ANavS Positioning and Map-Creation Systems – Reference Guide

Applicable to:

- Multi-Sensor RTK/PPP Module (MS-RTK)
  - Realtime Positioning Engine
  - Postprocessing Engine
- RTK/RTCM Reference Station
- ISP (Integrated Sensor Platform)
- ANavS Software Tools
Abstract

The following guide provides a detailed explanation for the usage of the different ANavS® positioning Systems, the Multi-Sensor RTK/PPP module (MS-RTK), the RTK/RTCM Reference Station and the Integrated-Sensor-Platform (ISP).

The **ANavS® MS-RTK module** (chapter 2) provides a precise position, velocity, and attitude information. It is a turnkey system with a very attractive price/performance ratio and can be easily integrated into users’ application. The module includes up to 3 Multi-frequency, Multi-GNSS (GPS, Galileo, Glonass, Beidou) receivers, a MEMS IMU, a barometer, a CAN interface for reception of vehicle data (wheel odometry and steering angle), an LTE module for reception of RTK/PPP corrections (e.g. from ANavS® RTK reference station), and the powerful ANavS® Sensor Fusion on a single board. The latter one performs a tight coupling of all sensor data with an Extended Kalman Filter (EKF), both in real time and in post-processing.

The **ANavS® RTK/RTCM Reference Station** (chapter 3) provides correction data in standard RTCM format to guarantee precise positioning in every situation without any integration effort. It only requires a power supply and connected GNSS antenna, thus it is ready to use after a short calibration phase of the fixed position.

The **Integrated Sensor Platform** (chapter 4) is a hardware-platform for easy integration of a large variety of sensors without any effort. It comes with a standard configuration of three GNSS receivers and integrated antennas, a high-grade MEMS IMU, a CAN interface for wheel odometry data and a barometer. On top of the standard sensors a fully integrated computer vision module is equipped, that can be flexibly configured with two cameras and/or a 3D-LiDAR with powerful processor for artificial intelligence (AI) applications.

Helpful **software tools** for system configuration, for solution visualization, for postprocessing and for converters are explained also in this guide.
Typographical Conventions

abc -param  Command-line instructions, e.g., in shell
abc -param  MSRTKF command-line instructions

List of Acronyms

AI  Artificial Intelligence
CAN  Controller Area Network
GNSS  Global Navigation Satellite System
GUI  Graphical User Interface, Graphical User Interface
IMU  Inertial Measurement Unit
ISP  Integrated Sensor Platform
LTE  Long Term Evolution
MEMS  Micro-Electro-Mechanical Systems
PPP  Precise-Point-Positioning
ROS  Realtime Operating System
RTCM  Radio Technical Commission for Maritime services
RTK  Real-Time-Kinematik
SLAM  Simultaneous Localization and Mapping
SSH  Secure Shell
VPU  Vision Processing Unit
## Document Change Log

