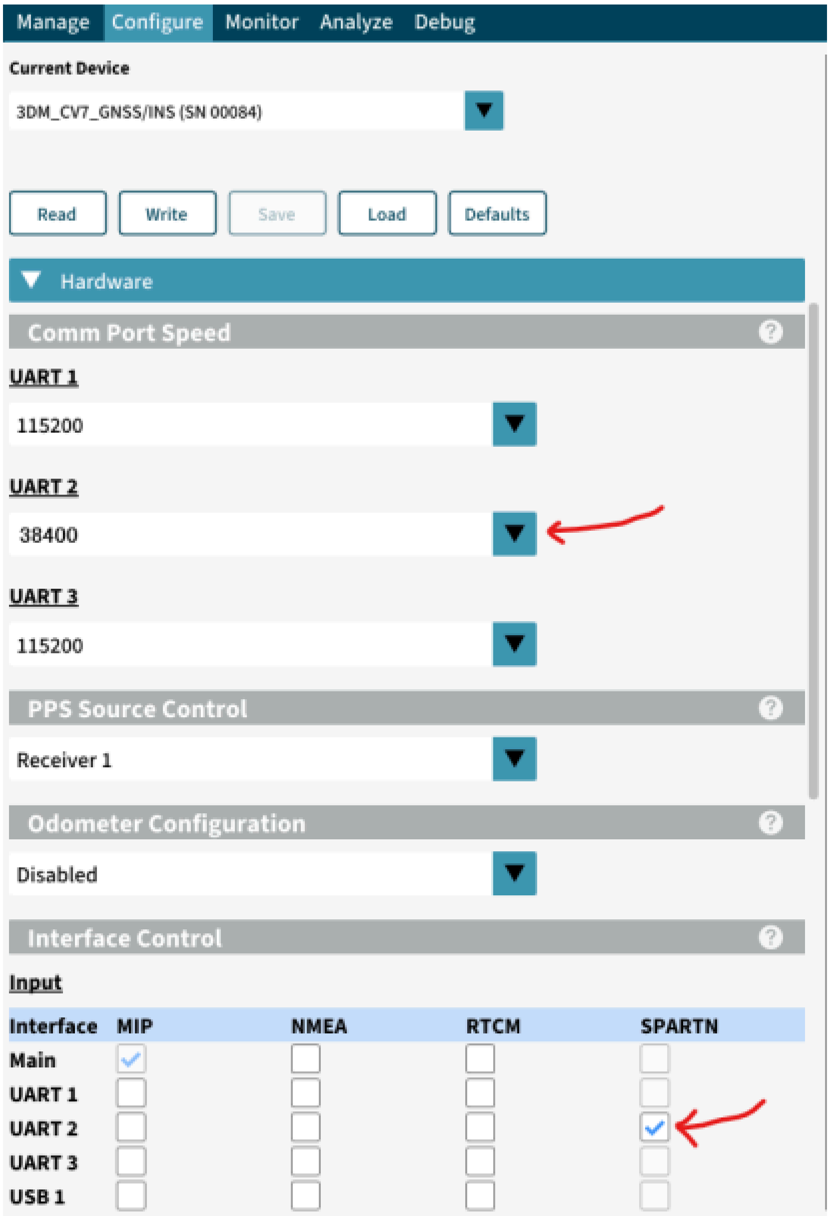
In this example, we will describe how to set up a 3DM-CV7-GNSS/INS to receive L-Band SPARTN corrections from and L-band demodulator. The demodulator we will be using is the Ublox D9S, which is compatible with Ublox's PointPerfect PPP-RTK service. We are using the C101-D9S-0 development board for this example.

First, we must configure the 3DM-CV7-GNSS/INS to be able to receive SPARTN corrections over its UART_2 port.

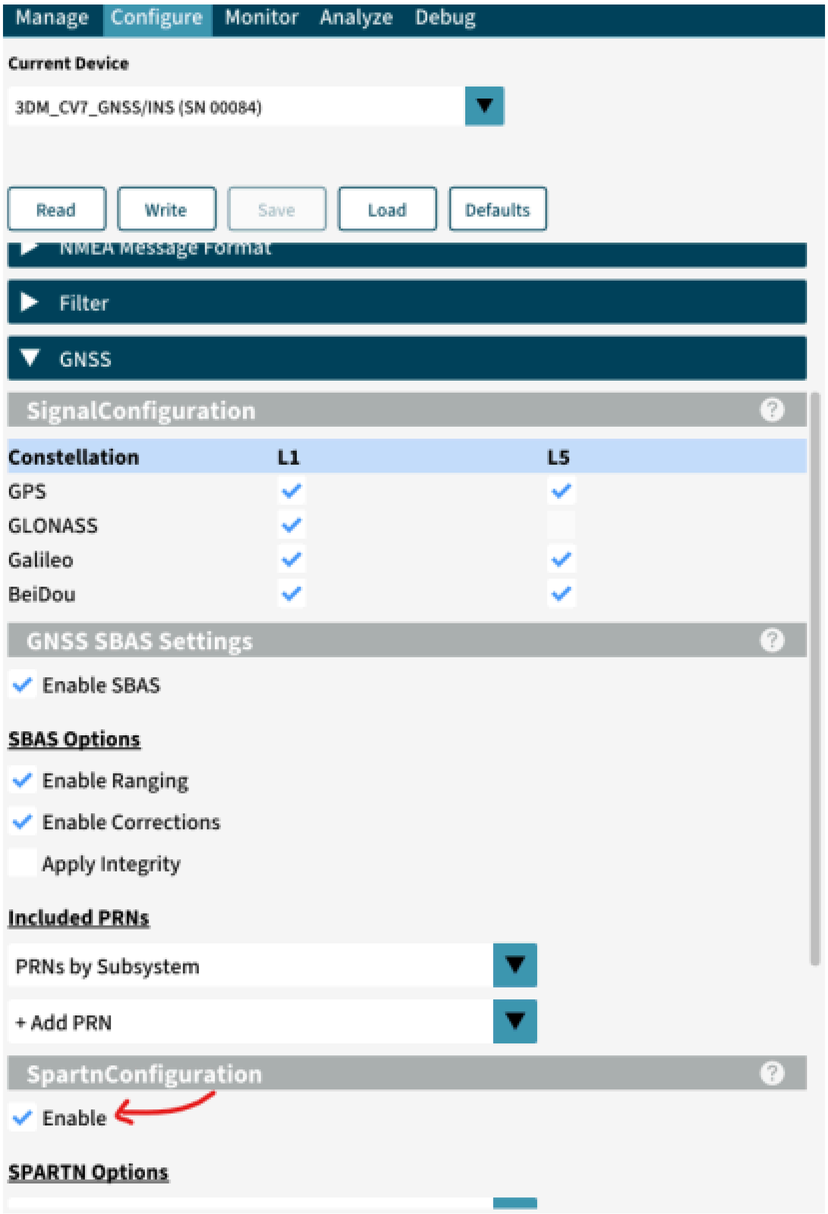
1. On a computer with InertialConnect installed, connect your 3DM-CV7-GNSS/INS to the computer with a USB cable.

2. Once connected, navigate to the Configure tab. Under the "Hardware" dropdown, configure the UART 2 baudrate to 38400 baud. Then, check the "SPARTN" box for UART 2 under the

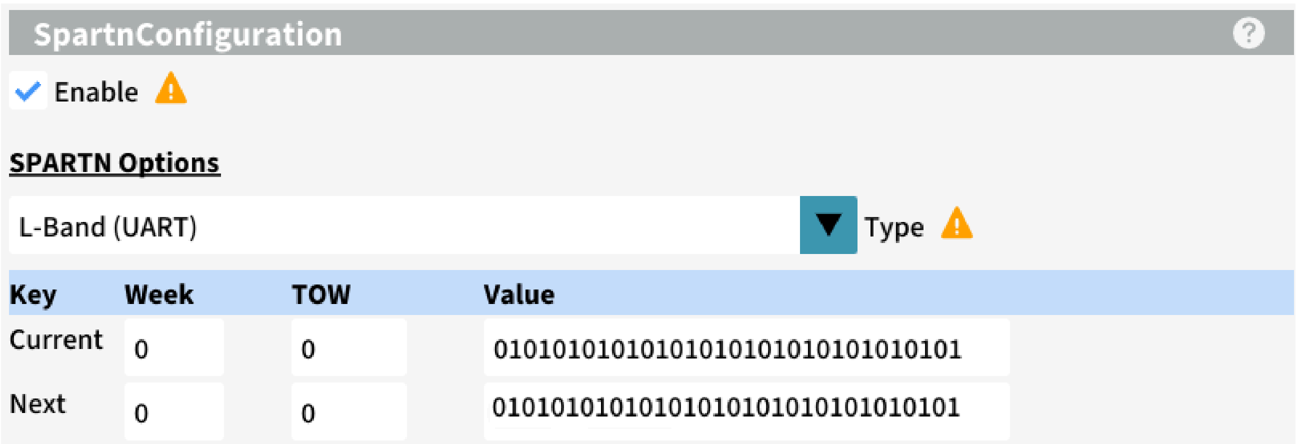
"Interface Control" section.



3. Next, scroll down to the GNSS dropdown. Under the "SPARTN" section, check "Enable SPARTN".



4. Once enabled, under the "SPARTN Options" dropdown select "L-band (UART)", and enter your keys. The "Week" and "TOW" boxes are that of the expiration date. Each key is a hexadecimal string of 16 non-zero bytes (32 total characters, each 0-9 or A-F).



NOTE: Enabling SPARTN corrections and selecting a source will block any RTCM messages from being received by the device.

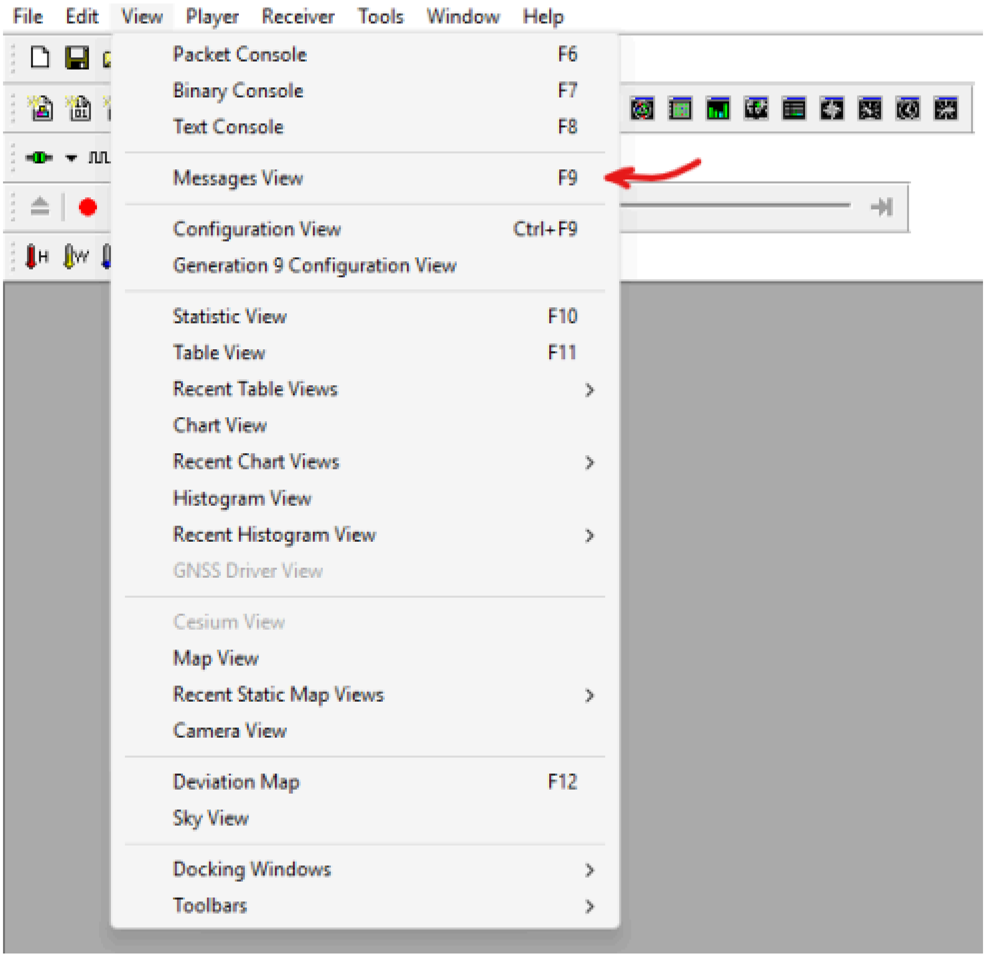
5. Lastly, configure a GPIO pin to act as a UART receive port for UART 2, and the device will now be ready to receive SPARTN corrections over UART 2.

6. Write the settings to the device, and save them if desired.

Next, the D9S needs to be configured to receive SPARTN data via L-band, and then forward it to the 3DM-CV7-GNSS/INS via UART. U-blox u-center will be required to send configuration strings to the device, instructions for installing u-center can be found in [Step 2 of this page]() of the manual.

1. Open U-center, and connect the D9S to your computer via USB.

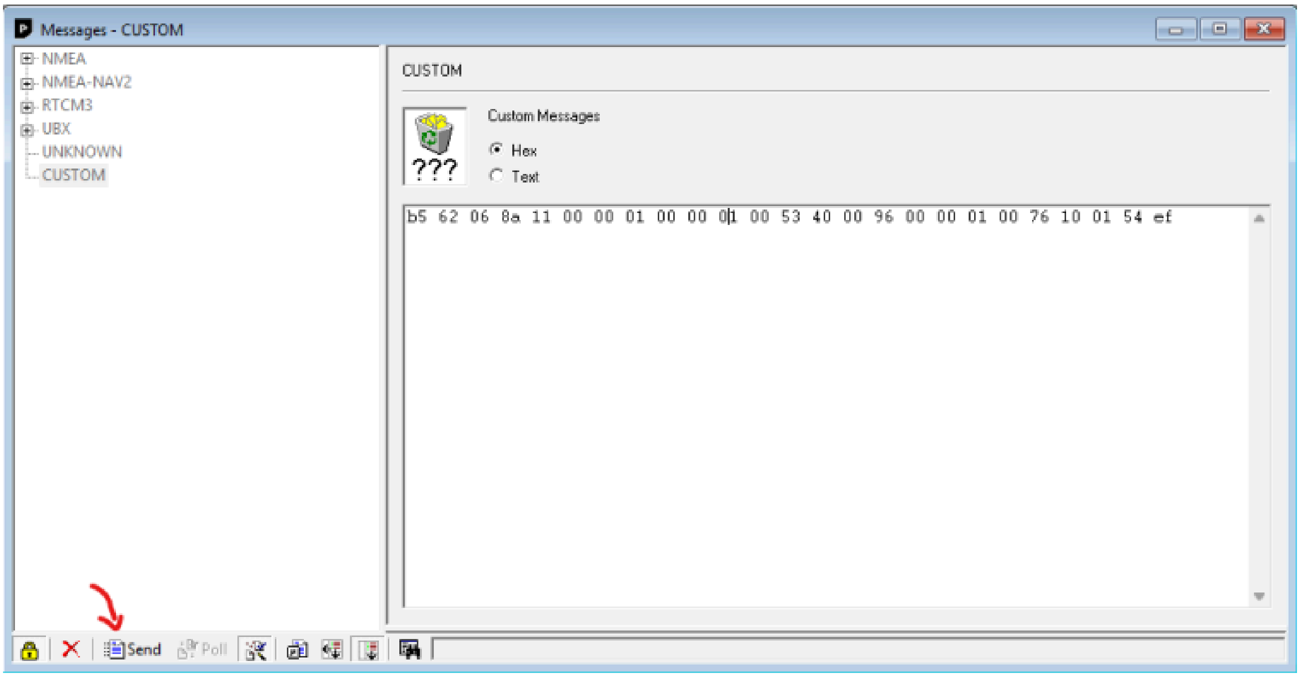
2. Under the "View" tab, click on "Messages View" to open the messages view window.



3. Under the "Custom" section, copy and paste the following two byte strings, and send them to the device. This will configure the D9S to output the required data over the UART 2 port. The settings will be saved to non-volatile memory, so you may unplug the device.

    1. b5 62 06 8a 11 00 00 01 00 00 01 00 53 40 00 96 00 00 01 00 76 10 01 54 ef

    2. b5 62 06 8a 11 00 00 04 00 00 01 00 53 40 00 96 00 00 01 00 76 10 01 57 1f



Finally, connect the TX2 port of the D9S to Pin 9 of the 3DM-CV7-GNSS/INS, and the appropriate power pins.

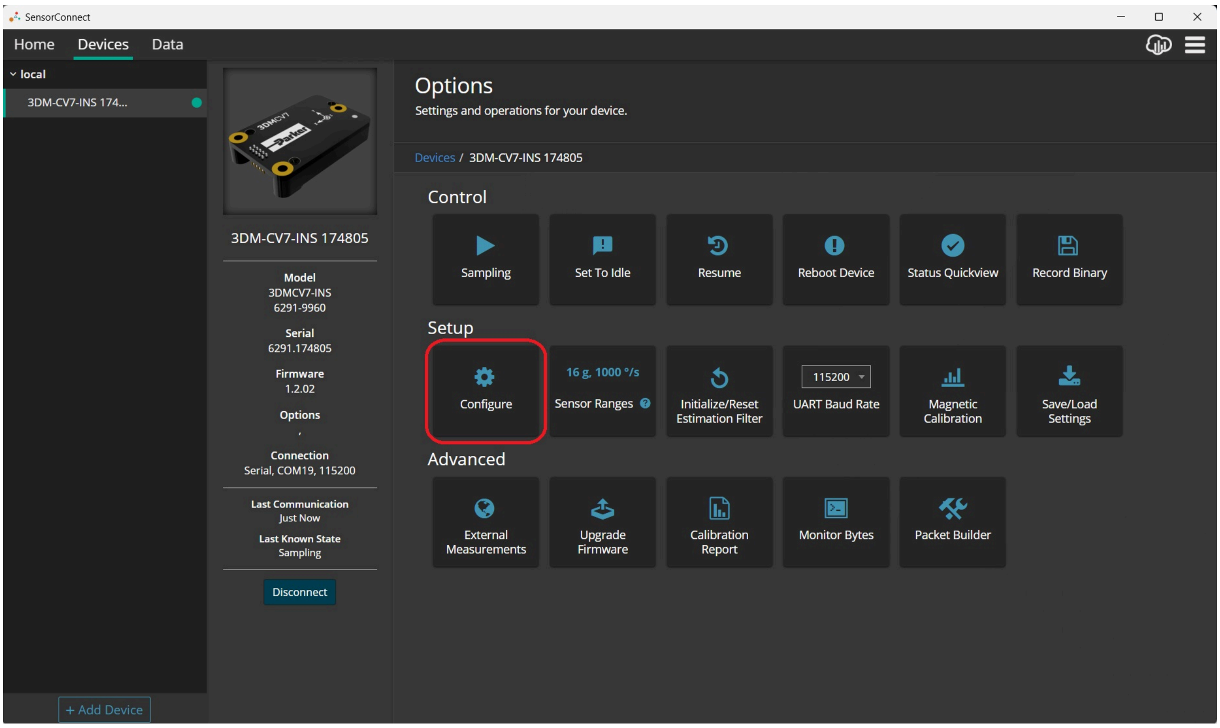
You are here: Coordinate Frames > Aiding Frames

Aiding Frames

The Aiding **Frame Configuration (0x13,0x01)** command enables the user to mount an **External Aiding Sensor** at a desired location on the vehicle and to specify the parameters of that frame. This includes an offset vector and a rotation with-respect-to the vehicle frame, eliminating the need to calculate complex frame transformations for the aiding data. These frames account for the effect of vehicle dynamics on external aiding measurements and their errors can be tracked during runtime.

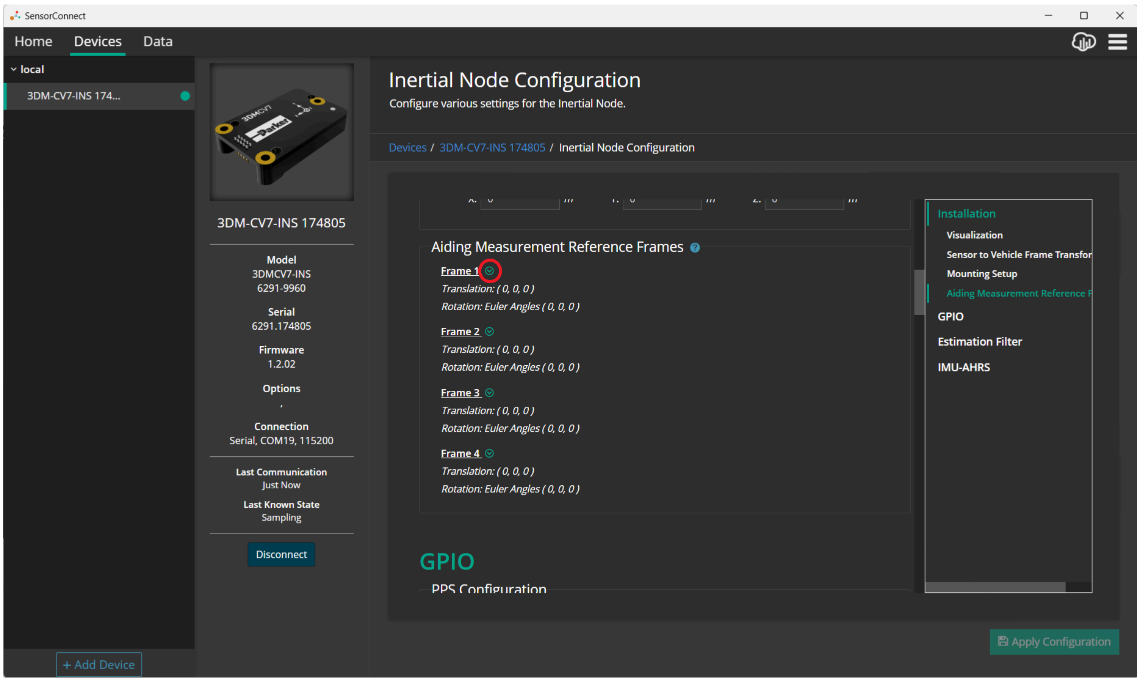
Example: Configure external GNSS, downward facing radar, external magnetometer, optical flow sensor

SensorConnect

1. Open **SensorConnect**
2. Connect the 3DM-CV7-GNSS/INS to your computer
3. Enter the Configure tile
    1. 

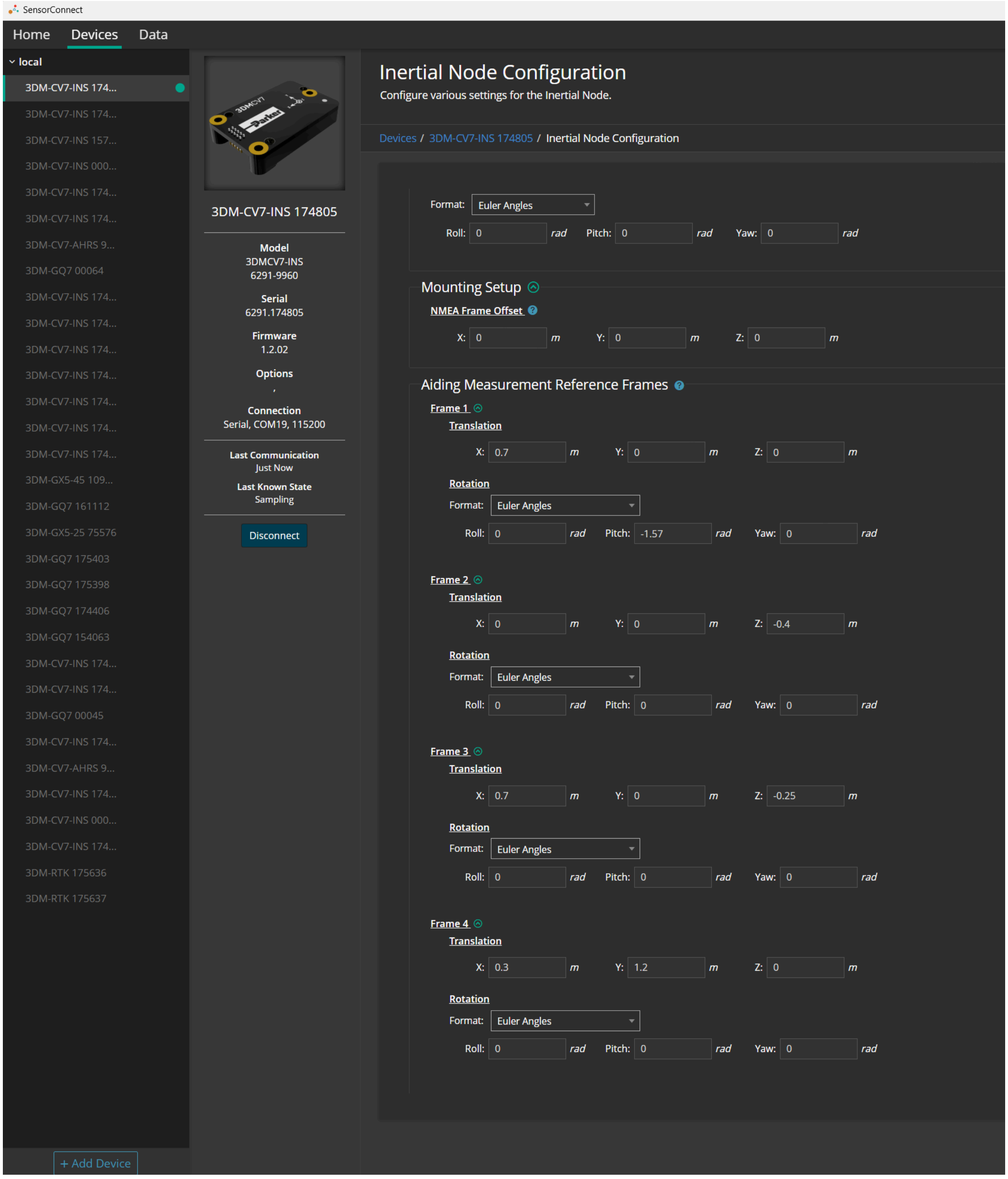
4. Scroll down to the Installation > Aiding Measurement Reference Frames section

    1. Click the Frame 1 down carrot to show the aiding frame configuration fields

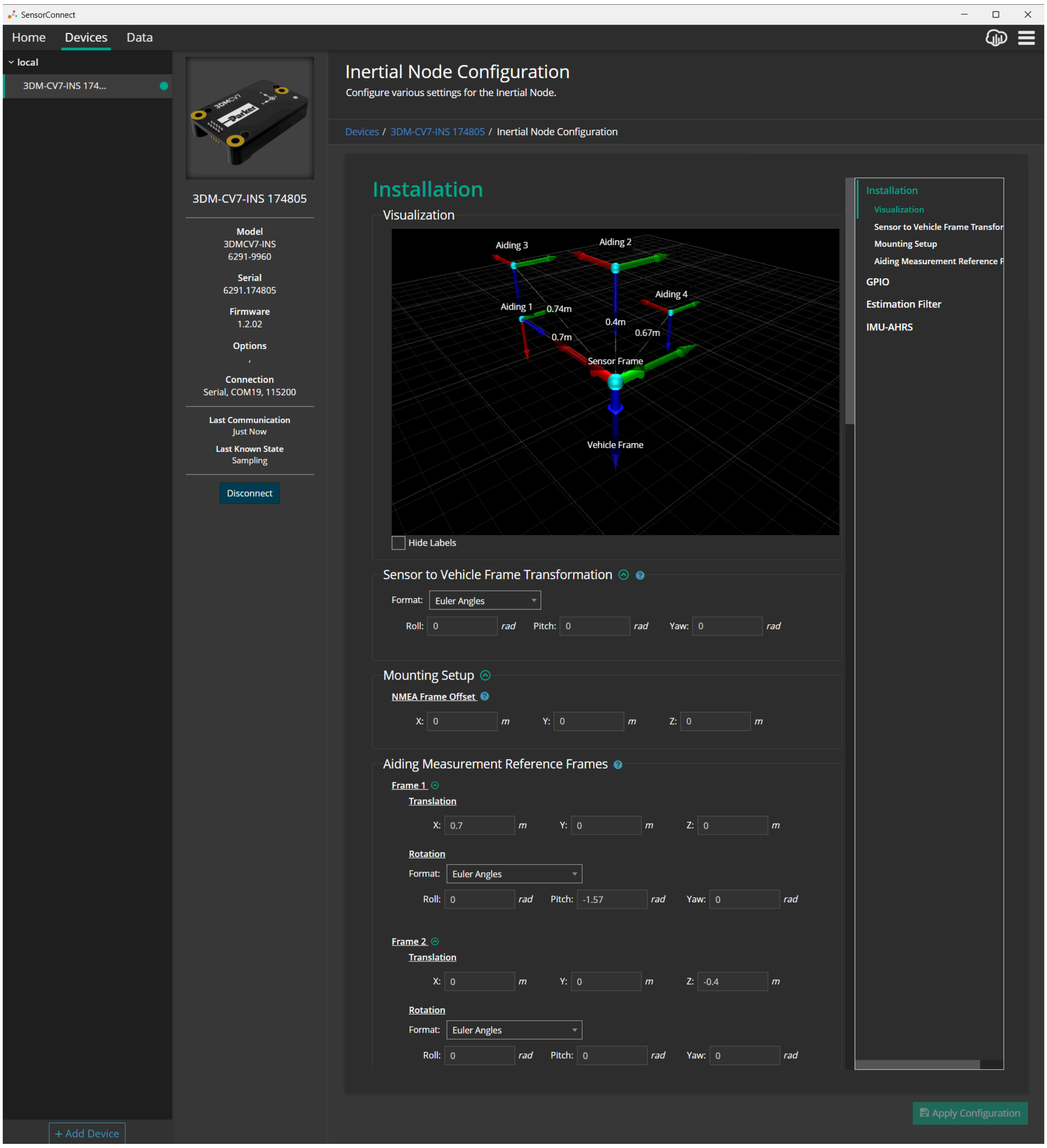
        1. 

        2. Assign each sensor a Sensor ID and a Aiding Frame as appropriate

        3. Enter the Translation and Rotation for Sensor 1 in Frame 1, Sensor 2 in Frame 2, ...

1. 

2. Verify the Translation and Rotation values are correct by scrolling up to view the Installation > Visualization window:

1. 

3. Click Apply Configuration

4. **Save Settings to Non-Volatile Memory:**

    1. If you intend for this configuration to be persistent through power cycles or if the device will be power cycled before connecting to the external GNSS receiver, it is recommended to "save as startup settings" before removing power. This can be performed using the **Device Settings (0x0C,0x30)** command with the SAVE function selector. For instructions using other interface methods, refer to the **FAQ: Do settings get erased from the** 3DM-CV7-GNSS/INS **if I unplug it?**

MIP SDK

The **Software** page contains a link to the MIP SDK Github, which has examples of the 3DM-CV7-GNSS/INS's aiding frames in use.

ROS

ROS employs aiding frames through the use of Transforms. This is described in detail on the microstrain_inertial_driver ROS Wiki Transform page.

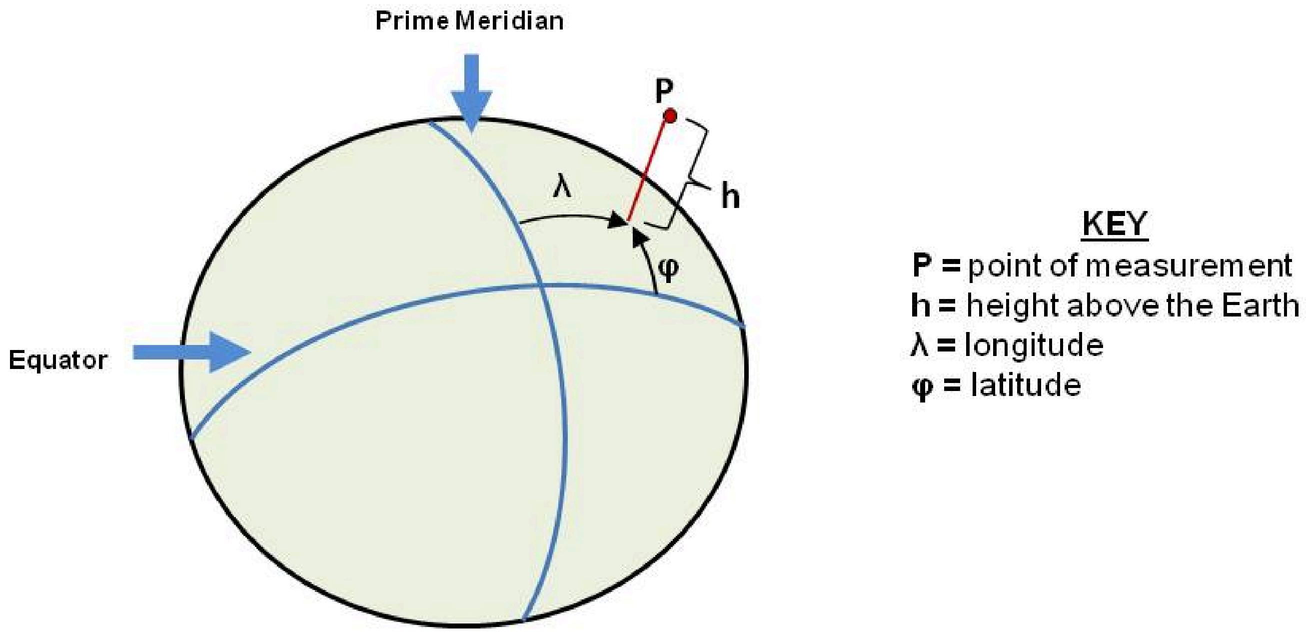
You are here: Coordinate Frames > Geodetic Frame

Geodetic Frame

The World Geodetic System is the standard for cartography and navigation. The latest revision, WGS84, is the reference coordinate system for GPS, and the 3DM-CV7-GNSS/INS reports position using this coordinate frame. It also calculates the magnitude of the local gravity vector using the WGS84 reference formulas.

The WGS coordinates are latitude (φ), longitude (λ), and height (h) above the reference ellipsoid. Latitude ranges from -90 degrees at the South Pole to 90 degrees at the North Pole. Longitude ranges from -180 to 180 degrees, with 0 degrees being the prime meridian. The -180/180 degree switchover occurs in the middle of the Pacific Ocean and includes a section of the International Date Line. The model takes into account the oblateness of the Earth’s surface.

A point (P) on or above the Earth in the WGS84 coordinate system is notated as: latitude (φ),longitude (λ), and height above the reference ellipsoid (h).



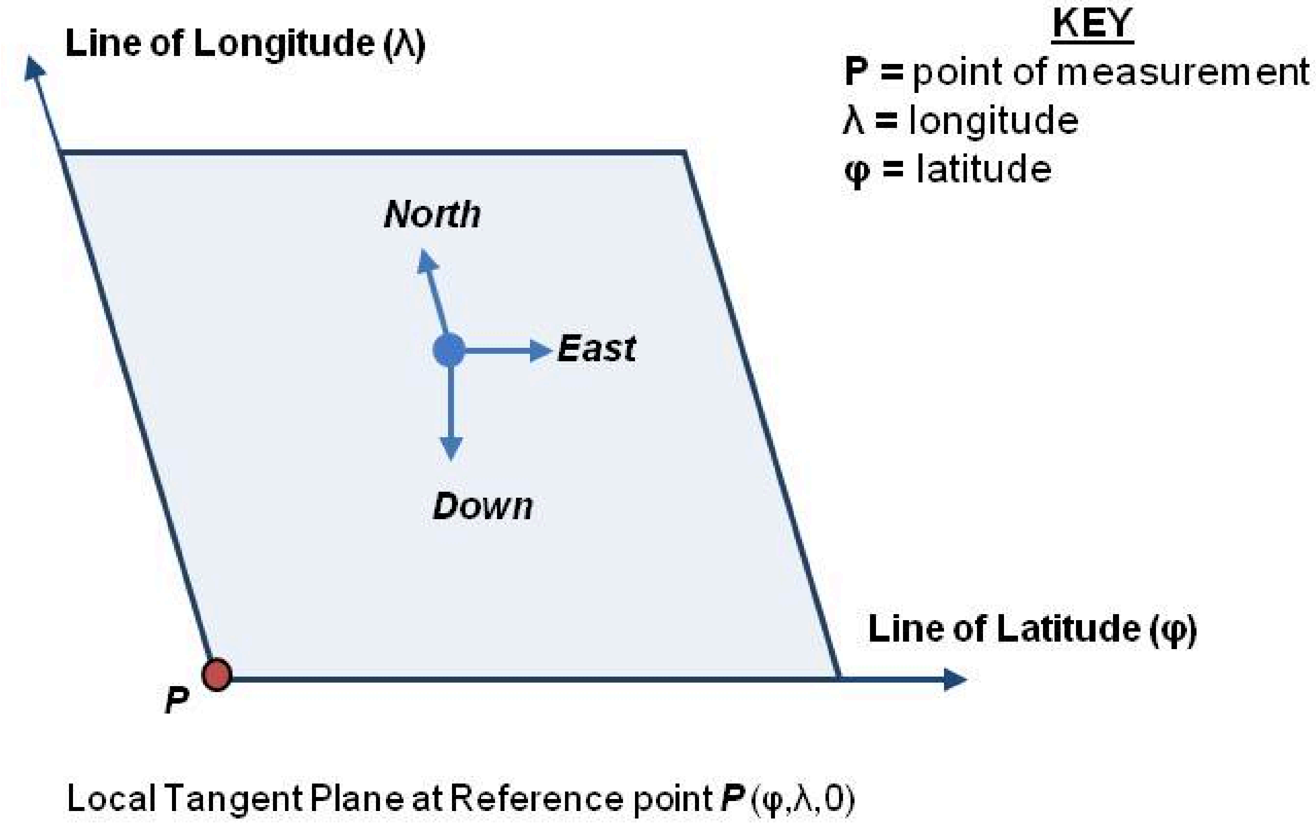
World Geodetic System (WGS84) Reference Ellipsoid

You are here: Coordinate Frames > NED Frame

NED Frame

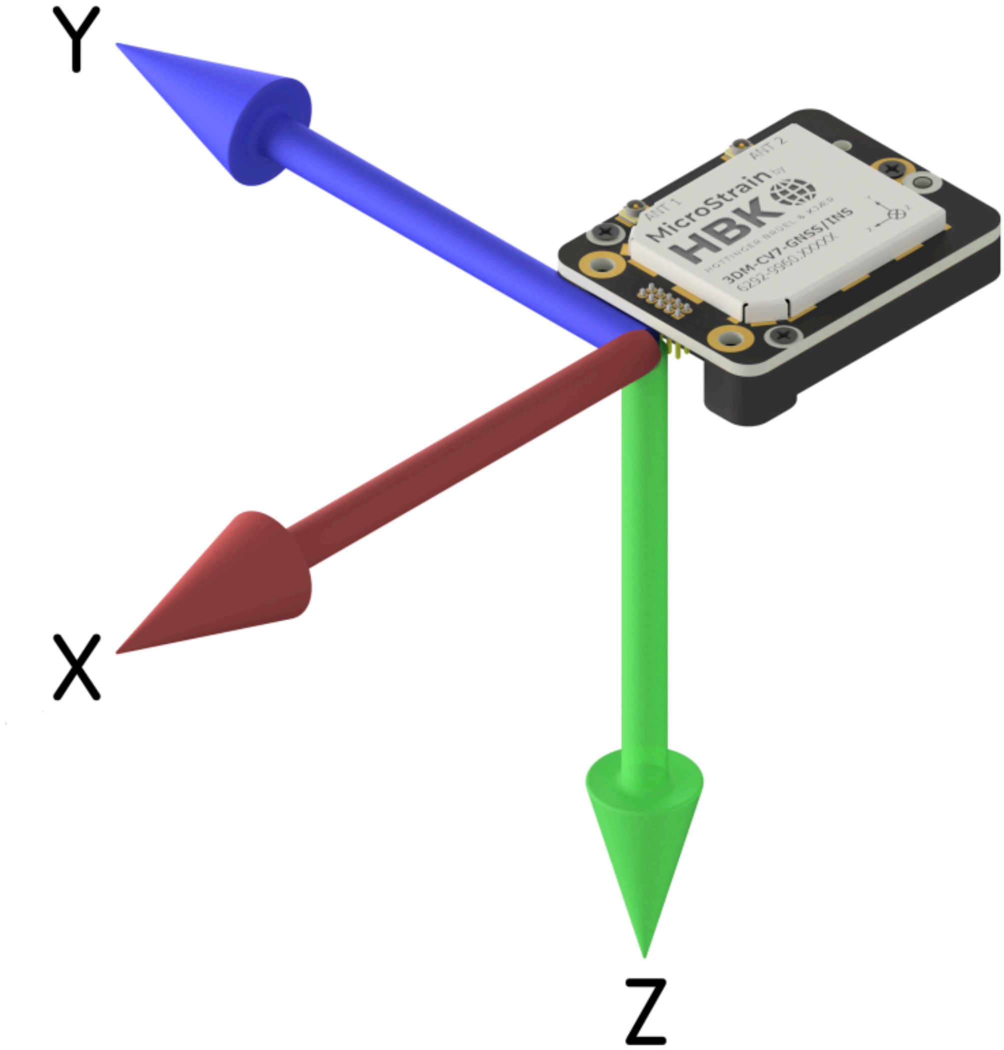
The North-East-Down (NED) frame is a local coordinate frame, which is formed by a tangent plane located at a particular point (current coordinates) on the WGS84 reference ellipse. The NED frame is constructed with the (true) North vector along the line of longitude, the East vector along the line of latitude, and the Down vector normal to and towards the tangent plane. The assumption when using the NED frame is that the local surface can be reasonably approximated by a flat plane. For most applications, this assumption is valid and provides a more intuitive reference frame for expressing velocity and attitude information than a global frame.

The 3DM-CV7 reports velocity in this frame and attitude with respect to this frame.



North East Down Frame

You are here: [Coordinate Frames](#) > Sensor Frame



Sensor Frame

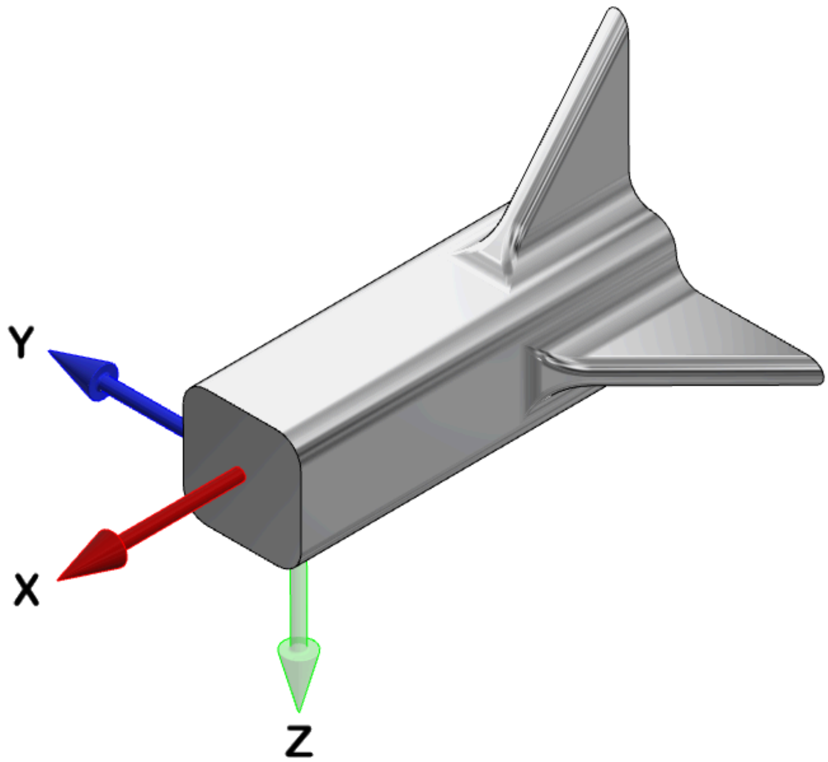
The sensor frame is indicated on the top of the device and shown in the diagram above. The sensor frame is oriented such that the x-axis vector is parallel with the long side of the sensor and points toward the sensor connector, the y-axis is 90° to the right of the x-axis, and the z-axis goes through the bottom of the sensor (outward). These axes were selected so that when the connector on the device is pointed north and the device is upright and level, the sensor frame will match the **NED Frame** exactly, giving zero rotation.

In Euler angles, the x-axis is the roll angle, the y-axis is the pitch angle, and the z-axis is the yaw angle.

If no sensor-to-vehicle transform has been specified (see **Vehicle Frame** for more details), then the 3DM-CV7 reports acceleration, angular rate, delta-theta, and delta-velocity in the sensor frame. Inertial sensor biases and corrections are always reported in the sensor frame.

Vehicle Frame

The image below defines the following vehicle coordinate frame: the x-axis points in the forward direction, the y-axis points out the right-hand side of the vehicle, and the z-axis points down.