<table>
<thead>
<tr>
<th>Issue</th>
<th>Revision</th>
<th>Sections Affected</th>
<th>Details of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>All</td>
<td>Initial version of document.</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>9</td>
<td>Added overview of NTP functionality.</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2.8.8</td>
<td>Added hint for new settings with already running navigation-service.</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>13</td>
<td>Added chapter for LTE-VPN configuration.</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>6.1.1</td>
<td>Updated binary protocol with new parameters (inserted in “reserved” fields), Added sub-chapter “Time-Synchronization with MS-RTK clock”.</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>Appendix 2/3/4/5/0</td>
<td>Added technical drawings for casing types including coordinates of IMU. Changed power-consumption rating for newer hardware versions (v9).</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>12</td>
<td>Added Java-Version for using Record-Extractor tool</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>8</td>
<td>Updated CAN-Interface description</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>2.4, 3.2</td>
<td>Recommendation to use triple frequency variant of ANavS positioning modules only with connected GNSS antennas.</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>4.2, 4.3, 4.4</td>
<td>Updated chapter “Getting-Started with the Integrated-Sensor-Platform (ISP)”</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>8.3.7</td>
<td>Added sub-chapter “Set up CAN hardware filter”</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>4.2, 4.4, 10</td>
<td>Updated 4.2 and 4.4 to the current ISP setup with optional two embedded GPU platforms. Added object detection as optional algorithm and corresponding subsection. Updated ROS-Ethernet Adapter description.</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>8.2</td>
<td>Updated description with endianness of CAN-Output of ANavS solution</td>
</tr>
<tr>
<td>2</td>
<td>11</td>
<td>Appendix 1</td>
<td>Added example DBC file for ANavS solution output</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>4.4</td>
<td>Updated section 4.4 with updated VPU setup with optional two embedded GPU platforms.</td>
</tr>
<tr>
<td>2</td>
<td>13</td>
<td>Appendix 3</td>
<td>Updated drawing with exact IMU position</td>
</tr>
<tr>
<td>2</td>
<td>14</td>
<td>6</td>
<td>Velocity in MSKF filter is given in body frame, not in navigation frame</td>
</tr>
<tr>
<td>2</td>
<td>15</td>
<td>4.2, 4.4</td>
<td>Updated ISP Setup and Computer Vision Operation Modes description, made compatible to setups with a single and two embedded computing platforms. Updated object detection module description with object detection and tracking, included object list message description.</td>
</tr>
<tr>
<td>Page</td>
<td>Section Numbers</td>
<td>Changes</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>-----------------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td>2.10</td>
<td>Added new section describing the tightly coupled Postprocessing Engine</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>16</td>
<td>0</td>
<td>Updated minimum voltage of power supply</td>
</tr>
<tr>
<td>2</td>
<td>17</td>
<td>4, 4.4.4, 4.4.6 Appendix 7</td>
<td>Updated the ISP chapter, especially added section 4.4.4 “Sensor Data Visualization in the ANavS GUI and in ROS” and section 4.4.6 “LiDAR Simultaneous Localization and Mapping (SLAM)”. Added Appendix 7 including the ISP extrinsic calibration.</td>
</tr>
<tr>
<td>2</td>
<td>18</td>
<td>2.1 - 2.3</td>
<td>Added mounting, powering and operating instructions for industrial variant</td>
</tr>
<tr>
<td>2</td>
<td>19</td>
<td>2.7</td>
<td>Added information about IMU orientation with industrial casing</td>
</tr>
</tbody>
</table>
1. Getting Started with the GUI tools.......................................................................................... 11
   1.1. System Requirements ......................................................................................................... 11
   1.2. Download .......................................................................................................................... 11
   1.3. Windows Installation .......................................................................................................... 11
   1.4. Linux Installation .............................................................................................................. 11
   1.5. The ANavS® Wizard Tool .................................................................................................. 12
   1.6. The ANavS® Visualizer Tool ............................................................................................ 14
2. Getting Started with the MS-RTK module.............................................................................. 19
   2.1. General .............................................................................................................................. 19
   2.2. Powering the MS-RTK Module ....................................................................................... 19
   2.3. Connections of the MS-RTK Module .............................................................................. 21
   2.4. The Setup for RTK- and Attitude-Determination ............................................................... 23
   2.5. GNSS Antenna Placement Guideline .............................................................................. 24
   2.6. MS-RTK module Placement Guideline .......................................................................... 25
   2.7. The ANavS®-Wizard to configure the MSRTK Module .................................................... 26
      2.7.1. Wizard Step-1 .............................................................................................................. 26
      2.7.2. Wizard Step-2 .............................................................................................................. 27
      2.7.3. Wizard Step-3 .............................................................................................................. 33
      2.7.4. Wizard Step-4 .............................................................................................................. 34
      2.7.5. Wizard Step-5 .............................................................................................................. 35
      2.7.6. Wizard Step-6 .............................................................................................................. 37
      2.7.7. Wizard Step-7 .............................................................................................................. 38
      2.7.8. Wizard Step-8 .............................................................................................................. 39
   2.8. The ANavS®-Wizard to Update the MS-RTK Module ....................................................... 41
   2.9. The ANavS®-Wizard for Postprocessing recorded sensor raw data .................................. 42
      2.9.1. Wizard Step-1 .............................................................................................................. 42
      2.9.2. Wizard Step-2 .............................................................................................................. 43
      2.9.3. Wizard Step-3 .............................................................................................................. 44
      2.9.4. Wizard Step-4 .............................................................................................................. 46
      2.9.5. Wizard Step-5 .............................................................................................................. 46
      2.9.6. Wizard Step-6 .............................................................................................................. 47
3. Getting Started with the RTK/RTCM Reference Station ............................................................... 48
   3.1. Powering the RTK/RTCM Reference Station ........................................................................ 48
   3.2. The Setup for the RTK/RTCM Reference Station .................................................................. 48
   3.3. The ANavS®-Wizard to configure the RTK/RTCM Reference Station ...................................... 50
       3.3.1. Wizard Step-1 ................................................................................................................ 50
       3.3.2. Wizard Step-2 ................................................................................................................ 51
       3.3.3. Wizard Step-3 ................................................................................................................ 53
       3.3.4. Wizard Step-4 ................................................................................................................ 54
       3.3.5. Wizard Step-5 ................................................................................................................ 55
   3.4. Receiving RTCM-Messages from the RTK/RTCM Reference Station ..................................... 56
       3.4.1. Broadcasting RTCM-Data via NTRIP-Caster hosted by ANavS cloud service ................. 57
       3.4.2. Broadcasting RTCM-Data via local NTRIP-Caster .......................................................... 58