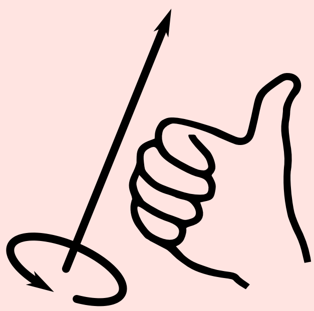


Transformation to Vehicle Frame

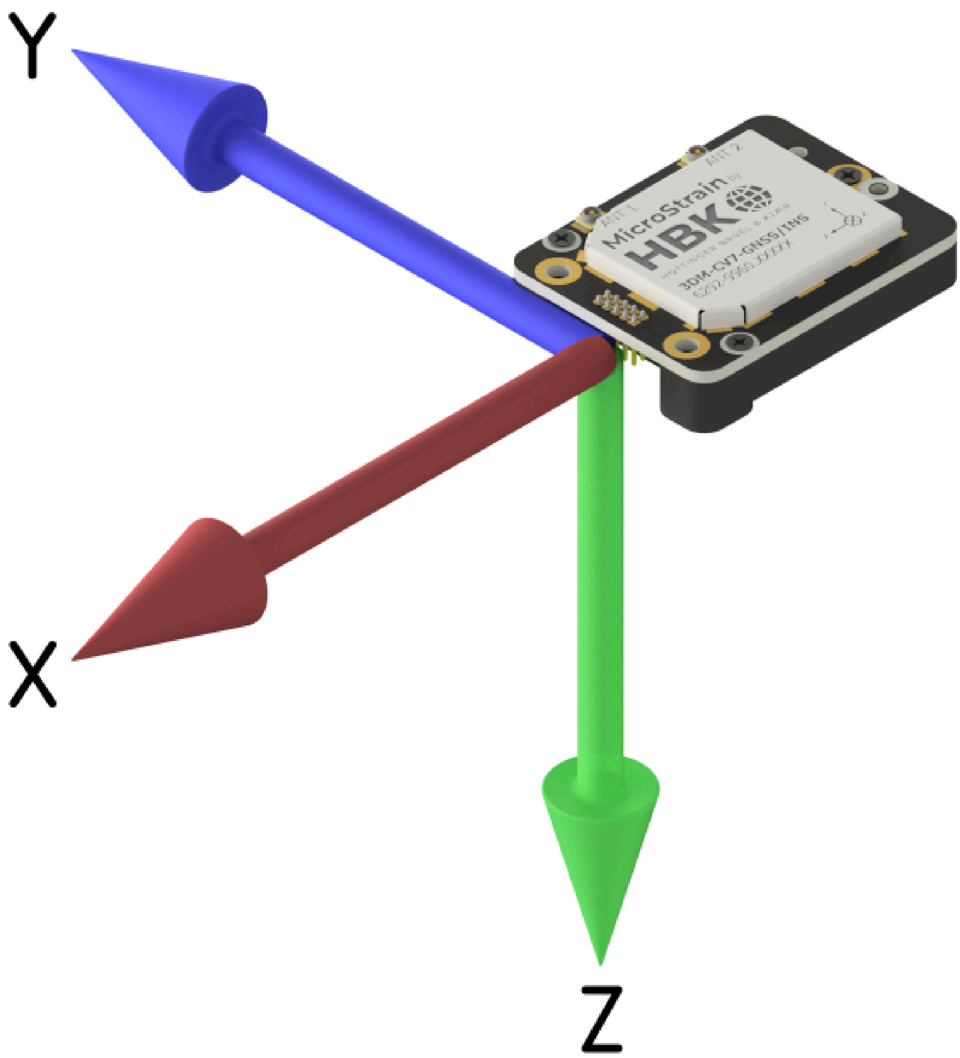
The 3DM-CV7 provides commands that define an orientation transformation that mathematically aligns the 3DM-CV7 sensing axes with the vehicle axes. These commands are useful when the sensor cannot be mounted in the same location or orientation as the desired reference point on the vehicle frame.

The transformation is expressed as a rotation from the sensor frame to the vehicle frame. The transform can be applied in either Euler angles using **Sensor to Vehicle Frame Transformation Euler (0x0C,0x31)** or with a direction cosine matrix using **Sensor to Vehicle Frame Transformation Direction Cosine Matrix (0x0C,0x33)**.

IMPORTANT: Transforms in Euler angles must be achieved by applying rotations around the sensor axes in the following order (starting with the sensor oriented in the vehicle frame and ending in its actual mounting orientation): yaw (z-axis), pitch (y-axis), and roll (x-axis). A positive rotation around an axis is defined by the right hand rule. The fingers curl in the positive direction.

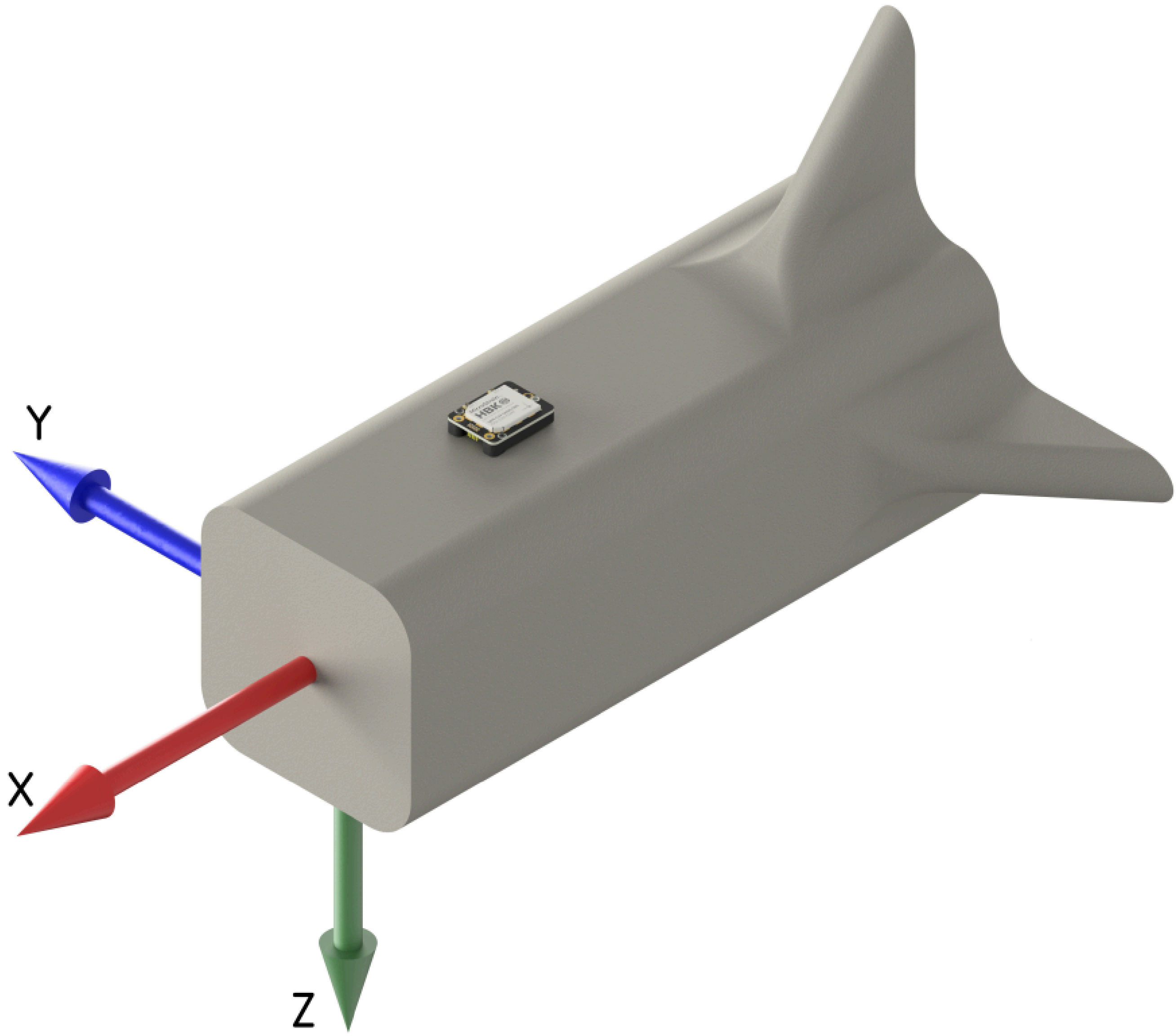


The following examples illustrate how to apply a transform using both sensor to vehicle transform commands. The **Sensor Frame** is shown in the diagram below.



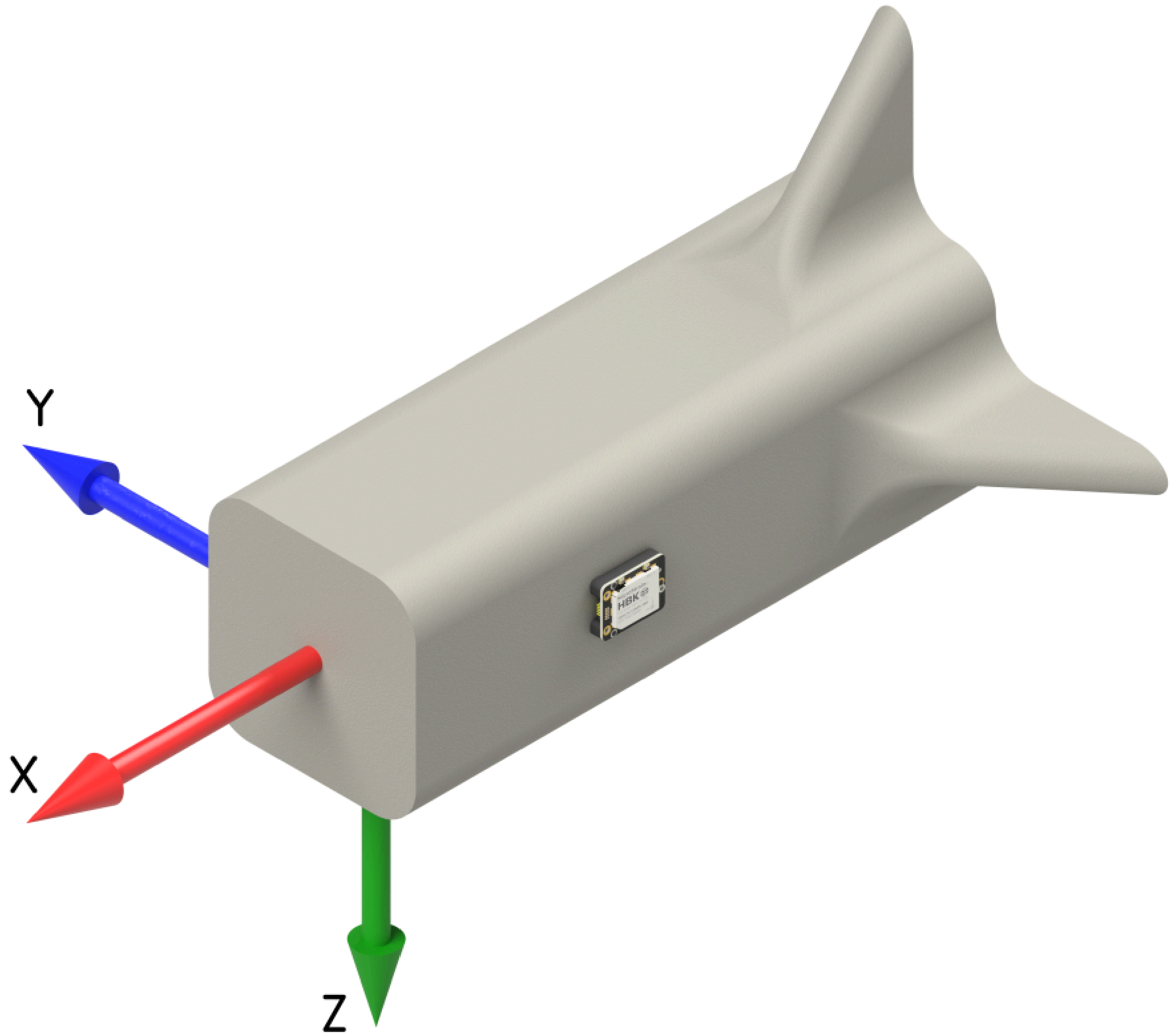
Example 1

In this example, the sensor frame is aligned with the vehicle frame. No transform has to be applied.



Example 2

In this example, the transform can be applied without a rotation around the sensor's yaw or pitch axes. A -90 degree (-1.5708 radian) rotation around the sensor's roll axis completes the transform.

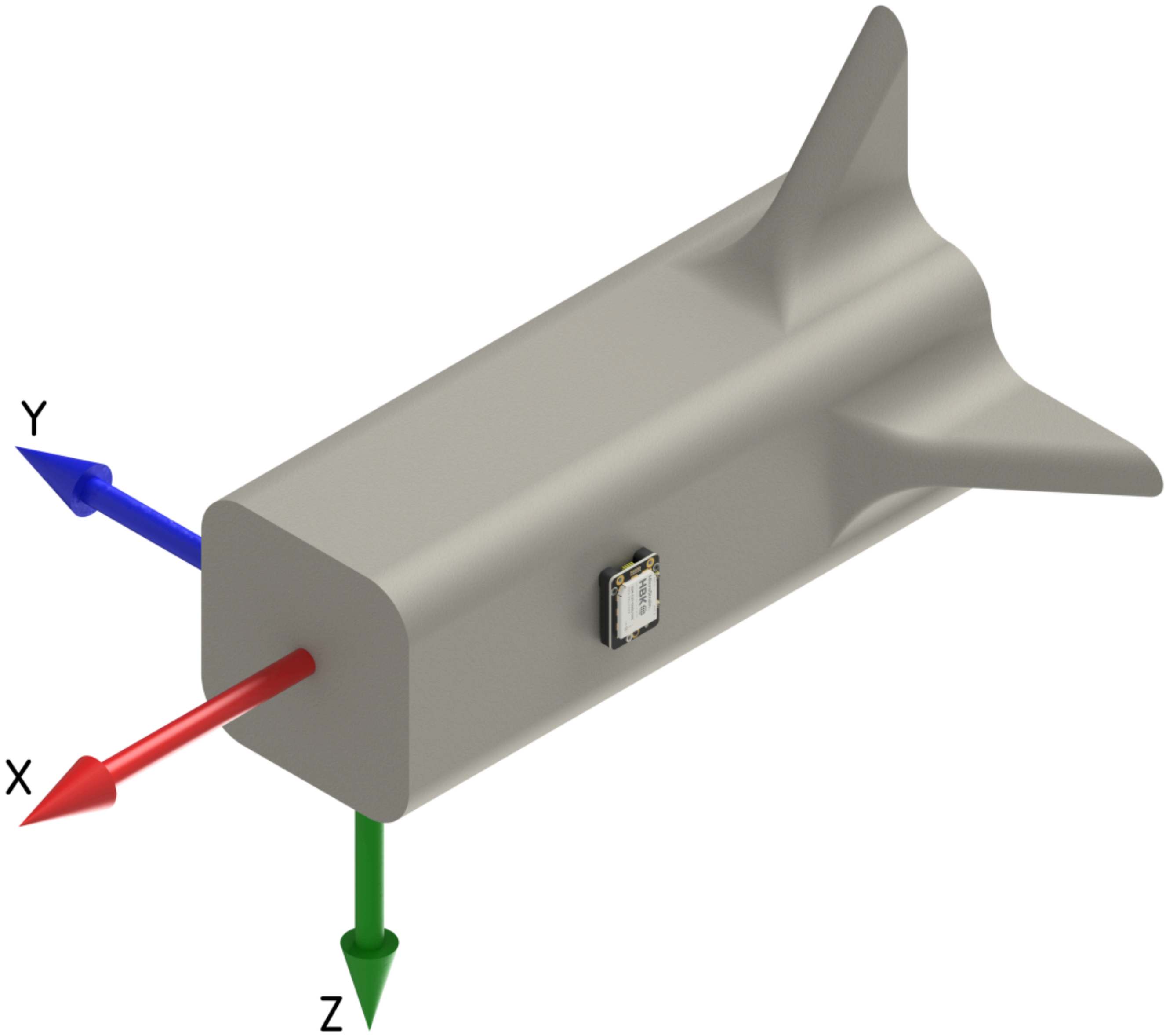


Sensor to Vehicle Frame Transformation Euler (0x0C,0x31) Command Parameters	Values	Units
Roll	-1.5708	radians
Pitch	0	
Yaw	0	

Sensor to Vehicle Frame Transformation Direction Cosine Matrix (0x0C,0x33) Command Parameters in Row Major Order (as required)									
Row, Col	**1,1**	**1,2**	**1,3**	**2,1**	**2,2**	**2,3**	**3,1**	**3,2**	**3,3**
Value	1.0	0.0	0.0	0.0	0.0	1.0	0.0	-1.0	0.0

Example 3

In this example, the transform is performed by first applying a +90 degree (+1.5708 radian) rotation around the sensor's yaw axis, followed by +90 degree (+1.5708 radian) rotation around the sensor's pitch axis. No rotation around the sensor's roll axis is required.

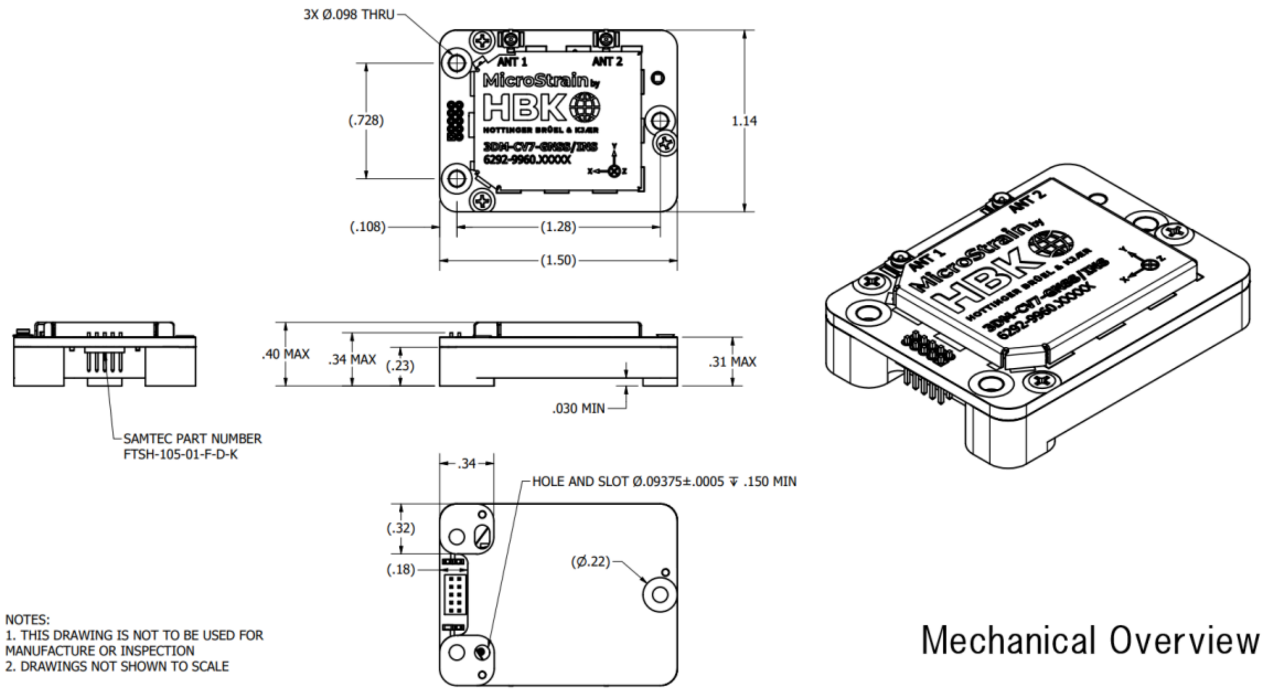


Sensor to Vehicle Frame Transformation Euler (0x0C,0x31) Command Parameters	Values	Units
Roll	**0**	radians
Pitch	**1.5708**	
Yaw	**1.5708**	

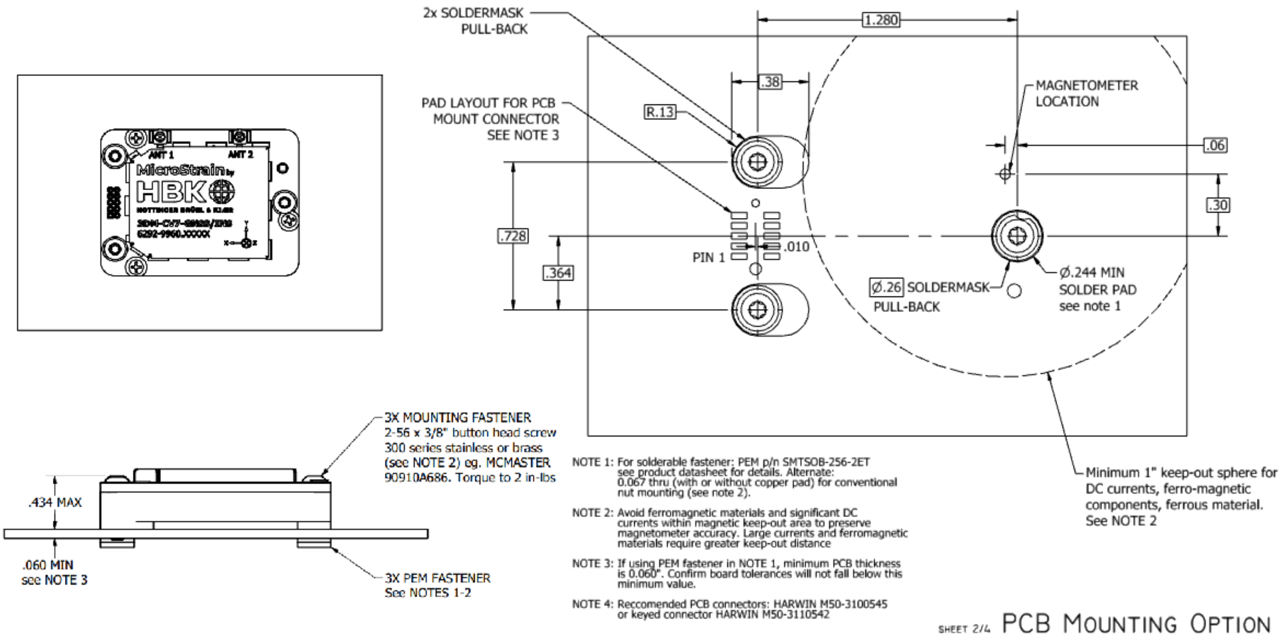
Sensor to Vehicle Frame Transformation Direction Cosine Matrix (0x0C,0x33) Command Parameters in Row Major Order (as required)									
Row, Col	**1,1**	**1,2**	**1,3**	**2,1**	**2,2**	**2,3**	**3,1**	**3,2**	**3,3**
Value	0.0	-1.0	0.0	0.0	0.0	1.0	-1.0	0.0	0.0

Specifications

Mechanical

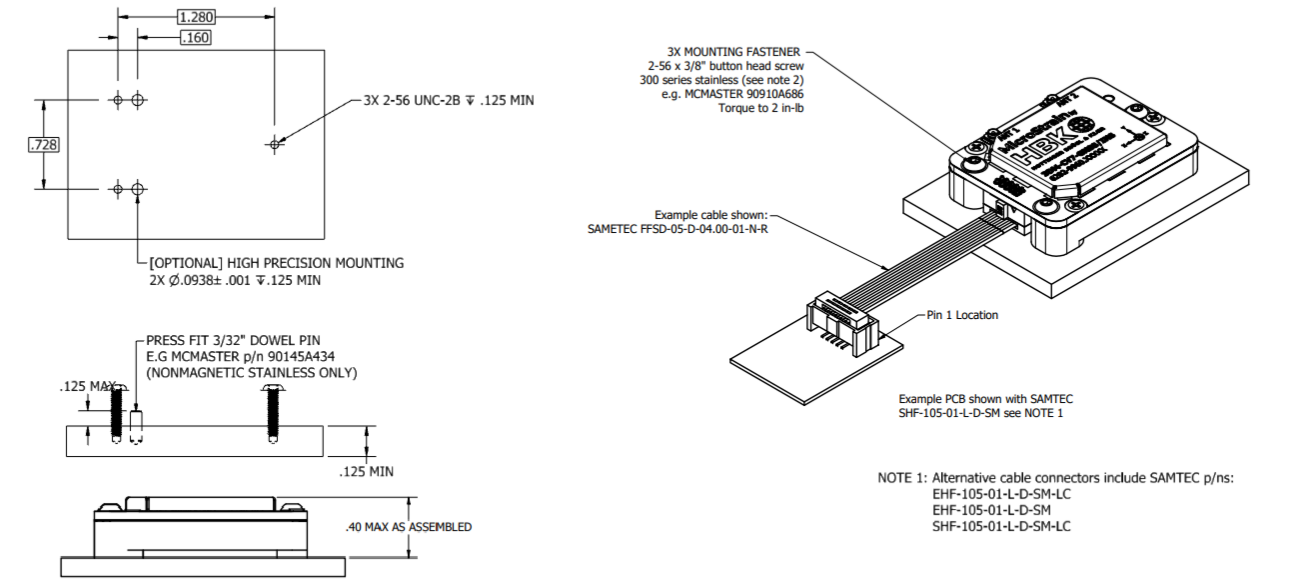


Mechanical Overview

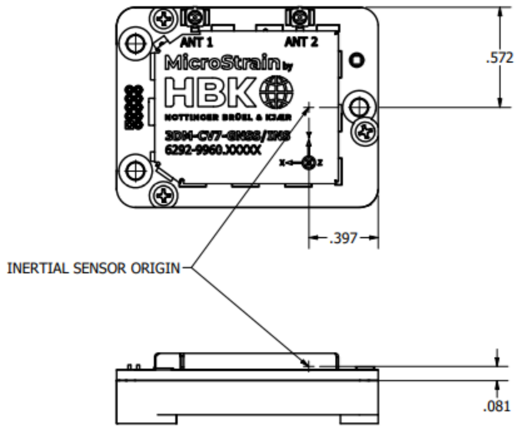


SHEET 2/4 PCB MOUNTING OPTION

CHASSIS MOUNTING HOLE PATTERN



SHEET 3/4 CHASSIS MOUNTING OPTION

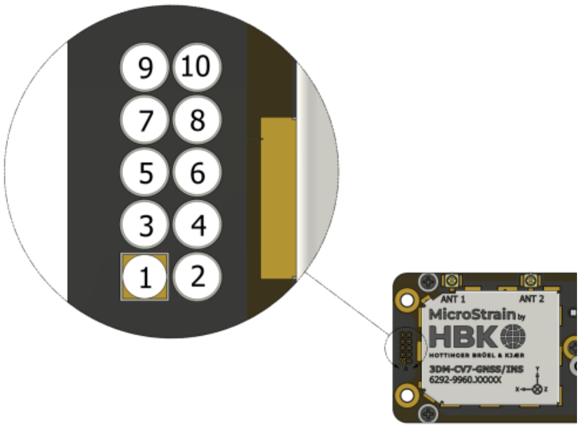

ANT 1
ANT 2
MicroStrain by
HBK
HOTTINGER BRÜEL & KJÆR
3DM-CV7-GNSS/INS
6292-9960.XXXXX
.572
.397
INERTIAL SENSOR ORIGIN
.081

You are here: Specifications > Electrical

Electrical

Pin Definitions

Pin #	Pin	Function
1	USB DM	USB Data -
2	USB DP	USB Data +
3	Vin	Power Supply
4	RxD	UART Receive (host to 3DM-CV7)
5	TxD	UART Transmit (3DM-CV7 to host)
6	GPIO3	Logic Level GPIO
7	GPIO1	Logic Level GPIO
8	GND	Signal & Power Ground
9	GPIO2	Logic Level GPIO
10	GPIO4	Logic Level GPIO



See **User GPIO** for details on how to use the GPIO pins and supported features.

Absolute Maximum Ratings

Exposure to stresses beyond those listed under **Absolute Maximum Ratings** may cause permanent damage to the device. Exposure to the **Absolute Maximum Ratings** for extended periods may affect device reliability and lifetime. **Absolute Maximum Ratings** are stress ratings only and do not impl functional operation of the device under any conditions beyond those listed in **Recommended Operating Conditions**.

Parameter[1]	Conditions	Min	Max	Units
Power Supply Voltage (Vin)[2]		10	10	V
GPIO, RxD Voltage [3]	Referenced to ground	-0.7	+7	V
TxD Current		-10	+10	mA
Voltage on USB pins	When device is operating	0	+5	V
Voltage on Chassis		-12	12	V

1. Reverse polarity and overvoltage protected. Operation is not guaranteed over this range.
2. Input voltage in this range may be applied even when the device is powered off.

Recommended Operating Conditions

Stresses beyond the limits in **Recommended Operating Conditions** will likely result in the improper functioning of the device. The voltages in **Recommended Operating Conditions** represent voltages as measured at the pins; they do not account for voltage drops in the cable.

Specification[1]	Min	Typ	Max	Units
Power Supply Voltage (Vin)	+3.2		+5.2	V
GPIO, RxD Input Voltage	0	0 to Vin	5.5	V

Power Requirements

Specification	Typ	Max	Units
Power Consumption	1.00	1.25	W

Electrical Characteristics

Parameter[1]	Conditions	Min	Typ	Max	Units
Power Consumption	-40°C		0.87		W
	25°C		**1.00**		
	85°C		1.25		
GPIO, TxD Output Low Voltage	No Load		0.01	0.4	V
GPIO, TxD Output High Voltage		2.6	2.99		
GPIO, TxD Output Low Voltage	300 uA load		0.3	0.7	V
GPIO, TxD Output High Voltage		2.3	2.7		
GPIO Output Impedance			1k		Ω
GPIO, RxD Input Low Threshold		1.1			V
GPIO, RxD Input High Threshold				2.1	
GPIO Input Pullup/Pulldown Resistance		30	40	50	kΩ
Over Voltage Threshold				5.70	V

Mechanical Shock Limits

The accelerometers used by the IMU are sensitive instruments that can suffer mechanical damage if subjected to large or frequent accelerations beyond their rated range while in operation. To protect themselves from occasional over-range events, the accelerometer's control circuitry will temporarily reduce drive power to the sense element if acceleration exceeds twice the full-scale range. While this protective mode is enabled, the reported acceleration values will be inaccurate and will appear to drift towards zero. Normal operation will resume after sensed acceleration returns to an acceptable level.

It is essential to note that, even with this automatic protection circuitry, repeated exposure of the accelerometer to shock and vibration acceleration levels exceeding twice its full-scale range will cause excessive mechanical wear. Over time, damage can accumulate and may cause the device to become defective. Care should be taken not to subject the sensor to accelerations beyond twice the full scale range during normal operating conditions.

The 3DM-CV7 has been tested to 500 g, 3 times with no calibration loss and outputs still within range.

Protocol Support and Specifications

This information is provided for reference purposes only. Data is subject to change between products and firmware versions. Unless noted elsewhere, MicroStrain by HBK reserves the right to make changes to these values without notice.

If you are developing software which configures the 3DM-CV7, where possible, it should always obtain information from the device itself. Avoid hard-coding these values in your software.

Descriptor Sets

Descriptor Value	Name	Base Rate
0x80	Sensor Data	1 kHz
0x82	Filter Data	1 kHz
0x91, 0x92	GNSS Data, Internal Receiver 1 / 2	5 Hz
0x93	RTK Data	0 (Data is transmitted as it is received)
0x94	External GNSS1	0 (Data is transmitted as it is received)
0xA0	System Data	1 kHz

Supported commands and data quantities (including descriptor sets) can be obtained programmatically via the **Get device descriptors (0x01,0x04)** and **Get device descriptors (extended) (0x01,0x07)** commands.

The actual base rate should be obtained using the **Get Data Base Rate (0x0C,0x0E)** command. It is **strongly recommended** to query the base rate directly from the device as it is model specific and could change with future firmware versions.

Anti-aliasing Lowpass Filter Support

The **Lowpass Filter (0x0C,0x54)** command can be used to configure the final low-pass filter on the following data quantities. Other quantities are unfiltered.

Descriptor Set	Data quantity
Sensor Data	Scaled Accel (0x80,0x04)
	Scaled Gyro (0x80,0x05)
	Scaled Mag (0x80,0x06)
	Scaled Pressure (0x80,0x17)
EKF Filter Data	Compensated Acceleration (0x82,0x1C)
	Comp Angular Rate (0x82,0x0E)
	Linear Accel (0x82,0x0D)

Comm Ports

These ports numbers are used with the **Comm Port Speed (0x01,0x09)** command.

Port	ID	Supported baud rates	Supported Protocols
UART 1	0x11	115200, 230400, 460800, 921600, 3000000	MIP, NMEA, RTCM, SPARTN
UART 2	0x12	4800, 9600, 19200, 38400, 57600, 115200, 230400, 460800, 921600	MIP, NMEA, RTCM, SPARTN
UART 3	0x13		

Events

Feature	Firmware Version 1.2.02+
Event Tick Rate	**1 kHz / 1 ms period** (always matches system data set base rate)
Max Triggers	**12**
Max GPIO Triggers	4
Max Threshold Triggers	8
Max Combination Triggers	8
Max Actions	**12**
Max GPIO Actions	6
Max Message Actions	8

Event support can be determined programmatically using the **Get Supported Events (0x0C,0x2A)** command.

FAQ

General:

- Should I use Serial or USB?
- Do settings get erased from the 3DM-CV7 if I unplug it?
- How do I upgrade firmware?

GNSS:

- Can I use a different GNSS antenna than the default U-Blox antenna?
- Does the 3DM-CV7 have anti-spoofing or anti-jamming capability?
- How do I measure the antenna offsets?
- What are the mounting requirements for using Dual-Antenna heading?
- How do I improve dual antenna heading performance?

RTK:

- Why did the 3DM-CV7 lose an RTK Fix?
- Is the 3DM-CV7's RTK solution guaranteed?
- Is the SensorCloud RTK corrections stream guaranteed?

Navigation Filter:

- Why is the filter stuck in Vertical Gyro Mode?
- How do I improve Navigation Filter performance?
- How do I know when I can trust the Navigation Filter solution?
- How do I interpret the Filter Status (0x82,0x10) message?
- Can the 3DM-CV7 navigate during a GPS outage?

Support

- How do I obtain Engineering Support for my device ?

Software:

- Does MicroStrain by HBK have data visualization tools?
- Does MicroStrain by HBK have ROS drivers?

- Does MicroStrain by HBK have PX4 drivers?
- Does MicroStrain by HBK have ArduPilot drivers?

General

Do settings get erased from the 3DM-CV7 if I unplug it?

- It depends. The 3DM-CV7 will use the existing "startup settings" stored in the device unless the settings are saved prior to power down. To save the settings, do one of the following:
  - In SensorConnect, click “Save as Startup Settings” in the “Save/Load” tile.
  - In SensorConnect, click the checkbox “Save as Startup Settings” in the “Streaming” tile.
  - In ROS, set the "save_settings" param to true and successfully launch the node
  - In MSCL, use the node.saveSettingsAsStartup() function.
  - Use the **Device Settings (0x0C,0x30)** command with the SAVE function selector.
- In SensorConnect, you can also export a .json settings file to your local computer that will save your full device configuration, which can be loaded again in the future.

How do I upgrade firmware?

- Please see the **Firmware Upgrades** page for methods and instructions on upgrading device firmware.

RTK

Why did the 3DM-CV7-GNSS/INS lose an RTK Fix?

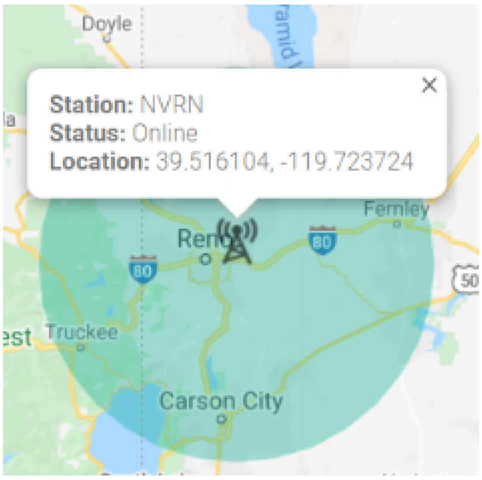
- The most common cases of losing an RTK Fix are:
  1. Loss of comms to RTK corrections: cell signal dropped, radio line of sight is obstructed, satcom obstructed, WiFi out of range, etc. Check comms
  2. Poor sky view / "high multipath": trees, buildings, parts of vehicle obstructing a 45° inverted cone above each antenna. Ref **Antenna(s)** page Multipath

3. EMI (Electro Magnetic Interference): From antennas or antenna cable coiled near compute, lights, motors, relays. Ref **Antenna(s)** page EMI
4. No groundplanes: groundplanes under patch antennas is critical to their satisfactory operation. The default 3DM-CV7-GNSS/INS antennas do not have groundplanes built in and unless mounted on a flat metal surface, like a roof of vehicle, can struggle. Ref **Antenna(s)** page Multipath
5. Incorrect credentials: manually selected base station mountpoint is incorrect and too far away (rover RTK error increases at 1cm + 1mm/km from base station). Check mountpoint is correct
6. Base station offline: review provider's coverage map

Is the SensorCloud RTK corrections stream guaranteed?

- We simplify the network and interface process to provide you an all-in-one navigation solution. This solution includes our 3DM-CV7, 3DM-RTK and **SensorCloudRTK** service. We work with RTK and cellular providers to deliver the SensorCloudRTK service. As we do not own those provider's networks we cannot control their operational behavior. **SensorCloudRTK Coverage Map** provides a real-time database showing base stations' ID, location, operational status and range as shown in Figure: SensorCloudRTK Example Base Station Info

Figure: SensorCloudRTK Example Base Station Info



Do I need to use the 3DM-RTK and SensorCloudRTK to receive RTK corrections?

- No, we have outlined several ways for the 3DM-CV7 to receive RTK corrections in the "**4 Ways to RTK**" app note and many more have been added since.

- For ROS users, head to the **ROS NTRIP client** node, compatible with our **ROS1** and **ROS2** drivers.

Navigation Filter

Why is the filter stuck in Vertical Gyro Mode?

- The filter is likely stuck in Vertical Gyro Mode because it does not have sufficient initial conditions. See **Filter States** for more info on Vertical Gyro Mode.
- In Vertical Gyro Mode, the Navigation Filter is only reporting valid gyro stabilized pitch, roll, and the change in heading relative to the initial orientation. To transition to Full Navigation mode, the Navigation Filter requires the initial conditions for all states shown in the table below. These can be provided to the Navigation Filter through the **Initialization Configuration (0x0D,0x52)** command. The time to transition from power on to Full Navigation mode under ideal conditions is typically less than 3 seconds.

Filter Mode	Filter State Value	Initial Conditions Required	Available Outputs
Initialization	1	None	None
Vertical Gyro	2	Roll, Pitch (from IMU)	Roll, pitch, relative heading
AHRS	3	Absolute heading	Roll, pitch, absolute heading
Full Navigation	4	Absolute heading, initial position, initial velocity	Roll, pitch, absolute heading, position, velocity

- Ensure 3DM-CV7 is configured correctly - see Filter Aiding Measurements Filter Aiding Measurements and **Filter Initialization**.
- If the 3DM-CV7 hasn't converged in 1 minute, review **How do I improve Navigation Filter performance?**, and **How do I know when I can trust the Navigation Filter solution?**

How do I improve Navigation Filter performance?

The time to transition to the Full Navigation filter mode is less than 3 seconds under ideal conditions. The following tips commonly resolve cases where convergence takes longer than expected:

- Ensure that your firmware is updated to the latest version.

- Ensure antennas are correctly installed, and if using dual antenna, that their offsets are not accidentally swapped (a common problem).
- Ensure the proper **Filter Aiding Measurements** are enabled.

How do I know when I can trust the Navigation Filter solution?

It is highly recommended to read the **Navigation Filter** section of the manual. You will need to configure the device to stream the data fields that contain the status information required for your application. Upon receipt of the status fields, you will need to set thresholds for the data based on your application's requirements, for example:

- The **Status (0x82,0x10)** message provides two important pieces of information for the user: the current **Filter State** (AKA mode) and **Status Flags** that provide real-time warnings about the filter's condition and its estimates.
  - The **Status (0x82,0x10)** message will report values of **4**, **1**, **1**, when the Navigation Filter is in normal operation. These numbers correspond to Full Navigation Filter State, Default Dynamics Mode, and Converged filter Status Flags.
  - See **"Interpreting the Status data channel"** below for more info.
- Filter Position Uncertainty Messages
  - The user must decide what position uncertainty values are required for their application. Typical 1σ values are:
    - Single point: 1.25-2.5m
    - RTK-Float: 0.1- 0.8m
    - RTK-Fixed: 0.01 - 0.06m (error model assumes maximum base station baseline of 50 km)
  - Position uncertainty is available in different frames in the following messages:
    - **LLH Position Uncertainty (0x82,0x08)**
    - **ECEF Position Uncertainty (0x82,0x36)**
- **Euler Angles Uncertainty (0x82,0x0A)**
  - Set application specific Yaw (heading) uncertainty threshold
  - Verify uncertainty is below threshold

How do I interpret the Status (0x82,0x10) message?

The data fields reported by the **Status (0x82,0x10)** message are the **Filter State**, Dynamics Mode, and **Status Flags**. The Status Flags is a bitfield that must be properly interpreted:

- SensorConnect's Status QuickView performs the conversion for the user into easy to understand text.
- The ROS drivers RQT Status QuickView GUI performs the conversion for the user into easy to understand text.
- If converting manually, a simple online decimal to binary converter or calculator application in "programmer mode" will work.
- For embedded applications, a simple bitshift calculation is required.
- We recommend the user checks for the Filter state = 0x04 and Status flags = 0x01 or 0x02 (i.e. no warnings are raised and the filter condition is either stable or converging) at minimum to verify satisfactory filter performance.

Ex: What does **4, 1, 1025** mean from **Status (0x82,0x10)**?

- One can use any decimal to binary converter, we see the 0th and 10th bit have values of 1.
- The 0th and 1st bits correspond to the Filter Condition,as described on the **Filter Status Flags** page. The 0th bit indicates a Filter condition of Stable. This is desired so is not the cause of concern.
- The 10th Status Flag bit corresponds to a time sync warning as seen in the table below with the 10th bit's row highlighted in yellow. This means no PPS source is detected and the recommended action is to check the PPS source is set correctly i.e. ensure an antenna is connected for the PPS source selected.

Bit #	Bit Value	Description	Potential Cause	Recommended Action
0-1	1	Filter condition (Stable/Converging/Unstable)	Insufficient initial conditions supplied.	Verify required conditions are properly enabled. Review **Filter States**.
2	0	Roll/Pitch Warning	Sensor is not aligned with vehicle frame.	Review **Vehicle Frame** page.
3	0	Heading Warning	Heading source is not valid. For Dual Antenna: Multipath, EMI, incorrect antenna offsets. For Magnetometer:time varying magnetic interference.	For Magnetometer:perform **Magnetometer Calibration,** check for time varying magnetic interference via **Scaled Mag (0x80,0x06)**, if present use external heading measurement.
4	0	Position Warning	Multipath, EMI, incorrect antenna offsets, antennas obstructed, antenna cable shorted. Review **How to improve filter performance** above.	Review: **Antenna** offsets, FAQ: Antenna sections, **FAQ:How to improve filter performance.**
5	0	Velocity Warning		
6	0	IMU Bias Warning	Gyro bias is high.	Perform **Capture Gyro Bias (0x0C,0x39)**

Bit #	Bit Value	Description	Potential Cause	Recommended Action
7	0	GNSS Clock Warning	Multipath, excessive vibration.	Not critical concern - continue navigation.
8	0	Antenna Lever Arm Warning	Antenna Lever Arm offsets are likely incorrect.	To improve performance, review **Antenna(s)** page.
9	0	Mounting Transform Warning	Transform is likely incorrect.	Review **Vehicle Frame** page.
10	1	Time Sync Warning	No PPS source detected.	Check **PPS** source is set correctly i.e. GNSS receiver 1 = PPS 1
12-15	0	Solution Error	Filter computation warning flags. If any bits 12-15 are set, all filter outputs will be invalid.	**Reset Navigation Filter (0x0D,0x01)**

- Note: the filter condition occupies 2 bits (bits 0 and 1) and can be interpreted as shown below (see **Filter Status Flags** page for more info).

Filter Condition Values	Description
1	Stable
2	Converging
3	Unstable/Recovering

Can the 3DM-CV7 navigate during a GNSS outage?

- Yes, the 3DM-CV7-GNSS/INS was designed with GNSS denied operation in mind. Although the 3DM-CV7 must have entered Full Navigation Mode (**Filter Mode 4**) prior the outage. If the 3DM-CV7 initially has a GNSS lock and experiences an outage, it will continue to provide a PVAT (Position, Velocity, Attitude, Time) solution throughout the outage.
- In cases where outages longer than 60 seconds are expected, and depending on the position accuracy requirements of the application, we typically recommend using additional **Filter Aiding Measurements** .
- For UAS and USV higher uncertainties may be allowable in which case the user may continue to navigate without these aiding measurements, while monitoring the navigation solution as described above. The 3DM-CV7 can continue to navigate but its position error will increase exponentially, as is the case on the majority of INS systems.

Data

Is the bias removed from Compensated Angular Rate (0x82,0x0E) and Compensated Acceleration (0x82,0x1C) data channels?

Yes, but the Compensated Angular Rate or Compensated Acceleration messages are not anti-aliased at this time. This means, if you choose a sample rate lower than 500 Hz, aliasing can occur, which looks like noise but is actually due to down-sampling. This shortcoming is being addressed in the next firmware release.

Support

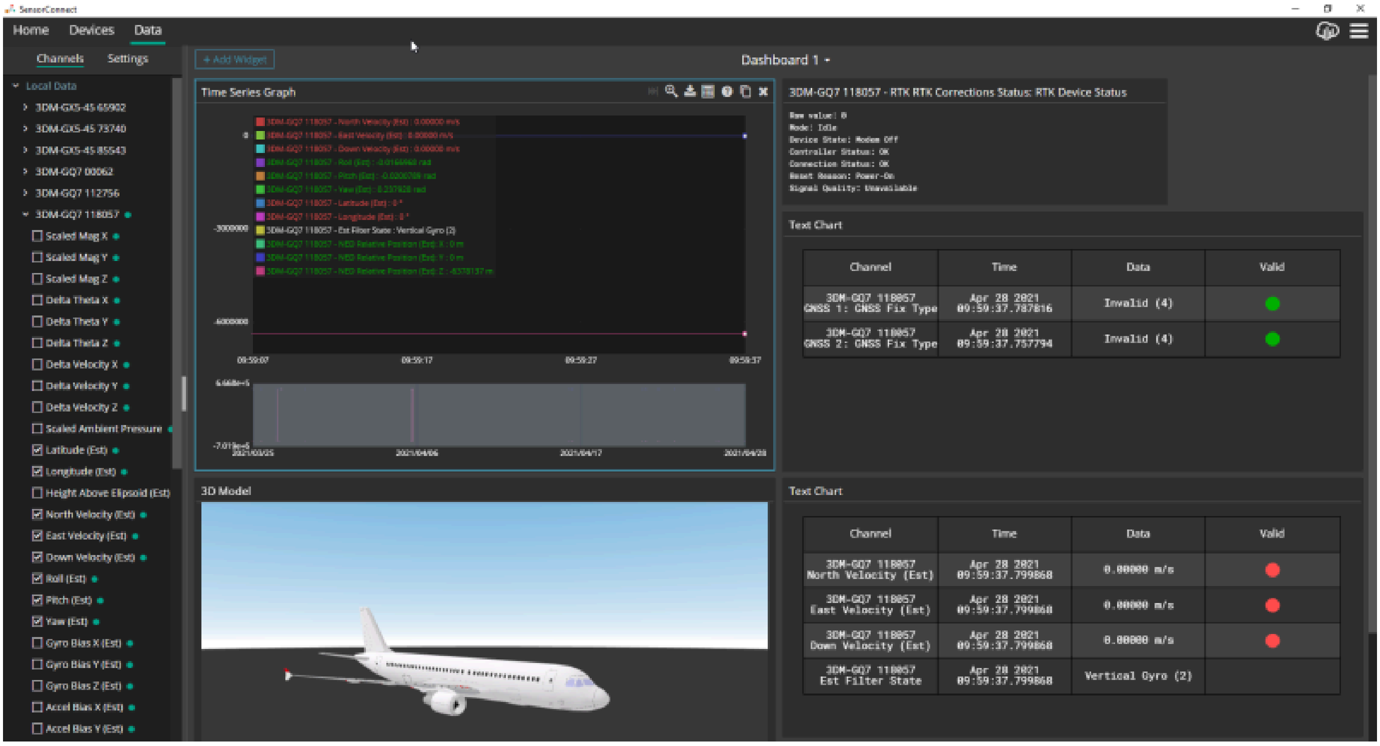
How do I obtain Engineering Support for my device ?

MicroStrain Support Engineers can be contacted through many methods. Please review the **Support** page for more information.

Software

Do you have data visualization tools?

- Yes. You can setup a real-time data visualization dashboard in **SensorConnect** as shown below. You can also export this data as a CSV file for easy data processing, or to a binary file.



Does MicroStrain by HBK have ROS drivers?

Yes, and we have examples, and a dedicated ROS Engineer to support them and support you! See the **Software** page for more information.

Does MicroStrain by HBKhave PX4 drivers?

Yes! Please see the **Software** page for more information.

Does MicroStrain by HBKhave ArduPilot drivers?

Yes! Please see the **Software** page for more information.

Application Notes

Enhancing INS with Radar

MicroStrain's Robotics R&D group has performed extensive testing and found radar to be an effective and robust aiding sensor for ground vehicle applications. An example of this integration and the notable3DM-CV7-GNSS/INS performance improvements during a GNSS outage has been published with example code in this white paper: Enhancing Inertial Navigation System Performance with Radar. This white paper provides example code using MicroStrain Inertial ROS driver, a custom ROS package for radar pre-processing, and configuration of a Ublox F9P GNSS receiver.

You are here: Application Notes > NMEA Input with u-blox F9P

NMEA Input with u-blox F9P

The following steps provide instructions for configuring the u-blox ZED-F9P GNSS receiver (F9P) to output NMEA data to be consumed by the 3DM-CV7-GNSS/INS as described on the **NMEA Input** page. A configuration file has been provided to ensure the receiver outputs the exact messages needed for the 3DM-CV7-GNSS/INS to compute a Full Navigation solution with the least effort.

The Ublox ZED-F9P has been tested extensively for GNSS aiding with the 3DM-CV7-GNSS/INS and provides great performance for low SWAP-C.

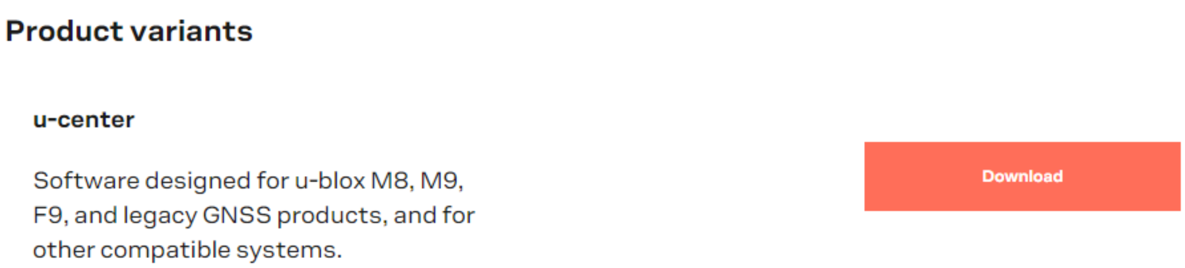
Note: MicroStrain by HBK recommends u-blox's GNSS receiver based on results from field testing and does not benefit from the sale of u-blox products.

Upload the Configuration File to the F9P

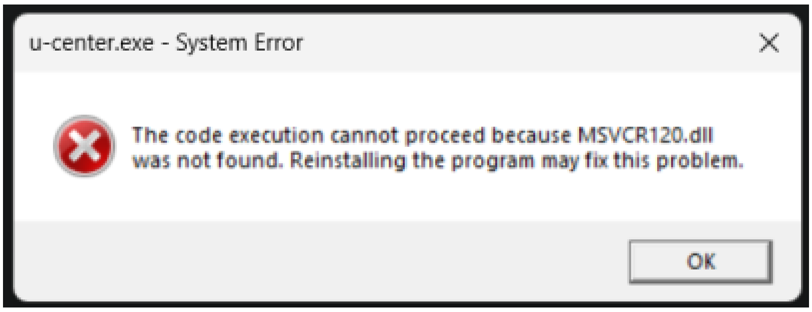
1. Click the attached F9P configuration file, enter ctrl + s to save the file as a .txt file, and store it in an accessible place on your computer. This configuration file will output ZDA, GGA, GSV, and VTG NMEA-0183 messages.

2. If you have not already, install u-blox u-center on your computer.

    1. Ensure to install the version for F9 GNSS products as shown below:



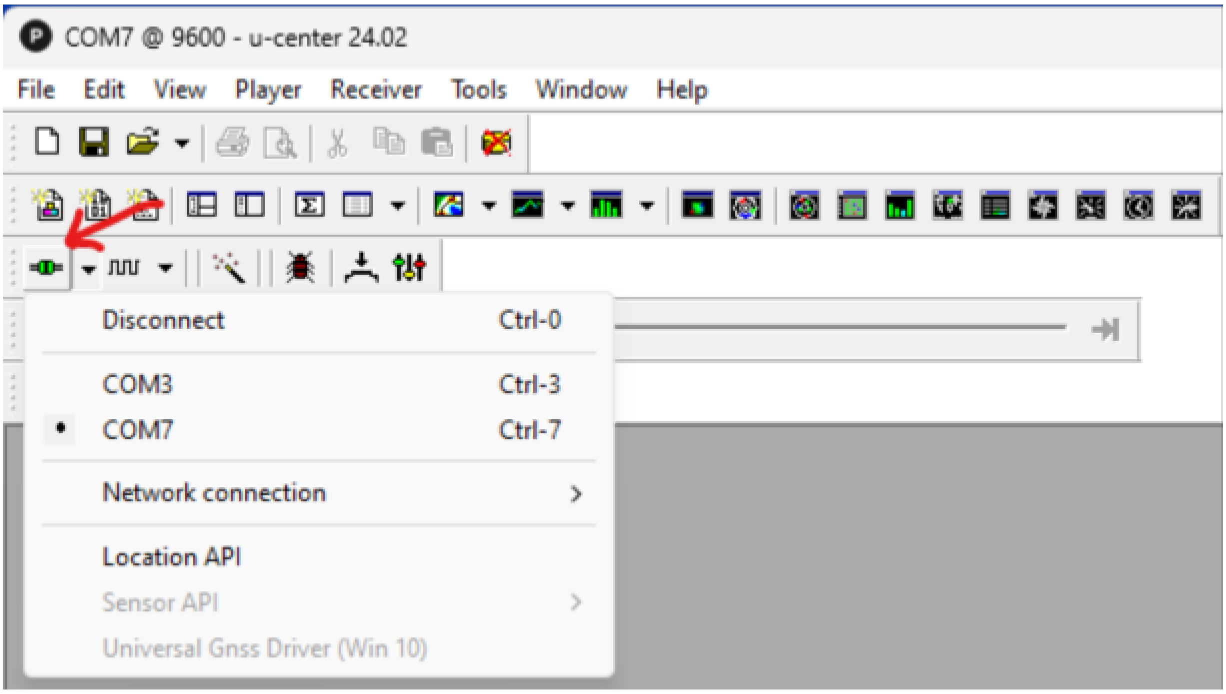
    2. NOTE: If you are installing u-center on a Windows computer and receive the following error message when you launch u-center:



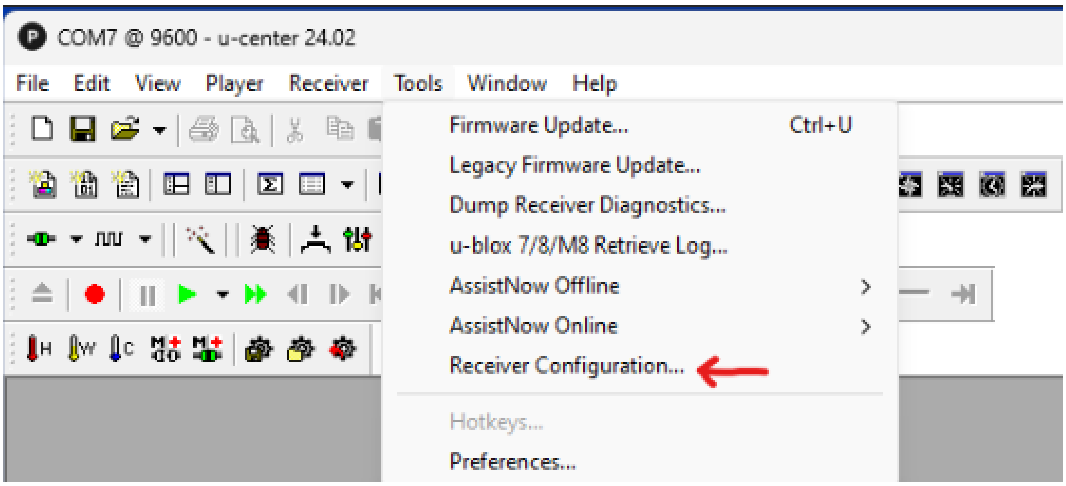
3. Download the x86 2013 Microsoft Visual C++ Redistributable package, which can be found here.

3. Launch u-center, and connect your F9P to your computer via USB.

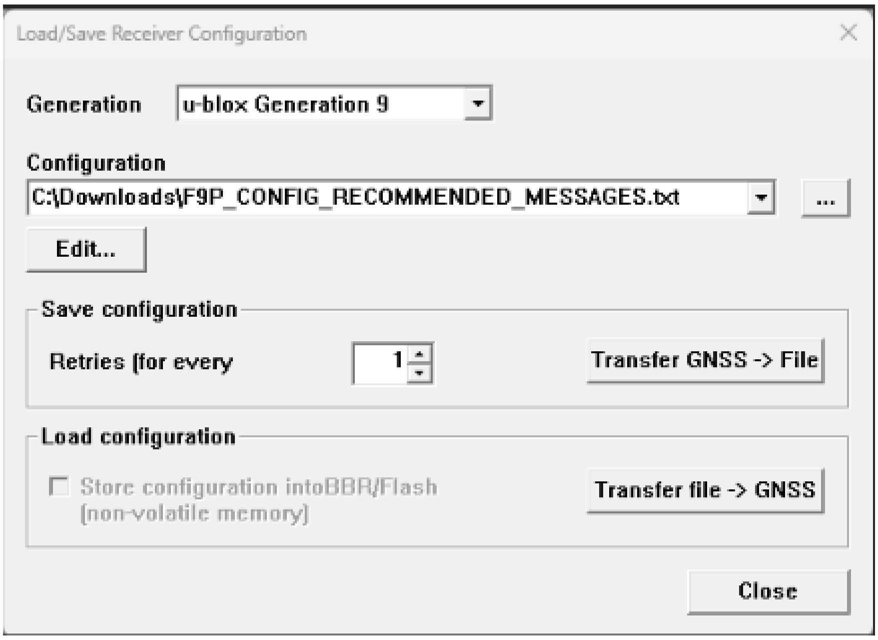
4. Connect the F9P to u-center by selecting the appropriate serial port.



5. Once connected, navigate to the “Tools” drop down menu in the toolbar, and select “Receiver Configuration…”.

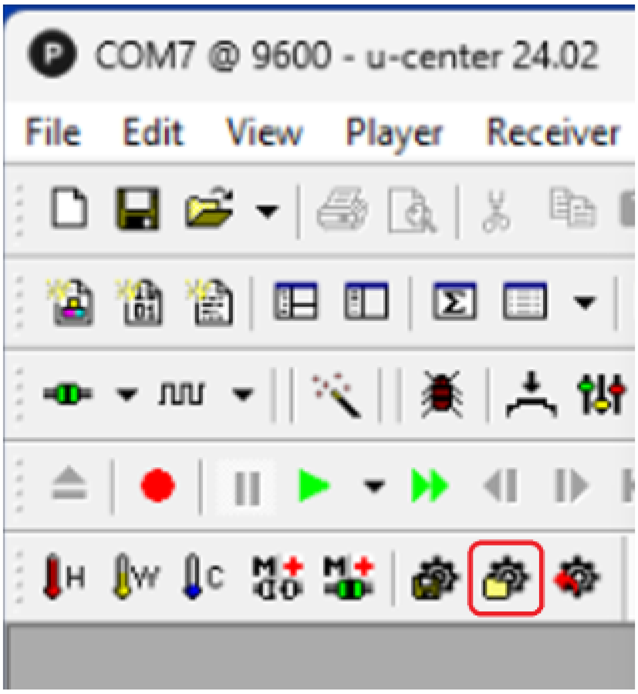


6. Select the button with the three dots to the right of the Configuration drop down box, navigate to where you stored the configuration file, and select it.



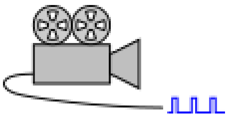
7. In the “Load configuration” box, press the “Transfer file -> GNSS” button.

8. Click the Save as Non Volatile Memory button (circled) in the toolbar. This will ensure configuration remains after power cycling the receiver.



Your F9P is now configured to send MicroStrain by HBK recommended NMEA messages over the TX2 port to enable a Full Navigation solution on the 3DM-CV7-GNSS/INS. Return to the **NMEA Input** page and the "Connect External GNSS Receiver to 3DM-CV7-GNSS/INS" section to continue setup.

Synchronizing Data Output with an External System



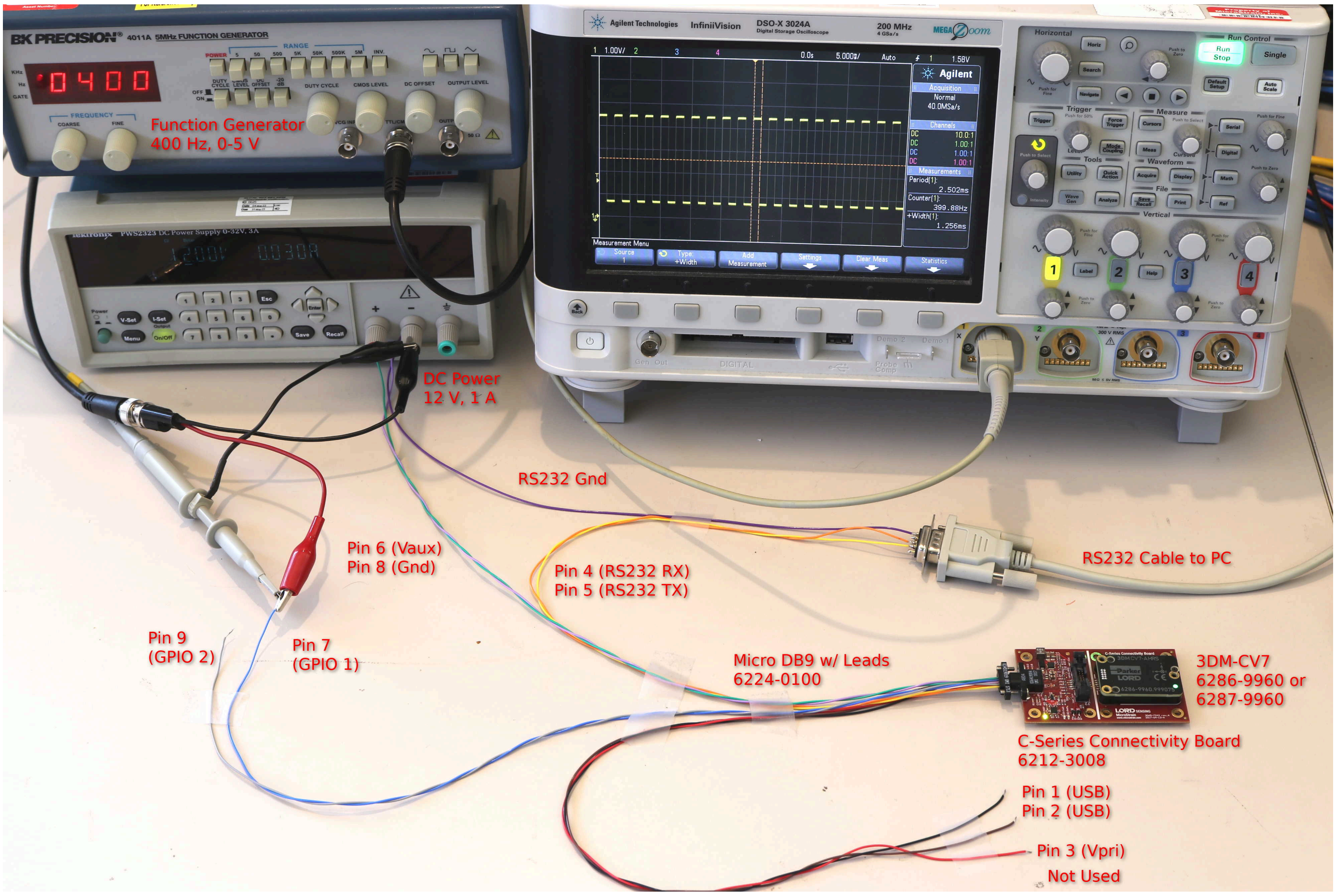
In this example, we'll show how to synchronize data output to an external device via a GPIO input. The output data will interpolated as if it were sampled at the falling edge of the input signal.

We'll stream **Delta Theta (0x80,0x07)** and **Delta Velocity (0x80,0x08)** from the **Sensor Data set** and **Euler Angles (0x82,0x05)** from the **Filter Data set**, as well as the **Event Source (0xFF,0xD0)**, Reference **Timestamp (0xFF,0xD5)**, and **Reference Time Delta (0xFF,0xD6)** for each.

Hardware Setup

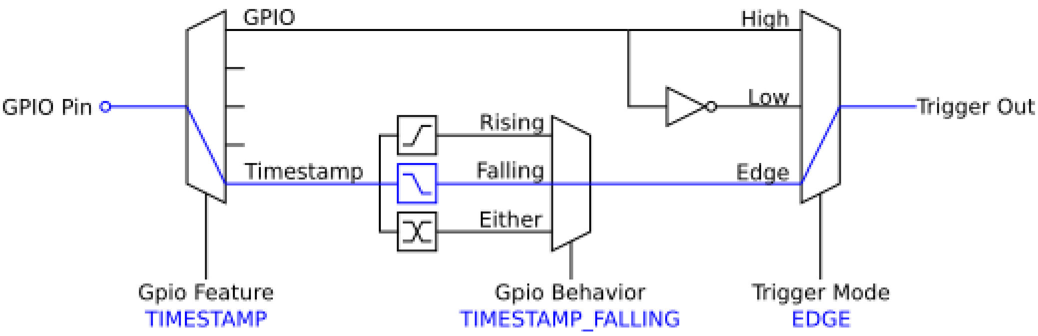
For this example, a function generator will be used to provide a timing signal. Set the output voltage for 0 to 3 volts and the output frequency to 400 Hz. Leave the output disabled for now.

Alternatively, a push button or other input may be used. Connect the button between GPIO 1 and ground, so that it pulls the pin low when pressed (for normally-open buttons).



GPIO Setup

In this example, we'll use GPIO 1 as the sync input. We need to configure the pin for hardware timestamps with the internal pullup enabled. The internal pullup will pull the pin to a default high state when left disconnected. The pullup is not necessary when the input will be actively driven, such as with a function generator. We use it in this example so a push button can be used. Similarly, we'll use the falling edge as that will coincide with normally-open push button presses (rising edges will occur when the button is released).



Command: GPIO Configuration (0x0C,0x41)

- Function: WRITE (0x01)

- Pin: 1
- Feature: TIMESTAMP (0x04)
- Behavior: TIMESTAMP_FALLING (0x02)
- Mode: pullup (0x04)

Command bytes: `0101040204`
MIP Packet: `75650c07 07410101040204 4171`

Inspection with a multimeter or oscilloscope will show that the voltage at GPIO 1 is hovering around 3 volts.

Event Trigger

We'll create a **Trigger: GPIO** set for EDGE mode on GPIO 1.

Command: **Event Trigger Configuration (0x0C,0x2E)**

- Function: WRITE (0x01)
- Instance: 1
- Type: GPIO (0x01)
- Parameters:
  - Pin: 1
  - Mode: EDGE (0x04)

Command bytes: `0101010104`
MIP Packet: `75650c07 072e0101010104 2af4`

This trigger will activate for one event cycle each time GPIO 1 transitions from high to low.

Message Action

To output data, we need one **Action: Message** per descriptor set and data rate. We'll set one up for the sensor data and one for the filter data.

Command: **Event Action Configuration (0x0C,0x2F)**

- Function: WRITE (0x01)
- Instance: 1
- Trigger: 1 (this must match the trigger set up previously)
- Type: MESSAGE (0x02)

- Parameters:
  - Descriptor Set: Sensor Data (0x80)
  - Decimation: 0 (oneshot)
  - Number of Fields: 5
  - Descriptors: [0xD0 (Event Source), 0xD5 (Reference Timestamp), 0xD6 (Delta Ref Time), 0x07 (Delta Theta), 0x08 (Delta Velocity)]

Command bytes: `0101010280000005d0d5d60708`

MIP Packet: `75650c0f 0f2f0101010280000005d0d5d60708 47ca`

Command: **[Event Action Configuration (0x0C,0x2F)]()**

- Function: WRITE (0x01)
- Instance: 2 (this must be different from the first message action)
- Trigger: 1 (this must match the trigger set up previously)
- Type: MESSAGE (0x02)
- Parameters:
  - Descriptor Set: Filter Data (0x82)
  - Decimation: 0 (oneshot)
  - Number of Fields: 4
  - Descriptors: [0xD0 (Event Source), 0xD5 (Reference Timestamp), 0xD6 (Delta Ref Time), 0x05 (Euler Angles)]

Command bytes: `0102010282000004d0d5d605`

MIP Packet: `75650c0e 0e2f0102010282000004d0d5d605 3d7a`

Enable and Test

Now that everything has been configured, we must enable the trigger and input signal.

Command: **[Event Control (0x0C,0x2B)]()**

- Function: WRITE (0x01)
- Instance: 1 (same as the trigger above)
- Mode: ENABLED (0x01)

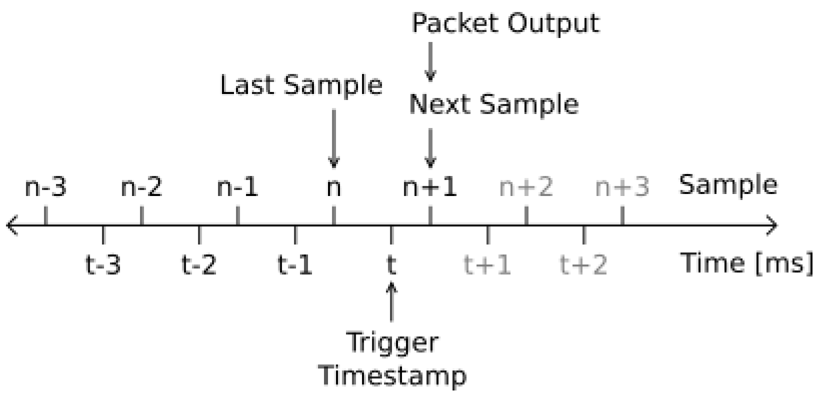
Command bytes: `010101`

MIP Packet: `75650c05 052b010101 1e82`

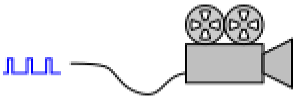
Enable the function generator now. The device will begin streaming two packets, one with the Sensor Data descriptor set (0x80) and one with the Filter Data descriptor set (0x82), at 400 Hz. The timestamps of each packet will reflect the exact time that the input signal changed from high to low. Both packets will have the same timestamp and event source. The delta time field of each will be 2.5 ms, or 1/400th of a second.

If using a push button, one of each type of packet will be emitted per push, assuming the button contacts do not bounce. If the button is pressed about once per second, the delta time fields will reflect this.

Note: Data is timestamped at the GPIO input signal edge and linearly interpolated to capture the sensed motion at that instance. However, the data is sent at intervals determined by the 3DM-CV7's internal clock.



Synchronized Square Wave Output



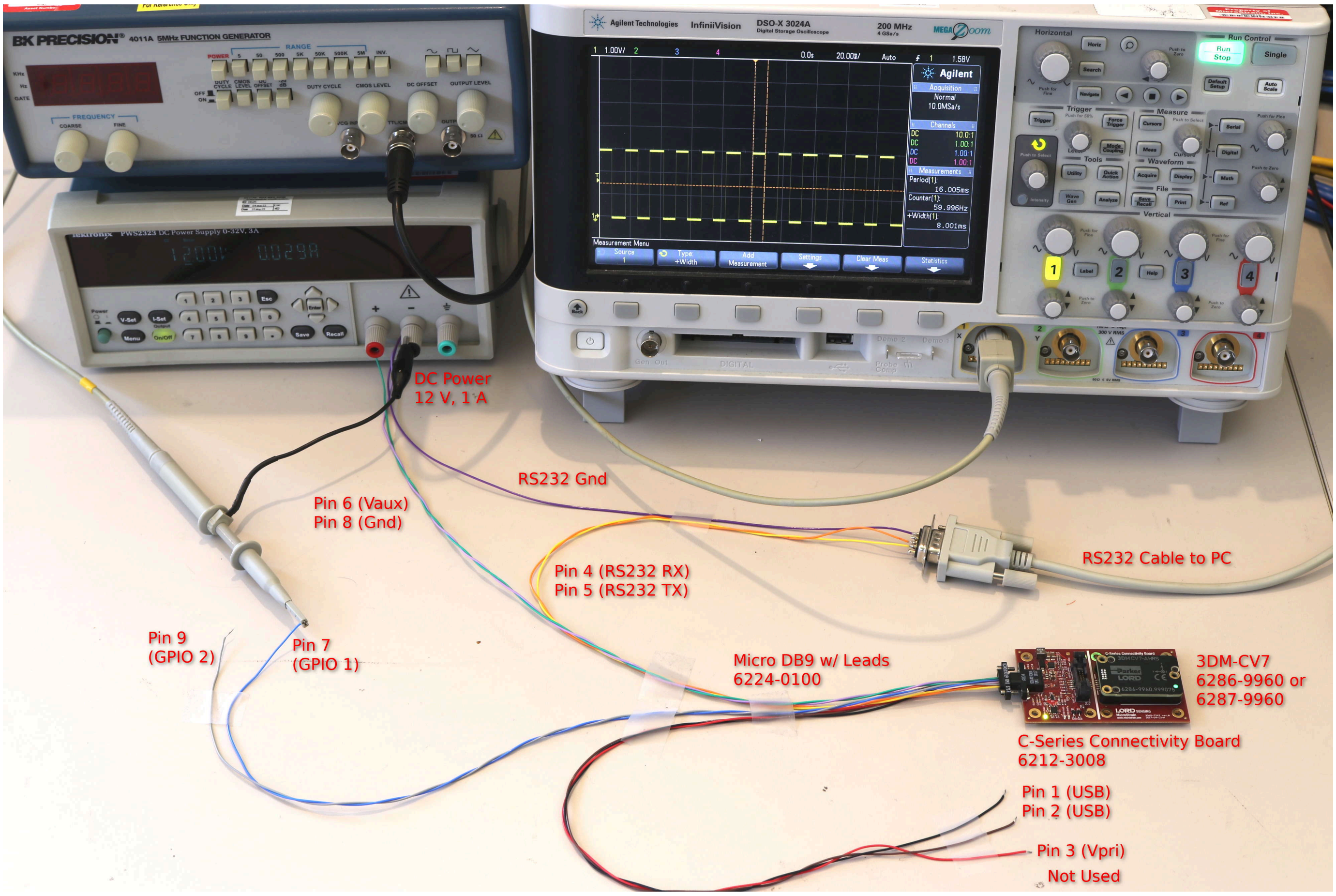
The event system can be used to produce a square wave synchronized to either the 3DM-CV7's internal reference time or an external timestamp. The square wave output can be used to trigger other devices at a consistent system-level time base.

For example, a PPS from a GNSS could be input into the 3DM-CV7 and a 100 Hz square wave could then be generated that is synchronized to the GNSS timing. This output could be used to drive a camera, thereby synchronizing the 3DM-CV7's data and the camera shutter with the GNSS timing signal.

In this example, we'll show you how to configure an interval trigger and GPIO action to produce a 60 Hz square wave synchronized to the 3DM-CV7's internal reference time.

Hardware Setup

Connect an oscilloscope to GPIO 1. Set the time scale to 20 ms per division and the amplitude to 1 V per division with DC coupling. Set the trigger to rising edge, DC coupled, and a threshold of 1.5 V. Alternatively, a frequency counter can be used. Another option is to use an LED, logic probe, or multimeter, but the frequency will have to be reduced to a few Hz in order for it to be visible.



GPIO Setup

GPIO 1 must be configured using the **GPIO Configuration (0x0C,0x41)** command. Since we'll be using it with a GPIO Action, we must select the GPIO feature and one of the output modes (either high or low).

Command: **GPIO Configuration (0x0C,0x41)**

- Function: WRITE (0x01)
- Pin: 1
- Feature: GPIO (0x01)
- Behavior: GPIO_OUTPUT_LOW (0x02) (GPIO_OUTPUT_HIGH could also be used, then the pin will be high by default when the action is not configured yet)
- Mode: None (0x00)

Command bytes: `0101010200`
MIP Packet: `75650c07 07410101010200 3a64`

There should be no noticeable change on the oscilloscope or frequency counter except possibly a reduction in the noise level. The pin will read close to 0 volts because it's now configured to output low. If you instead configure it to OUTPUT_HIGH mode, the pin will read about 3 volts (unless a load such as an LED is connected, then it will read a lower voltage). This is a good way to check if your measurement is working. Which mode is selected does not matter for this example as it will be overridden by the event system.

Event Action Setup

Next, we'll configure an event action to control GPIO 1 using the **Event Action Configuration (0x0C,0x2F)** command. We want it to directly control the state of the pin, so we'll use the ACTIVE_HIGH mode. ACTIVE_LOW could also be used if the opposite polarity is desired.

- Function: WRITE (0x01)
- Instance: 1
- Trigger: 1 (we'll use this as the trigger ID below)
- Type: GPIO (0x01)
- Parameters:
  - Pin: 1
  - Mode: ACTIVE_HIGH (0x01)

Command bytes: `010101010101`
MIP Packet: `75650c08 082f010101010101 2b31`

Event Trigger Setup

The final step is to configure an interval trigger. We'll set the data quantity to the **Reference Timestamp (0xFF,0xD5)** in the Sensor Data set. The interval will be the base rate divided by the frequency, or 1 kHz / 60 = 16.666667. For a 50% duty cycle, the threshold must be half of the interval, or 8.3333333. Note that since the trigger is updated once every event cycle, the state can only be updated with the granularity of one tick. As a result, for intervals that aren't exact integers, there will be some jitter from cycle to cycle. The average frequency however will remain accurate.

Command: **Event Trigger Configuration (0x0C,0x2E)**

- Function: WRITE (0x01)
- Instance: 1 (Must match the trigger ID from the action)
- Type: Threshold (0x02)

- Parameters:
  - Descriptor Set: 0x80 (Sensor Data)
  - Field Descriptor: 0xD5 (Shared Reference Timestamp)
  - Parameter ID: 1 (Timestamp in nanoseconds)
  - Type: INTERVAL (0x02)
  - Threshold: 8333333 (8.333333 ms, since the value is in nanoseconds)
  - Interval: 16666667 (16.666667 ms, since the value is in nanoseconds)

Command bytes: `010101010101`
MIP Packet: `75650c19 192e01010280d50102415fca0540000000416fca0560000000 308a`

No change in the output will be observed at this point as the trigger is still disabled.

Enable the Trigger

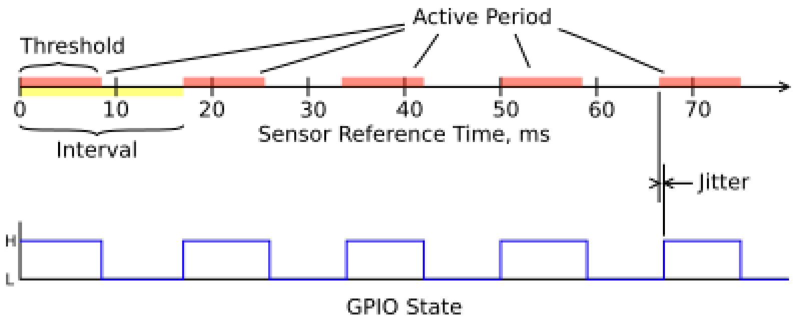
Send the **Event Control (0x0C,0x2B)** command to enable the trigger.

- Function: WRITE (0x01)
- Instance: 1
- Mode: ENABLED (0x01)

Command bytes: `010101`
MIP Packet: `75650c05 052b010101 1e82`

Conclusion

The 60-Hz square wave output should now be visible on the oscilloscope or frequency counter. The frequency may not read exactly 60 Hz but should be quite close. The difference is due to slight differences in the internal oscillators of the 3DM-CV7 and the test equipment. This is why time synchronization is important - no two devices will ever have perfectly aligned and adjusted clocks. The square wave output can be used to synchronize another device to the 3DM-CV7's clock. If the 3DM-CV7 is configured to sync to an external time source using the **PPS** input, the square wave can also be synchronized with the external time source. To do this, simply select the **External Timestamp (0xFF,0xD7)** instead of the reference timestamp in the interval trigger.


Active Period
Threshold
0
10
20
30
40
50
60
70
Sensor Reference Time, ms
Interval
Jitter
H
L
GPIO State

Gated Data Streaming using a GPIO

In this example, we'll show how to configure the 3DM-CV7 to stream scaled accelerometer, scaled gyro, and filtered Euler angles at 100 Hz only when a GPIO pin is pulled low. The data stream will stop when the GPIO is pulled high.

NOTE: This example can be easily adapted to the 3DM-CV7 by substituting GPIO 1 or GPIO 2 for GPIO 3.

Hardware Setup

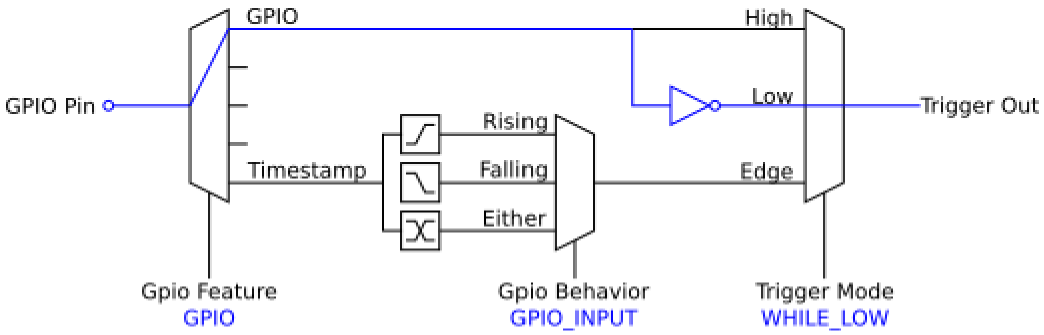
For this example, we'll assume there is a function generator connected to GPIO 3. The generator should be set for a square wave with the high voltage at 3 volts and the low voltage at 0. The frequency should be less than 2 Hz for easy observation. Leave the output disabled for now.

With slight modification, a jumper wire or push button can be used instead. Leave the wire floating on GPIO 3, or connect the button between GPIO 3 and ground.

To accommodate either situation, we'll set it up to enable streaming when the pin reads low.

GPIO Trigger Configuration

In this example, we'll use a **Trigger: GPIO** in "state" mode on GPIO 3 set to activate while the pin is in the low state.



Configure the trigger

Use the [Event Trigger Configuration (0x0C,0x2E)]() command to configure trigger 1 for GPIO.

- Function: WRITE (0x01)
- Instance: 1
- Type: GPIO (0x01)
- Parameters:
  - Pin: 3
  - Mode: WHILE_LOW (0x02)

Command bytes: `0101010302`

MIP Packet: `75650c07 072e0101010302 2af6`

GPIO Setup

Since we'll be using a GPIO pin in this example, it needs to be configured with the [GPIO Configuration (0x0C,0x41)]() command for GPIO Input mode. We'll enable the built-in pull-up resistor to ensure the pin floats high when disconnected. This helps avoid unintentional triggers when using a push button or jumper wire, or any time the pin can only be pulled low by the connected system.

- Function: WRITE (0x01)
- Pin: 3
- Feature: GPIO (0x01)
- Behavior: GPIO_INPUT (0x01)
- Pin Mode: Pullup (0x04)

Command bytes: `0103010104`

MIP Packet: `75650c07 07410103010104 3f6e`

Check the trigger is working

Enable the trigger

Since triggers are disabled by default, it must be enabled. Use the 3DM **Event Control (0x0C,0x02B)** command to enable it.

- Function: WRITE (0x01)
- Instance: 1 (this is the trigger ID used previously)
- Mode: ENABLED (0x01)

Command bytes: `010101`
MIP Packet: `75650c05 052b010101 1e82`

Check the status

The **Get Trigger Status (0x0C,0x2C)** command can report the status of the newly-configured and enabled trigger.

Command:

- Requested Count: 1
- Requested Instances: [1] (this is the trigger ID used previously)

Command Bytes: `0101`
MIP Packet: `75650c04 042c0101 1c5e`

Reply Packet: `75650c09 04f12c00 05b6010102 cf63`
Response data: `010102`

- Count: 1
- Type: 0x01 (GPIO)
- Status: 1 (enabled)

Turn on the function generator and issue the command a few more times until you see status code 3. If you're using a button. hold the button down for this step. If you're using a jumper wire, connect it to ground.

Reply packet: `75650c09 04f12c00 05b6010103 d064`
Response data: `010103`

- Count: 1
- Type: 0x01 (GPIO)

- Status: 3 (Active, Enabled)

Turn the signal generator off, release the button, or disconnect the jumper wire from ground.

Message Action setup

Use the **Event Action Configuration (0x0C,0x2F)** command to configure two message actions linked to the trigger. Two are required because we want to stream data from two MIP descriptor sets, **Sensor Data (0x80)** and **Filter Data (0x82)**.

Action 1

- Function: WRITE (0x01)
- Instance ID: 1 (new action, any value up to the maximum number of actions could be used here)
- Trigger ID: 1 (this must match the trigger ID we created above)
- Type: Message (0x02)
- Parameters:
  - Descriptor set: 0x80 (Sensor Data)
  - Decimation: 10 (This is the base rate of the sensor data, 1 kHz, divided by the desired streaming rate, in this case 100 Hz)
  - Count: 2
  - Descriptors: [0x04 (scaled accel), 0x05 (scaled gyro)]

Command bytes: `0101010280000A020405`
MIP Packet: `75650c0c 0c2f0101010280000a020405 c778`

Action 2

- Function: WRITE (0x01)
- Instance ID: 2 (this must be different from the one above)
- Trigger ID: 1 (this must match the trigger ID we created above)
- Type: Message (0x02)
- Parameters:
  - Descriptor set: 0x82 (Filter Data)
  - Decimation: 10 (This is the base rate of the sensor data, 1 kHz, divided by the desired streaming rate, in this case 100 Hz)

- Count: 1
- Descriptors: [0x05 (Euler angles attitude)]

Command bytes: `0102010282000A0105`

MIP Packet: `75650c0b 0b2f0102010282000a0105 c3ab`

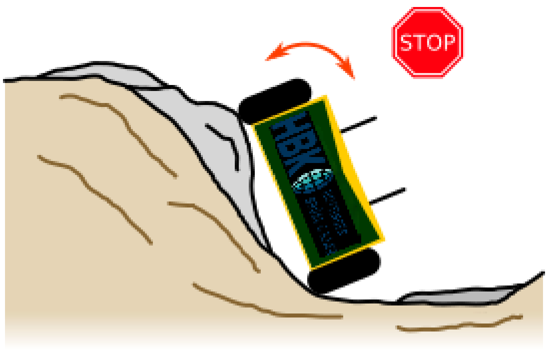
Test it

Verify the signal generator is turned off, or the jumper wire is disconnected. Send the resume command and verify that no data is currently streaming. If you do see data, make sure the scheduled streams are all turned off, that the pin is not pulled low, and that the trigger is configured properly.

Now, turn on the generator (or press the button or connect the jumper to ground) and observe that data beings streaming the configured fields. There should be two packets, one for sensor data with scaled accel and gyro fields, and another for filter data with euler angles.

The device will only stream while the pin is low, so if the function generator is set for 1 Hz and 50% duty cycle, it will stream for a half-second every second. If using a button, it will stream while the button is held down.

Stopping a Robot if It Tips Over



In this example, we'll show how the 3DM-CV7 event system can be used to trigger a robot's emergency stop feature in the event that it rolls over.

Warning: The 3DM-CV7 is not a safety-rated device. Do not use it as a safety device in applications where failure of such device could lead to personal injury, death, or significant property damage. The application in this example is for *additional* protection only, to improve the safety or convenience of an already-safe system.

NOTE: This example can be easily adapted to the 3DM-CV7 by substituting GPIO 1 or GPIO 2 for GPIO 4.

Hardware Setup

Let's assume the robot has a built-in emergency stop system which is triggered by a dedicated digital E-stop control signal. As a demonstration, we'll connect a buzzer to GPIO 4 to let us know if the E-stop has been triggered.

Important: Avoid connecting the 3DM-CV7 to an E-stop control line shared with other safety-critical E-stop signaling devices. Device failure or misconfiguration might prevent proper operation of the E-stop system (e.g. it may lock an shared active-low signal to the high state if configured improperly).

Device Configuration

GPIO

We'll set up GPIO 4 as a push-pull output to serve as our E-stop signal.

Command: **GPIO Configuration (0x0C,0x41)**

- Function: WRITE (0x01)
- Pin: 4
- Feature: GPIO (0x01)
- Behavior: GPIO_OUTPUT_LOW (0x02)
- Mode: None (0x00)

Command bytes: `0104010200`
MIP Packet: `75650c07 07410104010200 3d70`

Verify GPIO (Optional)

To verify the configuration and hardware connection, we can use the GPIO State command to manually set the pin to the high state (buzzer will sound) and the low state (buzzer quiet).

Command: GPIO State (0x0C,0x42)

- Function: WRITE (0x01)
- Pin: 4
- State: high (0x01)

Command bytes: `010401`
MIP Packet: `75650c05 0542010401 38e4`

Command: **GPIO State (0x0C,0x42)**

- Function: WRITE (0x01)
- Pin: 4
- State: low (0x00)

Command bytes: `010400`
MIP Packet: `75650c05 0542010400 37e3`

GPIO Action

Since we want to control a GPIO pin, we'll need a GPIO action. We'll use the oneshot mode so the pin will only be set to the stop state. We'll let a human operator manually reset the E-stop system (more on this to follow).

Command: **Event Action Configuration (0x0C,0x2F)**

- Function: WRITE (0x01)
- Instance: 1
- Trigger ID: 1 (must match trigger config below)
- Type: GPIO (0x01)
- Parameters:
  - Pin: 4
  - Mode: ONESHOT_HIGH (0x05)

Command bytes: `010101010405`
MIP Packet: `75650c08 082f010101010405 323b`

Threshold Trigger

Let's assume the robot is stable until it rolls 45 degrees to the left or right. The **Complementary Filter Euler Angles (0x80,0x0C)** contains the roll angle in radians from the **Complementary Filter**. We'll use this data field and set a **Trigger: Threshold** at +/- PI/4 radians. To make the trigger active when the roll is *outside*the limits, we must reverse the high and low thresholds.