   3.5. The ANavS®-Wizard to Update the RTK/RTCM Reference Station ....................................... 60
4. Getting Started with the Integrated-Sensor-Platform (ISP) .......................................................... 61
   4.1. General .................................................................................................................................. 61
   4.2. The ISP Setup ......................................................................................................................... 62
   4.3. The Basic-Configuration: GNSS, IMU and Odometry Sensor Fusion ..................................... 63
       4.3.1. The ANavS®-Wizard to Update the ISP Basic-Configuration ......................................... 64
   4.4. The Computer-Vision Operation-Modes ............................................................................... 64
       4.4.1. Prerequisites .................................................................................................................. 64
       4.4.2. Setup ............................................................................................................................. 64
       4.4.3. Basic Operation Mode with Data Acquisition ............................................................... 65
       4.4.4. Sensor Data Visualization in the ANavS GUI and in ROS ............................................... 70
       4.4.5. 2D Object Detection and Tracking ................................................................................ 73
       4.4.6. 3D LiDAR Simultaneous Localization and Mapping (SLAM) .......................................... 78
       4.4.7. Known Issues and Troubleshooting .............................................................................. 82
5. The Command Line API Reference Guide ..................................................................................... 85
   5.1. Navigation commands ........................................................................................................... 85
   5.2. System commands ................................................................................................................ 86
   5.3. Role commands ..................................................................................................................... 88
   5.4. CAN commands ..................................................................................................................... 89
5.5. Driver commands ................................................................. 91
5.6. GNSS commands ............................................................... 92
5.7. LTE commands ................................................................. 93
5.8. Network commands .......................................................... 94
5.9. Record commands ............................................................ 95
5.10. Server commands ............................................................ 96
5.11. Time commands .............................................................. 97
6. The ANavS Binary Solution Output Format .................................. 98
   6.1. The Standard Binary Solution Message ................................. 98
      6.1.1. The Payload ............................................................ 98
      6.1.2. The Result-Code ..................................................... 107
   6.2. The Extended Integrity Information Message ......................... 108
      6.2.1. The Payload ............................................................ 108
7. The NMEA Solution Output Format ......................................... 110
   7.1. The NMEA-Format .......................................................... 110
      7.1.1. Sentence Structure .................................................. 110
      7.1.2. Address field .......................................................... 110
      7.1.3. Data fields ............................................................. 111
      7.1.4. Checksum field ....................................................... 111
      7.1.5. Terminating field ..................................................... 111
      7.1.6. Satellite Numbering .................................................. 111
   7.2. Sentence specification ..................................................... 112
      7.2.1. GGA – Global positioning system (GPS) fix data .......... 112
      7.2.2. VTG – Course over ground and ground speed ............... 112
      7.2.3. GSA – GNSS DOP and active satellites ...................... 113
      7.2.4. GSV – GNSS satellites in view .................................... 113
      7.2.5. RMC – Recommended minimum specific GNSS data ....... 114
      7.2.6. ZDA – Time and date ............................................... 115
      7.2.7. PASHR – Attitude Data ............................................. 115
      7.2.8. ROT – Rate of Turn Data ......................................... 116
      7.2.9. THS – True Heading State Data ................................. 116
8. CAN-Interface ........................................................................................................................................ 117
   8.1. CAN Bus Settings ......................................................................................................................... 117
   8.2. CAN-Output: ANavS-Solution .................................................................................................... 118
   8.3. CAN-Input: Dynamic CAN Decoder with .dbc-file .................................................................. 120
       8.3.1. Overview .............................................................................................................................. 120
       8.3.2. Copy the DBC-file ............................................................................................................... 121
       8.3.3. Load the file ......................................................................................................................... 121
       8.3.4. Generate code with DBCC ............................................................................................... 121
       8.3.5. Select signals for sensors ................................................................................................. 121
       8.3.6. Compile generated code .................................................................................................... 123
       8.3.7. Setup CAN hardware filter ............................................................................................... 123
       8.3.8. Starting Sensor Fusion with ANavS Wizard ...................................................................... 124
9. The Network-Time-Protocol (NTP) ................................................................................................. 126
   9.1. Time Policy .................................................................................................................................. 126
   9.2. Time Server .................................................................................................................................. 127
   9.3. Status Information ...................................................................................................................... 127
   9.4. Time-Synchronization with MS-RTK clock ............................................................................... 128
10. ANavS ROS-Ethernet-Adapter (REA) ............................................................................................. 130
    10.1. System requirements and dependencies .............................................................................. 130
    10.2. REA Client ................................................................................................................................ 130
        10.2.1. Prerequisites ...................................................................................................................... 130
        10.2.2. Setup your ROS environment .......................................................................................... 131
        10.2.3. Sensor fusion solution (mode: padsolution2ros) ............................................................. 131
11. ANavS® Solution Decoder Tool .................................................................................................... 136
    11.1. Download ............................................................................................................................... 136
    11.2. Some Hints ............................................................................................................................. 137
    11.3. Hints especially for Linux users ............................................................................................. 137
    11.4. Decode to .csv-format ............................................................................................................. 137
    11.5. Decode to .kml-format ............................................................................................................. 138
12. ANavS® Record Extractor Tool ..................................................................................................... 140
    12.1. Download ............................................................................................................................... 140
1. Getting Started with the GUI tools