Command: **Event Trigger Configuration (0x0C,0x2E)**

- Function: WRITE (0x01)
- Instance: 1 (must match trigger ID in action)
- Type: Threshold (0x02)
- Parameters:
  - Descriptor Set: 0x80 (Sensor Dataset)
  - Field Descriptor: 0x0C (Euler Angles)
  - Parameter ID: 1 (Roll)
  - Mode: WINDOW (0x01)
  - Low Threshold: +0.7853981 (+45 degrees)
  - High Threshold: -0.7853981 (-45 degrees)

Command bytes: `010102800c01013fe921fb323ae5afbfe921fb323ae5af`
MIP Command: `75650c19 192e010102800c01013fe921fb323ae5afbfe921fb323ae5af e040`

Enable the Trigger

Ensure the 3DM-CV7 is lying face-up (z axis pointing down) on a level surface.

Command: **Event Control (0x0C,0x2B)**

- Function: WRITE (0x01)
- Instance: 1
- Mode: ENABLE (0x01)

Command bytes: `010101`
MIP Command: `75650c05 052b010101 1e82`

Test it

Tilt the 3DM-CV7 along the X axis (i.e. rolling) until the buzzer sounds. Since we used the oneshot mode in the GPIO action, it will not stop even if the device is returned to the upright position. To reset, send the following command which will set the pin low again:

Command: **GPIO State (0x0C,0x42)**

- Function: WRITE (0x01)
- Pin: 4
- State: low (0x00)

Command bytes: `010400`
MIP Packet: `75650c05 0542010400 37e3`

This is a "manual reset" command which could be issued by the robot during startup after an E-stop event. Note that if the pin is reset while the robot is still tipped over (rolled past 45 degrees), it will stay reset until the robot is (righted within 45 degrees) and tips over again.

Additional Features

Hysteresis

It may not be desirable to manually reset the E-stop signal with a MIP command. Instead, perhaps the robot should recover on its own, once righted. Typically the robot would have to return to the upright position within a small tolerance. The following two advanced examples describe ways to implement such hysteresis.

- **Using Two GPIO Actions in Oneshot mode** - Add a second and opposite pair of triggers and actions
- **Using a Latching Combination Trigger for Hysteresis** - Add a second threshold and latching behavior

Adding Pitch Detection

See the advanced example, **Using Two Thresholds with a Combination Trigger**, to learn how to set up multiple thresholds.

Advanced

Using Two Thresholds with a Combination Trigger

This is a continuation from the **Stopping a Robot if It Tips Over** example. In this page, we'll set up an additional threshold trigger and combination trigger which will add pitch detection. The GPIO action will need to be modified to reference a new trigger.

NOTE: This example can be easily adapted to the 3DM-CV7 by substituting GPIO 1 or GPIO 2 for GPIO 4.

Add Another Threshold Trigger

We'll add pitch thresholds just like we did for roll. This time we'll use trigger instance 2 and the pitch parameter.

Command: **Event Trigger Configuration (0x0C,0x2E)**

- Function: WRITE (0x01)
- Instance: **2**
- Type: Threshold (0x02)
- Parameters:
    - Descriptor Set: 0x80 (Sensor Dataset)
    - Field Descriptor: 0x0C (Euler Angles)
    - Parameter ID: **2 (Pitch)**
    - Mode: WINDOW (0x01)
    - Low Threshold: +0.7853981 (+45 degrees)
    - High Threshold: -0.7853981 (-45 degrees)

Command bytes: `010202800c02013fe921fb323ae5afbfe921fb323ae5af`

MIP Command: `75650c19 192e010202800c02013fe921fb323ae5afbfe921fb323ae5af e268`

Enable the new threshold trigger

Command: [Event Control (0x0C,0x2B)]

- Function: WRITE (0x01)
- Instance: **2**
- Mode: ENABLE (0x01)

Command bytes: `010201`
MIP Command: `75650c05 052b010201 1f84`

Combination Trigger

We want to trigger the E-stop if either axis exceeds the threshold, so we'll need a combination trigger set to activate if any of its inputs are active. Referencing the "Common Logic Values" table on the [Trigger: Combination] page, we'll select the value 0xFFFE which means "any input". We'll assign the roll axis to input A and the pitch axis to input B. If you had additional trigger sources, they could be assigned slots C or D, but for this example they are set to 0.

Command: [Event Trigger Configuration (0x0C,0x2E)]

- Function: WRITE (0x01)
- Instance: 3
- Type: Combination (0x03)
- Parameters:
  - Logic Table: 0xFFFE
  - Inputs: [1 (roll), 2 (pitch), 0, 0]

Command bytes: `010303fffe01020000`
MIP Packet: `75650c0b 0b2e010303fffe01020000 31f6`

Enable the Combo Trigger

Command: [Event Control (0x0C,0x2B)]

- Function: WRITE (0x01)
- Instance: 3
- Mode: ENABLE (0x01)

Command bytes: `010301`
MIP Command: `75650c05 052b010301 2086`

Modify the GPIO Action

The GPIO action needs to reference the new combination trigger.

Command: [Event Action Configuration (0x0C,0x2F)]

- Function: WRITE (0x01)
- Instance: 1
- Trigger ID: **3 (must match the combo trigger above)**
- Type: GPIO (0x01)
- Parameters:
  - Pin: 4
  - Mode: ONESHOT_HIGH

Command bytes: `010103010401`
MIP Packet: `75650c08 082f010103010401 303f`

Test it

Tipping the 3DM-CV7 past 45 degrees in any direction will set the GPIO output high. As in the original example, it must be reset with the **GPIO State (0x0C,0x42)** command.

Using a Latching Combination Trigger for Hysteresis

This is a continuation from the **Stopping a Robot if It Tips Over** example. In this page, we'll set up an additional threshold trigger and combination trigger which will perform the E-stop "reset" function. The GPIO action will need to be modified to no longer use the oneshot mode.

NOTE: This example can be easily adapted to the 3DM-CV7 by substituting GPIO 1 or GPIO 2 for GPIO 4.

GPIO Action Modification

For this method to work, the combination trigger must directly control the GPIO state. To do so, the GPIO action must use one of the ACTIVE_ modes, in this case ACTIVE_HIGH. Additionally, it will use a new trigger. The configuration command from the original example is changed to the following.

Command: **Event Action Configuration (0x0C,0x2F)**

- Function: WRITE (0x01)
- Instance: 1
- Trigger ID: **3 (must match the combo trigger below)**
- Type: GPIO (0x01)
- Parameters:
  - Pin: 4
  - Mode: **ACTIVE_HIGH (0x01)**

Command bytes: `010103010401`
MIP Packet: `75650c08 082f010103010401 303f`

Additional Threshold Trigger

Just like the other hysteresis example, **Using Two GPIO Actions in Oneshot mode**, we'll set up another threshold trigger.

We'll set up another threshold trigger just like the first but with a narrower window and the opposite logic. This will trigger when the robot is returned to the upright position, plus or minus 10 degrees. The trigger will be similar to the original one, but have a different instance ID, different thresholds, and reversed logic. This is identical to the one in the **Using Two GPIO Actions in Oneshot mode** example.

Command:

- Function: WRITE (0x01)
- Instance: **2** (must be different from the original)
- Type: Threshold (0x02)
- Parameters:
  - Descriptor Set: 0x80 (Sensor Dataset)
  - Field Descriptor: 0x0C (Euler Angles)
  - Parameter ID: 1 (Roll)
  - Mode: WINDOW (0x01)
  - Low Threshold: **-0.1745329 (-10 degrees)**
  - High Threshold: **+0.1745329 (+10 degrees)**

Command bytes: `010202800c0101bfc6571814c9bba03fc6571814c9bba0`
MIP Packet: `75650c19 192e010202800c0101bfc6571814c9bba03fc6571814c9bba0 b1fa`

Enable the new threshold trigger

Command: **Event Control (0x0C,0x2B)**

- Function: WRITE (0x01)
- Instance: **2**
- Mode: ENABLE (0x01)

Command bytes: `010201`
MIP Command: `75650c05 052b010201 1f84`

Combination Trigger

Now comes the final part which makes it all work. We need a latching combination trigger which will control the E-stop signal. Two inputs are needed, the original "stop" trigger (instance #1), and the new "reset" trigger (instance #2). The logic should follow these rules:

- If the "stop" trigger is active, the state will be active.
- If the "reset" trigger is active, the state will be inactive.
- If neither are active, the state doesn't change. It will be the same as the last event cycle.
- If both are active, the state will be active. This can't happen if the other triggers are set up properly because the thresholds do not overlap. However, it's good practice to specify all possible conditions.

An extra input is needed to obtain the state from the last event cycle to implement the third bullet point. We'll assign this "feedback" input to slot A. The "stop" and "reset" triggers will be assigned to slots B and C respectively. Input D is not needed, so it will be set to 0.

According to the rules above, this produces the following logic table:

C	B	A	Output	Comment
0	0	0	0	Neither "stop" nor "reset" active - no change in state so the output must match input A.
0	0	1	1	
0	1	0	1	"Stop" signal active - state should be active regardless of the other two
0	1	1	1	
1	0	0	0	"Reset" signal active - state should be inactive regardless of the previous state.
1	0	1	0	
1	1	0	1	Both signals active (impossible) - default to active
1	1	1	1	

The logic value for this table is 11001110 binary, or CE hexadecimal. To build the full u16 value, we'll duplicate the logic for the case where input D would theoretically be active. This results in 0xCECE. Now we have everything needed to build the command.

Command: **Event Trigger Configuration (0x0C,0x2E)**

- Function: WRITE (0x01)
- Instance: 3
- Type: Combination (0x03)
- Parameters:
  - Logic Table: 0xCECE
  - Inputs: [3 (this trigger), 1 (reset), 2 (stop), 0 (unused)]

Command bytes: `010303cece03010200`

MIP Packet: `75650c0b 0b2e010303cece03010200 d3e9`

Enable the combo trigger

Command: **Event Control (0x0C,0x2B)**

- Function: WRITE (0x01)
- Instance: 3
- Mode: ENABLE (0x01)

Command bytes: `010301`
MIP Command: `75650c05 052b010301 2086`

Test it

Now tilt the 3DM-CV7 as before. The buzzer should sound but once the device is returned to level it should stop. The result will be identical to the **Using Two GPIO Actions in Oneshot mode** example except if the pin is manually changed with the **GPIO State (0x0C,0x42)** command. In this case, the pin state may glitch briefly but will quickly return to the state determined by the event system.

Using Two GPIO Actions in Oneshot mode

This is a continuation from the **Stopping a Robot if It Tips Over** example. In this page, we'll set up an additional threshold trigger and GPIO action which will perform the E-stop "reset" function.

NOTE: This example can be easily adapted to the 3DM-CV7 by substituting GPIO 1 or GPIO 2 for GPIO 4.

GPIO Action #2

We'll set up a second GPIO action which will restore the pin to the inactive, low state. This action must have a different instance ID, use the opposite polarity oneshot mode, and reference a new trigger ID.

Command: **Event Action Configuration (0x0C,0x2F)**

- Function: WRITE (0x01)
- Instance: **2** (must be different from the original)
- Trigger ID: **2** (must match trigger config below and should not be the first trigger)
- Type: GPIO (0x01)
- Parameters:
  - Pin: 4 (the same pin)
  - Mode: ONESHOT_**LOW** (**0x06**)

Command bytes: `010202010406`
MIP Packet: `75650c08 082f010202010406 3545`

Threshold Trigger #2

We'll set up another threshold trigger just like the first but with a narrower window and the opposite logic. This will trigger when the robot is returned to the upright position, plus or minus 10 degrees. The trigger will be similar to the original one, but have a different instance ID, different thresholds, and reversed logic.

Command: [Event Trigger Configuration (0x0C,0x2E)]

- Function: WRITE (0x01)
- Instance: **2** (must be different from the original)
- Type: Threshold (0x02)
- Parameters:
  - Descriptor Set: 0x80 (Sensor Dataset)
  - Field Descriptor: 0x0C (Euler Angles)
  - Parameter ID: 1 (Roll)
  - Mode: WINDOW (0x01)
  - Low Threshold: **-0.1745329 (-10 degrees)**
  - High Threshold: **+0.1745329 (+10 degrees)**

Command bytes: `010202800c0101bfc6571814c9bba03fc6571814c9bba0`

MIP Packet: `75650c19 192e010202800c0101bfc6571814c9bba03fc6571814c9bba0 b1fa`

Enable the trigger

Command: [Event Control (0x0C,0x2B)]

- Function: WRITE (0x01)
- Instance: **2**
- Mode: ENABLE (0x01)

Command bytes: `010201`

MIP Command: `75650c05 052b010201 1f84`

Test it

Now tilt the 3DM-CV7 as before. The buzzer should sound but once the device is returned to level it should stop.

MIP Protocol

This section gives an overview of the MicroStrain by HBK "MIP" communications protocol.

MIP API documents the commands and data quantities available from the 3DM-CV7.

MIP Packet Overview provides a general overview of the MIP packet structure and implementation.

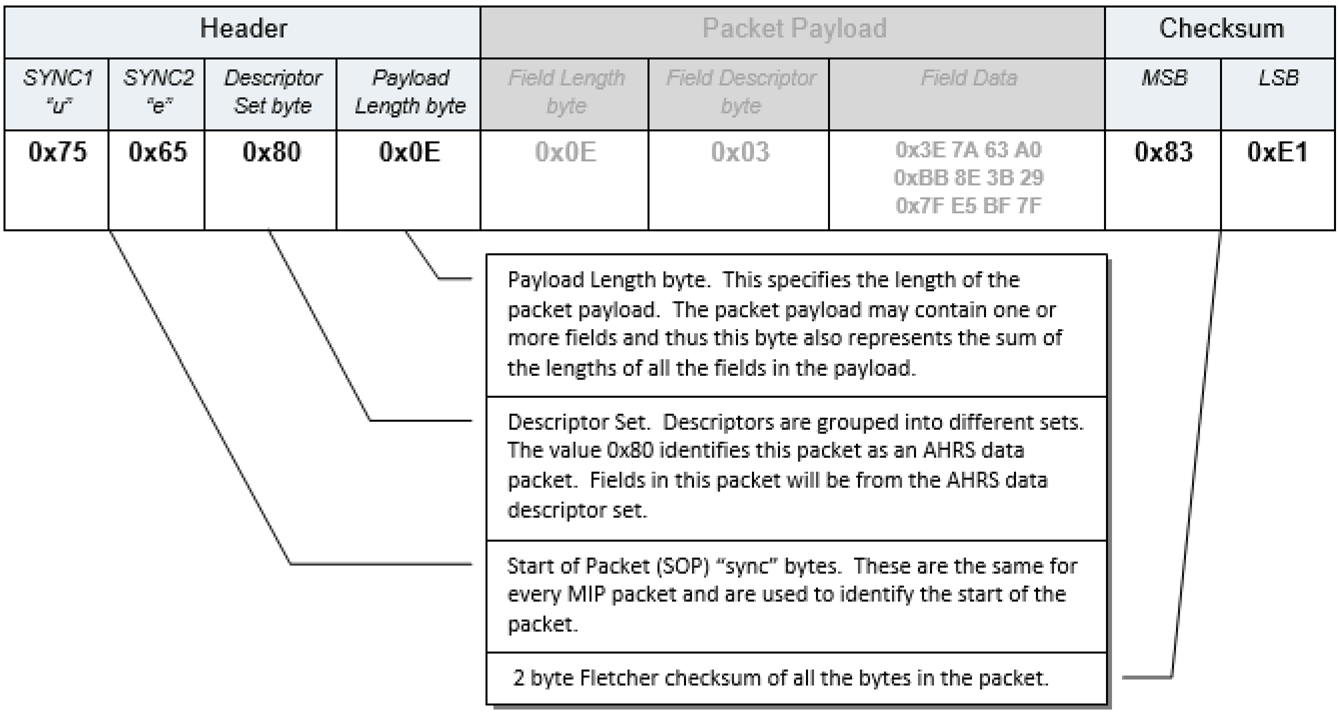
Command Overview and **Data Overview** provide a detailed description of the MIP command and data packet structures.

Parsing Incoming Packets, **Multiple Rate Data**, and **Communications Bandwidth Management** all give helpful tips on MIP data stream configuration, packet parsing, and more.

MIP Packet Overview

This is an overview of the 3DM-CV7 packet structure. The packet structure used is the MicroStrain by HBK “MIP” packet. An overview of the packet is presented here.

The MIP packet “wrapper” consists of a four byte header and two byte **Fletcher Checksum** footer:



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

Command Overview

MIP fields which have a descriptor set in the range 0x01 to 0x7F are commands. A command is issued by sending a MIP packet containing the intended command field to the device. Like all MIP fields, command fields contain the standard length and descriptor values, plus a payload. The payload is composed of zero or more parameters to the command, each of which has its own data type. Multiple commands may be issued in one packet, so long as the descriptor sets are all the same.

The device responds by sending a reply packet. The reply contains at minimum a standard ACK/NACK field for every command in the originating packet. This provides feedback as to whether the command was successfully executed, or why it failed. The ack/nack reply uses descriptor 0xF1, which is reserved in all command descriptor sets. Some commands can return additional data, for example to query the current setting. This response data comes as an additional field immediately after the corresponding ack field and will never be separated or split into a separate packet. The response field has a field descriptor in the range 0x81 to 0xEF. Many responses have a descriptor which matches the command descriptor + 0x80, but this is not always the case so make sure to check the DCP. Response data will not be sent for commands which are nacked (i.e., the reply field has a nonzero ack code).

Data types

Most commands take one or more parameters. These parameters may be represented by different data types. For example, the **Get Data Base Rate (0x0C,0x0E)** command has 1 parameter, a u8.

Some commands take enumerations or bitfields. These quantities are still represented by an underlying integral type (as listed in the standard types table linked above), but have special meaning assigned to particular values or bits, respectively. These meanings are described in the DCP by a table for the relevant enumeration or bitfield type.

Function Selectors

Most settings commands take a special first parameter called a *function selector*. Function selectors are a one-byte enum type and allow the setting to be set, read back, saved to non-volatile storage, reloaded from storage, or reset to the factory default. Many commands support all 5 options, while others support a subset such as just reading and writing. In read mode, a response field will be sent back along with the ack/nack as described above. No other modes send response data.

Settings commands often have several parameters which are only used when changing the setting (i.e., write mode). The length of the command field can be shorter in these cases, and many times the length is reduced to just include the function selector itself. Each command parameter documents the function modes for which it is required. Regardless, it is always permitted to send the full-length command with dummy values (typically zeros) as placeholders. For example, the **PPS Source (0x0C,0x28)** command accepts a function selector and one additional parameter called "source". Source must be specified for write mode, but may be omitted for all other modes. The field length would be 4 in write mode but could be 3 for read, save, load, or reset functions.

Reserved Values

Some fields and values are marked "reserved". Reserved fields and values are required to maintain compatibility and allow future expansion of the MIP interface and for factory programming. Unless otherwise specified, **the behavior of any undocumented or reserved command, field, enumeration value, or bitfield bit** is undefined and **may change between device models or in future firmware versions.** Do not use such values unless explicitly instructed to do so, even if the device behaves as desired. MicroStrain by HBK is not responsible for any consequences (e.g. required maintenance or equipment downtime) resulting from use of reserved or undocumented api elements or behavior.

DCP Format

Commands listed in the DCP follow a standardized table format which is shown here.

Description	**This is a brief description of what the command does.**	
Notes	More information may be provided here.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*The field length describes the entire length of this MIP field, including the field length, field descriptor, and all parameters.*
Descriptor	*u8*	*This is the field descriptor which identifies the command. These are unique within the same descriptor set.*
Function Selector	u8	For commands which include one, the function selector determines what action will be taken. This area will indicate what functions are available for this command. Example: `Write Read Save Load Default [WRSLD]`
Param_A `[WRSLD]`	u8	Each parameter will have a type and description entry in the table. The type of parameter can be a standard type or an enum or bitfield. For enums, the description will include a list of supported values. For bitfields, the description will list the behaviors of each bit. If a function selector is present, the parameter name will include one or more of the letters W, R, S, L, or D. These indicate if the parameter is required when the function selector is Write, Read, Save, Load, or Default, respectively. Parameters which are not required for a given function may be omitted from the command field, reducing its total length.
Mode `[W]`	u8 Enum	An example enum parameter. This parameter is only required when the function selector is set to Write. <table><tr><td>Name</td><td>Value</td><td>Description</td></tr><tr><td>MODE_1</td><td>1</td><td>Uses mode 1 for this feature.</td></tr></table>

		MODE_2 | 2 | Uses mode 2 for this feature. Only allowed if foobar is set to 0.
Ack/Nack Reply	[See standard MIP ack/nack reply format.]()	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*The field length describes the entire length of this MIP field, including the field length, field descriptor, and all response parameters.*
Response Descriptor	*u8*	*This is the response descriptor. For many commands, this matches the command field descriptor + 0x80.*
Param_A	u8	For most commands, the response parameters match the command parameters less the function selector.
Mode	u8 Enum	If this section is blank, refer to the command parameters above for description.

Example Command - Ping

Below is an example of a **Ping (0x01,0x01)** command from the Base Command set. A ping command has no parameters. 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: "75650102 0201 E0C6"								

The packet header has the "ue" starting sync bytes characteristic of all **MIP Packet Overview**. The descriptor set byte identifies the payload as being from the Base Command Set (0x01). The length of the packet payload consists of one field and is 2 bytes. The field starts with the length of the field (2) which is followed by the descriptor byte (0x01). The field descriptor value is the command identifier. 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 (see the **Fletcher Checksum** for instructions on how to compute a Fletcher two byte checksum).

Example Ping Reply Packet

All commands will generate a reply packet from the device containing an ACK/NACK field. The ACK/NACK field contains the command field descriptor plus an error code. An error code of 0x00 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 Packet Overview**. 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 (4) 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 descriptor (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 Unknown Command (0x01), Invalid Parameter (0x03) and Command Failed (0x04). The checksum is a two byte **Fletcher Checksum**

Example Command with Response Data

The **Get Data Base Rate (0x0C,0x0E)** command reports the base sample rate for a given data descriptor set. It takes 1 parameter, "descriptor set", and returns 2 parameters, "descriptor set" and "base rate". For this example, we'll choose the **Sensor Data (0x80)** descriptor set, 0x80. The command field would be `030E80`, corresponding to a length of 0x03, command descriptor 0x0E, and a "descriptor set" parameter of 0x80. Encoding this in a MIP packet with the 3DM command descriptor set (0x0C), we get `75650C03 030E80 7A7E`.

Upon receipt of that command, the device will reply with an ack/nack and some response data: `75650C09 04F10E00 058E8003E8 F058`. The reply packet is also from the 3DM descriptor set. The first field, `04F10E00`, is the reply field. Per the DCP, the first byte is the length (0x04), the second byte is the ack/nack descriptor (0xF1), the third byte is the corresponding command descriptor (0x0E), and the fourth and final byte is the ack code (0x00). An ack of 0x00 indicates success, so the response data is present and can be decoded.

The next field immediately following the reply is the response data, `058E8003E8`. The first value is again the length (0x05), followed by the field descriptor (0x8E). The requested descriptor set is echoed from the command (this is a common feature which allows the disambiguation of replies from multiple commands of the same type). Following that is the parameter which we're trying to obtain by this command: a u16 containing the base rate in Hz. In this case, the value is 0x03E8, which represents an AHRS base rate of 1000 Hz.

Example Command with a Function Selector

The **PPS Source (0x0C,0x28)** command controls how the device receives its pulse-per-second input. There are options: Disabled, User GPIO Input, and Internally Generated. In this example, it will be set to Generated, saved to non-volatile memory, then set to Disabled, reloaded from memory, and finally restored to the default setting.

The command takes a function selector and 1 parameter called "source". To start with, we'll set the source to Generated. To do this, we'll build the command field using the write function selector, 0x01. Source is a u8 (i.e. one byte) enum, with the value 0x04 corresponding to Internally Generated. The field bytes are `04280104`, and the resulting MIP packet is `75650c04 04280104 1b55`. The device will reply with an ack: `75650C04 04F12800 07FA`. Since the write function was used, no response data is included. If you issue this command to a 3DM-CV7, you should see the LED flash white once per second indicating that a PPS is being received.

Now that the PPS is configured how we'd like, we'll save it so that this setting is remembered after a reset or power cycle. The save function selector (0x03) is used for this purpose. This time, the source parameter is not required, so the field length can be shortened to 3 bytes: `032803`. (If desired for consistency, the source parameter could still be sent, in which case it would simply be ignored). The resulting packet is `75650C03 032803 1735`, and the device acknowledges with `75650C04 04F12800 07FA`. At this point, the device were power cycled or reset, the setting would persist and the LED would still flash once per second.

Now let's change the source to Disabled. Using the write mode again, this time with source 0x00, the command field is created: `04280100`. Sending the resulting packet `75650C04 04280100 1751` results in the same ack message again. The LED will stop flashing in this mode.

Disabling the PPS is not particularly useful, so let's load back the previous setting. Using the load function selector, 0x04, the resulting MIP packet is `75650c04 04f12804 0bfe`, and the device will ack the command. The LED will begin flashing once more.

Finally, if you're following along issuing commands to a real device, you'll want to reset the setting to the factory default to avoid later confusion. We'll load the factory default setting and then save it. While we're at it, we'll read back the default setting to see its value. We'll also take this opportunity to show how multiple commands can be sent in one packet.

For this task, three instances of the command are required, one each with function selectors default, save, and read. None of these are the write selector, so they can all have a field length of 3 by omitting the source parameter: `032805`, `032803`, `032802`. Notice only the function selector (the last byte of each) is different. Combining these into a single packet results in `75650C09 032805 032803 032802 7A4A`. The device will process each field in sequence and add the ack/nacks and any response data into a single reply packet. (In the event that the reply packet would exceed the maximum length of a MIP packet, the fields will be split into multiple reply packets as necessary). The packet bytes are `75650C0F 04F12800 04F12800 04F12800 03A801 F8D9`. Notice there are *four* fields and not three like the command packet. The first three are acks and the last is response data for the third command, which used the read function selector. Decoding the response data indicates that the default PPS source is 0x01, Internal Receiver 1, for the device used in this example.

Other Commands

See **Commands** for a list of supported commands.

Data Overview

This section gives an overview of the MIP data packets generated by the 3DM-CV7. When the device is powered up, it may be configured to immediately stream data immediately on startup 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 generated by the device will have the following format.

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. Next, descriptor set byte identifies the payload field as being from the **Sensor Data (0x80)** descriptor set. The final byte of the header is the length of the packet payload (15 bytes, 0x03). The payload portion of the packet comes next after the header and starts with the length of the field. The field descriptor byte identifies the field data as the quantity. 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**.

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.

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 and a continuous stream of data packets.

The data packets from each descriptor set can configured 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. See **Multiple Rate Data**.

If a large number of descriptors are selected for streaming or polling, the amount of data may exceed the size of a single MIP packet. In this case, additional packets, called **MIP Continuation Packets**, are emitted by the device.

MIP Continuation Packets

MIP packets are limited to 255 payload bytes. If enough fields are selected in the message format, this limit can be exceeded. When this happens, the payload is truncated at the end of the last whole field so that there are no fragmented fields. The remaining fields are added to a new packet of the same descriptor set, called a *continuation* packet. This process repeats until all fields have been transmitted. Here is an example using the descriptor set (to be concise, the 0x hexadecimal prefix has been omitted from the field descriptors):

Descriptors in Message Format: 11, 10, 01, 02, 04, 05, 06, 07, 08, 09, 40, 41, 36, 37, 42 (all at the same rate)
Required payload length: 290 bytes (requires 2 mip packets)
Packet #1 Field Descriptors: 11, 10, 01, 02, 04, 05, 06, 07, 08, 09, 40, 41, 36
Packet #2 Field Descriptors: 37, 42 *Continuation packet*
Packet #3 Field Descriptors: 11, 10, 01, 02, 04, 05, 06, 07, 08, 09, 40, 41, 36
Packet #4 Field Descriptors: 37, 42 *Continuation packet*
Packet #5 Field Descriptors: 11, 10, 01, 02, 04, 05, 06, 07, 08, 09, 40, 41, 36
Packet #6 Field Descriptors: 37, 42 *Continuation packet*
. . .

Determining which packets belong to the same group is important. In the example above, fields 0x37 and 0x42 are associated with the GPS timestamp (field descriptor 0x11) in the preceding packet. If the parser groups them the other way (e.g. packets 2 and 3, 4 and 5, etc.) then the timestamp for those fields will be effectively skewed. This can result in poor performance in real-time applications such as control loops.

To address this concern, typically the parser would be set up to look for the first and last fields in the message format. The data buffer would be reset upon receipt of the first field (0x11 from the example), and after the last field (0x42 from the example) is received the data would be sent on to the application for processing. This works well in embedded applications where multiple-rate data is not required and the message format and parsing code are part of the same system.

This doesn't work well in other situations, however. For example, parsing binary log files can be difficult when the message format is unknown. **Multiple Rate Data** also complicates the process because the MIP packets can contain different fields at different times even though the message format is fixed. While there are ways to deal with those situations, a simple solution may be to use one or more of the **Shared Data Descriptors**.

When placed at the beginning of the message format, fields belonging to the shared descriptor group will be copied to continuation packets. For example, the timestamp could be sent as the first field of every packet. All packets belonging to the same sample group would share the same timestamp, allowing the parser to group the data together. Here's the same example from before, modified to use the shared instead of the filter timestamp:

Descriptors in Message Format: D3, 10, 01, 02, 04, 05, 06, 07, 08, 09, 40, 41, 36, 37, 42 (all at the same rate)
Required payload length: 290 bytes (requires 2 mip packets)
Packet #1 Field Descriptors: D3, 10, 01, 02, 04, 05, 06, 07, 08, 09, 40, 41, 36
Packet #2 Field Descriptors: D3, 37, 42 *Continuation packet*
Packet #3 Field Descriptors: D3, 10, 01, 02, 04, 05, 06, 07, 08, 09, 40, 41, 36
Packet #4 Field Descriptors: D3, 37, 42 *Continuation packet*
Packet #5 Field Descriptors: D3, 10, 01, 02, 04, 05, 06, 07, 08, 09, 40, 41, 36
Packet #6 Field Descriptors: D3, 37, 42 *Continuation packet*
. . .

This time, the timestamp appears in every packet. The parser doesn't need to know the message format, and can simply group all of the data with the same timestamp. Notice that the additional field increases the total amount of data being transmitted. Most of the time this is acceptable, but it's important to factor this in when determining the available bandwidth.

Caution: The shared descriptors must be added to the beginning of the message format or poll request. If a message format or poll request contains standard descriptors preceding shared descriptors, the format of continuation packets is unspecified and subject to change. Therefore, place all shared descriptors at the front of the format or ensure the payload fits in one packet.

Shared Data Descriptors

In addition to the standard **data descriptor sets**, the MIP data protocol also has a set of **Shared Data (0xFF)** descriptors that are common across all **data descriptor sets**. The shared data descriptors start at 0xD0 and increment from there. These data quantities can be streamed or polled with any **data descriptor set**.

The shared descriptor set of 0xFF is not valid when configuring data streams. It is used as a placeholder to mean that the shared data descriptors can be used in any **data descriptor set**.

For example, to output the shared descriptor **GPS Timestamp (0xFF,0xD3)** in every sensor data packet (**data descriptor set** = 0x80), one would simply add the 0xD3 descriptor to the list of requested data descriptors in the Descriptors parameter when configuring the sensor data stream using **Message Format (0x0C,0x0F)**.

The shared data descriptors are also carried over in continuation packets, as described in **MIP Continuation Packets**.

Parsing Incoming Packets

Setup is usually the easy part of programming the 3DM-CV7. 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.

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

Multiple Rate Data

The message format commands allow you to set different data rates for different data quantities. This is a very useful feature because some data, such as accelerometer and gyroscope data, usually requires higher data rates (>100 Hz) than other IMU data quantities, such as pressure sensor output. 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 gyro data at 50 Hz, the gyro 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 gyro data field:

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

Fletcher Checksum

The checksum is a 2 byte Fletcher checksum and encompasses all the bytes in the packet (excluding the packet checksum bytes)

Fletcher Checksum algorithm

```
uint16_t fletcher_checksum(const uint8_t* packet, int packet_length)
{
  // Checksum covers from the first header byte to the last payload byte.
  // This should be equal to the payload length plus the 4 header bytes.
  const int checksum_length = packet_length - 2;

  uint8_t checksum_MSB = 0;
  uint8_t checksum_LSB = 0;

  // Iterate over the packet to compute the checksum.
  for(int i=0; i<checksum_length; i++)
  {
    checksum_MSB += packet[i];
    checksum_LSB += checksum_MSB;
  }
  return ((uint16_t)checksum_MSB << 8) | (uint16_t)checksum_LSB;
}
```

Communications Bandwidth Management

Because of the large amount and variety of data that is available from the 3DM-CV7, it is quite easy to overdrive the bandwidth of the communications channel. This can result in dropped packets. The 3DM-CV7 does not do analysis of the bandwidth requirements for any given output data configuration, it will simply drop a packet if its internal serial buffer is being filled faster than it is being emptied. It is up to the programmer to analyze the size of the data packets requested and the available bandwidth of the communications channel. Often the best way to determine this is empirically by trying different settings and watching for dropped packets. Below are some guidelines on how to determine maximum bandwidth for your application.

UART Bandwidth Calculation

USB vs. UART

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. In addition, it is critical to account for command responses in the data stream. Verify the baudrate can support the requested data output rate by streaming data and watching for dropped packets or failed command responses. 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:

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

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.

USB vs. UART

The 3DM-CV7 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-CV7 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-CV7) 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-CV7 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.

MIP API

Filter (0x82)

LLH Position (0x82,0x01)

Description	**Filter reported position in the WGS84 geodetic frame.**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*28*
Descriptor	*u8*	*0x01*
Latitude	double	[degrees]
Longitude	double	[degrees]
Ellipsoid Height	double	[meters]
Valid Flags	u16	0 - Invalid, 1 - valid

Velocity NED (0x82,0x02)

Description	**Filter reported velocity in the NED local-level frame.**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*16*
Descriptor	*u8*	*0x02*
North	float	[meters/second]
East	float	[meters/second]
Down	float	[meters/second]
Valid Flags	u16	0 - Invalid, 1 - valid

Attitude Quaternion (0x82,0x03)

Description	**4x1 vector representation of the quaternion describing the orientation of the device with respect to the NED local-level frame.**	
Notes	This quaternion satisfies the following relationship: Where: is the quaternion describing the rotation. and is a 3-element vector expressed in the NED frame. and is a 3-element vector expressed in the vehicle frame.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*4*
Descriptor	*u8*	*0x03*
Q	Quatf	Quaternion elements
Valid Flags	u16	0 - invalid, 1 - valid

Attitude DCM (0x82,0x04)

Description	**3x3 Direction Cosine Matrix describing the orientation of the device with respect to the NED local-level frame.**	
Notes	This matrix satisfies the following relationship: Where: is a 3-element vector expressed in the NED frame. is the same 3-element vector expressed in the vehicle frame. The matrix elements are stored is row-major order:	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*4*
Descriptor	*u8*	*0x04*
Dcm	Matrix3f	Matrix elements in row-major order.
Valid Flags	u16	0 - invalid, 1 - valid

Euler Angles (0x82,0x05)

Description	**Filter reported Euler angles describing the orientation of the device with respect to the NED local-level frame.**	
Notes	The Euler angles are reported in 3-2-1 (Yaw-Pitch-Roll, AKA Aircraft) order.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*16*
Descriptor	*u8*	*0x05*
Roll	float	[radians]
Pitch	float	[radians]
Yaw	float	[radians]
Valid Flags	u16	0 - invalid, 1 - valid

Gyro Bias (0x82,0x06)

Description	**Filter reported gyro bias expressed in the sensor frame.**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*4*
Descriptor	*u8*	*0x06*
Bias	Vector3f	(x, y, z) [radians/second]
Valid Flags	u16	0 - invalid, 1 - valid

Accel Bias (0x82,0x07)

Description	**Filter reported accelerometer bias expressed in the sensor frame.**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*4*
Descriptor	*u8*	*0x07*
Bias	Vector3f	(x, y, z) [meters/second^2]
Valid Flags	u16	0 - invalid, 1 - valid

LLH Position Uncertainty (0x82,0x08)

Description	**Filter reported 1-sigma position uncertainty in the NED local-level frame.**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*16*
Descriptor	*u8*	*0x08*
North	float	[meters]
East	float	[meters]
Down	float	[meters]
Valid Flags	u16	0 - invalid, 1 - valid

NED Velocity Uncertainty (0x82,0x09)

Description	**Filter reported 1-sigma velocity uncertainties in the NED local-level frame.**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*16*
Descriptor	*u8*	*0x09*
North	float	[meters/second]
East	float	[meters/second]
Down	float	[meters/second]
Valid Flags	u16	0 - invalid, 1 - valid

Euler Angles Uncertainty (0x82,0x0A)

Description	**Filter reported 1-sigma Euler angle uncertainties.**	
Notes	The uncertainties are reported in 3-2-1 (Yaw-Pitch-Roll, AKA Aircraft) order.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*16*
Descriptor	*u8*	*0x0A*
Roll	float	[radians]
Pitch	float	[radians]
Yaw	float	[radians]
Valid Flags	u16	0 - invalid, 1 - valid

Gyro Bias Uncertainty (0x82,0x0B)

Description	**Filter reported 1-sigma gyro bias uncertainties expressed in the sensor frame.**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*4*
Descriptor	*u8*	*0x0B*
Bias Uncert	Vector3f	(x,y,z) [radians/sec]
Valid Flags	u16	0 - invalid, 1 - valid

Accel Bias Uncertainty (0x82,0x0C)

Description	**Filter reported 1-sigma accelerometer bias uncertainties expressed in the sensor frame.**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*4*
Descriptor	*u8*	*0x0C*
Bias Uncert	Vector3f	(x,y,z) [meters/second^2]
Valid Flags	u16	0 - invalid, 1 - valid

Linear Accel (0x82,0x0D)

Description	**Filter-compensated linear acceleration expressed in the vehicle frame.**	
Notes	Note: The estimated gravity has been removed from this data leaving only linear acceleration.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*4*
Descriptor	*u8*	*0x0D*
Accel	Vector3f	(x,y,z) [meters/second^2]
Valid Flags	u16	0 - invalid, 1 - valid

Comp Angular Rate (0x82,0x0E)

Description	**Filter-compensated angular rate expressed in the vehicle frame.**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*4*
Descriptor	*u8*	*0x0E*
Gyro	Vector3f	(x, y, z) [radians/second]
Valid Flags	u16	0 - invalid, 1 - valid

Status (0x82,0x10)

Description	Device-specific filter status indicators.	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*8*
Descriptor	*u8*	*0x10*
Filter State	u16 enum	Device-specific filter state. Please consult the user manual for definition.
Dynamics Mode	u16 enum	Device-specific dynamics mode. Please consult the user manual for definition.
Status Flags	u16 bitfield	Device-specific status flags. Please consult the user manual for definition.

Filter State:

Name	Value	Descriptio
GX5_STARTUP	0	
GX5_INIT	1	
GX5_RUN_SOLUTION_VALID	2	
GX5_RUN_SOLUTION_ERROR	3	
INIT	1	
VERT_GYRO	2	
AHRS	3	
FULL_NAV	4	

Dynamics Mode:

Name	Value	Description
GX5_PORTABLE	1	
GX5_AUTOMOTIVE	2	
GX5_AIRBORNE	3	
GQ7_DEFAULT	1	

Status Flags:

Name	Bit(s)	Description
gx5_init_no_attitude	12	
gx5_init_no_position_velocity	13	
gx5_run_imu_unavailable	0	
gx5_run_gps_unavailable	1	
gx5_run_matrix_singularity	3	
gx5_run_position_covariance_warning	4	
gx5_run_velocity_covariance_warning	5	
gx5_run_attitude_covariance_warning	6	
gx5_run_nan_in_solution_warning	7	
gx5_run_gyro_bias_est_high_warning	8	
gx5_run_accel_bias_est_high_warning	9	
gx5_run_gyro_scale_factor_est_high_warning	10	
gx5_run_accel_scale_factor_est_high_warning	11	
gx5_run_mag_bias_est_high_warning	12	
gx5_run_ant_offset_correction_est_high_warning	13	

gx5_run_ant_offset_correction_est_high_warning	13	
gx5_run_mag_hard_iron_est_high_warning	14	
gx5_run_mag_soft_iron_est_high_warning	15	
gq7_filter_condition	0-1	
gq7_roll_pitch_warning	2	
gq7_heading_warning	3	
gq7_position_warning	4	
gq7_velocity_warning	5	
gq7_imu_bias_warning	6	
gq7_gnss_clk_warning	7	
gq7_antenna_lever_arm_warning	8	
gq7_mounting_transform_warning	9	
gq7_time_sync_warning	10	No time synchronization pulse detected
gq7_solution_error	12-15	Filter computation warning flags. If any bits 12 set, and all filter outputs will be invalid.

Timestamp (0x82,0x11)

Description	**GPS timestamp of the Filter data**	
Notes	Should the PPS become unavailable, the device will revert to its internal clock, which will cause the reported time to drift from true GPS time. Upon recovering from a PPS outage, the user should expect a jump in the reported GPS time due to the accumulation of internal clock error. If synchronization to an external clock or onboard GNSS receiver (for products that have one) is disabled, this time is equivalent to internal system time. Note: this data field may be deprecated in the future. The more flexible shared data field (0x82, 0xD3) should be used instead.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*14*
Descriptor	*u8*	*0x11*
Tow	double	GPS Time of Week [seconds]
Week Number	u16	GPS Week Number since 1980 [weeks]
Valid Flags	u16	0 - invalid, 1 - valid

Quaternion Attitude Uncertainty (0x82,0x12)

Description	**Filter reported quaternion uncertainties.**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*4*
Descriptor	*u8*	*0x12*
Q	Quatf	[dimensionless]
Valid Flags	u16	0 - invalid, 1 - valid

Gravity Vector (0x82,0x13)

Description	**Filter reported gravity vector expressed in the vehicle frame.**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*4*
Descriptor	*u8*	*0x13*
Gravity	Vector3f	(x, y, z) [meters/second^2]
Valid Flags	u16	0 - invalid, 1 - valid

Magnetic Model (0x82,0x15)

Description	**The World Magnetic Model is used for this data. Please refer to the device user manual for the current version of the model.**	
Notes	A valid GNSS location is required for the model to be valid.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*24*
Descriptor	*u8*	*0x15*
Intensity North	float	[Gauss]
Intensity East	float	[Gauss]
Intensity Down	float	[Gauss]
Inclination	float	[radians]
Declination	float	[radians]
Valid Flags	u16	0 - invalid, 1 - valid

Compensated Acceleration (0x82,0x1C)

Description	**Filter-compensated acceleration expressed in the vehicle frame.**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*4*
Descriptor	*u8*	*0x1C*
Accel	Vector3f	(x,y,z) [meters/second^2]
Valid Flags	u16	0 - invalid, 1 - valid

Pressure Altitude (0x82,0x21)

Description	**Filter reported 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.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*8*
Descriptor	*u8*	*0x21*
Pressure Altitude	float	[meters]
Valid Flags	u16	0 - invalid, 1 - valid

Multi Antenna Offset Correction (0x82,0x34)

Description	**Filter reported GNSS antenna offset in vehicle frame.**	
Notes	This offset added to any previously stored offset vector to compensate for errors in definition.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*5*
Descriptor	*u8*	*0x34*
Receiver Id	u8	Receiver ID for the receiver to which the antenna is attached
Offset	Vector3f	(x,y,z) [meters]
Valid Flags	u16	0 - invalid, 1 - valid

Multi Antenna Offset Correction Uncertainty (0x82,0x35)

Description	**Filter reported 1-sigma GNSS antenna offset uncertainties in vehicle frame.**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*5*
Descriptor	*u8*	*0x35*
Receiver Id	u8	Receiver ID for the receiver to which the antenna is attached
Offset Uncert	Vector3f	(x,y,z) [meters]
Valid Flags	u16	0 - invalid, 1 - valid

ECEF Position Uncertainty (0x82,0x36)

Description	**Filter reported 1-sigma position uncertainty in the ECEF frame.**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*4*
Descriptor	*u8*	*0x36*
Pos Uncertainty	Vector3f	[meters]
Valid Flags	u16	0 - invalid, 1 - valid

ECEF Velocity Uncertainty (0x82,0x37)

Description	**Filter reported 1-sigma velocity uncertainties in the ECEF frame.**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*4*
Descriptor	*u8*	*0x37*
Vel Uncertainty	Vector3f	[meters/second]
Valid Flags	u16	0 - invalid, 1 - valid

ECEF Position (0x82,0x40)

Description	**Filter reported ECEF position**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*4*
Descriptor	*u8*	*0x40*
Position Ecef	Vector3d	[meters, ECEF]
Valid Flags	u16	0 - invalid, 1 valid

ECEF Velocity (0x82,0x41)

Description	**Filter reported ECEF velocity**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*4*
Descriptor	*u8*	*0x41*
Velocity Ecef	Vector3f	[meters/second, ECEF]
Valid Flags	u16	0 - invalid, 1 valid

NED Relative Position (0x82,0x42)

Description	**Filter reported relative position, with respect to configured reference position**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*4*
Descriptor	*u8*	*0x42*
Relative Position	Vector3d	[meters, NED]
Valid Flags	u16	0 - invalid, 1 - valid

GNSS Position Aiding Status (0x82,0x43)

Description	Filter reported GNSS position aiding status	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*17*
Descriptor	*u8*	*0x43*
Receiver Id	u8	
Time Of Week	float	Last GNSS aiding measurement time of week [seconds]
Status	u16 bitfield	Aiding measurement status bitfield <table><tr><th>Name</th><th>Bit(s)</th><th>Description</th></tr><tr><td>tight_coupling</td><td>0</td><td>If 1, the Kalman filter is processing raw range information from this GNSS module</td></tr><tr><td>differential</td><td>1</td><td>If 1, the Kalman filter is processing RTK corrections from this GNSS module</td></tr><tr><td>integer_fix</td><td>2</td><td>If 1, the Kalman filter has an RTK integer fix from this GNSS module, indicating th position performance possible</td></tr><tr><td>GPS_L1</td><td>3</td><td>If 1, the Kalman filter is using GPS L1 measurements</td></tr><tr><td>GPS_L2</td><td>4</td><td>If 1, the Kalman filter is using GPS L2 measurements</td></tr><tr><td>GPS_L5</td><td>5</td><td>If 1, the Kalman filter is using GPS L5 measurements (not available on the GQ7)</td></tr><tr><td>GLO_L1</td><td>6</td><td>If 1, the Kalman filter is using GLONASS L1 measurements</td></tr><tr><td>GLO_L2</td><td>7</td><td>If 1, the Kalman filter is using GLONASS L2 measurements</td></tr><tr><td>GAL_E1</td><td>8</td><td>If 1, the Kalman filter is using Galileo E1 measurements</td></tr><tr><td>GAL_E5</td><td>9</td><td>If 1, the Kalman filter is using Galileo E5 measurements</td></tr><tr><td>GAL_E6</td><td>10</td><td>If 1, the Kalman filter is using Galileo E6 measurements</td></tr><tr><td>BEI_B1</td><td>11</td><td>If 1, the Kalman filter is using Beidou B1 measurements (not enabled on GQ7 cur</td></tr><tr><td>BEI_B2</td><td>12</td><td>If 1, the Kalman filter is using Beidou B2 measurements (not enabled on GQ7 cur</td></tr><tr><td>BEI_B3</td><td>13</td><td>If 1, the Kalman filter is using Beidou B3 measurements (not available on the GQ7</td></tr><tr><td>no_fix</td><td>14</td><td>If 1, this GNSS module is reporting no position fix</td></tr><tr><td>config_error</td><td>15</td><td>If 1, there is likely an issue with the antenna offset for this GNSS module</td></tr></table>
Reserved	u8[8]	

Aiding Measurement Summary (0x82,0x46)

Description	**Filter reported aiding measurement summary. This message contains a summary of the specified aidi measurement over the previous measurement interval ending at the specified time.**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*9*
Descriptor	*u8*	*0x46*
Time Of Week	float	[seconds]
Source	u8	
Type	u8 enum	(see product manual for supported types) Note: values 0x20 and above correspond to commanded aidi measurements in the 0x13 Aiding command set.
Indicator	u8 bitfield	

Type values:

Name	Value	Descripti
GNSS	1	
DUAL_ANTENNA	2	
HEADING	3	
PRESSURE	4	
MAGNETOMETER	5	
SPEED	6	
AIDING_POS_ECEF	33	
AIDING_POS_LLH	34	
AIDING_HEIGHT_ABOVE_ELLIPSOID	35	
AIDING_VEL_ECEF	40	
AIDING_VEL_NED	41	
AIDING_VEL_BODY_FRAME	42	
AIDING_HEADING_TRUE	49	
AIDING_MAGNETIC_FIELD	50	
AIDING_PRESSURE	51	

Indicator bits:

Name	Bit(s)	Description
enabled	0	
used	1	
residual_high_warning	2	
sample_time_warning	3	
configuration_error	4	
max_num_meas_exceeded	5	

Odometer Scale Factor Error (0x82,0x47)

Description	**Filter reported odometer scale factor error. The total scale factor estimate is the user indicated scale factor, plus the user indicated scale factor times the scale factor error.**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*8*
Descriptor	*u8*	*0x47*
Scale Factor Error	float	[dimensionless]
Valid Flags	u16	0 - invalid, 1 - valid

Odometer Scale Factor Error Uncertainty (0x82,0x48)

Description	**Filter reported odometer scale factor error uncertainty.**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*8*
Descriptor	*u8*	*0x48*
Scale Factor Error Uncertainty	float	[dimensionless]
Valid Flags	u16	0 - invalid, 1 - valid

GNSS Dual Antenna Status (0x82,0x49)

<table>
<tr><th>Description</th><th colspan="2">Summary information for status of GNSS dual antenna heading estimate.</th></tr>
<tr><td>Notes</td><td colspan="2"></td></tr>
<tr><th>Parameter Name</th><th>Data Type</th><th>Description</th></tr>
<tr><td>Field Length</td><td>u8</td><td>19</td></tr>
<tr><td>Descriptor</td><td>u8</td><td>0x49</td></tr>
<tr><td>Time Of Week</td><td>float</td><td>Last dual-antenna GNSS aiding measurement time of week [seconds]</td></tr>
<tr><td>Heading</td><td>float</td><td>[radians]</td></tr>
<tr><td>Heading Unc</td><td>float</td><td>[radians]</td></tr>
<tr><td>Fix Type</td><td>u8 enum</td><td>Fix type indicator
<table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td>FIX_NONE</td><td>0</td><td></td></tr><tr><td>FIX_DA_FLOAT</td><td>1</td><td></td></tr><tr><td>FIX_DA_FIXED</td><td>2</td><td></td></tr></table></td></tr>
<tr><td>Status Flags</td><td>u16 bitfield</td><td><table><tr><th>Name</th><th>Bit(s)</th><th>Description</th></tr><tr><td>rcv_1_data_valid</td><td>0</td><td></td></tr><tr><td>rcv_2_data_valid</td><td>1</td><td></td></tr><tr><td>antenna_offsets_valid</td><td>2</td><td></td></tr></table></td></tr>
<tr><td>Valid Flags</td><td>u16</td><td>0 - invalid, 1 - valid</td></tr>
</table>

Aiding Frame Configuration Error (0x82,0x50)

Description	**Filter reported aiding source frame configuration error**	
Notes	These estimates are used to compensate for small errors to the user-supplied aiding frame configurations (set with (0x13, 0x01) command).	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*3*
Descriptor	*u8*	*0x50*
Frame Id	u8	Frame ID for the receiver to which the antenna is attached
Translation	Vector3f	Translation config X, Y, and Z (m).
Attitude	Quatf	Attitude quaternion

Aiding Frame Configuration Error Uncertainty (0x82,0x51)

Description	**Filter reported aiding source frame configuration error uncertainty**	
Notes	These estimates are used to compensate for small errors to the user-supplied aiding frame configurations (set with (0x13, 0x01) command).	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*3*
Descriptor	*u8*	*0x51*
Frame Id	u8	Frame ID for the receiver to which the antenna is attached
Translation Unc	Vector3f	Translation uncertaint X, Y, and Z (m).
Attitude Unc	Vector3f	Attitude uncertainty, X, Y, and Z (radians).

GNSS Corrections (0x93)

Base Station Info (0x93,0x30)

Description	**RTCM reported base station information (sourced from RTCM Message 1005 or 1006)**	
Notes	Valid Flag Mapping:	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*22*
Descriptor	*u8*	*0x30*
Time Of Week	double	GPS Time of week the message was received [seconds]
Week Number	u16	GPS Week since 1980 [weeks]
Ecef Pos	Vector3d	Earth-centered, Earth-fixed [m]
Height	float	Antenna Height above the marker used in the survey [m]
Station Id	u16	Range: 0-4095
Indicators	u16 bitfield	Bitfield (see table below)
Valid Flags	u16 bitfield	(see table below)

Indicators bitfield:

Name	Bit(s)	Description
gps	0	
glonass	1	
galileo	2	
beidou	3	
ref_station	4	
single_receiver	5	
quarter_cycle_bit1	6	
quarter_cycle_bit2	7	
quarter_cycle_bits	6-7	

Valid Flags:

Name	Bit(s)	Description
tow	0	
week_number	1	
ecef_position	2	
height	3	
station_id	4	
indicators	5	
flags	0-5	

RTK Corrections Status (0x93,0x31)

<table>
<tr><th>Description</th><th colspan="2"></th></tr>
<tr><td>Notes</td><td colspan="2"></td></tr>
<tr><th>Parameter Name</th><th>Data Type</th><th>Description</th></tr>
<tr><td>Field Length</td><td>u8</td><td>52</td></tr>
<tr><td>Descriptor</td><td>u8</td><td>0x31</td></tr>
<tr><td>Time Of Week</td><td>double</td><td>GPS Time of week [seconds]</td></tr>
<tr><td>Week Number</td><td>u16</td><td>GPS Week since 1980 [weeks]</td></tr>
<tr><td>Epoch Status</td><td>u16 bitfield</td><td>Status of the corrections received during this epoch
<table>
<tr><th>Name</th><th>Bit(s)</th><th>Description</th></tr>
<tr><td>antenna_location_received</td><td>0</td><td></td></tr>
<tr><td>antenna_description_received</td><td>1</td><td></td></tr>
<tr><td>gps_received</td><td>2</td><td></td></tr>
<tr><td>glonass_received</td><td>3</td><td></td></tr>
<tr><td>galileo_received</td><td>4</td><td></td></tr>
<tr><td>beidou_received</td><td>5</td><td></td></tr>
<tr><td>using_gps_msm_messages</td><td>6</td><td>Using MSM messages for GPS corrections instead of RTCM messages 1001-1004</td></tr>
<tr><td>using_glonass_msm_messages</td><td>7</td><td>Using MSM messages for GLONASS corrections instead of R messages 1009-1012</td></tr>
<tr><td>dongle_status_read_failed</td><td>8</td><td>A read of the dongle status was attempted, but failed</td></tr>
</table>
</td></tr>
<tr><td>Dongle Status</td><td>u32</td><td>RTK Dongle Status Flags (valid only when using RTK dongle, see Get RTK Device Status Flags (0x0F,0 details)</td></tr>
<tr><td>Gps Correction Latency</td><td>float</td><td>Latency of last GPS correction [seconds]</td></tr>
<tr><td>Glonass Correction Latency</td><td>float</td><td>Latency of last GLONASS correction [seconds]</td></tr>
<tr><td>Galileo Correction Latency</td><td>float</td><td>Latency of last Galileo correction [seconds]</td></tr>
<tr><td>Beidou Correction Latency</td><td>float</td><td>Latency of last Beidou correction [seconds]</td></tr>
<tr><td>Reserved</td><td>u32[4]</td><td>Reserved for future use</td></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Bit(s)</th><th>Description</th></tr>
<tr><td>tow</td><td>0</td><td></td></tr>
<tr><td>week_number</td><td>1</td><td></td></tr>
<tr><td>epoch_status</td><td>2</td><td></td></tr>
</table>
</td></tr>
</table>

Valid Flags	u16 bitfield	dongle_status	3	
		gps_latency	4	
		glonass_latency	5	
		galileo_latency	6	
		beidou_latency	7	
		flags	0-7	

GNSS External (0x94)

GNSS LLH Position (0x94,0x03)

Description	**GNSS reported position in the WGS84 geodetic frame**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*44*
Descriptor	*u8*	*0x03*
Latitude	double	[degrees]
Longitude	double	[degrees]
Ellipsoid Height	double	[meters]
Msl Height	double	[meters]
Horizontal Accuracy	float	[meters]
Vertical Accuracy	float	[meters]
Valid Flags	u16 bitfield	see table below

Valid Flags bitfield:

Name	Bit(s)	Description
lat_lon	0	
ellipsoid_height	1	
msl_height	2	
horizontal_accuracy	3	
vertical_accuracy	4	
flags	0-4	

NED Velocity (0x94,0x05)

Description	**GNSS reported velocity in the NED frame**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*24*
Descriptor	*u8*	*0x05*
V	Vector3f	[meters/second]
Speed	float	[meters/second]
Ground Speed	float	[meters/second]
Heading	float	[degrees]
Speed Accuracy	float	[meters/second]
Heading Accuracy	float	[degrees]
Valid Flags	u16 bitfield	(see table below)

Name	Bit(s)	Description
velocity	0	
speed_3d	1	
ground_speed	2	
heading	3	
speed_accuracy	4	
heading_accuracy	5	
flags	0-5	

UTC Time (0x94,0x08)

Description	**GNSS reported Coordinated Universal Time**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*15*
Descriptor	*u8*	*0x08*
Year	u16	
Month	u8	Month (1-12)
Day	u8	Day (1-31)
Hour	u8	Hour (0-23)
Min	u8	Minute (0-59)
Sec	u8	Second (0-59)
Msec	u32	Millisecond(0-999)
Valid Flags	u16 bitfield	see table below

Name	Bit(s)	Description
gnss_date_time	0	
leap_seconds_known	1	
flags	0-1	

Fix Info (0x94,0x0B)

Description	**GNSS reported position fix type**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*8*
Descriptor	*u8*	*0x0B*
Fix Type	u8 enum	<table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td>FIX_3D</td><td>0</td><td></td></tr><tr><td>FIX_2D</td><td>1</td><td></td></tr><tr><td>FIX_TIME_ONLY</td><td>2</td><td></td></tr><tr><td>FIX_NONE</td><td>3</td><td></td></tr><tr><td>FIX_INVALID</td><td>4</td><td></td></tr><tr><td>FIX_RTK_FLOAT</td><td>5</td><td></td></tr><tr><td>FIX_RTK_FIXED</td><td>6</td><td></td></tr><tr><td>FIX_DIFFERENTIAL</td><td>7</td><td></td></tr></table>
Num Sv	u8	
Fix Flags	u16 bitfield	<table><tr><th>Name</th><th>Bit(s)</th><th>Description</th></tr><tr><td>sbas_used</td><td>0</td><td></td></tr><tr><td>dgnss_used</td><td>1</td><td></td></tr></table>
Valid Flags	u16 bitfield	<table><tr><th>Name</th><th>Bit(s)</th><th>Description</th></tr><tr><td>fix_type</td><td>0</td><td></td></tr><tr><td>num_sv</td><td>1</td><td></td></tr><tr><td>fix_flags</td><td>2</td><td></td></tr><tr><td>flags</td><td>0-2</td><td></td></tr></table>

Satellite Status (0x94,0x20)

Description	**Status information for a GNSS satellite.**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*27*
Descriptor	*u8*	*0x20*
Index	u8	Index of this field in this epoch.
Count	u8	Total number of fields in this epoch.
Time Of Week	double	GPS Time of week [seconds]
Week Number	u16	GPS Week since 1980 [weeks]
Gnss Id	u8 enum	<table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td>UNKNOWN</td><td>0</td><td></td></tr><tr><td>GPS</td><td>1</td><td></td></tr><tr><td>GLONASS</td><td>2</td><td></td></tr><tr><td>GALILEO</td><td>3</td><td></td></tr><tr><td>BEIDOU</td><td>4</td><td></td></tr><tr><td>SBAS</td><td>5</td><td></td></tr></table>
Satellite Id	u8	GNSS satellite id within the constellation
Elevation	float	Elevation of the satellite relative to the rover [degrees]
Azimuth	float	Azimuth of the satellite relative to the rover [degrees]
Health	bool	True if the satellite is healthy.
Valid Flags	u16 bitfield	<table><tr><th>Name</th><th>Bit(s)</th><th>Description</th></tr><tr><td>tow</td><td>0</td><td></td></tr><tr><td>week_number</td><td>1</td><td></td></tr><tr><td>gnss_id</td><td>2</td><td></td></tr><tr><td>satellite_id</td><td>3</td><td></td></tr><tr><td>elevation</td><td>4</td><td></td></tr><tr><td>azimuth</td><td>5</td><td></td></tr><tr><td>health</td><td>6</td><td></td></tr><tr><td>flags</td><td>0-6</td><td></td></tr></table>

GNSS Recv 1 (0x91)

GNSS LLH Position (0x91,0x03)

Description	**GNSS reported position in the WGS84 geodetic frame**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*44*
Descriptor	*u8*	*0x03*
Latitude	double	[degrees]
Longitude	double	[degrees]
Ellipsoid Height	double	[meters]
Msl Height	double	[meters]
Horizontal Accuracy	float	[meters]
Vertical Accuracy	float	[meters]
Valid Flags	u16 bitfield	(see nested table below)

Valid Flags (u16 bitfield):

Name	Bit(s)	Description
lat_lon	0	
ellipsoid_height	1	
msl_height	2	
horizontal_accuracy	3	
vertical_accuracy	4	
flags	0-4	

GNSS ECEF Position (0x91,0x04)

Description	**GNSS reported position in the Earth-centered, Earth-Fixed (ECEF) frame**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*8*
Descriptor	*u8*	*0x04*
X	Vector3d	[meters]
X Accuracy	float	[meters]
Valid Flags	u16 bitfield	(see nested table below)

Valid Flags bitfield:

Name	Bit(s)	Description
position	0	
position_accuracy	1	
flags	0-1	

NED Velocity (0x91,0x05)

Description	GNSS reported velocity in the NED frame	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*24*
Descriptor	*u8*	*0x05*
V	Vector3f	[meters/second]
Speed	float	[meters/second]
Ground Speed	float	[meters/second]
Heading	float	[degrees]
Speed Accuracy	float	[meters/second]
Heading Accuracy	float	[degrees]
Valid Flags	u16 bitfield	see table below

Name	Bit(s)	Description
velocity	0	
speed_3d	1	
ground_speed	2	
heading	3	
speed_accuracy	4	
heading_accuracy	5	
flags	0-5	

GNSS ECEF Velocity (0x91,0x06)

Description	**GNSS reported velocity in the Earth-centered, Earth-Fixed (ECEF) frame**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*8*
Descriptor	*u8*	*0x06*
V	Vector3f	[meters/second]
V Accuracy	float	[meters/second]
Valid Flags	u16 bitfield	(see bitfield table below)

Name	Bit(s)	Description
velocity	0	
velocity_accuracy	1	
flags	0-1	

DOP (0x91,0x07)

Description	**GNSS reported dilution of precision information.**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*32*
Descriptor	*u8*	*0x07*
Gdop	float	Geometric DOP
Pdop	float	Position DOP
Hdop	float	Horizontal DOP
Vdop	float	Vertical DOP
Tdop	float	Time DOP
Ndop	float	Northing DOP
Edop	float	Easting DOP
Valid Flags	u16 bitfield	(see table below)

Valid Flags (u16 bitfield):

Name	Bit(s)	Description
gdop	0	
pdop	1	
hdop	2	
vdop	3	
tdop	4	
ndop	5	
edop	6	
flags	0-6	

UTC Time (0x91,0x08)

Description	**GNSS reported Coordinated Universal Time**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*15*
Descriptor	*u8*	*0x08*
Year	u16	
Month	u8	Month (1-12)
Day	u8	Day (1-31)
Hour	u8	Hour (0-23)
Min	u8	Minute (0-59)
Sec	u8	Second (0-59)
Msec	u32	Millisecond(0-999)
Valid Flags	u16 bitfield	(see bitfield table below)

Name	Bit(s)	Description
gnss_date_time	0	
leap_seconds_known	1	
flags	0-1	

GPS Time (0x91,0x09)

Description	GNSS reported GPS Time	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*14*
Descriptor	*u8*	*0x09*
Tow	double	GPS Time of week [seconds]
Week Number	u16	GPS Week since 1980 [weeks]
Valid Flags	u16 bitfield	<table><tr><th>Name</th><th>Bit(s)</th><th>Description</th></tr><tr><td>tow</td><td>0</td><td></td></tr><tr><td>week_number</td><td>1</td><td></td></tr><tr><td>flags</td><td>0-1</td><td></td></tr></table>

Fix Info (0x91,0x0B)

<table>
<tr><th>Description</th><th colspan="2">GNSS reported position fix type</th></tr>
<tr><td>Notes</td><td colspan="2"></td></tr>
<tr><th>Parameter Name</th><th>Data Type</th><th>Description</th></tr>
<tr><td>Field Length</td><td>u8</td><td>8</td></tr>
<tr><td>Descriptor</td><td>u8</td><td>0x0B</td></tr>
<tr><td>Fix Type</td><td>u8 enum</td><td>
<table>
<tr><th>Name</th><th>Value</th><th>Description</th></tr>
<tr><td>FIX_3D</td><td>0</td><td></td></tr>
<tr><td>FIX_2D</td><td>1</td><td></td></tr>
<tr><td>FIX_TIME_ONLY</td><td>2</td><td></td></tr>
<tr><td>FIX_NONE</td><td>3</td><td></td></tr>
<tr><td>FIX_INVALID</td><td>4</td><td></td></tr>
<tr><td>FIX_RTK_FLOAT</td><td>5</td><td></td></tr>
<tr><td>FIX_RTK_FIXED</td><td>6</td><td></td></tr>
<tr><td>FIX_DIFFERENTIAL</td><td>7</td><td></td></tr>
</table>
</td></tr>
<tr><td>Num Sv</td><td>u8</td><td></td></tr>
<tr><td>Fix Flags</td><td>u16 bitfield</td><td>
<table>
<tr><th>Name</th><th>Bit(s)</th><th>Description</th></tr>
<tr><td>sbas_used</td><td>0</td><td></td></tr>
<tr><td>dgnss_used</td><td>1</td><td></td></tr>
</table>
</td></tr>
<tr><td>Valid Flags</td><td>u16 bitfield</td><td>
<table>
<tr><th>Name</th><th>Bit(s)</th><th>Description</th></tr>
<tr><td>fix_type</td><td>0</td><td></td></tr>
<tr><td>num_sv</td><td>1</td><td></td></tr>
<tr><td>fix_flags</td><td>2</td><td></td></tr>
<tr><td>flags</td><td>0-2</td><td></td></tr>
</table>
</td></tr>
</table>

GNSS Hardware Status (0x91,0x0D)

<table>
<tr><th>Description</th><th colspan="2">GNSS reported hardware status</th></tr>
<tr><td>Notes</td><td colspan="2"></td></tr>
<tr><td>Parameter Name</td><td>Data Type</td><td>Description</td></tr>
<tr><td>Field Length</td><td>u8</td><td>7</td></tr>
<tr><td>Descriptor</td><td>u8</td><td>0x0D</td></tr>
<tr><td>Receiver State</td><td>u8 enum</td><td><table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td>OFF</td><td>0</td><td></td></tr><tr><td>ON</td><td>1</td><td></td></tr><tr><td>UNKNOWN</td><td>2</td><td></td></tr></table></td></tr>
<tr><td>Antenna State</td><td>u8 enum</td><td><table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td>INIT</td><td>1</td><td></td></tr><tr><td>SHORT</td><td>2</td><td></td></tr><tr><td>OPEN</td><td>3</td><td></td></tr><tr><td>GOOD</td><td>4</td><td></td></tr><tr><td>UNKNOWN</td><td>5</td><td></td></tr></table></td></tr>
<tr><td>Antenna Power</td><td>u8 enum</td><td><table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td>OFF</td><td>0</td><td></td></tr><tr><td>ON</td><td>1</td><td></td></tr><tr><td>UNKNOWN</td><td>2</td><td></td></tr></table></td></tr>
<tr><td>Valid Flags</td><td>u16 bitfield</td><td><table><tr><th>Name</th><th>Bit(s)</th><th>Description</th></tr><tr><td>sensor_state</td><td>0</td><td></td></tr><tr><td>antenna_state</td><td>1</td><td></td></tr><tr><td>antenna_power</td><td>2</td><td></td></tr><tr><td>flags</td><td>0-2</td><td></td></tr></table></td></tr>
</table>

Clock Info 2 (0x91,0x10)

Description	**GNSS reported receiver clock parameters**	
Notes	This supersedes MIP_DATA_DESC_GNSS_CLOCK_INFO with additional information.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*36*
Descriptor	*u8*	*0x10*
Bias	double	
Drift	double	
Bias Accuracy Estimate	double	
Drift Accuracy Estimate	double	
Valid Flags	u16 bitfield	see below

Valid Flags (u16 bitfield):

Name	Bit(s)	Description
bias	0	
drift	1	
bias_accuracy	2	
drift_accuracy	3	
flags	0-3	

GPS Leap Seconds (0x91,0x11)

Description	GNSS reported leap seconds (difference between GPS and UTC Time)	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*5*
Descriptor	*u8*	*0x11*
Leap Seconds	u8	[s]
Valid Flags	u16 bitfield	**Name**: leap_seconds; **Bit(s)**: 1; **Description**:

Name	Bit(s)	Description
leap_seconds	1	

SBAS Info (0x91,0x12)

Description	GNSS SBAS status	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*18*
Descriptor	*u8*	*0x12*
Time Of Week	double	GPS Time of week [seconds]
Week Number	u16	GPS Week since 1980 [weeks]
Sbas System	u8 enum	SBAS system id <table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td>UNKNOWN</td><td>0</td><td></td></tr><tr><td>WAAS</td><td>1</td><td></td></tr><tr><td>EGNOS</td><td>2</td><td></td></tr><tr><td>MSAS</td><td>3</td><td></td></tr><tr><td>GAGAN</td><td>4</td><td></td></tr></table>
Sbas Id	u8	SBAS satellite id.
Count	u8	Number of SBAS corrections
Sbas Status	u8 bitfield	Status of the SBAS service <table><tr><th>Name</th><th>Bit(s)</th><th>Description</th></tr><tr><td>range_available</td><td>0</td><td></td></tr><tr><td>corrections_available</td><td>1</td><td></td></tr><tr><td>integrity_available</td><td>2</td><td></td></tr><tr><td>test_mode</td><td>3</td><td></td></tr></table>
Valid Flags	u16 bitfield	<table><tr><th>Name</th><th>Bit(s)</th><th>Description</th></tr><tr><td>tow</td><td>0</td><td></td></tr><tr><td>week_number</td><td>1</td><td></td></tr><tr><td>sbas_system</td><td>2</td><td></td></tr><tr><td>sbas_id</td><td>3</td><td></td></tr><tr><td>count</td><td>4</td><td></td></tr><tr><td>sbas_status</td><td>5</td><td></td></tr><tr><td>flags</td><td>0-5</td><td></td></tr></table>

Satellite Status (0x91,0x20)

Description	**Status information for a GNSS satellite.**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*27*
Descriptor	*u8*	*0x20*
Index	u8	Index of this field in this epoch.
Count	u8	Total number of fields in this epoch.
Time Of Week	double	GPS Time of week [seconds]
Week Number	u16	GPS Week since 1980 [weeks]
Gnss Id	u8 enum	<table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td>UNKNOWN</td><td>0</td><td></td></tr><tr><td>GPS</td><td>1</td><td></td></tr><tr><td>GLONASS</td><td>2</td><td></td></tr><tr><td>GALILEO</td><td>3</td><td></td></tr><tr><td>BEIDOU</td><td>4</td><td></td></tr><tr><td>SBAS</td><td>5</td><td></td></tr></table>
Satellite Id	u8	GNSS satellite id within the constellation
Elevation	float	Elevation of the satellite relative to the rover [degrees]
Azimuth	float	Azimuth of the satellite relative to the rover [degrees]
Health	bool	True if the satellite is healthy.
Valid Flags	u16 bitfield	<table><tr><th>Name</th><th>Bit(s)</th><th>Description</th></tr><tr><td>tow</td><td>0</td><td></td></tr><tr><td>week_number</td><td>1</td><td></td></tr><tr><td>gnss_id</td><td>2</td><td></td></tr><tr><td>satellite_id</td><td>3</td><td></td></tr><tr><td>elevation</td><td>4</td><td></td></tr><tr><td>azimuth</td><td>5</td><td></td></tr><tr><td>health</td><td>6</td><td></td></tr><tr><td>flags</td><td>0-6</td><td></td></tr></table>

Raw (0x91,0x22)

<table>
<tr><th>Description</th><th colspan="2">GNSS Raw observation.</th></tr>
<tr><td>Notes</td><td colspan="2"></td></tr>
<tr><th>Parameter Name</th><th>Data Type</th><th>Description</th></tr>
<tr><td>Field Length</td><td>u8</td><td>63</td></tr>
<tr><td>Descriptor</td><td>u8</td><td>0x22</td></tr>
<tr><td>Index</td><td>u8</td><td>Index of this field in this epoch.</td></tr>
<tr><td>Count</td><td>u8</td><td>Total number of fields in this epoch.</td></tr>
<tr><td>Time Of Week</td><td>double</td><td>GPS Time of week [seconds]</td></tr>
<tr><td>Week Number</td><td>u16</td><td>GPS Week since 1980 [weeks]</td></tr>
<tr><td>Receiver Id</td><td>u16</td><td>When the measurement comes from RTCM, this will be the reference station ID; otherwise, it's the rece
number (1,2,...)</td></tr>
<tr><td>Tracking Channel</td><td>u8</td><td>Channel the receiver is using to track this satellite.</td></tr>
<tr><td>Gnss Id</td><td>u8
enum</td><td>
<table>
<tr><th>Name</th><th>Value</th><th>Description</th></tr>
<tr><td>UNKNOWN</td><td>0</td><td></td></tr>
<tr><td>GPS</td><td>1</td><td></td></tr>
<tr><td>GLONASS</td><td>2</td><td></td></tr>
<tr><td>GALILEO</td><td>3</td><td></td></tr>
<tr><td>BEIDOU</td><td>4</td><td></td></tr>
<tr><td>SBAS</td><td>5</td><td></td></tr>
</table>
</td></tr>
<tr><td>Satellite Id</td><td>u8</td><td>GNSS satellite id within the constellation.</td></tr>
<tr><td></td><td></td><td>Signal identifier for the satellite.
<table>
<tr><th>Name</th><th>Value</th><th>Description</th></tr>
<tr><td>UNKNOWN</td><td>0</td><td></td></tr>
<tr><td>GPS_L1CA</td><td>1</td><td></td></tr>
<tr><td>GPS_L1P</td><td>2</td><td></td></tr>
<tr><td>GPS_L1Z</td><td>3</td><td></td></tr>
<tr><td>GPS_L2CA</td><td>4</td><td></td></tr>
<tr><td>GPS_L2P</td><td>5</td><td></td></tr>
<tr><td>GPS_L2Z</td><td>6</td><td></td></tr>
<tr><td>GPS_L2CL</td><td>7</td><td></td></tr>
<tr><td>GPS_L2CM</td><td>8</td><td></td></tr>
<tr><td>GPS_L2CML</td><td>9</td><td></td></tr>
<tr><td>GPS_L5I</td><td>10</td><td></td></tr>
<tr><td>GPS_L5Q</td><td>11</td><td></td></tr>
<tr><td>GPS_L5IQ</td><td>12</td><td></td></tr>
</table>
</td></tr>
</table>

Signal Id	u8 enum	GPS_L1CD	13	
		GPS_L1CP	14	
		GPS_L1CDP	15	
		GLONASS_G1CA	32	
		GLONASS_G1P	33	
		GLONASS_G2C	34	
		GLONASS_G2P	35	
		GALILEO_E1C	64	
		GALILEO_E1A	65	
		GALILEO_E1B	66	
		GALILEO_E1BC	67	
		GALILEO_E1ABC	68	
		GALILEO_E6C	69	
		GALILEO_E6A	70	
		GALILEO_E6B	71	
		GALILEO_E6BC	72	
		GALILEO_E6ABC	73	
		GALILEO_E5BI	74	
		GALILEO_E5BQ	75	
		GALILEO_E5BIQ	76	
		GALILEO_E5ABI	77	
		GALILEO_E5ABQ	78	
		GALILEO_E5ABIQ	79	
		GALILEO_E5AI	80	
		GALILEO_E5AQ	81	
		GALILEO_E5AIQ	82	
		SBAS_L1CA	96	
		SBAS_L5I	97	
		SBAS_L5Q	98	
		SBAS_L5IQ	99	
		QZSS_L1CA	128	
		QZSS_LEXS	129	
		QZSS_LEXL	130	
		QZSS_LEXSL	131	
		QZSS_L2CM	132	
		QZSS_L2CL	133	
		QZSS_L2CML	134	
		QZSS_L5I	135	
		QZSS_L5Q	136	
		QZSS_L5IQ	137	
		QZSS_L1CD	138	
		QZSS_L1CP	139	
		QZSS_L1CDP	140	
		BEIDOU_B1I	160	
		BEIDOU_B1Q	161	
		BEIDOU_B1IQ	162	
		BEIDOU_B3I	163	
		BEIDOU_B3Q	164	
		BEIDOU_B3IQ	165	
		BEIDOU_B2I	166	
		BEIDOU_B2Q	167	
		BEIDOU_B2IQ	168	
		BEIDOU_B2A	169	
Signal Strength	float	Carrier to noise ratio [dBHz].		

<table>
<tr><td>Quality</td><td>u8
enum</td><td>Indicator of signal quality.<table>
<tr><th>Name</th><th>Value</th><th>Description</th></tr>
<tr><td>NONE</td><td>0</td><td></td></tr>
<tr><td>SEARCHING</td><td>1</td><td></td></tr>
<tr><td>ACQUIRED</td><td>2</td><td></td></tr>
<tr><td>UNUSABLE</td><td>3</td><td></td></tr>
<tr><td>TIME_LOCKED</td><td>4</td><td></td></tr>
<tr><td>FULLY_LOCKED</td><td>5</td><td></td></tr>
</table></td></tr>
<tr><td>Pseudorange</td><td>double</td><td>Pseudo-range measurement [meters].</td></tr>
<tr><td>Carrier Phase</td><td>double</td><td>Carrier phase measurement [Carrier periods].</td></tr>
<tr><td>Doppler</td><td>float</td><td>Measured doppler shift [Hz].</td></tr>
<tr><td>Range Uncert</td><td>float</td><td>Uncertainty of the pseudo-range measurement [m].</td></tr>
<tr><td>Phase Uncert</td><td>float</td><td>Uncertainty of the phase measurement [Carrier periods].</td></tr>
<tr><td>Doppler Uncert</td><td>float</td><td>Uncertainty of the measured doppler shift [Hz].</td></tr>
<tr><td>Lock Time</td><td>float</td><td>DOC Minimum carrier phase lock time [s]. Note: the maximum value is dependent on the receiver.</td></tr>
<tr><td>Valid Flags</td><td>u16
bitfield</td><td><table>
<tr><th>Name</th><th>Bit(s)</th><th>Descriptior</th></tr>
<tr><td>tow</td><td>0</td><td></td></tr>
<tr><td>week_number</td><td>1</td><td></td></tr>
<tr><td>receiver_id</td><td>2</td><td></td></tr>
<tr><td>tracking_channel</td><td>3</td><td></td></tr>
<tr><td>gnss_id</td><td>4</td><td></td></tr>
<tr><td>satellite_id</td><td>5</td><td></td></tr>
<tr><td>signal_id</td><td>6</td><td></td></tr>
<tr><td>signal_strength</td><td>7</td><td></td></tr>
<tr><td>quality</td><td>8</td><td></td></tr>
<tr><td>pseudorange</td><td>9</td><td></td></tr>
<tr><td>carrier_phase</td><td>10</td><td></td></tr>
<tr><td>doppler</td><td>11</td><td></td></tr>
<tr><td>range_uncertainty</td><td>12</td><td></td></tr>
<tr><td>carrier_phase_uncertainty</td><td>13</td><td></td></tr>
<tr><td>doppler_uncertainty</td><td>14</td><td></td></tr>
<tr><td>lock_time</td><td>15</td><td></td></tr>
<tr><td>flags</td><td>0-15</td><td></td></tr>
</table></td></tr>
</table>

GPS Ephemeris (0x91,0x61)

Description	GPS Ephemeris Data	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*220*
Descriptor	*u8*	*0x61*
Index	u8	Index of this field in this epoch.
Count	u8	Total number of fields in this epoch.
Time Of Week	double	GPS Time of week [seconds]
Week Number	u16	GPS Week since 1980 [weeks]
Satellite Id	u8	GNSS satellite id within the constellation.
Health	u8	Satellite and signal health
Iodc	u8	Issue of Data Clock. This increments each time the data changes and rolls over at 4. It is used to make s various raw data elements from different sources line up correctly.
Iode	u8	Issue of Data Ephemeris.
T Oc	double	Reference time for clock data.
Af0	double	Clock bias in [s].
Af1	double	Clock drift in [s/s].
Af2	double	Clock drift rate in [s/s^2].
T Gd	double	T Group Delay [s].
Isc L1ca	double	Inter-signal correction (L1).
Isc L2c	double	Inter-signal correction (L2, or L5 if isc_l5 flag is set).
T Oe	double	Reference time for ephemeris in [s].
A	double	Semi-major axis [m].
A Dot	double	Semi-major axis rate [m/s].
Mean Anomaly	double	[rad].
Delta Mean Motion	double	[rad].
Delta Mean Motion Dot	double	[rad/s].
Eccentricity	double	
Argument Of Perigee	double	[rad].
Omega	double	Longitude of Ascending Node [rad].

<table>
<tr><td>Omega Dot</td><td>double</td><td>Rate of Right Ascension [rad/s].</td></tr>
<tr><td>Inclination</td><td>double</td><td>Inclination angle [rad].</td></tr>
<tr><td>Inclination Dot</td><td>double</td><td>Inclination angle rate of change [rad/s].</td></tr>
<tr><td>C Ic</td><td>double</td><td>Harmonic Correction Term.</td></tr>
<tr><td>C Is</td><td>double</td><td>Harmonic Correction Term.</td></tr>
<tr><td>C Uc</td><td>double</td><td>Harmonic Correction Term.</td></tr>
<tr><td>C Us</td><td>double</td><td>Harmonic Correction Term.</td></tr>
<tr><td>C Rc</td><td>double</td><td>Harmonic Correction Term.</td></tr>
<tr><td>C Rs</td><td>double</td><td>Harmonic Correction Term.</td></tr>
<tr><td>Valid Flags</td><td>u16 bitfield</td><td><table>
<tr><th>Name</th><th>Bit(s)</th><th>Description</th></tr>
<tr><td>ephemeris</td><td>0</td><td></td></tr>
<tr><td>modern_data</td><td>1</td><td></td></tr>
<tr><td>isc_l5</td><td>2</td><td></td></tr>
<tr><td>flags</td><td>0-2</td><td></td></tr>
</table></td></tr>
</table>

Glonass Ephemeris (0x91,0x62)

Description	Glonass Ephemeris Data	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*56*
Descriptor	*u8*	*0x62*
Index	u8	Index of this field in this epoch.
Count	u8	Total number of fields in this epoch.
Time Of Week	double	GPS Time of week [seconds]
Week Number	u16	GPS Week since 1980 [weeks]
Satellite Id	u8	GNSS satellite id within the constellation.
Freq Number	s8	GLONASS frequency number (-7 to 24)
Tk	u32	Frame start time within current day [seconds]
Tb	u32	Ephemeris reference time [seconds]
Sat Type	u8	Type of satellite (M) GLONASS = 0, GLONASS-M = 1
Gamma	double	Relative deviation of carrier frequency from nominal [dimensionless]
Tau N	double	Time correction relative to GLONASS Time [seconds]
X	Vector3d	Satellite PE-90 position [m]
V	Vector3f	Satellite PE-90 velocity [m/s]
A	Vector3f	Satellite PE-90 acceleration due to perturbations [m/s^2]
Health	u8	Satellite Health (Bn), Non-zero indicates satellite malfunction
P	u8	Satellite operation mode (See GLONASS ICD)
Nt	u8	Day number within a 4 year period.
Delta Tau N	float	Time difference between L1 and L2[m/s]
Ft	u8	User Range Accuracy (See GLONASS ICD)
En	u8	Age of current information [days]
P1	u8	Time interval between adjacent values of tb [minutes]
P2	u8	Oddness "1" or evenness "0" of the value of tb.
P3	u8	Number of satellites in almanac for this frame
P4	u8	Flag indicating ephemeris parameters are present
Valid Flags	u16 bitfield	see table below

Name	Bit(s)	Description
ephemeris	0	
flags	0	

Galileo Ephemeris (0x91,0x63)

Description	Galileo Ephemeris Data	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*220*
Descriptor	*u8*	*0x63*
Index	u8	Index of this field in this epoch.
Count	u8	Total number of fields in this epoch.
Time Of Week	double	GPS Time of week [seconds]
Week Number	u16	GPS Week since 1980 [weeks]
Satellite Id	u8	GNSS satellite id within the constellation.
Health	u8	Satellite and signal health
Iodc	u8	Issue of Data Clock. This increments each time the data changes and rolls over at 4. It is used to make s various raw data elements from different sources line up correctly.
Iode	u8	Issue of Data Ephemeris.
T Oc	double	Reference time for clock data.
Af0	double	Clock bias in [s].
Af1	double	Clock drift in [s/s].
Af2	double	Clock drift rate in [s/s^2].
T Gd	double	T Group Delay [s].
Isc L1ca	double	Inter-signal correction (L1).
Isc L2c	double	Inter-signal correction (L2, or L5 if isc_l5 flag is set).
T Oe	double	Reference time for ephemeris in [s].
A	double	Semi-major axis [m].
A Dot	double	Semi-major axis rate [m/s].
Mean Anomaly	double	[rad].
Delta Mean Motion	double	[rad].
Delta Mean Motion Dot	double	[rad/s].
Eccentricity	double	
Argument Of Perigee	double	[rad].
Omega	double	Longitude of Ascending Node [rad].

<table>
<tr><td>Omega Dot</td><td>double</td><td>Rate of Right Ascension [rad/s].</td></tr>
<tr><td>Inclination</td><td>double</td><td>Inclination angle [rad].</td></tr>
<tr><td>Inclination Dot</td><td>double</td><td>Inclination angle rate of change [rad/s].</td></tr>
<tr><td>C Ic</td><td>double</td><td>Harmonic Correction Term.</td></tr>
<tr><td>C Is</td><td>double</td><td>Harmonic Correction Term.</td></tr>
<tr><td>C Uc</td><td>double</td><td>Harmonic Correction Term.</td></tr>
<tr><td>C Us</td><td>double</td><td>Harmonic Correction Term.</td></tr>
<tr><td>C Rc</td><td>double</td><td>Harmonic Correction Term.</td></tr>
<tr><td>C Rs</td><td>double</td><td>Harmonic Correction Term.</td></tr>
<tr><td>Valid Flags</td><td>u16 bitfield</td><td><table>
<tr><th>Name</th><th>Bit(s)</th><th>Description</th></tr>
<tr><td>ephemeris</td><td>0</td><td></td></tr>
<tr><td>modern_data</td><td>1</td><td></td></tr>
<tr><td>isc_l5</td><td>2</td><td></td></tr>
<tr><td>flags</td><td>0-2</td><td></td></tr>
</table></td></tr>
</table>

BeiDou Ephemeris (0x91,0x64)

Description	BeiDou Ephemeris Data	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*220*
Descriptor	*u8*	*0x64*
Index	u8	Index of this field in this epoch.
Count	u8	Total number of fields in this epoch.
Time Of Week	double	GPS Time of week [seconds]
Week Number	u16	GPS Week since 1980 [weeks]
Satellite Id	u8	GNSS satellite id within the constellation.
Health	u8	Satellite and signal health
Iodc	u8	Issue of Data Clock. This increments each time the data changes and rolls over at 4. It is used to make s various raw data elements from different sources line up correctly.
Iode	u8	Issue of Data Ephemeris.
T Oc	double	Reference time for clock data.
Af0	double	Clock bias in [s].
Af1	double	Clock drift in [s/s].
Af2	double	Clock drift rate in [s/s^2].
T Gd	double	T Group Delay [s].
Isc L1ca	double	Inter-signal correction (L1).
Isc L2c	double	Inter-signal correction (L2, or L5 if isc_l5 flag is set).
T Oe	double	Reference time for ephemeris in [s].
A	double	Semi-major axis [m].
A Dot	double	Semi-major axis rate [m/s].
Mean Anomaly	double	[rad].
Delta Mean Motion	double	[rad].
Delta Mean Motion Dot	double	[rad/s].
Eccentricity	double	
Argument Of Perigee	double	[rad].
Omega	double	Longitude of Ascending Node [rad].