The suite of GUI tools contains the so called ANavS® Wizard for configuring the hardware and the ANavS®-Visualizer for showing the real-time sensor fusion solution with all estimated parameters. How to use the programs is explained in the chapters for each ANavS Positioning System.

1.1. System Requirements

The following minimum requirements should be fulfilled:

- **Processor:** Intel®Core™ i5 with 1.60GHz or comparable processors
- **RAM:** Minimum 4 GB
- **System type:** Windows 64-Bit architecture or Ubuntu 18.04
- **Hard-Disk space:** Minimum 500 MB

1.2. Download

Download the latest ANavS® Online Installer, containing the ANavS® Wizard and the ANavS®-Visualizer, for your desired Operating System (Windows 10/11 or Ubuntu 18.04) from the following link:

https://anavs.com/knowledgebase/anavs-app/

1.3. Windows Installation

- Unzip and double click the installer executable file
  ANavS_Wizard_Online_Installer_Win64Bit and follow the installation-wizard step-by-step.
- Once it is finished, the installer displays an option to run the ANavS® Wizard after completion. If you choose this option, the application should start immediately. Otherwise, you must navigate to your previously defined installation-folder and run the Wizard-executable ANavS_Wizard.exe. Besides, it could be important to start the software with admin-rights or to give the software the rights if asked for it.
- The ANavS® Wizard is always proving for new published versions of this tool in the online repository after starting the software.

1.4. Linux Installation

- Unzip the download file
• Launch the executable file ANavS_Wizard_Online_Installer_Unix64Bit via command line. If this does not work, please change the file permissions to be an executable file.
• After starting, follow the installation-wizard step-by-step.
• Once it is finished, the installer displays an option to run the ANavS®-Wizard after completion. If you choose this option, the application should start immediately. Otherwise, you must navigate to your installation-destination and run the Wizard-executable ANavS_Wizard.exe via command line. Besides, it could be important to start the software with admin-rights or to give the software the rights if asked for it.
• The ANavS® Wizard is always proving for new published versions of this tool after starting the software.

1.5. The ANavS® Wizard Tool

The ANavS® Wizard tool connects to the ANavS® Positioning Systems using a TCP/IP data stream connection. This connection allows the ANavS®-Wizard to operate and control a positioning system remotely via Wi-Fi, Ethernet, or LTE.

Defining a TCP/IP connection is simply done by specifying the IP address of the ANavS Positioning System. This can be done by inserting the IP in the settings dialog (very left bottom button, see Figure 1). On default, the user should be connected via the established Wi-Fi Access-Point of the Positioning System with a SSID named with a substring “ANAVS”. The default static IP of the Positioning System is

- 192.168.42.1 or
- 192.168.43.1

The default Ethernet configuration is DHCP. This brings the user the possibility to connect the ANavS Positioning System to a router with a running DHCP-server. Contact your network administrator for more information on how to map or get the IP address to/from a DNS server. A short introduction searching for IP address is also explained in the ANavS knowledge base (https://anavs.com/knowledgebase/using-the-ip-scanning-tool-nmap/)

The status field in the right bottom corner of the ANavS® Wizard shows the status of TCP/IP-connection. The message “TCP-Conn OK” says that the ANavS® Wizard can communicate with the MS-RTK module, RTK Reference Station or ISP. The message “TCP-Conn N/A” says the ANavS®-Wizard is NOT able to communicate with the ANavS Positioning Syst.

HINT: To get the best possible user-experience with the ANavS® Wizard, please change your global Windows/Linux Display settings of the Symbols to 100%. See Figure 2 for this.
Figure 1: Defining the TCP/IP connection for controlling the positioning systems. The button is found in the left bottom corner. The different Modes of the ANavS® Wizard are explained in chapter 2 with the usage of the different positioning systems.

Figure 2: Changing display settings on Windows to 100%
1.6. The ANavS® Visualizer Tool

The ANavS® Visualizer is normally started by the ANavS® Wizard. After pushing the green start-button for triggering the ANavS® sensor fusion within the ANavS® Wizard at the end of a stepwise Positioning-System configuration, the ANavS® Visualizer (Figure 3) pops-up and the sensor fusion starts to run. The ANavS® GUI visualizes all sensor fusion estimates via text, graphs and on a map.

**HINT:** To get the best possible user-experience with the ANavS® Visualizer, please change your global Windows/Linux Display settings of the Symbols to 100%. See Figure 2 for this.

Besides, in case of already running sensor fusion on a positioning system, the user can directly start the ANavS® Visualizer tool. For this, please navigate to the subfolder `/bin` in your installation folder and start the visualizer via double click on the `ANavS_GUI.exe`.

![ANavS® Visualizer with graphs, constellation plot and map](image)

**Figure 3: ANavS® Visualizer with graphs, constellation plot and map**

At startup time of the application, the visualizer tries to connect via default or user-defined IPs with an ANavS® Positioning System. In addition to this Auto-Connect, the user can also manually insert the IP and Port within the Tab “Settings”, shown in Figure 4. Please note, the Port is in the standard configuration always 6001, which is reserved for the proprietary binary protocol, used by the ANavS® Visualizer decoder to gather all the solution and system
information. Changing the port will prevent getting the solution stream into the ANavS® Visualizer in default settings without customer adjustments.
Figure 4: Defining the TCP/IP connection of the Positioning System for data streaming.

Figure 5: Positioning Accuracy in Centimeter-Level (see scale of the map).
Figure 6: Phase-Residuals and many more information about the internal Kalman filter states.

Hints for ANavS® GUI handling:

- A Left-Mouse clicking in the Google-Earth plot zooms-in.
- A Right-Mouse clicking in the Google-Earth plot zooms-out.
- The map through the Google-Earth API is only visible with scale-level 5 meter or higher. Below only a white map with scaling is visible.
- In the Tab “Settings”, one can define the IP-address of an ANavS Positioning System to which the GUI should connect.
- The “Data rate” shows the amount of solution output data streamed to your Laptop/PC. With a successful connection, the data-rate should be always greater than zero.
- The “Relative Position” shows the baseline-vector from RTCM/RTK Reference Station to your Rover.
- The “Last corrections” in the left upper corner shows the time since the last RTK correction data was received. The counter should be under 10 seconds to get best RTK performance. If no counter appears but only “N/A”, there is no valid connection between the ANavS Positioning System and the selected RTK correction source. Please prove again your settings in that case.
Important to get best performance:

- The gray or red light for the Attitude-LED signals that the attitude-solution is not fixed with GNSS phase measurements. That means, there is NO highly accurate attitude information available.
- The gray or red light for the RTK-LED signals that the position-solution is not fixed with GNSS phase measurements. That means, there is NO highly accurate position information available.
- The orange light for Attitude/RTK signals that the sensor fusion solution lost the fixed state and the Kalman filters performs in a float mode.
- The green light for Attitude/RTK signal that the position and/or attitude is fixed with GNSS phase measurements, and a highly accurate solution is available.
- To get best experience with the Positioning-Systems, please start movement only with float (orange signal) or fixed (green signal) mode.
2. Getting Started with the MS-RTK module

2.1. General

This guide is intended for first time Multi-Sensor RTK module (MS-RTK) users and provides an overview of how to handle with the required software, connect to and configure the MSRTK module, and acquire position and attitude solutions. By the end of this guide, you will be able to acquire a fixed RTK and Attitude solution using ANavS® MS-RTK modules. The steps in this guide should be performed outdoors.

![MS-RTK module](image)

**Figure 7:** MS-RTK module in industrial casing with touch panel (left) and 3D-printed casing (right)

**Important:** Setting it up and operation of MSRTK must be performed by experienced persons only

2.2. Mounting the MS-RTK industrial casing variant

To measure the correct movement of the rover, the industrial casing variant must be mounted on the rover according to the axis of the integrated IMU.

To mount the industrial variant, use the four bolt holes as depicted in Figure 8. The following screws are recommended (not included). There are no dedicated torque requirements:

<table>
<thead>
<tr>
<th>Part</th>
<th>Qty.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIN 912 M5</td>
<td>4</td>
</tr>
<tr>
<td>DIN 125 M5</td>
<td>4</td>
</tr>
</tbody>
</table>
2.3. Powering the MS-RTK Module

The MS-RTK module has three different options for power supply which are explained in the following:

Important:

- All options should deliver a minimum current supply of **2.1A@5V (15 Watt)** until hardware-version msrtkv8. From hardware-version msrtkv9 onwards (from 2021 on), all options should deliver minimum current supply of **2.2A@9V (19.8 Watt)**.
- It is possible to use the MS-RTK Module powered with a power bank. Please make sure that you use a power bank with USB-C PD with the minimum power ratings stated above.
- Don’t use the USB-Type-A connector for powering the MS-RTK module as that does not support USB PD and will hence not deliver sufficient power.
Power can be supplied via a standard AC power plug (see Figure 10). Please use the delivered power adapter and the USB-C to USB-C cable. If you want to use your own USB PD supply, make sure that the supply and cable can handle the required power ratings. The device cannot detect when insufficient power is supplied, and outages or bad measurements can happen.

![Figure 10: Power-Supply with AC Power Plug](image)

Power supply is additionally as well via a DC power header possible (see Figure 11). Please take care of the polarity (+/-) and the allowed nominal voltage-range of **12V – 24V**. The absolute maximum voltage range is 9V – 36V. The power source should always deliver a **minimum of 16 W**.

![Figure 11: DC Power-Supply per Wires](image)

When power is simultaneously provided via USB-C and DC-in, the module will drain from both in an unspecified ratio as no priority switching is implemented.

### 2.4. Connections of the MS-RTK Module

The following section describes the different standard connectors and interfaces of the MS-RTK module. Figure 12 shows the antenna-connections with up to 3 GNSS antennas/receivers.
Please connect for a One-Antenna setup to GNSS 1, for Two-Antenna setup to GNSS 1+2 and for a Three-Antenna setup to GNSS 1+2+3.

Figure 12: GNSS Antenna connections

Figure 13 shows the standard connectors of the casing:

- **SMA Connectors for GNSS**: As already described in Figure 12, on the left side of the casing are the connectors for the GNSS antennas. Please take care to connect in the appropriate order (GNSS 1, GNSS 2, GNSS 3).

- **SMA Connector for LTE/Mobile Network**: Please connect the delivered LTE Antenna in case of using the integrated mobile network chip for internet access (for example to receive correction data from your RTK Reference Station).

- **USB-Typ C for Power-Supply**: As already described, the USB-Typ-C connector is used for the Power-Plug or Powerbank for power supply.