<table>
<tr><td>Omega Dot</td><td>double</td><td>Rate of Right Ascension [rad/s].</td></tr>
<tr><td>Inclination</td><td>double</td><td>Inclination angle [rad].</td></tr>
<tr><td>Inclination Dot</td><td>double</td><td>Inclination angle rate of change [rad/s].</td></tr>
<tr><td>C Ic</td><td>double</td><td>Harmonic Correction Term.</td></tr>
<tr><td>C Is</td><td>double</td><td>Harmonic Correction Term.</td></tr>
<tr><td>C Uc</td><td>double</td><td>Harmonic Correction Term.</td></tr>
<tr><td>C Us</td><td>double</td><td>Harmonic Correction Term.</td></tr>
<tr><td>C Rc</td><td>double</td><td>Harmonic Correction Term.</td></tr>
<tr><td>C Rs</td><td>double</td><td>Harmonic Correction Term.</td></tr>
<tr><td>Valid Flags</td><td>u16 bitfield</td><td><table>
<tr><th>Name</th><th>Bit(s)</th><th>Description</th></tr>
<tr><td>ephemeris</td><td>0</td><td></td></tr>
<tr><td>modern_data</td><td>1</td><td></td></tr>
<tr><td>isc_l5</td><td>2</td><td></td></tr>
<tr><td>flags</td><td>0-2</td><td></td></tr>
</table></td></tr>
</table>

GNSS Recv 2 (0x92)

GNSS LLH Position (0x92,0x03)

Description	GNSS reported position in the WGS84 geodetic frame	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*44*
Descriptor	*u8*	*0x03*
Latitude	double	[degrees]
Longitude	double	[degrees]
Ellipsoid Height	double	[meters]
Msl Height	double	[meters]
Horizontal Accuracy	float	[meters]
Vertical Accuracy	float	[meters]
Valid Flags	u16 bitfield	see table below

Valid Flags (u16 bitfield):

Name	Bit(s)	Description
lat_lon	0	
ellipsoid_height	1	
msl_height	2	
horizontal_accuracy	3	
vertical_accuracy	4	
flags	0-4	

GNSS ECEF Position (0x92,0x04)

Description	**GNSS reported position in the Earth-centered, Earth-Fixed (ECEF) frame**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*8*
Descriptor	*u8*	*0x04*
X	Vector3d	[meters]
X Accuracy	float	[meters]
Valid Flags	u16 bitfield	see table below

Valid Flags:

Name	Bit(s)	Description
position	0	
position_accuracy	1	
flags	0-1	

NED Velocity (0x92,0x05)

Description	GNSS reported velocity in the NED frame
Notes	

Parameter Name	Data Type	Description
Field Length	*u8*	*24*
Descriptor	*u8*	*0x05*
V	Vector3f	[meters/second]
Speed	float	[meters/second]
Ground Speed	float	[meters/second]
Heading	float	[degrees]
Speed Accuracy	float	[meters/second]
Heading Accuracy	float	[degrees]
Valid Flags	u16 bitfield	(see table below)

Name	Bit(s)	Description
velocity	0	
speed_3d	1	
ground_speed	2	
heading	3	
speed_accuracy	4	
heading_accuracy	5	
flags	0-5	

GNSS ECEF Velocity (0x92,0x06)

Description	GNSS reported velocity in the Earth-centered, Earth-Fixed (ECEF) frame	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*8*
Descriptor	*u8*	*0x06*
V	Vector3f	[meters/second]
V Accuracy	float	[meters/second]
Valid Flags	u16 bitfield	see table below

Name	Bit(s)	Description
velocity	0	
velocity_accuracy	1	
flags	0-1	

DOP (0x92,0x07)

Description	**GNSS reported dilution of precision information.**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*32*
Descriptor	*u8*	*0x07*
Gdop	float	Geometric DOP
Pdop	float	Position DOP
Hdop	float	Horizontal DOP
Vdop	float	Vertical DOP
Tdop	float	Time DOP
Ndop	float	Northing DOP
Edop	float	Easting DOP
Valid Flags	u16 bitfield	(see table below)

Name	Bit(s)	Description
gdop	0	
pdop	1	
hdop	2	
vdop	3	
tdop	4	
ndop	5	
edop	6	
flags	0-6	

UTC Time (0x92,0x08)

Description	GNSS reported Coordinated Universal Time	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*15*
Descriptor	*u8*	*0x08*
Year	u16	
Month	u8	Month (1-12)
Day	u8	Day (1-31)
Hour	u8	Hour (0-23)
Min	u8	Minute (0-59)
Sec	u8	Second (0-59)
Msec	u32	Millisecond(0-999)
Valid Flags	u16 bitfield	see table below

Name	Bit(s)	Description
gnss_date_time	0	
leap_seconds_known	1	
flags	0-1	

GPS Time (0x92,0x09)

Description	**GNSS reported GPS Time**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*14*
Descriptor	*u8*	*0x09*
Tow	double	GPS Time of week [seconds]
Week Number	u16	GPS Week since 1980 [weeks]
Valid Flags	u16 bitfield	see nested table below

Name	Bit(s)	Description
tow	0	
week_number	1	
flags	0-1	

Fix Info (0x92,0x0B)

Description	**GNSS reported position fix type**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*8*
Descriptor	*u8*	*0x0B*
Fix Type	u8 enum	<table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td>FIX_3D</td><td>0</td><td></td></tr><tr><td>FIX_2D</td><td>1</td><td></td></tr><tr><td>FIX_TIME_ONLY</td><td>2</td><td></td></tr><tr><td>FIX_NONE</td><td>3</td><td></td></tr><tr><td>FIX_INVALID</td><td>4</td><td></td></tr><tr><td>FIX_RTK_FLOAT</td><td>5</td><td></td></tr><tr><td>FIX_RTK_FIXED</td><td>6</td><td></td></tr><tr><td>FIX_DIFFERENTIAL</td><td>7</td><td></td></tr></table>
Num Sv	[u8]	
Fix Flags	u16 bitfield	<table><tr><th>Name</th><th>Bit(s)</th><th>Description</th></tr><tr><td>sbas_used</td><td>0</td><td></td></tr><tr><td>dgnss_used</td><td>1</td><td></td></tr></table>
Valid Flags	u16 bitfield	<table><tr><th>Name</th><th>Bit(s)</th><th>Description</th></tr><tr><td>fix_type</td><td>0</td><td></td></tr><tr><td>num_sv</td><td>1</td><td></td></tr><tr><td>fix_flags</td><td>2</td><td></td></tr><tr><td>flags</td><td>0-2</td><td></td></tr></table>

GNSS Hardware Status (0x92,0x0D)

<table>
<tr><th>Description</th><th colspan="2">GNSS reported hardware status</th></tr>
<tr><td>Notes</td><td colspan="2"></td></tr>
<tr><td>Parameter Name</td><td>Data Type</td><td>Description</td></tr>
<tr><td>Field Length</td><td>u8</td><td>7</td></tr>
<tr><td>Descriptor</td><td>u8</td><td>0x0D</td></tr>
<tr><td>Receiver State</td><td>u8 enum</td><td><table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td>OFF</td><td>0</td><td></td></tr><tr><td>ON</td><td>1</td><td></td></tr><tr><td>UNKNOWN</td><td>2</td><td></td></tr></table></td></tr>
<tr><td>Antenna State</td><td>u8 enum</td><td><table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td>INIT</td><td>1</td><td></td></tr><tr><td>SHORT</td><td>2</td><td></td></tr><tr><td>OPEN</td><td>3</td><td></td></tr><tr><td>GOOD</td><td>4</td><td></td></tr><tr><td>UNKNOWN</td><td>5</td><td></td></tr></table></td></tr>
<tr><td>Antenna Power</td><td>u8 enum</td><td><table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td>OFF</td><td>0</td><td></td></tr><tr><td>ON</td><td>1</td><td></td></tr><tr><td>UNKNOWN</td><td>2</td><td></td></tr></table></td></tr>
<tr><td>Valid Flags</td><td>u16 bitfield</td><td><table><tr><th>Name</th><th>Bit(s)</th><th>Description</th></tr><tr><td>sensor_state</td><td>0</td><td></td></tr><tr><td>antenna_state</td><td>1</td><td></td></tr><tr><td>antenna_power</td><td>2</td><td></td></tr><tr><td>flags</td><td>0-2</td><td></td></tr></table></td></tr>
</table>

Clock Info 2 (0x92,0x10)

Description	**GNSS reported receiver clock parameters**	
Notes	This supersedes MIP_DATA_DESC_GNSS_CLOCK_INFO with additional information.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*36*
Descriptor	*u8*	*0x10*
Bias	double	
Drift	double	
Bias Accuracy Estimate	double	
Drift Accuracy Estimate	double	
Valid Flags	u16 bitfield	(see table below)

Name	Bit(s)	Description
bias	0	
drift	1	
bias_accuracy	2	
drift_accuracy	3	
flags	0-3	

GPS Leap Seconds (0x92,0x11)

Description	**GNSS reported leap seconds (difference between GPS and UTC Time)**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*5*
Descriptor	*u8*	*0x11*
Leap Seconds	u8	[s]
Valid Flags	u16 bitfield	(see nested table below)

Valid Flags bitfield:

Name	Bit(s)	Description
leap_seconds	1	

SBAS Info (0x92,0x12)

<table>
<tr><th>Description</th><th colspan="2">GNSS SBAS status</th></tr>
<tr><td>Notes</td><td colspan="2"></td></tr>
<tr><th>Parameter Name</th><th>Data Type</th><th>Description</th></tr>
<tr><td>Field Length</td><td>u8</td><td>18</td></tr>
<tr><td>Descriptor</td><td>u8</td><td>0x12</td></tr>
<tr><td>Time Of Week</td><td>double</td><td>GPS Time of week [seconds]</td></tr>
<tr><td>Week Number</td><td>u16</td><td>GPS Week since 1980 [weeks]</td></tr>
<tr><td>Sbas System</td><td>u8 enum</td><td>SBAS system id
<table>
<tr><th>Name</th><th>Value</th><th>Description</th></tr>
<tr><td>UNKNOWN</td><td>0</td><td></td></tr>
<tr><td>WAAS</td><td>1</td><td></td></tr>
<tr><td>EGNOS</td><td>2</td><td></td></tr>
<tr><td>MSAS</td><td>3</td><td></td></tr>
<tr><td>GAGAN</td><td>4</td><td></td></tr>
</table></td></tr>
<tr><td>Sbas Id</td><td>u8</td><td>SBAS satellite id.</td></tr>
<tr><td>Count</td><td>u8</td><td>Number of SBAS corrections</td></tr>
<tr><td>Sbas Status</td><td>u8 bitfield</td><td>Status of the SBAS service
<table>
<tr><th>Name</th><th>Bit(s)</th><th>Description</th></tr>
<tr><td>range_available</td><td>0</td><td></td></tr>
<tr><td>corrections_available</td><td>1</td><td></td></tr>
<tr><td>integrity_available</td><td>2</td><td></td></tr>
<tr><td>test_mode</td><td>3</td><td></td></tr>
</table></td></tr>
<tr><td>Valid Flags</td><td>u16 bitfield</td><td>
<table>
<tr><th>Name</th><th>Bit(s)</th><th>Description</th></tr>
<tr><td>tow</td><td>0</td><td></td></tr>
<tr><td>week_number</td><td>1</td><td></td></tr>
<tr><td>sbas_system</td><td>2</td><td></td></tr>
<tr><td>sbas_id</td><td>3</td><td></td></tr>
<tr><td>count</td><td>4</td><td></td></tr>
<tr><td>sbas_status</td><td>5</td><td></td></tr>
<tr><td>flags</td><td>0-5</td><td></td></tr>
</table></td></tr>
</table>

Satellite Status (0x92,0x20)

Description	**Status information for a GNSS satellite.**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*27*
Descriptor	*u8*	*0x20*
Index	u8	Index of this field in this epoch.
Count	u8	Total number of fields in this epoch.
Time Of Week	double	GPS Time of week [seconds]
Week Number	u16	GPS Week since 1980 [weeks]
Gnss Id	u8 enum	(see enum table below)
Satellite Id	u8	GNSS satellite id within the constellation
Elevation	float	Elevation of the satellite relative to the rover [degrees]
Azimuth	float	Azimuth of the satellite relative to the rover [degrees]
Health	bool	True if the satellite is healthy.
Valid Flags	u16 bitfield	(see bitfield table below)

Gnss Id:

Name	Value	Description
UNKNOWN	0	
GPS	1	
GLONASS	2	
GALILEO	3	
BEIDOU	4	
SBAS	5	

Valid Flags:

Name	Bit(s)	Description
tow	0	
week_number	1	
gnss_id	2	
satellite_id	3	
elevation	4	
azimuth	5	
health	6	
flags	0-6	

Raw (0x92,0x22)

Description	GNSS Raw observation.	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*63*
Descriptor	*u8*	*0x22*
Index	u8	Index of this field in this epoch.
Count	u8	Total number of fields in this epoch.
Time Of Week	double	GPS Time of week [seconds]
Week Number	u16	GPS Week since 1980 [weeks]
Receiver Id	u16	When the measurement comes from RTCM, this will be the reference station ID; otherwise, it's the rece number (1,2,...)
Tracking Channel	u8	Channel the receiver is using to track this satellite.
Gnss Id	u8 enum	(see table below)
Satellite Id	u8	GNSS satellite id within the constellation.
		Signal identifier for the satellite. (see table below)

Gnss Id:

Name	Value	Description
UNKNOWN	0	
GPS	1	
GLONASS	2	
GALILEO	3	
BEIDOU	4	
SBAS	5	

Signal identifier for the satellite.

Name	Value	Description
UNKNOWN	0	
GPS_L1CA	1	
GPS_L1P	2	
GPS_L1Z	3	
GPS_L2CA	4	
GPS_L2P	5	
GPS_L2Z	6	
GPS_L2CL	7	
GPS_L2CM	8	
GPS_L2CML	9	
GPS_L5I	10	
GPS_L5Q	11	
GPS_L5IQ	12	

Signal Id	u8 enum	GPS_L1CD	13	
		GPS_L1CP	14	
		GPS_L1CDP	15	
		GLONASS_G1CA	32	
		GLONASS_G1P	33	
		GLONASS_G2C	34	
		GLONASS_G2P	35	
		GALILEO_E1C	64	
		GALILEO_E1A	65	
		GALILEO_E1B	66	
		GALILEO_E1BC	67	
		GALILEO_E1ABC	68	
		GALILEO_E6C	69	
		GALILEO_E6A	70	
		GALILEO_E6B	71	
		GALILEO_E6BC	72	
		GALILEO_E6ABC	73	
		GALILEO_E5BI	74	
		GALILEO_E5BQ	75	
		GALILEO_E5BIQ	76	
		GALILEO_E5ABI	77	
		GALILEO_E5ABQ	78	
		GALILEO_E5ABIQ	79	
		GALILEO_E5AI	80	
		GALILEO_E5AQ	81	
		GALILEO_E5AIQ	82	
		SBAS_L1CA	96	
		SBAS_L5I	97	
		SBAS_L5Q	98	
		SBAS_L5IQ	99	
		QZSS_L1CA	128	
		QZSS_LEXS	129	
		QZSS_LEXL	130	
		QZSS_LEXSL	131	
		QZSS_L2CM	132	
		QZSS_L2CL	133	
		QZSS_L2CML	134	
		QZSS_L5I	135	
		QZSS_L5Q	136	
		QZSS_L5IQ	137	
		QZSS_L1CD	138	
		QZSS_L1CP	139	
		QZSS_L1CDP	140	
		BEIDOU_B1I	160	
		BEIDOU_B1Q	161	
		BEIDOU_B1IQ	162	
		BEIDOU_B3I	163	
		BEIDOU_B3Q	164	
		BEIDOU_B3IQ	165	
		BEIDOU_B2I	166	
		BEIDOU_B2Q	167	
		BEIDOU_B2IQ	168	
		BEIDOU_B2A	169	
Signal Strength	float	Carrier to noise ratio [dBHz].		

Quality	u8 enum	Indicator of signal quality. <table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td>NONE</td><td>0</td><td></td></tr><tr><td>SEARCHING</td><td>1</td><td></td></tr><tr><td>ACQUIRED</td><td>2</td><td></td></tr><tr><td>UNUSABLE</td><td>3</td><td></td></tr><tr><td>TIME_LOCKED</td><td>4</td><td></td></tr><tr><td>FULLY_LOCKED</td><td>5</td><td></td></tr></table>
Pseudorange	double	Pseudo-range measurement [meters].
Carrier Phase	double	Carrier phase measurement [Carrier periods].
Doppler	float	Measured doppler shift [Hz].
Range Uncert	float	Uncertainty of the pseudo-range measurement [m].
Phase Uncert	float	Uncertainty of the phase measurement [Carrier periods].
Doppler Uncert	float	Uncertainty of the measured doppler shift [Hz].
Lock Time	float	DOC Minimum carrier phase lock time [s]. Note: the maximum value is dependent on the receiver.
Valid Flags	u16 bitfield	<table><tr><th>Name</th><th>Bit(s)</th><th>Descriptior</th></tr><tr><td>tow</td><td>0</td><td></td></tr><tr><td>week_number</td><td>1</td><td></td></tr><tr><td>receiver_id</td><td>2</td><td></td></tr><tr><td>tracking_channel</td><td>3</td><td></td></tr><tr><td>gnss_id</td><td>4</td><td></td></tr><tr><td>satellite_id</td><td>5</td><td></td></tr><tr><td>signal_id</td><td>6</td><td></td></tr><tr><td>signal_strength</td><td>7</td><td></td></tr><tr><td>quality</td><td>8</td><td></td></tr><tr><td>pseudorange</td><td>9</td><td></td></tr><tr><td>carrier_phase</td><td>10</td><td></td></tr><tr><td>doppler</td><td>11</td><td></td></tr><tr><td>range_uncertainty</td><td>12</td><td></td></tr><tr><td>carrier_phase_uncertainty</td><td>13</td><td></td></tr><tr><td>doppler_uncertainty</td><td>14</td><td></td></tr><tr><td>lock_time</td><td>15</td><td></td></tr><tr><td>flags</td><td>0-15</td><td></td></tr></table>

GPS Ephemeris (0x92,0x61)

Description	**GPS Ephemeris Data**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*220*
Descriptor	*u8*	*0x61*
Index	u8	Index of this field in this epoch.
Count	u8	Total number of fields in this epoch.
Time Of Week	double	GPS Time of week [seconds]
Week Number	u16	GPS Week since 1980 [weeks]
Satellite Id	u8	GNSS satellite id within the constellation.
Health	u8	Satellite and signal health
Iodc	u8	Issue of Data Clock. This increments each time the data changes and rolls over at 4. It is used to make s various raw data elements from different sources line up correctly.
Iode	u8	Issue of Data Ephemeris.
T Oc	double	Reference time for clock data.
Af0	double	Clock bias in [s].
Af1	double	Clock drift in [s/s].
Af2	double	Clock drift rate in [s/s^2].
T Gd	double	T Group Delay [s].
Isc L1ca	double	Inter-signal correction (L1).
Isc L2c	double	Inter-signal correction (L2, or L5 if isc_l5 flag is set).
T Oe	double	Reference time for ephemeris in [s].
A	double	Semi-major axis [m].
A Dot	double	Semi-major axis rate [m/s].
Mean Anomaly	double	[rad].
Delta Mean Motion	double	[rad].
Delta Mean Motion Dot	double	[rad/s].
Eccentricity	double	
Argument Of Perigee	double	[rad].
Omega	double	Longitude of Ascending Node [rad].

Omega Dot	double	Rate of Right Ascension [rad/s].
Inclination	double	Inclination angle [rad].
Inclination Dot	double	Inclination angle rate of change [rad/s].
C Ic	double	Harmonic Correction Term.
C Is	double	Harmonic Correction Term.
C Uc	double	Harmonic Correction Term.
C Us	double	Harmonic Correction Term.
C Rc	double	Harmonic Correction Term.
C Rs	double	Harmonic Correction Term.
Valid Flags	u16 bitfield	(see nested table below)

Name	Bit(s)	Description
ephemeris	0	
modern_data	1	
isc_l5	2	
flags	0-2	

You are here:

Glonass Ephemeris (0x92,0x62)

Description	**Glonass Ephemeris Data**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*56*
Descriptor	*u8*	*0x62*
Index	u8	Index of this field in this epoch.
Count	u8	Total number of fields in this epoch.
Time Of Week	double	GPS Time of week [seconds]
Week Number	u16	GPS Week since 1980 [weeks]
Satellite Id	u8	GNSS satellite id within the constellation.
Freq Number	s8	GLONASS frequency number (-7 to 24)
Tk	u32	Frame start time within current day [seconds]
Tb	u32	Ephemeris reference time [seconds]
Sat Type	u8	Type of satellite (M) GLONASS = 0, GLONASS-M = 1
Gamma	double	Relative deviation of carrier frequency from nominal [dimensionless]
Tau N	double	Time correction relative to GLONASS Time [seconds]
X	Vector3d	Satellite PE-90 position [m]
V	Vector3f	Satellite PE-90 velocity [m/s]
A	Vector3f	Satellite PE-90 acceleration due to perturbations [m/s^2]
Health	u8	Satellite Health (Bn), Non-zero indicates satellite malfunction
P	u8	Satellite operation mode (See GLONASS ICD)
Nt	u8	Day number within a 4 year period.
Delta Tau N	float	Time difference between L1 and L2[m/s]
Ft	u8	User Range Accuracy (See GLONASS ICD)
En	u8	Age of current information [days]
P1	u8	Time interval between adjacent values of tb [minutes]
P2	u8	Oddness "1" or evenness "0" of the value of tb.
P3	u8	Number of satellites in almanac for this frame
P4	u8	Flag indicating ephemeris parameters are present
Valid Flags	u16 bitfield	(see bitfield table below)

Name	**Bit(s)**	**Description**
ephemeris	0	
flags	0	

Galileo Ephemeris (0x92,0x63)

Description	Galileo Ephemeris Data	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*220*
Descriptor	*u8*	*0x63*
Index	u8	Index of this field in this epoch.
Count	u8	Total number of fields in this epoch.
Time Of Week	double	GPS Time of week [seconds]
Week Number	u16	GPS Week since 1980 [weeks]
Satellite Id	u8	GNSS satellite id within the constellation.
Health	u8	Satellite and signal health
Iodc	u8	Issue of Data Clock. This increments each time the data changes and rolls over at 4. It is used to make s various raw data elements from different sources line up correctly.
Iode	u8	Issue of Data Ephemeris.
T Oc	double	Reference time for clock data.
Af0	double	Clock bias in [s].
Af1	double	Clock drift in [s/s].
Af2	double	Clock drift rate in [s/s^2].
T Gd	double	T Group Delay [s].
Isc L1ca	double	Inter-signal correction (L1).
Isc L2c	double	Inter-signal correction (L2, or L5 if isc_l5 flag is set).
T Oe	double	Reference time for ephemeris in [s].
A	double	Semi-major axis [m].
A Dot	double	Semi-major axis rate [m/s].
Mean Anomaly	double	[rad].
Delta Mean Motion	double	[rad].
Delta Mean Motion Dot	double	[rad/s].
Eccentricity	double	
Argument Of Perigee	double	[rad].
Omega	double	Longitude of Ascending Node [rad].

<table>
<tr><td>Omega Dot</td><td>double</td><td>Rate of Right Ascension [rad/s].</td></tr>
<tr><td>Inclination</td><td>double</td><td>Inclination angle [rad].</td></tr>
<tr><td>Inclination Dot</td><td>double</td><td>Inclination angle rate of change [rad/s].</td></tr>
<tr><td>C Ic</td><td>double</td><td>Harmonic Correction Term.</td></tr>
<tr><td>C Is</td><td>double</td><td>Harmonic Correction Term.</td></tr>
<tr><td>C Uc</td><td>double</td><td>Harmonic Correction Term.</td></tr>
<tr><td>C Us</td><td>double</td><td>Harmonic Correction Term.</td></tr>
<tr><td>C Rc</td><td>double</td><td>Harmonic Correction Term.</td></tr>
<tr><td>C Rs</td><td>double</td><td>Harmonic Correction Term.</td></tr>
<tr><td>Valid Flags</td><td>u16 bitfield</td><td><table>
<tr><th>Name</th><th>Bit(s)</th><th>Description</th></tr>
<tr><td>ephemeris</td><td>0</td><td></td></tr>
<tr><td>modern_data</td><td>1</td><td></td></tr>
<tr><td>isc_l5</td><td>2</td><td></td></tr>
<tr><td>flags</td><td>0-2</td><td></td></tr>
</table></td></tr>
</table>

BeiDou Ephemeris (0x92,0x64)

Description	BeiDou Ephemeris Data	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*220*
Descriptor	*u8*	*0x64*
Index	u8	Index of this field in this epoch.
Count	u8	Total number of fields in this epoch.
Time Of Week	double	GPS Time of week [seconds]
Week Number	u16	GPS Week since 1980 [weeks]
Satellite Id	u8	GNSS satellite id within the constellation.
Health	u8	Satellite and signal health
Iodc	u8	Issue of Data Clock. This increments each time the data changes and rolls over at 4. It is used to make s various raw data elements from different sources line up correctly.
Iode	u8	Issue of Data Ephemeris.
T Oc	double	Reference time for clock data.
Af0	double	Clock bias in [s].
Af1	double	Clock drift in [s/s].
Af2	double	Clock drift rate in [s/s^2].
T Gd	double	T Group Delay [s].
Isc L1ca	double	Inter-signal correction (L1).
Isc L2c	double	Inter-signal correction (L2, or L5 if isc_l5 flag is set).
T Oe	double	Reference time for ephemeris in [s].
A	double	Semi-major axis [m].
A Dot	double	Semi-major axis rate [m/s].
Mean Anomaly	double	[rad].
Delta Mean Motion	double	[rad].
Delta Mean Motion Dot	double	[rad/s].
Eccentricity	double	
Argument Of Perigee	double	[rad].
Omega	double	Longitude of Ascending Node [rad].

Omega Dot	double	Rate of Right Ascension [rad/s].
Inclination	double	Inclination angle [rad].
Inclination Dot	double	Inclination angle rate of change [rad/s].
C Ic	double	Harmonic Correction Term.
C Is	double	Harmonic Correction Term.
C Uc	double	Harmonic Correction Term.
C Us	double	Harmonic Correction Term.
C Rc	double	Harmonic Correction Term.
C Rs	double	Harmonic Correction Term.
Valid Flags	u16 bitfield	<table><tr><th>Name</th><th>Bit(s)</th><th>Description</th></tr><tr><td>ephemeris</td><td>0</td><td></td></tr><tr><td>modern_data</td><td>1</td><td></td></tr><tr><td>isc_l5</td><td>2</td><td></td></tr><tr><td>flags</td><td>0-2</td><td></td></tr></table>

Sensor (0x80)

Scaled Accel (0x80,0x04)

Description	**3-element vector representing the sensed acceleration.**	
Notes	This quantity is temperature compensated and expressed in the vehicle frame.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*2*
Descriptor	*u8*	*0x04*
Scaled Accel	Vector3f	(x, y, z)[g]

Scaled Gyro (0x80,0x05)

Description	**3-element vector representing the sensed angular rate.**	
Notes	This quantity is temperature compensated and expressed in the vehicle frame.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*2*
Descriptor	*u8*	*0x05*
Scaled Gyro	Vector3f	(x, y, z) [radians/second]

Scaled Mag (0x80,0x06)

Description	**3-element vector representing the sensed magnetic field.**	
Notes	This quantity is temperature compensated and expressed in the vehicle frame.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*2*
Descriptor	*u8*	*0x06*
Scaled Mag	Vector3f	(x, y, z) [Gauss]

Delta Theta (0x80,0x07)

Description	**3-element vector representing the time integral of angular rate.**	
Notes	This quantity is the integral of sensed angular rate over the period set by the IMU message format. It is expressed in the vehicle frame.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*2*
Descriptor	*u8*	*0x07*
Delta Theta	Vector3f	(x, y, z) [radians]

Delta Velocity (0x80,0x08)

Description	**3-element vector representing the time integral of acceleration.**	
Notes	This quantity is the integral of sensed acceleration over the period set by the IMU message format. It is expressed in the vehicle frame.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*2*
Descriptor	*u8*	*0x08*
Delta Velocity	Vector3f	(x, y, z) [g*sec]

Complementary Filter Orientation Matrix (0x80,0x09)

Description	**3x3 Direction Cosine Matrix describing the orientation of the device with respect to the NED local-level frame.**	
Notes	This matrix satisfies the following relationship: Where: is a 3-element vector expressed in the NED frame. is the same 3-element vector expressed in the vehicle frame. The matrix elements are stored is row-major order:	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*2*
Descriptor	*u8*	*0x09*
M	Matrix3f	Matrix elements in row-major order.

Complementary Filter Quaternion (0x80,0x0A)

Description	**4x1 vector representation of the quaternion describing the orientation of the device with respect to the NED local-level frame.**	
Notes	This quaternion satisfies the following relationship: Where: is the quaternion describing the rotation. and is a 3-element vector expressed in the NED frame. and is a 3-element vector expressed in the vehicle frame.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*2*
Descriptor	*u8*	*0x0A*
Q	Quatf	Quaternion elements

Complementary Filter Euler Angles (0x80,0x0C)

Description	**Euler angles describing the orientation of the device with respect to the NED local-level frame.**	
Notes	The Euler angles are reported in 3-2-1 (Yaw-Pitch-Roll, AKA Aircraft) order.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*14*
Descriptor	*u8*	*0x0C*
Roll	float	[radians]
Pitch	float	[radians]
Yaw	float	[radians]

GPS Timestamp (0x80,0x12)

<table>
<tr><th>Description</th><th colspan="2">GPS timestamp of the SENSOR data</th></tr>
<tr><td>Notes</td><td colspan="2">Should the PPS become unavailable, the device will revert to its internal clock, which will cause the reported time from true GPS time. Upon recovering from a PPS outage, the user should expect a jump in the reported GPS tim the accumulation of internal clock error. If synchronization to an external clock or onboard GNSS receiver (for pro that have one) is disabled, this time is equivalent to internal system time.

Note: this data field may be deprecated in the future. The more flexible shared data field (0x80, 0xD3) should be instead.</td></tr>
<tr><th>Parameter Name</th><th>Data Type</th><th>Description</th></tr>
<tr><td>Field Length</td><td>u8</td><td>14</td></tr>
<tr><td>Descriptor</td><td>u8</td><td>0x12</td></tr>
<tr><td>Tow</td><td>double</td><td>GPS Time of Week [seconds]</td></tr>
<tr><td>Week Number</td><td>u16</td><td>GPS Week Number since 1980 [weeks]</td></tr>
<tr><td>Valid Flags</td><td>u16 bitfield</td><td>
<table>
<tr><th>Name</th><th>Bit(s)</th><th>Description</th></tr>
<tr><td>pps_valid</td><td>0</td><td>True when the PPS signal is present.</td></tr>
<tr><td>time_refresh</td><td>1</td><td>Toggles each time the time is updated via internal GPS or the GPS Time U command (0x01, 0x72).</td></tr>
<tr><td>time_initialized</td><td>2</td><td>True if the time has ever been set.</td></tr>
<tr><td>tow_valid</td><td>3</td><td>True if the time of week is valid.</td></tr>
<tr><td>week_number_valid</td><td>4</td><td>True if the week number is valid.</td></tr>
</table>
</td></tr>
</table>

Temperature Statistics (0x80,0x14)

Description	**SENSOR reported temperature statistics**	
Notes	Temperature may originate from the MEMS sensors, or be calculated in combination with board temperature sensors. All quantities are calculated with respect to the last power on or reset, whichever is later.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*14*
Descriptor	*u8*	*0x14*
Min Temp	float	[degC]
Max Temp	float	[degC]
Mean Temp	float	[degC]

Scaled Pressure (0x80,0x17)

Description	**Scalar value representing the sensed ambient pressure.**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*6*
Descriptor	*u8*	*0x17*
Scaled Pressure	float	[mBar]

Overrange Status (0x80,0x18)

Description		
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*4*
Descriptor	*u8*	*0x18*
Status	u16 bitfield	(see bitfield table below)

Name	Bit(s)	Description
accel_x	0	
accel_y	1	
accel_z	2	
gyro_x	4	
gyro_y	5	
gyro_z	6	
mag_x	8	
mag_y	9	
mag_z	10	
press	12	

Odometer Data (0x80,0x40)

Description		
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*12*
Descriptor	*u8*	*0x40*
Speed	float	Average speed over the time interval [m/s]. Can be negative for quadrature encoders.
Uncertainty	float	Uncertainty of velocity [m/s].
Valid Flags	u16	If odometer is configured, bit 0 will be set to 1.

Shared (0xFF)

Event Source (0xFF,0xD0)

GPS Timestamp (0xFF,0xD3)

Delta Time (0xFF,0xD4)

Reference Timestamp (0xFF,0xD5)

Reference Time Delta (0xFF,0xD6)

External Timestamp (0xFF,0xD7)

External Time Delta (0xFF,0xD8)

Event Source (0xFF,0xD0)

Description	Identifies which event trigger caused this packet to be emitted.	
Notes	Generally this is used to determine whether a packet was emitted due to scheduled streaming or due to an event.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*3*
Descriptor	*u8*	*0xD0*
Trigger Id	u8	Trigger ID number. If 0, this message was emitted due to being scheduled in the 3DM Message Format Command (0x0C,0x0F).

GPS Timestamp (0xFF,0xD3)

Description	Outputs the current GPS system time in time-of-week and week number format.
Notes	For events, this is the time of the event trigger. In order to be valid, a PPS signal needs to be present, and both a GPS time-of-week and week number command (0x0C, 0x72) need to be received after PPS sync has been ach

Parameter Name	Data Type	Description
Field Length	*u8*	*14*
Descriptor	*u8*	*0xD3*
Tow	double	GPS Time of Week [seconds]
Week Number	u16	GPS Week Number since 1980 [weeks]
Valid Flags	u16 bitfield	see table below

Name	Bit(s)	Description
tow	0	Whole number seconds TOW has been set
week_number	1	Week number has been set
time_valid	0-1	Both TOW and Week Number have been set

Delta Time (0xFF,0xD4)

Description	**Time in the synchronized clock domain since the last output of this field within the same descriptor set and event instance.**	
Notes	This can be used to track the amount of time passed between event occurrences. See the manual page on delta time quantities. This field contains the same value as the delta external time field, 0xD8, but is expressed in seconds. Transmission of either of these fields restarts a shared counter, so only one should be streamed at a time to avoid confusion. The counter is not shared across descriptors sets or between event instances.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*10*
Descriptor	*u8*	*0xD4*
Seconds	double	Seconds since last output.

Reference Timestamp (0xFF,0xD5)

Description	Internal reference timestamp.	
Notes	This timestamp represents the time at which the corresponding data was sampled, according to the internal reference clock. This is a monotonic clock which never jumps. The value is always valid. For events, this is the time of the event trigger.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*10*
Descriptor	*u8*	*0xD5*
Nanoseconds	u64	Nanoseconds since initialization.

Reference Time Delta (0xFF,0xD6)

Description	Delta time since the last packet.	
Notes	Difference between the time as reported by the shared reference time field, 0xD5, and the previous output of this delta quantity within the same descriptor set and event instance. The delta is based on the reference time which never jumps. The value is always valid. This can be used to track the amount of time passed between event occurrences. See the manual page on delta time quantities.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*10*
Descriptor	*u8*	*0xD6*
Dt Nanos	u64	Nanoseconds since the last occurrence of this field in a packet of the same descriptor set and event source.

External Timestamp (0xFF,0xD7)

Description	**External timestamp in nanoseconds.**	
Notes	This timestamp represents the time at which the corresponding data was sampled in the external clock domain. Equivalent to the GPS Timestamp but in nanoseconds. For events, this is the time of the event trigger. To be valid, external clock sync must be achieved using the PPS input.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*12*
Descriptor	*u8*	*0xD7*
Nanoseconds	u64	
Valid Flags	u16 bitfield	<table><tr><th>Name</th><th>Bit(s)</th><th>Description</th></tr><tr><td>nanoseconds</td><td>0</td><td></td></tr></table>

External Time Delta (0xFF,0xD8)

Description	Delta time since the last packet containing delta external (0xFF,0xD4) or delta gps time (0xFF,0xD8).	
Notes	Difference between the time as reported by the shared external time field, 0xD7, and the previous output of this c quantity within the same descriptor set and event instance. This can be used to track the amount of time passed between event occurrences. See the manual page on delta quantities. This field contains the same value as the delta gps time field, 0xD4, but is expressed in nanoseconds. Transmiss either of these fields restarts a shared counter, so only one should be streamed at a time to avoid confusion. The is not shared across descriptors sets or between event instances.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*12*
Descriptor	*u8*	*0xD8*
Dt Nanos	u64	Nanoseconds since the last occurrence of this field in a packet of the same descriptor set and event sou
Valid Flags	u16 bitfield	<table><tr><th>Name</th><th>Bit(s)</th><th>Description</th></tr><tr><td>dt_nanos</td><td>0</td><td></td></tr></table>

System (0xA0)

Built In Test (0xA0,0x01)

Description	**Contains the continuous built-in-test (BIT) results.**	
Notes	Due to the large size of this field, it is recommended to stream it at a low rate or poll it on demand. These bits are "sticky" until the next output message. If a fault occurs in between scheduled messages or while the device is idle, the next packet with this field will have the corresponding flags set. The flag is then cleared unless the fault persists. Unlike the commanded BIT, some bits may be 1 in certain non-fault situations, so simply checking if the result is all 0s is not very useful. For example, on devices with a built-in GNSS receiver, a "solution fault" bit may be set before the receiver has obtained a position fix. Consult the device manual to determine which bits are of interest for your application. All unspecified bits are reserved for future use and must be ignored.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*18*
Descriptor	*u8*	*0x01*
Result	u8[16]	Device-specific bitfield (128 bits). See device user manual. Bits are least-significant-byte first. For example, bit 0 is located at bit 0 of result[0], bit 1 is located at bit 1 of result[0], bit 8 is located at bit 0 of result[1], and bit 127 is located at bit 7 of result[15].

Time Sync Status (0xA0,0x02)

Description	**Indicates whether a sync has been achieved using the PPS signal.**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*4*
Descriptor	*u8*	*0x02*
Time Sync	bool	True if sync with the PPS signal is currently valid. False if PPS feature is disabled or a PPS signal is not detected.
Last Pps Rcvd	u8	Elapsed time in seconds since last PPS was received, with a maximum value of 255.

GPIO State (0xA0,0x03)

Description	**Indicates the state of all of the user GPIO pins.**	
Notes	This message can be used to correlate external signals with the device time or other data quantities. It should generally be used with slow GPIO signals as brief pulses shorter than the scheduled data rate will be missed. To synchronize with faster signals and pulses, or for more accurate timestamping, utilize the event system and set the GPIO feature to TIMESTAMP in the 3DM GPIO Configuration command (0x0C,0x41). These GPIO states are sampled within one base period of the system data descriptor set. To obtain valid readings, the desired pin(s) must be configured to the GPIO feature (either input or output behavior) using the 3DM GPIO Configuration command (0x0C,0x41). Other gpio features may work on some devices but this is not guaranteed. Consult the factory before producing a design relying on reading pins configured to other feature types.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*3*
Descriptor	*u8*	*0x03*
States	u8	Bitfield containing the states for each GPIO pin. Bit 0 (0x01): pin 1 Bit 1 (0x02): pin 2 Bit 2 (0x04): pin 3 Bit 3 (0x08): pin 4 Bits for pins that don't exist will read as 0.

3DM (0x0C)

Poll NMEA Data (0x0C,0x04)

Description	Poll the device for a NMEA message with the specified format.	
Notes	This function polls for a NMEA message using the provided format. If the format is not provided, the device will attempt to use the stored format (set with the Set NMEA 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 normal NMEA messages.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*4 + 5*count*
Descriptor	*u8*	*0x04*
Suppress Ack	bool	Suppress the usual ACK/NACK reply.
Count	u8	Number of format entries (limited by payload size)
Format Entries	NmeaMessage[count]	List of format entries.
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	

NMEA Message Format (0x0C,0x0C)

Description	**Set, read, or save the NMEA message format.**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*4 + 5*count*
Descriptor	*u8*	*0x0C*
Function Selector	u8	This command supports the following MIP function selectors: `Write Read Save Load Default [WRSLD]`
Count `[W]`	u8	Number of format entries (limited by payload size)
Format Entries `[W]`	NmeaMessage[count]	List of format entries.
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*3 + 5*count*
Response Descriptor	*u8*	*0x8C*
Count	u8	Number of format entries (limited by payload size)
Format Entries	NmeaMessage[count]	List of format entries.

Poll Data (0x0C,0x0D)

Description	Poll the device for a message with the specified descriptor set and format.	
Notes	This function polls for a 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 Message Format command.) If no format is provided and there is no stored format, the device will respond with a NACK. The reply packet contains an ACK/NACK field. The polled data packet is sent separately as a normal Data packet.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*5 + 1*num_descriptors*
Descriptor	*u8*	*0x0D*
Desc Set	u8	Data descriptor set. Must be supported.
Suppress Ack	bool	Suppress the usual ACK/NACK reply.
Num Descriptors	u8	Number of descriptors in the format list.
Descriptors	u8[num_descriptors]	Descriptor format list.
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	

Get Data Base Rate (0x0C,0x0E)

Description	**Get the base rate for the specified descriptor set in Hz.**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*3*
Descriptor	*u8*	*0x0E*
Desc Set	u8	This is the data descriptor set. It must be a supported descriptor.
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*5*
Response Descriptor	*u8*	*0x8E*
Desc Set	u8	Echoes the parameter in the command.
Rate	u16	Base rate in Hz (0 = variable, unknown, or user-defined rate. Data will be sent when received).

Message Format (0x0C,0x0F)

Description	**Set, read, or save the format for a given data packet.**	
Notes	The resulting data messages will maintain the order of descriptors sent in the command.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*5 + 3*num_descriptors*
Descriptor	*u8*	*0x0F*
Function Selector	u8	This command supports the following MIP function selectors: `Write Read Save Load Default [WRSLD]`
Desc Set `[WRSLD]`	u8	Data descriptor set. Must be supported. When function is SAVE, LOAD, or DEFAULT, can be 0 to apply to all descriptor sets.
Num Descriptors `[W]`	u8	Number of descriptors (limited by payload size)
Descriptors `[W]`	DescriptorRate[num_descriptors]	List of descriptors and decimations.
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*4 + 3*num_descriptors*
Response Descriptor	*u8*	*0x8F*
Desc Set	u8	Echoes the descriptor set from the command.
Num Descriptors	u8	Number of descriptors in the list.
Descriptors	DescriptorRate[num_descriptors]	List of descriptors and decimations.

Factory Streaming (0x0C,0x10)

Description	**Configures the device for recording data for technical support.**	
Notes	This command will configure all available data streams to predefined formats designed to be used with technical support.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*4*
Descriptor	*u8*	*0x10*
Action	u8 enum	**Name** / **Value** / **Description**: OVERWRITE / 0 / Replaces the message format(s), removing any existing descriptors. MERGE / 1 / Merges support descriptors into existing format(s). May reorder descriptors. ADD / 2 / Adds descriptors to the current message format(s) without changing existing desc May result in duplicates.
Reserved	u8	Reserved. Set to 0x00.
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	

Action values:

Name	Value	Description
OVERWRITE	0	Replaces the message format(s), removing any existing descriptors.
MERGE	1	Merges support descriptors into existing format(s). May reorder descriptors.
ADD	2	Adds descriptors to the current message format(s) without changing existing desc May result in duplicates.

Data Stream Control (0x0C,0x11)

Description	**Enable/disable the selected data stream.**	
Notes	Each data stream (descriptor set) can be enabled or disabled. The default for the device is all streams enabled. For all functions except 0x01 (use new setting), the new enable flag value is ignored and can be omitted.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*5*
Descriptor	*u8*	*0x11*
Function Selector	u8	This command supports the following MIP function selectors: `Write Read Save Load Default [WRSLD]`
Desc Set `[WRSLD]`	u8	The descriptor set of the stream to control. When function is SAVE, LOAD, or DEFAULT, can be ALL_STREAMS(0) to apply to all descriptor sets. On Generation 5 products, this must be one of the above legacy constants.
Enable `[W]`	bool	True or false to enable or disable the stream.
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*4*
Response Descriptor	*u8*	*0x85*
Desc Set	u8	
Enabled	bool	

You are here:

GNSS SBAS Settings (0x0C,0x22)

<table>
<tr><th>Description</th><th colspan="2">Configure the GNSS SBAS subsystem</th></tr>
<tr><td>Notes</td><td colspan="2"></td></tr>
<tr><th>Parameter Name</th><th>Data Type</th><th>Description</th></tr>
<tr><td>Field Length</td><td>u8</td><td>7 + 2*num_included_prns</td></tr>
<tr><td>Descriptor</td><td>u8</td><td>0x22</td></tr>
<tr><td>Function Selector</td><td>u8</td><td>This command supports the following MIP function selectors:
<code>Write Read Save Load Default [WRSLD]</code></td></tr>
<tr><td>Enable Sbas <code>[W]</code></td><td>u8</td><td>0 - SBAS Disabled, 1 - SBAS enabled</td></tr>
<tr><td>Sbas Options <code>[W]</code></td><td>u16 bitfield</td><td>SBAS options, see definition
<table>
<tr><th>Name</th><th>Bit(s)</th><th>Description</th></tr>
<tr><td>enable_ranging</td><td>0</td><td>Use SBAS pseudoranges in position solution</td></tr>
<tr><td>enable_corrections</td><td>1</td><td>Use SBAS differential corrections</td></tr>
<tr><td>apply_integrity</td><td>2</td><td>Use SBAS integrity information. If enabled, only GPS sate
information is available will be used.</td></tr>
</table></td></tr>
<tr><td>Num Included Prns <code>[W]</code></td><td>u8</td><td>Number of SBAS PRNs to include in search (0 = include all)</td></tr>
<tr><td>Included Prns <code>[W]</code></td><td>u16[num_included_prns]</td><td>List of specific SBAS PRNs to search for</td></tr>
<tr><th>Ack/Nack Reply</th><td colspan="2">See standard MIP ack/nack reply format.</td></tr>
<tr><th>Response Data</th><th>Data Type</th><th>Description</th></tr>
<tr><td>Response Length</td><td>u8</td><td>6 + 2*num_included_prns</td></tr>
<tr><td>Response Descriptor</td><td>u8</td><td>0xA1</td></tr>
<tr><td>Enable Sbas</td><td>u8</td><td>0 - SBAS Disabled, 1 - SBAS enabled</td></tr>
<tr><td>Sbas Options</td><td>u16 bitfield</td><td>SBAS options, see definition
<table>
<tr><th>Name</th><th>Bit(s)</th><th>Description</th></tr>
<tr><td>enable_ranging</td><td>0</td><td>Use SBAS pseudoranges in position solution</td></tr>
<tr><td>enable_corrections</td><td>1</td><td>Use SBAS differential corrections</td></tr>
<tr><td>apply_integrity</td><td>2</td><td>Use SBAS integrity information. If enabled, only GPS sate
information is available will be used.</td></tr>
</table></td></tr>
<tr><td>Num Included Prns</td><td>u8</td><td>Number of SBAS PRNs to include in search (0 = include all)</td></tr>
<tr><td>Included Prns</td><td>u16[num_included_prns]</td><td>List of specific SBAS PRNs to search for</td></tr>
</table>

PPS Source Control (0x0C,0x28)

<table>
<tr><th>Description</th><th colspan="2">Controls the Pulse Per Second (PPS) source.</th></tr>
<tr><td>Notes</td><td colspan="2"></td></tr>
<tr><th>Parameter Name</th><th>Data Type</th><th>Description</th></tr>
<tr><td>Field Length</td><td>u8</td><td>4</td></tr>
<tr><td>Descriptor</td><td>u8</td><td>0x28</td></tr>
<tr><td>Function Selector</td><td>u8</td><td>This command supports the following MIP function selectors:
<code>Write Read Save Load Default [WRSLD]</code></td></tr>
<tr><td>Source <code>[W]</code></td><td>u8 enum</td><td>
<table>
<tr><th>Name</th><th>Value</th><th>Description</th></tr>
<tr><td>DISABLED</td><td>0</td><td>PPS output is disabled. Not valid for PPS source command.</td></tr>
<tr><td>RECEIVER_1</td><td>1</td><td>PPS is provided by GNSS receiver 1.</td></tr>
<tr><td>RECEIVER_2</td><td>2</td><td>PPS is provided by GNSS receiver 2.</td></tr>
<tr><td>GPIO</td><td>3</td><td>PPS is provided to an external GPIO pin. Use the GPIO Setup command to choos configure the pin.</td></tr>
<tr><td>GENERATED</td><td>4</td><td>PPS is generated from the system oscillator.</td></tr>
</table>
</td></tr>
<tr><th>Ack/Nack Reply</th><th colspan="2">See standard MIP ack/nack reply format.</th></tr>
<tr><th>Response Data</th><th>Data Type</th><th>Description</th></tr>
<tr><td>Response Length</td><td>u8</td><td>3</td></tr>
<tr><td>Response Descriptor</td><td>u8</td><td>0xA8</td></tr>
<tr><td>Source</td><td>u8 enum</td><td>
<table>
<tr><th>Name</th><th>Value</th><th>Description</th></tr>
<tr><td>DISABLED</td><td>0</td><td>PPS output is disabled. Not valid for PPS source command.</td></tr>
<tr><td>RECEIVER_1</td><td>1</td><td>PPS is provided by GNSS receiver 1.</td></tr>
<tr><td>RECEIVER_2</td><td>2</td><td>PPS is provided by GNSS receiver 2.</td></tr>
<tr><td>GPIO</td><td>3</td><td>PPS is provided to an external GPIO pin. Use the GPIO Setup command to choos configure the pin.</td></tr>
<tr><td>GENERATED</td><td>4</td><td>PPS is generated from the system oscillator.</td></tr>
</table>
</td></tr>
</table>

Get Supported Events (0x0C,0x2A)

Description	Lists the available trigger or action types.	
Notes	There are a limited number of trigger and action slots available in the device. Up to M triggers and N actions can once in slots 1..M and 1..N respectively. M and N are identified by the max_instances field in the response with th query selector. Each slot can be configured as one of a variety of different types of triggers or actions. The supported types are response to this command. Additionally, there is a limit on the number of a given type. In other words, while the d M triggers in total, only a few of them maybe usable as a given type. This limit helps optimize device resources. T identified in the count field. All of the information in this command is available in the user manual. This command provides a programmatic m the information.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*3*
Descriptor	*u8*	*0x2A*
Query	u8 enum	What type of information to retrieve. **Name** / **Value** / **Description** TRIGGER_TYPES / 1 / Query the supported trigger types and max count for each ACTION_TYPES / 2 / Query the supported action types and max count for each
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*Unavailable*
Response Descriptor	*u8*	*0xB4*
Query	u8 enum	Query type specified in the command. **Name** / **Value** / **Description** TRIGGER_TYPES / 1 / Query the supported trigger types and max count for each ACTION_TYPES / 2 / Query the supported action types and max count for each
Max Instances	u8	Number of slots available. The 'instance' number for the configuration or control commands mu and this value.
Num Entries	u8	Number of supported types.
Entries	Info[num_entries]	List of supported types.