- **Camera Interface 1/2**: The interfaces for ANavS® visual odometry sensor (VIS). Camera 1 can for example be used for the forward movement direction and Camera 2 for backward movement direction.¹

- **CAN**: The CAN-2.0A Interface is used for receiving vehicle data for example for wheel-speed information and to output the sensor-fusion solution. More information about adjusting DBC-files can be found in chapter 2.8.1 and chapter 8.

- **USB-Interface**: The standard USB interface can be used for example for data-storage, Wi-Fi-dongle, LTE-dongle, data-transmission, etc.

- **Ethernet**: The ethernet-port can be used for internet/network-connections to stream for example the ANavS sensor fusion solution.

¹ The Camera-Interfaces are only available until hardware version MSRTKv8.
- **Power-Supply per Wires:** As already described in Figure 11, the module can be powered with DC-voltage (Nominal: 12V – 24V, Max: 9V – 36V) via wires.

Special interfaces and pins like UART, FPGA for incremental-decoder, PPS-Signal and GPIOs are not forwarded to the casing and need special customer adjustments.

![Diagram showing standard interfaces lead to the casing]

**Figure 13: Standard interfaces lead to the casing**

### 2.5. The Setup for RTK- and Attitude-Determination

This chapter describes a typical setup of the MS-RTK module on a rover platform (car, robot, UAV, etc.). The heading and pitch of a vehicle can be derived from the orientation of the baseline between a base and a rover antenna when both antennas are attached to the vehicle. The base GNSS-antenna is connected to the first GNSS-receiver configured as RTK antenna. The second and third GNSS antenna is connected to the second and third receiver configured as Attitude antennas. This is illustrated in Figure 14.

With three GNSS-antennas on top of the car/rover the software can calculate the yaw, pitch, and roll angle of the rover in a very accurate way without the need of movement. The RTK correction data is typically streamed from an ANavS® RTK Reference Station or from an external service-provider via Ethernet, LTE or Wi-Fi connection. The RTK-Baseline defines the centimeter-accurate position of your rover.
2.6. GNSS Antenna Placement Guideline

This is a very important step of the setup and has direct impact on the Position- and Attitude performance of the sensor fusion. The placement of the GNSS antennas defines the attitude-baseline, that means the yaw, pitch, and roll angle of your rover. Please mount the antennas in accordance with your use-case and carefully measure the distances between the GNSS antennas (measurement-error <= 1 cm). The configuration of the sensor fusion with these measured distances is done in the ANavS® Wizard and is described in the section 2.8.5.

Please pay attention to the following:

- The minimum distance between two GNSS antennas is 30 cm.
- Please note that the attitude accuracy increases with the distance between the GNSS antennas (sigma=0.25° absolute attitude-accuracy per meter antenna spacing). That means at 2-meter baseline-length a sigma of 0.125° for the heading/pitch/roll.
Please align your antennas in the same way to minimize the impact of antenna phase center offsets. The high-grade antennas have typically an arrow marked on the bottom or rear side of the antenna. The orientation of the low-cost patch antennas is simply defined by the orientation of the cable-connector.

- Please consider and measure also a height-offset between the mounted GNSS antennas. We recommend a placement at the same height level for the beginning as this prevents any sign errors of the differential height and typically enables the best performance.
- For setups with three GNSS antennas (3D-Setup for yaw, pitch, and roll angle determination), please make sure that distance measurements in perpendicular directions are having a 90° angular spacing as otherwise the distance measurements do not fit to the actual geometry. This would prevent any reliable solution.

2.7. MS-RTK module Placement Guideline

The following Figure 15 shows a placement example of the MS-RTK module and GNSS antennas in the body-frame of your rover (top-view).

Please pay attention on the following:
The movement direction of your rover is defining the orientation of the positive x-axis.

The MS-RTK module includes an IMU, which needs to be oriented in a certain manner: The MS-RTK module must be oriented such that the SMA-connector of the LTE/mobile-network is pointing towards the movement direction. With the industrial casing, the connector panel (SMA connectors, USB, Ethernet) is defining the positive x-axis IMU direction.

To get the best performance, it is good practice to mount the MS-RTK module in the center of rotation of your rover.

### 2.8. The ANavS®-Wizard to configure the MSRTK Module

In the previous section, the user became familiar with the hardware and the setup of the MS-RTK module. This section describes the use of the ANavS® Wizard software. To be able to configure and receive data of the module, please connect with the Wi-Fi Access-Point “ANAVS_MSRTK_AP” (Password: anavsrtk) of your switched-on MS-RTK module. Please note that the boot time of the Linux-OS is approximately 1-2 min. As described in chapter 1.5, the default static IP of the MS-RTK module is:

- 192.168.42.1 or
- 192.168.43.1 (on older hardware)

#### 2.8.1. Wizard Step-1

Figure 16 shows Step-1 of the ANavS® Wizard application. Please select the first option “Starting ANavS Sensor Fusion on MSRTK module”.