Structures

Info

Description		
Parameter Name	**Data Type**	**Description**
Type	u8	Trigger or action type, as defined in the respective setup command.
Count	u8	This is the maximum number of instances supported for this type.

Event Control (0x0C,0x2B)

Description	**Enables or disables event triggers.**	
Notes	Triggers can be disabled, enabled, and tested. While disabled, a trigger will not evaluate its logic and effective b like no trigger is configured. A disabled trigger will not activate any actions. Triggers are disabled by default. Use this command to enable (or disable) a trigger, or to place it into a test mode. When in test mode, the trigger disabled but the output is forced to the active state, meaning that it will behave as if the trigger logic is satisfied a associated actions will execute.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*5*
Descriptor	*u8*	*0x2B*
Function Selector	u8	This command supports the following MIP function selectors: `Write Read Save Load Default [WRSLD]`
Instance `[WRSLD]`	u8	Trigger instance to affect. 0 can be used to apply the mode to all configured triggers, except when the func selector is READ.
Mode `[W]`	u8 enum	How to change the trigger state. Except when instance is 0, the corresponding trigger must be configured have type 0. **Name** / **Value** / **Description** DISABLED / 0 / Trigger is disabled. ENABLED / 1 / Trigger is enabled and will work normally. TEST / 2 / Forces the trigger to the active state for testing purposes. TEST_PULSE / 3 / Trigger is forced to the active state for one event cycle only. After the test cycle, th mode reverts to the previous state (either enabled or disabled).
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*4*
Response Descriptor	*u8*	*0xB5*
Instance	u8	Trigger instance to affect. 0 can be used to apply the mode to all configured triggers, except when the func selector is READ.
Mode	u8 enum	How to change the trigger state. Except when instance is 0, the corresponding trigger must be configured have type 0. **Name** / **Value** / **Description** DISABLED / 0 / Trigger is disabled. ENABLED / 1 / Trigger is enabled and will work normally. TEST / 2 / Forces the trigger to the active state for testing purposes. TEST_PULSE / 3 / Trigger is forced to the active state for one event cycle only. After the test cycle, th mode reverts to the previous state (either enabled or disabled).

Get Event Trigger Status (0x0C,0x2C)

Description		
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*3 + 1*requested_count*
Descriptor	*u8*	*0x2C*
Requested Count	u8	Number of entries requested. If 0, requests all trigger slots.
Requested Instances	u8[requested_count]	List of trigger instances to query.
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*3 + 2*count*
Response Descriptor	*u8*	*0xB6*
Count	u8	Number of entries requested. If requested_count was 0, this is the number of supported trigger slots.
Triggers	Entry[count]	A list of the configured triggers. Entries are in the order requested, or in increasing order if count was 0.

Structures

Entry

<table>
<tr><th>Description</th><th colspan="2"></th></tr>
<tr><th>Parameter Name</th><th>Data Type</th><th>Description</th></tr>
<tr><td>Type</td><td>u8</td><td>Configured trigger type.</td></tr>
<tr><td>Status</td><td>u8 bitfield</td><td>Trigger status.
<table>
<tr><th>Name</th><th>Bit(s)</th><th>Description</th></tr>
<tr><td>active</td><td>0</td><td>True if the trigger is currently active (either due to its logic or being in test mode).</td></tr>
<tr><td>enabled</td><td>1</td><td>True if the trigger is enabled.</td></tr>
<tr><td>test</td><td>2</td><td>True if the trigger is in test mode.</td></tr>
</table></td></tr>
</table>

Get Event Action Status (0x0C,0x2D)

Description		
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*3 + 1*requested_count*
Descriptor	*u8*	*0x2D*
Requested Count	u8	Number of entries requested. If 0, requests all action slots.
Requested Instances	u8[requested_count]	List of action instances to query.
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*3 + 2*count*
Response Descriptor	*u8*	*0xB7*
Count	u8	Number of entries requested. If requested_count was 0, this is the number of supported action slots.
Actions	Entry[count]	A list of the configured actions. Entries are in the order requested, or in increasing order if count was 0.

Structures

Entry

Description		
Parameter Name	**Data Type**	**Description**
Action Type	u8	Configured action type.
Trigger Id	u8	Associated trigger instance.

Event Trigger Configuration (0x0C,0x2E)

Description	**Configures various types of event triggers.**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*5*
Descriptor	*u8*	*0x2E*
Function Selector	u8	This command supports the following MIP function selectors: `Write Read Save Load Default [WRSLD]`
Instance `[WRSLD]`	u8	Trigger number. When function is SAVE, LOAD, or DEFAULT, this can be 0 to apply to all instance
Type `[W]`	u8 enum	Type of trigger to configure. **Name** / **Value** / **Description** NONE / 0 / No trigger selected. The state will always be inactive. GPIO / 1 / Trigger based on the state of a GPIO pin. See GpioParams. THRESHOLD / 2 / Compare a data quantity against a high and low threshold. See Threshol COMBINATION / 3 / Logical combination of two or more triggers. See CombinationParams.
Parameters `[W]`	Parameters	
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*4*
Response Descriptor	*u8*	*0xB8*
Instance	u8	Trigger number. When function is SAVE, LOAD, or DEFAULT, this can be 0 to apply to all instance
Type	u8 enum	Type of trigger to configure. **Name** / **Value** / **Description** NONE / 0 / No trigger selected. The state will always be inactive. GPIO / 1 / Trigger based on the state of a GPIO pin. See GpioParams. THRESHOLD / 2 / Compare a data quantity against a high and low threshold. See Threshol COMBINATION / 3 / Logical combination of two or more triggers. See CombinationParams.
Parameters	Parameters	

Structures

GpioParams

<table>
<tr><th>Description</th><th colspan="2"></th></tr>
<tr><th>Parameter Name</th><th>Data Type</th><th>Description</th></tr>
<tr><td>Pin</td><td>u8</td><td>GPIO pin number.</td></tr>
<tr><td>Mode</td><td>u8 enum</td><td>How the pin state affects the trigger.
<table>
<tr><th>Name</th><th>Value</th><th>Description</th></tr>
<tr><td>DISABLED</td><td>0</td><td>The pin will have no effect and the trigger will never activate.</td></tr>
<tr><td>WHILE_HIGH</td><td>1</td><td>The trigger will be active while the pin is high.</td></tr>
<tr><td>WHILE_LOW</td><td>2</td><td>The trigger will be active while the pin is low.</td></tr>
<tr><td>EDGE</td><td>4</td><td>Use if the pin is configured for timestamping via the 3DM Gpio Configuration comı (0x0C41).</td></tr>
</table></td></tr>
</table>

ThresholdParams

<table>
<tr><th>Description</th><th colspan="2">Comparison of a supported MIP field parameter against a set of thresholds.</th></tr>
<tr><td>Notes</td><td colspan="2">Triggers when a data quantity meets the comparison criteria. The comparison can be either a window compariso high and low thresholds or a periodic interval. The data quantity is identified by the MIP descriptor set, field descı and parameter number. E.g. Scaled acceleration in the Z direction is specified with desc_set=0x80 (sensor data field_desc=0x04 (scaled accel), and param_id=3 (the third parameter and Z axis). The window comparison can for a variety of purposes, such as disabling a robot's drive motors if it tips over. In this case, a window compariso be set up to monitor the roll angle, (0x80,0x0C,3). The lower threshold would be set to -pi/2 radians and the uppe threshold to pi/2 radians. The interval trigger can be used to perform an action periodically if used with a time fiel to execute the action every 16 ms, set an interval comparison on the GPS time of week parameter (0x80,0xD3,1 high_thres set to 0.016. The lower threshold determines how long the trigger is active within the 16-ms period. Ei comparison type can be inverted by reversing the threshold values; setting low_thres > high_thres will result in the reverse condition.</td></tr>
<tr><th>Parameter Name</th><th>Data Type</th><th>Description</th></tr>
<tr><td>Desc Set</td><td>u8</td><td>Descriptor set of target data quantity.</td></tr>
<tr><td>Field Desc</td><td>u8</td><td>Field descriptor of target data quantity.</td></tr>
<tr><td>Param Id</td><td>u8</td><td>1-based index of the target parameter within the MIP field. E.g. for Scaled Accel (0x80,0x04) a value of 2 represent the Y axis.</td></tr>
<tr><td>Type</td><td>u8 enum</td><td>Determines the type of comparison.
<table>
<tr><th>Name</th><th>Value</th><th>Description</th></tr>
<tr><td>WINDOW</td><td>1</td><td>Window comparison. Trigger is active if low_thres <= value <= high_thres. If the thres are reversed, the trigger is active when value < high_thres or value > low_thres.</td></tr>
<tr><td>INTERVAL</td><td>2</td><td>Trigger at evenly spaced intervals. Normally used with time fields to trigger periodica Trigger is active when (value % interval) <= int_thres. If the thresholds are reversed (high_thres < low_thres) then the trigger is active when (value % low_thres) > high_thı</td></tr>
</table></td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td></tr>
</table>

CombinationParams

Description		
Parameter Name	**Data Type**	**Description**
Logic Table	u16	The last column of a truth table describing the output given the state of each input.
Input Triggers	u8[4]	List of trigger IDs for inputs. Use 0 for unused inputs.

Parameters

Description		
Parameter Name	**Data Type**	**Description**
Gpio	GpioParams	

Threshold	ThresholdParams	
Combination	CombinationParams	

Event Action Configuration (0x0C,0x2F)

Description	**Configures various types of event actions.**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*6*
Descriptor	*u8*	*0x2F*
Function Selector	u8	This command supports the following MIP function selectors: `Write Read Save Load Default [WRSLD]`
Instance `[WRSLD]`	u8	Action number. When function is SAVE, LOAD, or DEFAULT, this can be 0 to apply to all instances
Trigger `[W]`	u8	Trigger ID number.
Type `[W]`	u8 enum	Type of action to configure. <table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td>NONE</td><td>0</td><td>No action. Parameters should be empty.</td></tr><tr><td>GPIO</td><td>1</td><td>Control the state of a GPIO pin. See GpioParameters.</td></tr><tr><td>MESSAGE</td><td>2</td><td>Output a data packet. See MessageParameters.</td></tr></table>
Parameters `[W]`	Parameters	
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*5*
Response Descriptor	*u8*	*0xB9*
Instance	u8	Action number. When function is SAVE, LOAD, or DEFAULT, this can be 0 to apply to all instances
Trigger	u8	Trigger ID number.
Type	u8 enum	Type of action to configure. <table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td>NONE</td><td>0</td><td>No action. Parameters should be empty.</td></tr><tr><td>GPIO</td><td>1</td><td>Control the state of a GPIO pin. See GpioParameters.</td></tr><tr><td>MESSAGE</td><td>2</td><td>Output a data packet. See MessageParameters.</td></tr></table>
Parameters	Parameters	

Structures

GpioParams

<table>
<tr><th colspan="3">Description</th></tr>
<tr><th>Parameter Name</th><th>Data Type</th><th>Description</th></tr>
<tr><td>Pin</td><td>u8</td><td>GPIO pin number.</td></tr>
<tr><td>Mode</td><td>u8 enum</td><td>Behavior of the pin.
<table>
<tr><th>Name</th><th>Value</th><th>Description</th></tr>
<tr><td>DISABLED</td><td>0</td><td>Pin state will not be changed.</td></tr>
<tr><td>ACTIVE_HIGH</td><td>1</td><td>Pin will be set high when the trigger is active and low otherwise.</td></tr>
<tr><td>ACTIVE_LOW</td><td>2</td><td>Pin will be set low when the trigger is active and high otherwise.</td></tr>
<tr><td>ONESHOT_HIGH</td><td>5</td><td>Pin will be set high each time the trigger activates. It will not be set low.</td></tr>
<tr><td>ONESHOT_LOW</td><td>6</td><td>Pin will be set low each time the trigger activates. It will not be set high.</td></tr>
<tr><td>TOGGLE</td><td>7</td><td>Pin will change to the opposite state each time the trigger activates.</td></tr>
</table>
</td></tr>
</table>

MessageParams

Description		
Parameter Name	**Data Type**	**Description**
Desc Set	u8	MIP data descriptor set.
Decimation	u16	Decimation from the base rate. If 0, a packet is emitted each time the trigger activates. Otherwise, packets will be streamed while the trigger is active. The internal decimation counter is reset if the trigger deactivates.
Num Fields	u8	Number of mip fields in the packet. Limited to 12.
Descriptors	u8[num_fields]	List of field descriptors.

Parameters

Description		
Parameter Name	**Data Type**	**Description**
Gpio	GpioParams	Gpio parameters, if type == GPIO. Ignore otherwise.
Message	MessageParams	Message parameters, if type == MESSAGE. Ignore otherwise.

Device Start Up Settings (0x0C,0x30)

Description	**Save, Load, or Reset to Default the values for all device settings.**	
Notes	When a save current settings command is issued, a brief data disturbance may occur while all settings are written to non-volatile memory. This command should have a long timeout as it may take up to 1 second to complete.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*3*
Descriptor	*u8*	*0x30*
Function Selector	u8	This command supports the following MIP function selectors: `Save Load Default [SLD]`
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	

Sensor-to-Vehicle Frame Transformation Euler (0x0C,0x31)

Description	**Sets the sensor-to-vehicle frame transformation using Yaw, Pitch, and Roll Euler angles.**
Notes	These are the Yaw, Pitch, and Roll mounting angles of the sensor with respect to vehicle frame of reference, and describe the transformation of vectors from the sensor body frame to the vehicle frame. Note: This is the transformation, the inverse of the rotation defined in our legacy products. The transformation may be stored in the device as a matrix or quaternion. When Euler angles are read back from the device, they may not be exactly equal to the Euler angles used to set the transformation, but they are functionally equivalent, such that they result in the same transformation. This transformation to the vehicle frame will be applied to the following output quantities: IMU: Scaled Acceleration Scaled Gyro Scaled Magnetometer Delta Theta Delta Velocity Complementary Filter Orientation Estimation Filter: Estimated Orientation, Quaternion Estimated Orientation, Matrix Estimated Orientation, Euler Angles Estimated Linear Acceleration Estimated Angular Rate Estimated Gravity Vector Changing this setting will force all low-pass filters, the complementary filter, and the estimation filter to reset.

Parameter N...	Data Type	Description

Name		
Field Length	*u8*	*15*
Descriptor	*u8*	*0x31*
Function Selector	u8	This command supports the following MIP function selectors: `Write Read Save Load Default [WRSLD]`
Roll `[W]`	float	[radians]
Pitch `[W]`	float	[radians]
Yaw `[W]`	float	[radians]
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*14*
Response Descriptor	*u8*	*0xB1*
Roll	float	[radians]
Pitch	float	[radians]
Yaw	float	[radians]

Sensor-to-Vehicle Frame Transformation Quaternion (0x0C,0x32)

Description	Set the sensor-to-vehicle frame transformation using unit length quaternion.
Notes	Note: This is the transformation, the inverse of the rotation. This quaternion describes the transformation of vectors from the sensor body frame to the vehicle frame of reference, and satisfies the following relationship: Where: is the quaternion describing the transformation. and is a 3-element vector expressed in the sensor body frame. and is a 3-element vector expressed in the vehicle frame. The transformation may be stored in the device as a matrix or a quaternion. When the quaternion is read back from the device, it may not be exactly equal to the quaternion used to set the transformation, but it is functionally equivalent. 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 Changing this setting will force all low-pass filters, the complementary filter, and the estimation filter to reset.

	filter to reset.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*3*
Descriptor	*u8*	*0x32*
Function Selector	u8	This command supports the following MIP function selectors: `Write Read Save Load Default [WRSLD]`
Q `[W]`	Quatf	Unit length quaternion representing transform [w, i, j, k]
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*2*
Response Descriptor	*u8*	*0xB2*
Q	Quatf	Unit length quaternion representing transform [w, i, j, k]

Sensor-to-Vehicle Frame Transformation Direction Cosine Matrix (0x0C,0x33)

Description	Set the sensor to vehicle frame transformation using a using a 3 x 3 direction cosine matrix , stored in row-major order in a 9-element array.
Notes	These angles define the transformation of vectors from the sensor body frame to the fixed vehicle frame, according to: Where: is a 3-element vector expressed in the sensor body frame. is the same 3-element vector expressed in the vehicle frame. The matrix elements are stored is row-major order: The transformation may be stored in the device as a matrix or a quaternion. When is read back from the device, it may not be exactly equal to array used to set the transformation, but it is functionally equivalent. 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 Changing this setting will force all low-pass filters, the complementary filter, and the estimation

	filter to reset.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*3*
Descriptor	*u8*	*0x33*
Function Selector	u8	This command supports the following MIP function selectors: `Write Read Save Load Default [WRSLD]`
Dcm `[W]`	Matrix3f	3 x 3 direction cosine matrix, stored in row-major order
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*2*
Response Descriptor	*u8*	*0xB3*
Dcm	Matrix3f	3 x 3 direction cosine matrix, stored in row-major order

Accelerometer Bias Configuration (0x0C,0x37)

Description	**Configures the user specified accelerometer bias**	
Notes	The user specified bias is subtracted from the calibrated accelerometer output. Value is input in the sensor frame.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*3*
Descriptor	*u8*	*0x37*
Function Selector	u8	This command supports the following MIP function selectors: `Write Read Save Load Default [WRSLD]`
Bias `[W]`	Vector3f	accelerometer bias in the sensor frame (x,y,z) [g]
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*2*
Response Descriptor	*u8*	*0x9A*
Bias	Vector3f	accelerometer bias in the sensor frame (x,y,z) [g]

Gyroscope Bias Configuration (0x0C,0x38)

Description	**Configures the user specified gyroscope bias**	
Notes	The user specified bias is subtracted from the calibrated angular rate output. Value is input in the sensor frame.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*3*
Descriptor	*u8*	*0x38*
Function Selector	u8	This command supports the following MIP function selectors: `Write Read Save Load Default [WRSLD]`
Bias `[W]`	Vector3f	gyro bias in the sensor frame (x,y,z) [radians/second]
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*2*
Response Descriptor	*u8*	*0x9B*
Bias	Vector3f	gyro bias in the sensor frame (x,y,z) [radians/second]

Capture Gyroscope Bias (0x0C,0x39)

Description	**Samples gyro for a specified time range and writes the averaged result to the Gyro Bias vector in RAM**	
Notes	The device will average the gyro output for the duration of "averaging_time_ms." To store the resulting vector in non-volatile memory, use the Set Gyro Bias command. IMPORTANT: The device must be stationary and experiencing minimum vibration for the duration of "averaging_time_ms" Averaging Time range: 1000 to 30,000	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*4*
Descriptor	*u8*	*0x39*
Averaging Time Ms	u16	Averaging time [milliseconds]
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*2*
Response Descriptor	*u8*	*0x9B*
Bias	Vector3f	gyro bias in the sensor frame (x,y,z) [radians/second]

Magnetometer Hard Iron Offset (0x0C,0x3A)

Description	**Configure the user specified magnetometer hard iron offset vector**	
Notes	The values for this offset are determined empirically by external software algorithms based on calibration data taken after the device is installed in its application. These values can be obtained and set by using Microstrain software tools. Alternatively, on some systems, the auto-mag calibration feature may be used to capture these values in-run. The offset is applied to the scaled magnetometer vector prior to output.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*3*
Descriptor	*u8*	*0x3A*
Function Selector	u8	This command supports the following MIP function selectors: `Write Read Save Load Default [WRSLD]`
Offset `[W]`	Vector3f	hard iron offset in the sensor frame (x,y,z) [Gauss]
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*2*
Response Descriptor	*u8*	*0x9C*
Offset	Vector3f	hard iron offset in the sensor frame (x,y,z) [Gauss]

Magnetometer Soft Iron Matrix (0x0C,0x3B)

Description	**Configure the user specified magnetometer soft iron offset matrix**	
Notes	The values for this matrix are determined empirically by external software algorithms based on calibration data taken after the device is installed in its application. These values can be obtained and set by using Microstrain software tools. Alternatively, on some systems, the auto-mag calibration feature may be used to capture these values in-run. The matrix is applied to the scaled magnetometer vector prior to output. The matrix is in row major order:	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*3*
Descriptor	*u8*	*0x3B*
Function Selector	u8	This command supports the following MIP function selectors: `Write Read Save Load Default [WRSLD]`
Offset `[W]`	Matrix3f	soft iron matrix [dimensionless]
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*2*
Response Descriptor	*u8*	*0x9D*
Offset	Matrix3f	soft iron matrix [dimensionless]

UART Baudrate (0x0C,0x40)

Description	**Read, Save, Load, or Reset to Default the baud rate of the main communication channel.**	
Notes	For all functions except 0x01 (use new settings), the new baud rate value is ignored. Please see the device user manual for supported baud rates. The device will wait until all incoming and outgoing data has been sent, up to a maximum of 250 ms, before applying any change. No guarantee is provided as to what happens to commands issued during this delay period; They may or may not be processed and any responses aren't guaranteed to be at one rate or the other. The same applies to data packets. It is highly recommended that the device be idle before issuing this command and that it be issued in its own packet. Users should wait 250 ms after sending this command before further interaction.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*7*
Descriptor	*u8*	*0x40*
Function Selector	u8	This command supports the following MIP function selectors: `Write Read Save Load Default [WRSLD]`
Baud `[W]`	u32	
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*6*
Response Descriptor	*u8*	*0x87*
Baud	u32	

GPIO Configuration (0x0C,0x41)

Description	**Configures the user GPIO pins on the connector for use with several built-in functions or for general i output.**
Notes	GPIO pins are device-dependent. Some features are only available on certain pins. Some behaviors require spe configurations. Consult the device user manual for restrictions and default settings. To avoid glitches on GPIOs configured as an output in a mode other than GPIO, always configure the relevant fu before setting up the pin with this command. Otherwise, the pin state will be undefined between this command a one to set up the feature. For input pins, use this command first so the state is well-defined when the feature is in Some configurations can only be active on one pin at a time. If such configuration is applied to a second pin, the one will take precedence and the original pin's configuration will be reset.

Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*7*
Descriptor	*u8*	*0x41*
Function Selector	u8	This command supports the following MIP function selectors: `Write Read Save Load Default [WRSLD]`
Pin `[WRSLD]`	u8	GPIO pin number counting from 1. For save, load, and default function selectors, this can be 0 to select a
Feature `[W]`	u8 enum	Determines how the pin will be used. (see table below)
		Select an appropriate value from the enumeration based on the selected feature (e.g. for PPS, select on values prefixed with PPS_.) (see table below)

Determines how the pin will be used.

Name	Value	Description
UNUSED	0	The pin is not used. It may be technically possible to read the pin state in this mod this is not guaranteed to be true of all devices or pins.
GPIO	1	General purpose input or output. Use this for direct control of pin output state or to the state of the pin.
PPS	2	Pulse per second input or output.
ENCODER	3	Motor encoder/odometer input.
TIMESTAMP	4	Precision Timestamping. Use with Event Trigger Configuration (0x0C,0x2E).
UART	5	UART data or control lines.

Select an appropriate value from the enumeration based on the selected feature (e.g. for PPS, select on values prefixed with PPS_.)

Name	Value	Description
UNUSED	0	Use 0 unless otherwise specified.
GPIO_INPUT	1	Pin will be an input. This can be used to stream or poll the value and is default setting.
GPIO_OUTPUT_LOW	2	Pin is an output initially in the LOW state. This state will be restored du system startup if the configuration is saved.
GPIO_OUTPUT_HIGH	3	Pin is an output initially in the HIGH state. This state will be restored du

Behavior [W] — u8 enum (continued)

		system startup if the configuration is saved.
PPS_INPUT	1	Pin will receive the pulse-per-second signal. Only one pin can have this behavior. This will only work if the PPS Source command is configured GPIO.
PPS_OUTPUT	2	Pin will transmit the pulse-per-second signal from the device.
ENCODER_A	1	Encoder "A" quadrature input. Only one pin can have this behavior. The command to set this behavior will take precedence.
ENCODER_B	2	Encoder "B" quadrature input. Only one pin can have this behavior. The command to set this behavior will take precedence.
TIMESTAMP_RISING	1	Rising edges will be timestamped.
TIMESTAMP_FALLING	2	Falling edges will be timestamped.
TIMESTAMP_EITHER	3	Both rising and falling edges will be timestamped.
UART_PORT2_TX	33	(0x21) UART port 2 transmit.
UART_PORT2_RX	34	(0x22) UART port 2 receive.
UART_PORT3_TX	49	(0x31) UART port 3 transmit.
UART_PORT3_RX	50	(0x32) UART port 3 receive.

Pin Mode [W] — u8 bitfield

GPIO configuration. May be restricted depending on device, pin, feature, and behavior. See device user

Name	Bit(s)	Description
open_drain	0	The pin will be an open-drain output. The state will be either LOW or FLOATING inst LOW or HIGH, respectively. This is used to connect multiple open-drain outputs from devices. An internal or external pull-up resistor is typically used in combination. The maximum voltage of an open drain output is subject to the device maximum input vo range found in the specifications.
pulldown	1	The pin will have an internal pull-down resistor enabled. This is useful for connecting signals which can only be pulled high such as mechanical switches. Cannot be used combination with pull-up. See the device specifications for the resistance value.
pullup	2	The pin will have an internal pull-up resistor enabled. Useful for connecting inputs to which can only be pulled low such as mechanical switches, or in combination with ar drain output. Cannot be used in combination with pull-down. See the device specific for the resistance value. Use of this mode may restrict the maximum allowed input vo See the device datasheet for details.

Ack/Nack Reply	See standard MIP ack/nack reply format.	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*6*
Response Descriptor	*u8*	*0xC1*
Pin	u8	GPIO pin number counting from 1. For save, load, and default function selectors, this can be 0 to select a

Feature — u8 enum

Determines how the pin will be used.

Name	Value	Description
UNUSED	0	The pin is not used. It may be technically possible to read the pin state in this mod this is not guaranteed to be true of all devices or pins.
GPIO	1	General purpose input or output. Use this for direct control of pin output state or to the state of the pin.
PPS	2	Pulse per second input or output.
ENCODER	3	Motor encoder/odometer input.
TIMESTAMP	4	Precision Timestamping. Use with Event Trigger Configuration (0x0C,0x2E).
UART	5	UART data or control lines.

Select an appropriate value from the enumeration based on the selected feature (e.g. for PPS, select on values prefixed with PPS_.)

Name	Value	Description
UNUSED	0	Use 0 unless otherwise specified.
GPIO_INPUT	1	Pin will be an input. This can be used to stream or poll the value and is default setting.

Field	Type	Name	Value	Description
Behavior	u8 enum	GPIO_OUTPUT_LOW	2	Pin is an output initially in the LOW state. This state will be restored dur system startup if the configuration is saved.
		GPIO_OUTPUT_HIGH	3	Pin is an output initially in the HIGH state. This state will be restored du system startup if the configuration is saved.
		PPS_INPUT	1	Pin will receive the pulse-per-second signal. Only one pin can have this behavior. This will only work if the PPS Source command is configured GPIO.
		PPS_OUTPUT	2	Pin will transmit the pulse-per-second signal from the device.
		ENCODER_A	1	Encoder "A" quadrature input. Only one pin can have this behavior. The command to set this behavior will take precedence.
		ENCODER_B	2	Encoder "B" quadrature input. Only one pin can have this behavior. The command to set this behavior will take precedence.
		TIMESTAMP_RISING	1	Rising edges will be timestamped.
		TIMESTAMP_FALLING	2	Falling edges will be timestamped.
		TIMESTAMP_EITHER	3	Both rising and falling edges will be timestamped.
		UART_PORT2_TX	33	(0x21) UART port 2 transmit.
		UART_PORT2_RX	34	(0x22) UART port 2 receive.
		UART_PORT3_TX	49	(0x31) UART port 3 transmit.
		UART_PORT3_RX	50	(0x32) UART port 3 receive.

Pin Mode — u8 bitfield

GPIO configuration. May be restricted depending on device, pin, feature, and behavior. See device user

Name	Bit(s)	Description
open_drain	0	The pin will be an open-drain output. The state will be either LOW or FLOATING inst LOW or HIGH, respectively. This is used to connect multiple open-drain outputs from devices. An internal or external pull-up resistor is typically used in combination. The maximum voltage of an open drain output is subject to the device maximum input vo range found in the specifications.
pulldown	1	The pin will have an internal pull-down resistor enabled. This is useful for connecting signals which can only be pulled high such as mechanical switches. Cannot be used combination with pull-up. See the device specifications for the resistance value.
pullup	2	The pin will have an internal pull-up resistor enabled. Useful for connecting inputs to which can only be pulled low such as mechanical switches, or in combination with ar drain output. Cannot be used in combination with pull-down. See the device specific for the resistance value. Use of this mode may restrict the maximum allowed input v See the device datasheet for details.

GPIO State (0x0C,0x42)

Description	**Allows the state of the pin to be read or controlled.**	
Notes	This command serves two purposes: 1) To allow reading the state of a pin via command, rather than polling a data quantity, and 2) to provide a way to set the output state without also having to specify the operating mode. The state read back from the pin is the physical state of the pin, rather than a configuration value. The state can be read regardless of its configuration as long as the device supports GPIO input on that pin. If the pin is set to an output, the read value would match the output value. While the state of a pin can always be set, it will only have an observable effect if the pin is set to output mode. This command does not support saving, loading, or resetting the state. Instead, use the GPIO Configuration command, which allows the initial state to be configured.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*5*
Descriptor	*u8*	*0x42*
Function Selector	u8	This command supports the following MIP function selectors: `Write Read [WR]`
Pin `[WR]`	u8	GPIO pin number counting from 1. Cannot be 0.
State `[W]`	bool	The pin state.
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*4*
Response Descriptor	*u8*	*0xC2*
Pin	u8	GPIO pin number counting from 1. Cannot be 0.
State	bool	The pin state.

Odometer Configuration (0x0C,0x43)

<table>
<tr><th>Description</th><th colspan="2">Configures the hardware odometer interface.</th></tr>
<tr><td>Notes</td><td colspan="2"></td></tr>
<tr><td>Parameter Name</td><td>Data Type</td><td>Description</td></tr>
<tr><td>Field Length</td><td>u8</td><td>12</td></tr>
<tr><td>Descriptor</td><td>u8</td><td>0x43</td></tr>
<tr><td>Function Selector</td><td>u8</td><td>This command supports the following MIP function selectors:
<code>Write Read Save Load Default [WRSLD]</code></td></tr>
<tr><td>Mode <code>[W]</code></td><td>u8 enum</td><td>Mode setting.<table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td>DISABLED</td><td>0</td><td>Encoder is disabled.</td></tr><tr><td>QUADRATURE</td><td>2</td><td>Quadrature encoder mode.</td></tr></table></td></tr>
<tr><td>Scaling <code>[W]</code></td><td>float</td><td>Encoder pulses per meter of distance traveled [pulses/m]. Distance traveled is computed using the formu / N * 2R * pi, where d is distance, p is the number of pulses received, N is the encoder resolution, and R is wheel radius. By simplifying all of the parameters into one, the formula d = p / S is obtained, where s is the odometer scaling factor passed to this command. S is equivalent to N / (2R * pi) and has units of pulses / N is in units of "A" pulses per revolution and R is in meters. Make this value negative if the odometer is mo so that it rotates backwards.</td></tr>
<tr><td>Uncertainty <code>[W]</code></td><td>float</td><td>Uncertainty in encoder counts to distance translation (1-sigma value) [m/m].</td></tr>
<tr><td>Ack/Nack Reply</td><td colspan="2">See standard MIP ack/nack reply format.</td></tr>
<tr><td>Response Data</td><td>Data Type</td><td>Description</td></tr>
<tr><td>Response Length</td><td>u8</td><td>11</td></tr>
<tr><td>Response Descriptor</td><td>u8</td><td>0xC3</td></tr>
<tr><td>Mode</td><td>u8 enum</td><td>Mode setting.<table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td>DISABLED</td><td>0</td><td>Encoder is disabled.</td></tr><tr><td>QUADRATURE</td><td>2</td><td>Quadrature encoder mode.</td></tr></table></td></tr>
<tr><td>Scaling</td><td>float</td><td>Encoder pulses per meter of distance traveled [pulses/m]. Distance traveled is computed using the formu / N * 2R * pi, where d is distance, p is the number of pulses received, N is the encoder resolution, and R is wheel radius. By simplifying all of the parameters into one, the formula d = p / S is obtained, where s is the odometer scaling factor passed to this command. S is equivalent to N / (2R * pi) and has units of pulses / N is in units of "A" pulses per revolution and R is in meters. Make this value negative if the odometer is mo so that it rotates backwards.</td></tr>
<tr><td>Uncertainty</td><td>float</td><td>Uncertainty in encoder counts to distance translation (1-sigma value) [m/m].</td></tr>
</table>

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

Description	**Advanced configuration for the IMU data quantity low-pass filters.**	
Notes	Deprecated, use the lowpass filter (0x0C,0x54) command instead. 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 data descriptors: 0x04 - Scaled accelerometer data 0x05 - Scaled gyro data 0x06 - Scaled magnetometer data (if applicable) 0x17 - Scaled pressure data (if applicable)	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*9*
Descriptor	*u8*	*0x50*
Function Selector	u8	This command supports the following MIP function selectors: `Write Read Save Load Default [WRSLD]`
Target Descriptor `[WRSLD]`	u8	Field descriptor of filtered quantity within the Sensor data set. Supported values are accel (0x04), gyro (0x05), mag (0x06), and pressure (0x17), provided the data is supported by the device. Except with the READ function selector, this can be 0 to apply to all of the above quantities.
Enable `[W]`	bool	The target data will be filtered if this is true.
Manual `[W]`	bool	If false, the cutoff frequency is set to half of the streaming rate as configured by the message format command. Otherwise, the cutoff frequency is set according to the following 'frequency' parameter.
Frequency `[W]`	u16	-3dB cutoff frequency in Hz. Will not affect filtering if 'manual' is false.
Reserved `[W]`	u8	Reserved, set to 0x00.
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*8*
Response Descriptor	*u8*	*0x8B*

Target Descriptor	u8	
Enable	bool	True if the filter is currently enabled.
Manual	bool	True if the filter cutoff was manually configured.
Frequency	u16	The cutoff frequency of the filter. If the filter is in auto mode, this value is unspecified.
Reserved	u8	Reserved and must be ignored.

Complementary Filter Configuration (0x0C,0x51)

Description	**Configure the settings for the complementary filter which produces the following (0x80) descriptor set values: attitude matrix (0x80,09), quaternion (0x80,0A), and Euler angle (0x80,0C) outputs.**	
Notes	The filter can be configured to correct for pitch and roll using the accelerometer (with the assumption that linear acceleration is minimal), and to correct for heading using the magnetometer (with the assumption that the local magnetic field is dominated by the Earth's own magnetic field). Pitch/roll and heading corrections each have their own configurable time constants, with a valid range of 1-1000 seconds. The default time constant is 10 seconds.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*13*
Descriptor	*u8*	*0x51*
Function Selector	u8	This command supports the following MIP function selectors: `Write Read Save Load Default [WRSLD]`
Pitch Roll Enable `[W]`	bool	Enable Pitch/Roll corrections
Heading Enable `[W]`	bool	Enable Heading corrections (only available on devices with magnetometer)
Pitch Roll Time Constant `[W]`	float	Time constant associated with the pitch/roll corrections [s]
Heading Time Constant `[W]`	float	Time constant associated with the heading corrections [s]
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*12*
Response Descriptor	*u8*	*0x97*
Pitch Roll Enable	bool	Enable Pitch/Roll corrections
Heading Enable	bool	Enable Heading corrections (only available on devices with magnetometer)
Pitch Roll		

Pitch Roll Time Constant	float	Time constant associated with the pitch/roll corrections [s]
Heading Time Constant	float	Time constant associated with the heading corrections [s]

Sensor Range (0x0C,0x52)

Description	**Changes the IMU sensor gain.**	
Notes	This allows you to optimize the range to get the best accuracy and performance while minimizing over-range eve Use the 3DM Get Calibrated Sensor Ranges (0x0C,0x53) command to determine the appropriate setting value f application. Using values other than those specified may result in a NACK or inaccurate measurement data.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*5*
Descriptor	*u8*	*0x52*
Function Selector	u8	This command supports the following MIP function selectors: `Write Read Save Load Default [WRSLD]`
Sensor `[WRSLD]`	u8 enum	Which type of sensor will get the new range value. **Name** / **Value** / **Description** ALL / 0 / Only allowed for SAVE, LOAD, and DEFAULT function selectors. ACCEL / 1 / Accelerometer. Range is specified in g. GYRO / 2 / Gyroscope. Range is specified in degrees/s. MAG / 3 / Magnetometer. Range is specified in Gauss. PRESS / 4 / Pressure sensor. Range is specified in hPa.
Setting `[W]`	u8	Use the 3DM Get Calibrated Sensor Ranges (0x0C,0x53) command to determine this value.
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*4*
Response Descriptor	*u8*	*0xD2*
Sensor	u8 enum	Which type of sensor will get the new range value. **Name** / **Value** / **Description** ALL / 0 / Only allowed for SAVE, LOAD, and DEFAULT function selectors. ACCEL / 1 / Accelerometer. Range is specified in g. GYRO / 2 / Gyroscope. Range is specified in degrees/s. MAG / 3 / Magnetometer. Range is specified in Gauss. PRESS / 4 / Pressure sensor. Range is specified in hPa.
Setting	u8	Use the 3DM Get Calibrated Sensor Ranges (0x0C,0x53) command to determine this value.

Get Calibrated Sensor Ranges (0x0C,0x53)

Description	**Returns the supported sensor ranges which may be used with the 3DM Sensor Range (0x0C,0x52) co**	
Notes	The response includes an array of (u8, float) pairs which map each allowed setting to the corresponding maximu units. See SensorRangeType for units.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*3*
Descriptor	*u8*	*0x53*
Sensor	u8 enum	The sensor to query. Cannot be ALL. **Name** | **Value** | **Description** ALL | 0 | Only allowed for SAVE, LOAD, and DEFAULT function selectors. ACCEL | 1 | Accelerometer. Range is specified in g. GYRO | 2 | Gyroscope. Range is specified in degrees/s. MAG | 3 | Magnetometer. Range is specified in Gauss. PRESS | 4 | Pressure sensor. Range is specified in hPa.
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*Unavailable*
Response Descriptor	*u8*	*0xD3*
Sensor	u8 enum	The sensor type from the command. **Name** | **Value** | **Description** ALL | 0 | Only allowed for SAVE, LOAD, and DEFAULT function selectors. ACCEL | 1 | Accelerometer. Range is specified in g. GYRO | 2 | Gyroscope. Range is specified in degrees/s. MAG | 3 | Magnetometer. Range is specified in Gauss. PRESS | 4 | Pressure sensor. Range is specified in hPa.
Num Ranges	u8	Number of supported ranges.
Ranges	Entry[num_ranges]	List of possible range settings.

Structures

Entry

Description	

Parameter Name	Data Type	Description
Setting	u8	The value used in the 3DM Sensor Range command and response.
Range	float	The actual range value. Units depend on the sensor type.

Low-Pass Anti-Aliasing Filter (0x0C,0x54)

Description	This command controls the low-pass anti-aliasing filter supported data quantities.	
Notes	See the device user manual for data quantities which support the anti-aliasing filter. If set to automatic mode, the frequency will track half of the transmission rate of the target descriptor according to the configured message format (0x0C,0x0F). For example, if scaled accel (0x80,0x04) is set to stream at 100 Hz, the filter would be set to 50 Hz. Changing the message format to 200 Hz would automatically adjust the filter to 100 Hz. For WRITE, SAVE, LOAD, and DEFAULT function selectors, the descriptor set and/or field descriptor may be 0x00 to set, save, load, or reset the setting for all supported descriptors. The field descriptor must be 0x00 if the descriptor set is 0x00.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*11*
Descriptor	*u8*	*0x54*
Function Selector	u8	This command supports the following MIP function selectors: `Write Read Save Load Default [WRSLD]`
Desc Set `[WRSLD]`	u8	Descriptor set of the quantity to be filtered.
Field Desc `[WRSLD]`	u8	Field descriptor of the quantity to be filtered.
Enable `[W]`	bool	The filter will be enabled if this is true.
Manual `[W]`	bool	If false, the frequency parameter is ignored and the filter will track to half of the configured message format frequency.
Frequency `[W]`	float	Cutoff frequency in Hz. This will return the actual frequency when read out in automatic mode.
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*10*
Response Descriptor	*u8*	*0xD4*
Desc Set	u8	Descriptor set of the quantity to be filtered.
Field Desc	u8	Field descriptor of the quantity to be filtered.
Enable	bool	The filter will be enabled if this is true.
Manual	bool	If false, the frequency parameter is ignored and the filter will track to half of the configured message format frequency.
Frequency	float	Cutoff frequency in Hz. This will return the actual frequency when read out in automatic mode.

Aiding (0x13)

Frame Configuration (0x13,0x01)

Description	Defines an aiding frame associated with a specific sensor frame ID.
Notes	The frame ID used in this command should mirror the frame ID used in the aiding command (if that aiding measu measured in this reference frame). This transform satisfies the following relationship: Where: is rotation matrix defined by the rotation component and is the translation vector is a 3-element position vector expressed in the external sensor frame is a 3-element position vector expressed in the vehicle frame Rotation can be defined using Euler angles OR quaternions. If Format selector is set to Euler Angles, the fourth e the rotation vector is ignored and should be set to 0. When the tracking_enabled flag is 1, the Kalman filter will track errors in the provided frame definition; when 0, nc tracked. Example: GNSS antenna lever arm Frame ID: 1 Format: 1 (Euler) Translation: [0,1,] (GNSS with a 1 meter Y offset in the vehicle frame) Rotation: [0,((Rotational component is not relevant for GNSS measurements, set to zero)

Parameter Name	Data Type	Description
Field Length	*u8*	*6*
Descriptor	*u8*	*0x01*
Function Selector	u8	This command supports the following MIP function selectors: `Write Read Save Load Default [WRSLD]`
Frame Id `[WRSLD]`	u8	Reference frame number. Limit 4.
Format `[WR]`	u8 enum	Format of the transformation. <table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td>EULER</td><td>1</td><td>Translation vector followed by euler angles (roll, pitch, yaw).</td></tr><tr><td>QUATERNION</td><td>2</td><td>Translation vector followed by quaternion (w, x, y, z).</td></tr></table>
Tracking Enabled	bool	If enabled, the Kalman filter will track errors.

[W]		
Translation [W]	Vector3f	Translation X, Y, and Z.
Rotation [W]	Rotation	Rotation as specified by format.
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*5*
Response Descriptor	*u8*	*0x81*
Frame Id	u8	Reference frame number. Limit 4.
Format	u8 enum	Format of the transformation. **Name** / **Value** / **Description** EULER / 1 / Translation vector followed by euler angles (roll, pitch, yaw). QUATERNION / 2 / Translation vector followed by quaternion (w, x, y, z).
Tracking Enabled	bool	If enabled, the Kalman filter will track errors.
Translation	Vector3f	Translation X, Y, and Z.
Rotation	Rotation	Rotation as specified by format.

Structures

Rotation

Description		
Parameter Name	**Data Type**	**Description**
Euler	Vector3f	Rotation represented as euler angles in RPY format [rad]. Range +/- pi.
Quaternion	Quatf	Rotation represented as a quaternion in WXYZ format.

Echo Control (0x13,0x1F)

<table>
<tr><th>Description</th><th colspan="2">Controls command response behavior to external aiding commands</th></tr>
<tr><td>Notes</td><td colspan="2"></td></tr>
<tr><td>Parameter Name</td><td>Data Type</td><td>Description</td></tr>
<tr><td>Field Length</td><td>u8</td><td>4</td></tr>
<tr><td>Descriptor</td><td>u8</td><td>0x1F</td></tr>
<tr><td>Function Selector</td><td>u8</td><td>This command supports the following MIP function selectors:
<code>Write Read Save Load Default [WRSLD]</code></td></tr>
<tr><td>Mode <code>[W]</code></td><td>u8 enum</td><td>Controls data echoing.
<table>
<tr><th>Name</th><th>Value</th><th>Description</th></tr>
<tr><td>SUPPRESS_ACK</td><td>0</td><td>Suppresses the usual command ack field for aiding messages.</td></tr>
<tr><td>STANDARD</td><td>1</td><td>Normal ack/nack behavior.</td></tr>
<tr><td>RESPONSE</td><td>2</td><td>Echo the data back as a response.</td></tr>
</table></td></tr>
<tr><td>Ack/Nack Reply</td><td colspan="2">See standard MIP ack/nack reply format.</td></tr>
<tr><td>Response Data</td><td>Data Type</td><td>Description</td></tr>
<tr><td>Response Length</td><td>u8</td><td>3</td></tr>
<tr><td>Response Descriptor</td><td>u8</td><td>0x9F</td></tr>
<tr><td>Mode</td><td>u8 enum</td><td>Controls data echoing.
<table>
<tr><th>Name</th><th>Value</th><th>Description</th></tr>
<tr><td>SUPPRESS_ACK</td><td>0</td><td>Suppresses the usual command ack field for aiding messages.</td></tr>
<tr><td>STANDARD</td><td>1</td><td>Normal ack/nack behavior.</td></tr>
<tr><td>RESPONSE</td><td>2</td><td>Echo the data back as a response.</td></tr>
</table></td></tr>
</table>

ECEF Position (0x13,0x21)

Description	Cartesian vector position aiding command. Coordinates are given in the WGS84 ECEF system.	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*5*
Descriptor	*u8*	*0x21*
Time	Time	Timestamp of the measurement.
Frame Id	u8	Source ID for this estimate (source_id == 0 indicates this sensor, source_id > 0 indicates an external
Position	Vector3d	ECEF position [m].
Uncertainty	Vector3f	ECEF position uncertainty [m]. Cannot be 0 unless the corresponding valid flags are 0.
Valid Flags	u16 bitfield	Valid flags. Axes with 0 will be completely ignored. (see nested table below)
Ack/Nack Reply	See standard MIP ack/nack reply format.	

Valid Flags:

Name	Bit(s)	Description
X	0	
Y	1	
Z	2	

LLH Position (0x13,0x22)

Description	**Geodetic position aiding command.**	
Notes	Coordinates are given in WGS84 geodetic latitude, longitude, and height above the ellipsoid. Uncertainty is give coordinates, which are parallel to incremental changes in latitude, longitude, and height.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*29*
Descriptor	*u8*	*0x22*
Time	Time	Timestamp of the measurement.
Frame Id	u8	Source ID for this estimate (source_id == 0 indicates this sensor, source_id > 0 indicates an external e
Latitude	double	[deg]
Longitude	double	[deg]
Height	double	[m]
Uncertainty	Vector3f	NED position uncertainty. Cannot be 0 unless the corresponding valid flags are 0.
Valid Flags	u16 bitfield	Valid flags. Axes with 0 will be completely ignored. (see nested table below)
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	

Valid Flags:

Name	Bit(s)	Description
Latitude	0	
Longitude	1	
Height	2	

Height Above Ellipsoid (0x13,0x23)

Description	**Estimated value of the height above ellipsoid.**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*13*
Descriptor	*u8*	*0x23*
Time	Time	Timestamp of the measurement.
Frame Id	u8	Source ID for this estimate (source_id == 0 indicates this sensor, source_id > 0 indicates an external estimate).
Height	float	[m]
Uncertainty	float	[m]
Valid Flags	u16	
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	

ECEF Velocity (0x13,0x28)

Description	**ECEF velocity aiding command. Coordinates are given in the WGS84 ECEF frame.**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*5*
Descriptor	*u8*	*0x28*
Time	Time	Timestamp of the measurement.
Frame Id	u8	Source ID for this estimate (source_id == 0 indicates this sensor, source_id > 0 indicates an external e
Velocity	Vector3f	ECEF velocity [m/s].
Uncertainty	Vector3f	ECEF velocity uncertainty [m/s]. Cannot be 0 unless the corresponding valid flags are 0.
Valid Flags	u16 bitfield	Valid flags. Axes with 0 will be completely ignored. <table><tr><th>Name</th><th>Bit(s)</th><th>Description</th></tr><tr><td>X</td><td>0</td><td></td></tr><tr><td>Y</td><td>1</td><td></td></tr><tr><td>Z</td><td>2</td><td></td></tr></table>
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	

NED Velocity (0x13,0x29)

Description	**NED velocity aiding command. Coordinates are given in the local North East Down frame.**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*5*
Descriptor	*u8*	*0x29*
Time	Time	Timestamp of the measurement.
Frame Id	u8	Source ID for this estimate (source_id == 0 indicates this sensor, source_id > 0 indicates an external e
Velocity	Vector3f	NED velocity [m/s].
Uncertainty	Vector3f	NED velocity uncertainty [m/s]. Cannot be 0 unless the corresponding valid flags are 0.
Valid Flags	u16 bitfield	Valid flags. Axes with 0 will be completely ignored.
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	

Valid Flags bitfield:

Name	Bit(s)	Description
X	0	
Y	1	
Z	2	

Body Frame Velocity (0x13,0x2A)

Description	**Estimated of velocity of the vehicle in the frame associated with the given sensor ID, relative to the veh frame.**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*5*
Descriptor	*u8*	*0x2A*
Time	Time	Timestamp of the measurement.
Frame Id	u8	Source ID for this estimate (source_id == 0 indicates this sensor, source_id > 0 indicates an external e
Velocity	Vector3f	[m/s]
Uncertainty	Vector3f	[m/s] 1-sigma uncertainty. Cannot be 0 unless the corresponding valid flags are 0.
Valid Flags	u16 bitfield	Valid flags. Axes with 0 will be completely ignored. (see table below)
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	

Name	Bit(s)	Description
X	0	
Y	1	
Z	2	

True Heading (0x13,0x31)

Description		
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*13*
Descriptor	*u8*	*0x31*
Time	Time	Timestamp of the measurement.
Frame Id	u8	Source ID for this estimate (source_id == 0 indicates this sensor, source_id > 0 indicates an external estimate).
Heading	float	Heading [radians]. Range +/- Pi.
Uncertainty	float	Cannot be 0 unless the valid flags are 0.
Valid Flags	u16	
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	

Magnetic Field (0x13,0x32)

Description	**Estimate of magnetic field in the frame associated with the given sensor ID.**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*5*
Descriptor	*u8*	*0x32*
Time	Time	Timestamp of the measurement.
Frame Id	u8	Source ID for this estimate (source_id == 0 indicates this sensor, source_id > 0 indicates an external e
Magnetic Field	Vector3f	[G]
Uncertainty	Vector3f	[G] 1-sigma uncertainty. Cannot be 0 unless the corresponding valid flags are 0.
Valid Flags	u16 bitfield	Valid flags. Axes with 0 will be completely ignored. (see table below)
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	

Name	Bit(s)	Description
X	0	
Y	1	
Z	2	

Pressure (0x13,0x33)

Description	**Estimated value of air pressure.**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*13*
Descriptor	*u8*	*0x33*
Time	Time	Timestamp of the measurement.
Frame Id	u8	Source ID for this estimate (source_id == 0 indicates this sensor, source_id > 0 indicates an external estimate).
Pressure	float	[mbar]
Uncertainty	float	[mbar] 1-sigma uncertainty. Cannot be 0 unless the valid flags are 0.
Valid Flags	u16	
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	

Base (0x01)

Ping (0x01,0x01)

Set to idle (0x01,0x02)

Get device information (0x01,0x03)

Get device descriptors (0x01,0x04)

Built in test (0x01,0x05)

Resume (0x01,0x06)

Get device descriptors (extended) (0x01,0x07)

Continuous built-in test (0x01,0x08)

Comm Port Speed (0x01,0x09)

GPS Time Update Command (0x01,0x72)

Reset device (0x01,0x7E)

Ping (0x01,0x01)

Description	**Test Communications with a device.**	
Notes	The Device will respond with an ACK, if present and operating correctly. If the device is not in a normal operating mode, it may NACK.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*2*
Descriptor	*u8*	*0x01*
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	

Set to idle (0x01,0x02)

Description	**Turn off all device data streams.**	
Notes	The Device will respond with an ACK, if present and operating correctly. 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.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*2*
Descriptor	*u8*	*0x02*
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	

Get device information (0x01,0x03)

Description	**Get the device ID strings and firmware version number.**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*2*
Descriptor	*u8*	*0x03*
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*2*
Response Descriptor	*u8*	*0x81*
Device Info	base_device_info	

Get device descriptors (0x01,0x04)

Description	**Get the command and data descriptors supported by the device.**	
Notes	Reply has two fields: "ACK/NACK" and "Descriptors". The "Descriptors" field is an array of 16 bit values. The MSB specifies the descriptor set and the LSB specifies the descriptor.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*2*
Descriptor	*u8*	*0x04*
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*Unavailable*
Response Descriptor	*u8*	*0x82*
Descriptors	u16[0]	

Built in test (0x01,0x05)

Description	**Run the device Built-In Test (BIT).**	
Notes	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. Reference the device user manual to decode the result.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*2*
Descriptor	*u8*	*0x05*
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*6*
Response Descriptor	*u8*	*0x83*
Result	u32	

Resume (0x01,0x06)

Description	**Take the device out of idle mode.**	
Notes	The device responds with ACK upon success.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*2*
Descriptor	*u8*	*0x06*
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	

Get device descriptors (extended) (0x01,0x07)

Description	**Get the command and data descriptors supported by the device.**	
Notes	Reply has two fields: "ACK/NACK" and "Descriptors". The "Descriptors" field is an array of 16 bit values. The MSB specifies the descriptor set and the LSB specifies the descriptor.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*2*
Descriptor	*u8*	*0x07*
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*Unavailable*
Response Descriptor	*u8*	*0x86*
Descriptors	u16[0]	

Continuous built-in test (0x01,0x08)

Description	**Report result of continuous built-in test.**	
Notes	This test is non-disruptive but is not as thorough as the commanded BIT.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*2*
Descriptor	*u8*	*0x08*
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*18*
Response Descriptor	*u8*	*0x88*
Result	u8[16]	Device-specific bitfield (128 bits). See device user manual. Bits are least-significant-byte first. For example, bit 0 is located at bit 0 of result[0], bit 1 is located at bit 1 of result[0], bit 8 is located at bit 0 of result[1], and bit 127 is located at bit 7 of result[15].

Comm Port Speed (0x01,0x09)

Description	**Controls the baud rate of a specific port on the device.**	
Notes	Please see the device user manual for supported baud rates on each port. The device will wait until all incoming and outgoing data has been sent, up to a maximum of 250 ms, before applying any change. No guarantee is provided as to what happens to commands issued during this delay period; They may or may not be processed and any responses aren't guaranteed to be at one rate or the other. The same applies to data packets. It is highly recommended that the device be idle before issuing this command and that it be issued in its own packet. Users should wait 250 ms after sending this command before further interaction.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*8*
Descriptor	*u8*	*0x09*
Function Selector	u8	This command supports the following MIP function selectors: `Write Read Save Load Default [WRSLD]`
Port `[WRSLD]`	u8	Port ID number, starting with 1. When function is SAVE, LOAD, or DEFAULT, this can be 0 to apply to all ports. See the device user manual for details.
Baud `[W]`	u32	Port baud rate. Must be a supported rate.
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*7*
Response Descriptor	*u8*	*0x89*
Port	u8	Port ID number, starting with 1. When function is SAVE, LOAD, or DEFAULT, this can be 0 to apply to all ports. See the device user manual for details.
Baud	u32	Port baud rate. Must be a supported rate.

GPS Time Update Command (0x01,0x72)

Description	**Set device internal GPS time**	
Notes	When combined with a PPS input signal applied to the I/O connector, this command enables complete synchroni of data outputs with an external time base, such as GPS system time. Since the hardware PPS synchronization c detect the fractional number of seconds when pulses arrive, complete synchronization requires that the user prov whole number of seconds via this command. After achieving PPS synchronization, this command should be sent once to set the time-of-week and once to set the week number. PPS synchronization can be verified by monitorir time sync status message (0xA0, 0x02) or the valid flags of any shared external timestamp (0x--, D7) data field.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*8*
Descriptor	*u8*	*0x72*
Function Selector	u8	This command supports the following MIP function selectors: `Write [W]`
Field Id `[W]`	u8 enum	Determines how to interpret value. <table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td>WEEK_NUMBER</td><td>1</td><td>Week number.</td></tr><tr><td>TIME_OF_WEEK</td><td>2</td><td>Time of week in seconds.</td></tr></table>
Value `[W]`	u32	Week number or time of week, depending on the field_id.
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	

Reset device (0x01,0x7E)

Description	**Resets the device.**	
Notes	Device responds with ACK and immediately resets.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*2*
Descriptor	*u8*	*0x7E*
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	

Filter (0x0D)

Reset Navigation Filter (0x0D,0x01)

Description	**Resets the filter to the initialization 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.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*2*
Descriptor	*u8*	*0x01*
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	

Set Initial Heading Control (0x0D,0x03)

Description	**Set the initial heading angle.**	
Notes	The estimation filter will reset the heading estimate to provided value. If the product supports magnetometer aiding and this feature has been enabled, the heading argument will be ignored and the filter will initialize using the inferred magnetic heading.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*6*
Descriptor	*u8*	*0x03*
Heading	float	Initial heading in radians [-pi, pi]
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	

Run Navigation Filter (0x0D,0x05)

Description	**Manual run command.**	
Notes	If the initialization configuration has the "wait_for_run_command" option enabled, the filter will wait until it receives this command before commencing integration and enabling the Kalman filter. Prior to the receipt of this command, the filter will remain in the filter initialization mode.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*2*
Descriptor	*u8*	*0x05*
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	

GNSS Antenna Offset Control (0x0D,0x13)

Description	**Configure the GNSS antenna offset.**	
Notes	For 5-series products, this is expressed in the sensor frame, from the sensor origin to the GNSS antenna RF center. For 7-series products, this is expressed in the vehicle frame, from the sensor origin to the GNSS antenna RF center. This command should also be used for CV7 / GV7-INS NMEA Input over GPIO. The magnitude of the offset vector is limited to 10 meters	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*3*
Descriptor	*u8*	*0x13*
Function Selector	u8	This command supports the following MIP function selectors: `Write Read Save Load Default [WRSLD]`
Offset `[W]`	Vector3f	[meters]
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*2*
Response Descriptor	*u8*	*0x83*
Offset	Vector3f	[meters]

GNSS Aiding Source Control (0x0D,0x15)

Description	**Control the source of GNSS information used to update the Kalman Filter.**	
Notes	Changing the GNSS source while the sensor is in the "running" state may temporarily place it back in the "init" st the new source of GNSS data is received.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*4*
Descriptor	*u8*	*0x15*
Function Selector	u8	This command supports the following MIP function selectors: `Write Read Save Load Default [WRSLD]`
Source `[W]`	u8 enum	**Name** / **Value** / **Description**: ALL_INT / 1 / All internal receivers EXT / 2 / External GNSS messages provided by user INT_1 / 3 / Internal GNSS Receiver 1 only INT_2 / 4 / Internal GNSS Receiver 2 only
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*3*
Response Descriptor	*u8*	*0x86*
Source	u8 enum	**Name** / **Value** / **Description**: ALL_INT / 1 / All internal receivers EXT / 2 / External GNSS messages provided by user INT_1 / 3 / Internal GNSS Receiver 1 only INT_2 / 4 / Internal GNSS Receiver 2 only

External Heading Update (0x0D,0x17)

Description	Provide a filter measurement from an external heading source	
Notes	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; otherwise it is NACK'd. Heading angle uncertainties of <= 0.0 will be NACK'd Please refer to your device user manual for information on the maximum rate of this message. On -25 models, if the declination source (0x0D, 0x43) is not valid, true heading updates will be NACK'd. On -45 models, if the declination source is invalid, magnetic heading updates will be NACK'd.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*11*
Descriptor	*u8*	*0x17*
Heading	float	Bounded by +-PI [radians]
Heading Uncertainty	float	1-sigma [radians]
Type	u8	1 - True, 2 - Magnetic
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	

Accelerometer Noise Standard Deviation (0x0D,0x1A)

Description	**Accelerometer Noise Standard Deviation**	
Notes	Each of the noise values must be greater than 0.0. The noise value represents process noise in the 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.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*3*
Descriptor	*u8*	*0x1A*
Function Selector	u8	This command supports the following MIP function selectors: `Write Read Save Load Default [WRSLD]`
Noise `[W]`	Vector3f	Accel Noise 1-sigma [meters/second^2]
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*2*
Response Descriptor	*u8*	*0x89*
Noise	Vector3f	Accel Noise 1-sigma [meters/second^2]

Gyroscope Noise Standard Deviation (0x0D,0x1B)

Description	**Gyroscope Noise Standard Deviation**	
Notes	Each of the noise values must be greater than 0.0 The noise value represents process noise in the 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.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*3*
Descriptor	*u8*	*0x1B*
Function Selector	u8	This command supports the following MIP function selectors: `Write Read Save Load Default [WRSLD]`
Noise `[W]`	Vector3f	Gyro Noise 1-sigma [rad/second]
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*2*
Response Descriptor	*u8*	*0x8A*
Noise	Vector3f	Gyro Noise 1-sigma [rad/second]

Accelerometer Bias Model Parameters (0x0D,0x1C)

Description	**Accelerometer Bias Model Parameters**	
Notes	Noise values must be greater than 0.0	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*3*
Descriptor	*u8*	*0x1C*
Function Selector	u8	This command supports the following MIP function selectors: `Write Read Save Load Default [WRSLD]`
Beta `[W]`	Vector3f	Accel Bias Beta [1/second]
Noise `[W]`	Vector3f	Accel Noise 1-sigma [meters/second^2]
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*2*
Response Descriptor	*u8*	*0x8B*
Beta	Vector3f	Accel Bias Beta [1/second]
Noise	Vector3f	Accel Noise 1-sigma [meters/second^2]

Gyroscope Bias Model Parameters (0x0D,0x1D)

Description	**Gyroscope Bias Model Parameters**	
Notes	Noise values must be greater than 0.0	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*3*
Descriptor	*u8*	*0x1D*
Function Selector	u8	This command supports the following MIP function selectors: `Write Read Save Load Default [WRSLD]`
Beta `[W]`	Vector3f	Gyro Bias Beta [1/second]
Noise `[W]`	Vector3f	Gyro Noise 1-sigma [rad/second]
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*2*
Response Descriptor	*u8*	*0x8C*
Beta	Vector3f	Gyro Bias Beta [1/second]
Noise	Vector3f	Gyro Noise 1-sigma [rad/second]

External Heading Update With Time (0x0D,0x1F)

Description	**Provide a filter measurement from an external heading source at a specific GPS time**	
Notes	This is more accurate than the External Heading Update (0x0D, 0x17) and should be used in applications where the rate of heading change will cause significant measurement error due to the sampling, transmission, and processing time required. Accurate time stamping of the heading information is important. 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; otherwise it is NACK'd. Heading angle uncertainties of <= 0.0 will be NACK'd Please refer to your device user manual for information on the maximum rate of this message. On -25 models, if the declination source (0x0D, 0x43) is not valid, true heading updates will be NACK'd. On -45 models, if the declination source is invalid, magnetic heading updates will be NACK'd.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*21*
Descriptor	*u8*	*0x1F*
Gps Time	double	[seconds]
Gps Week	u16	[GPS week number, not modulus 1024]
Heading	float	Relative to true north, bounded by +-PI [radians]
Heading Uncertainty	float	1-sigma [radians]
Type	u8	1 - True, 2 - Magnetic
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	

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

Description	**Zero Angular Rate Update**	
Notes	The ZUPT 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.)	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*8*
Descriptor	*u8*	*0x20*
Function Selector	u8	This command supports the following MIP function selectors: `Write Read Save Load Default [WRSLD]`
Enable `[W]`	u8	0 - Disable, 1 - Enable
Threshold `[W]`	float	[radians/second]
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*7*
Response Descriptor	*u8*	*0x8E*
Enable	u8	0 - Disable, 1 - Enable
Threshold	float	[radians/second]

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.**	
Notes	Note: Noise values must be greater than 0.0 The noise value represents process noise in the Estimation Filter. Changing this value modifies how the filter responds to dynamic input and can be used to tune filter performance. Default values provide good performance for most laboratory conditions.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*3*
Descriptor	*u8*	*0x28*
Function Selector	u8	This command supports the following MIP function selectors: `Write Read Save Load Default [WRSLD]`
Noise `[W]`	Vector3f	Gravity Noise 1-sigma [gauss]
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*2*
Response Descriptor	*u8*	*0x93*
Noise	Vector3f	Gravity Noise 1-sigma [gauss]

Magnetic Field Declination Source Control (0x0D,0x43)

Description	Set/Get the local magnetic field declination angle source.	
Notes	This can be used to correct for the local value of declination of the earthmagnetic field. Having a correct value is important for best performance of the auto-mag calibration feature. If you do not have an accurate inclination ang source, it is recommended that you leave the auto-mag calibration feature off.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*8*
Descriptor	*u8*	*0x43*
Function Selector	u8	This command supports the following MIP function selectors: `Write Read Save Load Default [WRSLD]`
Source `[W]`	u8 enum	Magnetic field declination angle source <table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td>NONE</td><td>1</td><td>No source. See command documentation for default behavior</td></tr><tr><td>WMM</td><td>2</td><td>Magnetic field is assumed to conform to the World Magnetic Model, calculated using c location estimate as an input to the model.</td></tr><tr><td>MANUAL</td><td>3</td><td>Magnetic field is assumed to have the parameter specified by the user.</td></tr></table>
Declination `[W]`	float	Declination angle [radians] (only required if source = MANUAL)
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*7*
Response Descriptor	*u8*	*0xB2*
Source	u8 enum	Magnetic field declination angle source <table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td>NONE</td><td>1</td><td>No source. See command documentation for default behavior</td></tr><tr><td>WMM</td><td>2</td><td>Magnetic field is assumed to conform to the World Magnetic Model, calculated using c location estimate as an input to the model.</td></tr><tr><td>MANUAL</td><td>3</td><td>Magnetic field is assumed to have the parameter specified by the user.</td></tr></table>
Declination	float	Declination angle [radians] (only required if source = MANUAL)

Aiding Measurement Control (0x0D,0x50)

Description	**Enables / disables the specified aiding measurement source.**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*6*
Descriptor	*u8*	*0x50*
Function Selector	u8	This command supports the following MIP function selectors: `Write Read Save Load Default [WRSLD]`
Aiding Source `[WRSLD]`	u16 enum	Aiding measurement source <table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td>GNSS_POS_VEL</td><td>0</td><td>GNSS Position and Velocity</td></tr><tr><td>GNSS_HEADING</td><td>1</td><td>GNSS Heading (dual antenna)</td></tr><tr><td>ALTIMETER</td><td>2</td><td>Pressure altimeter (built-in sensor)</td></tr><tr><td>SPEED</td><td>3</td><td>Speed sensor / Odometer</td></tr><tr><td>MAGNETOMETER</td><td>4</td><td>Magnetometer (built-in sensor)</td></tr><tr><td>EXTERNAL_HEADING</td><td>5</td><td>External heading input</td></tr><tr><td>EXTERNAL_ALTIMETER</td><td>6</td><td>External pressure altimeter input</td></tr><tr><td>EXTERNAL_MAGNETOMETER</td><td>7</td><td>External magnetomer input</td></tr><tr><td>BODY_FRAME_VEL</td><td>8</td><td>External body frame velocity input</td></tr><tr><td>ALL</td><td>65535</td><td>Save/load/reset all options</td></tr></table>
Enable `[W]`	bool	Controls the aiding source
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*5*
Response Descriptor	*u8*	*0xD0*
Aiding Source	u16 enum	Aiding measurement source <table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td>GNSS_POS_VEL</td><td>0</td><td>GNSS Position and Velocity</td></tr><tr><td>GNSS_HEADING</td><td>1</td><td>GNSS Heading (dual antenna)</td></tr><tr><td>ALTIMETER</td><td>2</td><td>Pressure altimeter (built-in sensor)</td></tr><tr><td>SPEED</td><td>3</td><td>Speed sensor / Odometer</td></tr><tr><td>MAGNETOMETER</td><td>4</td><td>Magnetometer (built-in sensor)</td></tr><tr><td>EXTERNAL_HEADING</td><td>5</td><td>External heading input</td></tr><tr><td>EXTERNAL_ALTIMETER</td><td>6</td><td>External pressure altimeter input</td></tr><tr><td>EXTERNAL_MAGNETOMETER</td><td>7</td><td>External magnetomer input</td></tr><tr><td>BODY_FRAME_VEL</td><td>8</td><td>External body frame velocity input</td></tr><tr><td>ALL</td><td>65535</td><td>Save/load/reset all options</td></tr></table>
Enable	bool	Controls the aiding source

Navigation Filter Initialization (0x0D,0x52)

Description	Controls the source and values used for initial conditions of the navigation solution.
Notes	Notes: Initial conditions are the position, velocity, and attitude of the platform used when the filter starts running or For the user specified position array, the units are meters if the ECEF frame is selected, and degrees latitude, d longitude, and meters above ellipsoid if the latitude/longitude/height frame is selected. For the user specified vel the units are meters per second, but the reference frame depends on the reference frame selector (ECEF or NE

Parameter Name	Data Type	Description
Field Length	*u8*	*19*
Descriptor	*u8*	*0x52*
Function Selector	u8	This command supports the following MIP function selectors: `Write Read Save Load Default [WRSLD]`
Wait For Run Command `[W]`	u8	Initialize filter only after receiving "run" command
Initial Cond Src `[W]`	u8 enum	Initial condition source: <table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td>AUTO_POS_VEL_ATT</td><td>0</td><td>Automatic position, velocity and attitude</td></tr><tr><td>AUTO_POS_VEL_PITCH_ROLL</td><td>1</td><td>Automatic position and velocity, automatic pitch and roll, ar specified heading</td></tr><tr><td>AUTO_POS_VEL</td><td>2</td><td>Automatic position and velocity, with fully user-specified atti</td></tr><tr><td>MANUAL</td><td>3</td><td>User-specified position, velocity, and attitude.</td></tr></table>
Auto Heading Alignment Selector `[W]`	u8 bitfield	Bitfield specifying the allowed automatic heading alignment methods for automatic initial conditions. B to 1 to enable, and the correspond to the following: <table><tr><th>Name</th><th>Bit(s)</th><th>Description</th></tr><tr><td>dual_antenna</td><td>0</td><td>Dual-antenna GNSS alignment</td></tr><tr><td>kinematic</td><td>1</td><td>GNSS kinematic alignment (GNSS velocity determines initial heading)</td></tr><tr><td>magnetometer</td><td>2</td><td>Magnetometer heading alignment (Internal magnetometer determines initial he</td></tr><tr><td>external</td><td>3</td><td>External heading alignment (External heading input determines heading)</td></tr></table>
Initial Heading `[W]`	float	User-specified initial platform heading (degrees).
Initial Pitch `[W]`	float	User-specified initial platform pitch (degrees)
Initial Roll `[W]`	float	User-specified initial platform roll (degrees)
Initial Position	Vector3f	User-specified initial platform position (units determined by reference frame selector, see note.)

[W]		
Initial Velocity [W]	Vector3f	User-specified initial platform velocity (units determined by reference frame selector, see note.)
Reference Frame Selector [W]	u8 enum	User-specified initial position/velocity reference frames <table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td>ECEF</td><td>1</td><td>WGS84 Earth-fixed, earth centered coordinates</td></tr><tr><td>LLH</td><td>2</td><td>WGS84 Latitude, longitude, and height above ellipsoid</td></tr></table>
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*18*
Response Descriptor	*u8*	*0xD2*
Wait For Run Command	u8	Initialize filter only after receiving "run" command
Initial Cond Src	u8 enum	Initial condition source: <table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td>AUTO_POS_VEL_ATT</td><td>0</td><td>Automatic position, velocity and attitude</td></tr><tr><td>AUTO_POS_VEL_PITCH_ROLL</td><td>1</td><td>Automatic position and velocity, automatic pitch and roll, ar specified heading</td></tr><tr><td>AUTO_POS_VEL</td><td>2</td><td>Automatic position and velocity, with fully user-specified atti</td></tr><tr><td>MANUAL</td><td>3</td><td>User-specified position, velocity, and attitude.</td></tr></table>
Auto Heading Alignment Selector	u8 bitfield	Bitfield specifying the allowed automatic heading alignment methods for automatic initial conditions. B to 1 to enable, and the correspond to the following: <table><tr><th>Name</th><th>Bit(s)</th><th>Description</th></tr><tr><td>dual_antenna</td><td>0</td><td>Dual-antenna GNSS alignment</td></tr><tr><td>kinematic</td><td>1</td><td>GNSS kinematic alignment (GNSS velocity determines initial heading)</td></tr><tr><td>magnetometer</td><td>2</td><td>Magnetometer heading alignment (Internal magnetometer determines initial he</td></tr><tr><td>external</td><td>3</td><td>External heading alignment (External heading input determines heading)</td></tr></table>
Initial Heading	float	User-specified initial platform heading (degrees).
Initial Pitch	float	User-specified initial platform pitch (degrees)
Initial Roll	float	User-specified initial platform roll (degrees)
Initial Position	Vector3f	User-specified initial platform position (units determined by reference frame selector, see note.)
Initial Velocity	Vector3f	User-specified initial platform velocity (units determined by reference frame selector, see note.)
Reference Frame Selector	u8 enum	User-specified initial position/velocity reference frames <table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td>ECEF</td><td>1</td><td>WGS84 Earth-fixed, earth centered coordinates</td></tr><tr><td>LLH</td><td>2</td><td>WGS84 Latitude, longitude, and height above ellipsoid</td></tr></table>

Adaptive Filter Control (0x0D,0x53)

Description	**Configures the basic setup for auto-adaptive filtering. See product manual for a detailed description of this feature.**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*6*
Descriptor	*u8*	*0x53*
Function Selector	u8	This command supports the following MIP function selectors: `Write Read Save Load Default [WRSLD]`
Level `[W]`	u8	Auto-adaptive operating level: 0=Off, 1=Conservative, 2=Moderate (default), 3=Aggressive.
Time Limit `[W]`	u16	Maximum duration of measurement rejection before entering recovery mode (ms)
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*5*
Response Descriptor	*u8*	*0xD3*
Level	u8	Auto-adaptive operating level: 0=Off, 1=Conservative, 2=Moderate (default), 3=Aggressive.
Time Limit	u16	Maximum duration of measurement rejection before entering recovery mode (ms)

GNSS Multi-Antenna Offset Control (0x0D,0x54)

Description	**Set the antenna lever arm.**	
Notes	This command works with devices that utilize multiple antennas. **Offset Limit**: 10 m magnitude (default)	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*4*
Descriptor	*u8*	*0x54*
Function Selector	u8	This command supports the following MIP function selectors: `Write Read Save Load Default [WRSLD]`
Receiver Id `[WRSLD]`	u8	Receiver: 1, 2, etc...
Antenna Offset `[W]`	Vector3f	Antenna lever arm offset vector in the vehicle frame (m)
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*3*
Response Descriptor	*u8*	*0xD4*
Receiver Id	u8	
Antenna Offset	Vector3f	

Relative Position Configuration (0x0D,0x55)

<table>
<tr><th>Description</th><th colspan="2">Configure the reference location for filter relative positioning outputs</th></tr>
<tr><td>Notes</td><td colspan="2"></td></tr>
<tr><td>Parameter Name</td><td>Data Type</td><td>Description</td></tr>
<tr><td>Field Length</td><td>u8</td><td>5</td></tr>
<tr><td>Descriptor</td><td>u8</td><td>0x55</td></tr>
<tr><td>Function Selector</td><td>u8</td><td>This command supports the following MIP function selectors:
<code>Write Read Save Load Default [WRSLD]</code></td></tr>
<tr><td>Source <code>[W]</code></td><td>u8</td><td>0 - auto (RTK base station), 1 - manual</td></tr>
<tr><td>Reference Frame Selector <code>[W]</code></td><td>u8 enum</td><td>ECEF or LLH
<table>
<tr><th>Name</th><th>Value</th><th>Description</th></tr>
<tr><td>ECEF</td><td>1</td><td>WGS84 Earth-fixed, earth centered coordinates</td></tr>
<tr><td>LLH</td><td>2</td><td>WGS84 Latitude, longitude, and height above ellipsoid</td></tr>
</table></td></tr>
<tr><td>Reference Coordinates <code>[W]</code></td><td>Vector3d</td><td>reference coordinates, units determined by source selection</td></tr>
<tr><td>Ack/Nack Reply</td><td colspan="2">See standard MIP ack/nack reply format.</td></tr>
<tr><td>Response Data</td><td>Data Type</td><td>Description</td></tr>
<tr><td>Response Length</td><td>u8</td><td>4</td></tr>
<tr><td>Response Descriptor</td><td>u8</td><td>0xD5</td></tr>
<tr><td>Source</td><td>u8</td><td>0 - auto (RTK base station), 1 - manual</td></tr>
<tr><td>Reference Frame Selector</td><td>u8 enum</td><td>ECEF or LLH
<table>
<tr><th>Name</th><th>Value</th><th>Description</th></tr>
<tr><td>ECEF</td><td>1</td><td>WGS84 Earth-fixed, earth centered coordinates</td></tr>
<tr><td>LLH</td><td>2</td><td>WGS84 Latitude, longitude, and height above ellipsoid</td></tr>
</table></td></tr>
<tr><td>Reference Coordinates</td><td>Vector3d</td><td>reference coordinates, units determined by source selection</td></tr>
</table>

Reference point lever arm (0x0D,0x56)

Description	**Lever arm offset with respect to the sensor for the indicated point of reference.**	
Notes	This is used to change the location of the indicated point of reference, and will affect filter position and velocity ou Changing this setting from default will result in a global position offset that depends on vehicle attitude, and a velo that depends on vehicle attitude and angular rate. The lever arm is defined by a 3-element vector that points from the sensor to the desired reference point, with (x,y components given in the vehicle's reference frame. Note, if the reference point selector is set to VEH (1), this setting will affect the following data fields: (0x82, 0x01) 0x02), (0x82, 0x40), (0x82, 0x41), and (0x82, 42) **Offset Limits** Reference Point VEH (1): 10 m magnitude (default)	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*4*
Descriptor	*u8*	*0x56*
Function Selector	u8	This command supports the following MIP function selectors: `Write Read Save Load Default [WRSLD]`
Ref Point Sel `[W]`	u8 enum	Reserved, must be 1 <table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td>VEH</td><td>1</td><td>Defines the origin of the vehicle</td></tr></table>
Lever Arm Offset `[W]`	Vector3f	[m] Lever arm offset vector in the vehicle's reference frame.
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*3*
Response Descriptor	*u8*	*0xD6*
Ref Point Sel	u8 enum	Reserved, must be 1 <table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td>VEH</td><td>1</td><td>Defines the origin of the vehicle</td></tr></table>
Lever Arm Offset	Vector3f	[m] Lever arm offset vector in the vehicle's reference frame.

Measurement speed lever arm (0x0D,0x61)

Description	**Lever arm offset for speed measurements.**	
Notes	This is used to compensate for an off-center measurement point having a different speed due to rotation of the vehicle. The typical use case for this would be an odometer attached to a wheel on a standard 4-wheeled vehicle. If the odometer is on the left wheel, it will report higher speed on right turns and lower speed on left turns. This is because the outside edge of the curve is longer than the inside edge.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*4*
Descriptor	*u8*	*0x61*
Function Selector	u8	This command supports the following MIP function selectors: `Write Read Save Load Default [WRSLD]`
Source `[WRSLD]`	u8	Reserved, must be 1.
Lever Arm Offset `[W]`	Vector3f	[m] Lever arm offset vector in the vehicle's reference frame.
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*3*
Response Descriptor	*u8*	*0xE1*
Source	u8	Reserved, must be 1.
Lever Arm Offset	Vector3f	[m] Lever arm offset vector in the vehicle's reference frame.

Wheeled Vehicle Constraint Control (0x0D,0x63)

Description	**Configure the wheeled vehicle kinematic constraint.**	
Notes	When enabled, the filter uses the assumption that velocity is constrained to the primary vehicle axis. By convention, the primary vehicle axis is the vehicle X-axis (note: the sensor may be physically installed in any orientation on the vehicle if the appropriate mounting transformation has been specified). This constraint will typically improve heading estimates for vehicles where the assumption is valid, such as an automobile, particularly when GNSS coverage is intermittent.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*4*
Descriptor	*u8*	*0x63*
Function Selector	u8	This command supports the following MIP function selectors: `Write Read Save Load Default [WRSLD]`
Enable `[W]`	u8	0 - Disable, 1 - Enable
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*3*
Response Descriptor	*u8*	*0xE3*
Enable	u8	0 - Disable, 1 - Enable

GNSS Antenna Offset Calibration Control (0x0D,0x64)

Description	**Configure the GNSS antenna lever arm calibration.**	
Notes	When enabled, the filter will enable lever arm error tracking, up to the maximum offset specified.	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*8*
Descriptor	*u8*	*0x64*
Function Selector	u8	This command supports the following MIP function selectors: `Write Read Save Load Default [WRSLD]`
Enable `[W]`	u8	0 - Disable, 1 - Enable
Max Offset `[W]`	float	Maximum absolute value of lever arm offset error in the vehicle frame [meters]. See device user manual for the valid range of this parameter.
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*7*
Response Descriptor	*u8*	*0xE4*
Enable	u8	0 - Disable, 1 - Enable
Max Offset	float	Maximum absolute value of lever arm offset error in the vehicle frame [meters]. See device user manual for the valid range of this parameter.

GNSS (0x0E)

Receiver Info (0x0E,0x01)

Signal Configuration (0x0E,0x02)

Spartn Configuration (0x0E,0x20)

Receiver Info (0x0E,0x01)

Description	**Return information about the GNSS receivers in the device.**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*2*
Descriptor	*u8*	*0x01*
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*3 + 34*num_receivers*
Response Descriptor	*u8*	*0x81*
Num Receivers	u8	Number of physical receivers in the device
Receiver Info	Info[num_receivers]	

Structures

Info

Description		
Parameter Name	**Data Type**	**Description**
Receiver Id	u8	Receiver id: e.g. 1, 2, etc.
Mip Data Descriptor Set	u8	MIP descriptor set associated with this receiver
Description	char[32]	Ascii description of receiver. Contains the following info (comma-delimited): Module name/model Firmware version info

Signal Configuration (0x0E,0x02)

Description	**Configure the GNSS signals used by the device.**	
Notes		
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*11*
Descriptor	*u8*	*0x02*
Function Selector	u8	This command supports the following MIP function selectors: `Write Read Save Load Default [WRSLD]`
Gps Enable `[W]`	u8	Bitfield 0: Enable L1CA, 1: Enable L2C, 2: Enable L5
Glonass Enable `[W]`	u8	Bitfield 0: Enable L1OF, 1: Enable L2OF
Galileo Enable `[W]`	u8	Bitfield 0: Enable E1, 1: Enable E5B, 2: Enable E5A
Beidou Enable `[W]`	u8	Bitfield 0: Enable B1, 1: Enable B2, 2: Enable B2A
Reserved `[W]`	u8[4]	
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*10*
Response Descriptor	*u8*	*0x82*
Gps Enable	u8	Bitfield 0: Enable L1CA, 1: Enable L2C, 2: Enable L5
Glonass Enable	u8	Bitfield 0: Enable L1OF, 1: Enable L2OF
Galileo Enable	u8	Bitfield 0: Enable E1, 1: Enable E5B, 2: Enable E5A
Beidou Enable	u8	Bitfield 0: Enable B1, 1: Enable B2, 2: Enable B2A
Reserved	u8[4]	

Spartn Configuration (0x0E,0x20)

Description	Configure the SPARTN corrections service parameters.	
Notes	Notes: - Enable and type settings will only update after a power cycle - Type settings will only take effect after a power cycle - Key information can be updated while running	
Parameter Name	**Data Type**	**Description**
Field Length	*u8*	*81*
Descriptor	*u8*	*0x20*
Function Selector	u8	This command supports the following MIP function selectors: `Write Read Save Load Default [WRSLD]`
Enable `[W]`	u8	Enable/Disable the SPARTN subsystem (0 = Disabled, 1 = Enabled)
Type `[W]`	u8	Connection type (0 - None, 1 = Network, 2 = L-Band)
Current Key Tow `[W]`	u32	The GPS time of week the current key is valid until
Current Key Week `[W]`	u16	The GPS week number the current key is valid until
Current Key `[W]`	u8[32]	32 character string of ASCII hex values for the current key (e.g. "bc" for 0xBC)
Next Key Tow `[W]`	u32	The GPS time of week the next key is valid until
Next Key Week `[W]`	u16	The GPS week number the next key is valid until
Next Key `[W]`	u8[32]	32 character string of ASCII hex valuesfor the next key (e.g. "bc" for 0xBC)
Ack/Nack Reply	**See standard MIP ack/nack reply format.**	
Response Data	**Data Type**	**Description**
Response Length	*u8*	*80*
Response Descriptor	*u8*	*0xA0*
Enable	u8	Enable/Disable the SPARTN subsystem (0 = Disabled, 1 = Enabled)
Type	u8	Connection type (0 - None, 1 = Network, 2 = L-Band)
Current Key Tow	u32	The GPS time of week the current key is valid until
Current Key Week	u16	The GPS week number the current key is valid until
Current Key	u8[32]	32 character string of ASCII hex values for the current key (e.g. "bc" for 0xBC)
Next Key Tow	u32	The GPS time of week the next key is valid until
Next Key Week	u16	The GPS week number the next key is valid until
Next Key	u8[32]	32 character string of ASCII hex valuesfor the next key (e.g. "bc" for 0xBC)

Standard MIP Types and Values

Standard Types

Type	Size [bytes]	Description
bool	1	A single-byte value which can be either true (0x01) or false (0x00).
char	1	A single-byte value which represents an ASCII character. Typically used as an array to form a string.
u8	1	Unsigned 8-bit byte.
u16	2	Unsigned 16-bit value.
u32	4	Unsigned 32-bit value.
u64	8	Unsigned 64-bit value.
s8	1	Signed 8-bit value (two's complement).
s16	2	Signed 16-bit value (two's complement).
s32	4	Signed 32-bit value (two's complement).
s64	8	Signed 64-bit value (two's complement).
float	4	IEEE-754 floating point value.
double	8	IEEE-754 double-precision floating point value.

Vector Types

Type	Size [bytes]	Description
vector3f	12	3D vector of floats
vector3d	24	3D vector of doubles
vector4f	16	4D vector of floats
vector4d	32	4D vector of doubles
matrix3f	36	3x3 matrix of floats in row major order
matrix3d	72	3x3 matrix of doubles in row major order
quatf	16	4D quaternion vector of floats (qw, qx, qy, qz)

Ack/Nack Field Specification

Description	This field is used to indicate the result of a received command.	
Parameter Name	Data Type	Description
Field Length	u8	4
Descriptor	u8	0xF1
Command Descriptor	u8	Matches command descriptor.

Reply Code	u8	Ack/Nack Reply Code

Ack/Nack Codes

0x00 - Success
0x01 - Unknown Command
0x02 - Bad Checksum
0x03 - Invalid Parameter
0x04 - Command Failed
0x05 - Command Timeout
0x06 - Unknown descriptor set

Function Selectors

Most settings commands support a function selector, which allows the setting to be changed, read back, saved, loaded, or reset to the default. Some commands don't support all 5 options. For most settings, no additional parameters are needed when the function selector is not 0x01. However certain commands need more information (for example, which subset of settings to read back).

0x01 - Apply new setting
0x02 - Read current setting
0x03 - Save current setting to non-volatile memory
0x04 - Load current setting from non-volatile memory
0x05 - Restore factory default setting