2.8.2. Wizard Step-2

Figure 17 and Figure 18 shows Step-2 of the ANavS® Wizard application. In this step, the Wizard is proving the communication between the users Laptop/PC and the MS-RTK module. In case the error-dialog as shown in Figure 17 appears, please check again your Wi-Fi/Ethernet connection settings with the button in the bottom left corner (“Settings”).

Figure 18 shows the typical window with correct Wi-Fi/Ethernet settings. One can adjust here the settings for the mobile network module, the Wi-Fi module, the Ethernet module and the CAN-transceiver module (described in chapter 8 in more detail).
Figure 17: Step-2, Wi-Fi/Ethernet connection error message

Figure 18: Step-2, Mobile-Network/Wi-Fi/CAN settings of MS-RTK module
Mobile Network Settings:

ANavS delivers all Positioning Systems with already equipped SIM-cards, which are provider-independent and dial-in into the best available mobile network at users’ location. The region (Europe, North America, Asia, …) is selectable by the customer in the ordering-process. The fee of the SIM-card is part of a service-contract or directly bookable by ANavS. The following settings are needed for the SIMs:

- APN: em
- Dial-Up Number: T*99#

Own SIM-cards can also be used. With the 3D printed casing, the user can exchange the SIM-card without any help and extra cost. With the industrial casing, the user must send the SIM to ANavS before delivering the Positioning System. If this change has to be done afterwards, ANavS reserves the right to charge for the replacement.

In case of using an own SIM-card, please change the settings here accordingly and click afterwards to the button “SEND CONFIG” to transmit the new settings to the MS-RTK module.

In the next step, click “(RE-)START 4G” button. If all provided information for the SIM-card are correct and LTE-reception is available at your position, the field “4G Is Inactive” becomes “4G is Active” and the color turns from red to green.

The configuration is saved to enable an automatic re-connection with the mobile network after each reboot.

Wi-Fi Settings (obsolete):

The left section is showing the current Wi-Fi settings. The right section is for the Wi-Fi configuration. The Wi-Fi-module can be used in four different Modis.

- **Access-Point (AP-STATIC Profile):**
  The standard/default way is the Access-Point (AP) profile. This means that the MS-RTK module is creating its own Wi-Fi-network with the SSID “ANAVS_MSRTK_AP” with password “anavsrtk” and static IP 192.168.42.1. Please use this mode for your first steps to get familiar with the system.

- **Client DHCP-Mode (CLI-DHCP Profile):**
  With this selected profile, the MS-RTK module tries to connect to an existing Wi-Fi network with a running DHCP-server running on it. To give the information to the MSRTK module, please use the “SCAN” Button and select the suitable SSID from the dropdown-list. After this, please define the Password in the dialog-window, activate the “ADD” radio button and click “APPLY NEW CONFIGURATION”. It is possible to define a lot of known Wi-Fi-networks. But take care, you must know the assigned IP-
address to your MSRTK module in your network to be able to communicate with the ANavS® Wizard to the MSRTK module. The new IP must be signed in the message box by clicking on the “Settings”-button in the bottom left corner. A detailed description of how to find the IP-address of your MSRTK module is described in the ANavS knowledge base “Using the IP scanning-tool NMAP” ². Another knowledge base article is explaining how to broadcast RTK-Data from your ANavS RTK reference station directly via Wi-Fi to your MS-RTK module ³.

- **Client Static-Mode (CLI-STATIC Profile):**
  With this selected profile, the MS-RTK module tries to connect to an existing Wi-Fi network without a running DCHP-server running on it. The procedure for configuration is the same as for the CLI-DHCP profile.

- **Powered-Off Wi-Fi module (OFF Profile):**
  Use this mode to save power consumption or to reduce traffic in the Wi-Fi frequency range. **BUT NOT RECOMMENDED**, as it increases the risk of a lockout

**Ethernet Settings:**

The left section is showing the current Ethernet settings. The right section is for the Ethernet configuration. The Ethernet module can be used in three different Modis.

- **DHCP-Mode (DHCP Profile):**
  With this default setting, the Ethernet port is waiting to get an IP address of a DHCP server (e.g., from a router). After connecting the MS-RTK module via Ethernet cable with a router/DHCP-server, you can identify the MSRTK modules IP-address via the Ethernet information section in the Wizard window, the dashboard of your router or scanning the network as explained in the knowledge base article “Using the IP scanning-tool NMAP” ².

- **Static-Mode (STATIC Profile):**
  Another option is to set a static IP for the MSRTK module and directly connect it with another device or laptop with same static IP range address.

- **Powered-Off Ethernet port (OFF Profile):**
  Use this mode to save power consumption.

**CAN Settings:**

---


CAN is a robust and widely spread data bus standard which is among other applications also used in almost all modern vehicles for communication between the various controllers in it.

The MS-RTK Module comes with a CAN Interface that allows the user to input wheel odometry values and/or to access the solution data over that interface. The CAN-2.0A Interface can be fully configured in the ANavS Wizard (Figure 19). The settings are hereby divided in two categories: the bus settings and the solution output (CAN-Output).

**CAN Bus Settings:**

- **Bus-Speed:** Select the Bit rate of the CAN bus. By default, 500 Kbit/s are selected.
- **Termination enabled:** CAN requires a 120 Ohm hardware termination at the last transceiver. The MS-RTK System has a built in switchable 120 Ohm termination which can be enabled by using this slider.
- **Bus mode:** The normal mode is the default mode in which the CAN Interface is fully operating. In the listen-only mode, the CAN-Controller does not transmit any data and does not acknowledge the received CAN-frames. This mode can be used to monitor the data bus without participate or interfere with the traffic on the bus. In the listen-only mode it is not possible to output solution information.
- **Output enabled:** This function is intended as an extra security layer to avoid unintentional writing on the CAN bus. It must be enabled to use the solution output functionality of the CAN Interface. Otherwise, it is recommended to switch it off.

**Solution Output via CAN:**

The CAN Interface can be used to receive the output of the position and attitude solution. The CAN Interface sends for each solution variable a unique message containing the value. The CAN address corresponds to the ID of the variable. The variables are bundled in groups which can be activated or deactivated by checking the corresponding boxes. As standard, each packet has a unique ID which is also shown in the table below. To change the IDs of the CAN messages (e.g., to avoid collisions with the information of other devices on the bus), there is the possibility to add an offset to all messages. To use the solution output via CAN, the bus must be configured in the normal mode and the output must be enabled (see also above section “CAN Bus settings”).

<table>
<thead>
<tr>
<th>ID</th>
<th>Format</th>
<th>Name</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>uint16</td>
<td>resCode</td>
<td>–</td>
<td>Result code bitfield, which keeps the system status and information.</td>
</tr>
<tr>
<td>2</td>
<td>uint16</td>
<td>week</td>
<td>–</td>
<td>Week number of the current epoch.</td>
</tr>
<tr>
<td>3</td>
<td>double</td>
<td>tow</td>
<td>s</td>
<td>Time of Week of the current epoch.</td>
</tr>
<tr>
<td>4</td>
<td>uint16</td>
<td>weekInit</td>
<td>–</td>
<td>Week number of the epoch when the system was started.</td>
</tr>
<tr>
<td>Index</td>
<td>Type</td>
<td>Field</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>double</td>
<td>towInit</td>
<td>Time of Week of the epoch when the system was started.</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>double</td>
<td>lat</td>
<td>deg Latitude.</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>double</td>
<td>lon</td>
<td>deg Longitude.</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>double</td>
<td>height</td>
<td>m Height</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>double</td>
<td>ECEF-X</td>
<td>m X-position in ECEF-coordinate frame</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>double</td>
<td>ECEF-Y</td>
<td>m Y-position in ECEF-coordinate frame</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>double</td>
<td>ECEF-Z</td>
<td>m Z-position in ECEF-coordinate frame</td>
<td></td>
</tr>
<tr>
<td>13–15</td>
<td>3*double</td>
<td>b</td>
<td>m Baseline in NED frame spanned by the position given by lat, lon and height, and by the position of the reference station.</td>
<td></td>
</tr>
<tr>
<td>16–18</td>
<td>3*double</td>
<td>bStdDev</td>
<td>m Standard deviation of the baseline.</td>
<td></td>
</tr>
<tr>
<td>19–21</td>
<td>3*double</td>
<td>vel</td>
<td>m/s Velocity in NED frame.</td>
<td></td>
</tr>
<tr>
<td>22–24</td>
<td>3*double</td>
<td>velStdDev</td>
<td>m/s Standard deviation of the velocity.</td>
<td></td>
</tr>
<tr>
<td>25–27</td>
<td>3*double</td>
<td>acc</td>
<td>m/s² Acceleration in body frame.</td>
<td></td>
</tr>
<tr>
<td>28–30</td>
<td>3*double</td>
<td>accStdDev</td>
<td>m/s² Standard deviation of the acceleration.</td>
<td></td>
</tr>
<tr>
<td>31–33</td>
<td>3*double</td>
<td>att</td>
<td>deg Attitude/Euler angles (heading, pitch, roll).</td>
<td></td>
</tr>
<tr>
<td>34–36</td>
<td>3*double</td>
<td>attStdDev</td>
<td>deg Standard deviation of the attitude.</td>
<td></td>
</tr>
<tr>
<td>39–43</td>
<td>5*double</td>
<td>timing-Info</td>
<td>s CPU-Load of used sensors. The sum (green line in the GUI) of elapsed time should not over 1 second (→ to slow processor). First double: Elapsed time GNSS; Second double: Elapsed time IMU; Third double: Elapsed time Baro; Fourth double: Elapsed time Odometry; Fifth double: Reserved</td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>double</td>
<td>latency-info</td>
<td>s Overall end-to-end positioning latency</td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>double</td>
<td>gnssReception</td>
<td>– Scalar, which indicates the GNSS signal reception. The value is between 0 and 20, where 20 is the best, i.e. very good conditions.</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>uint8</td>
<td>numSats</td>
<td>– Number of satellites.</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: CAN-Bus output messages
Odometry input via CAN:

The ANavS sensor fusion can handle different forms of vehicle odometry. Up to 5 variables can therefore be defined. These are either one, two or four independent wheel speeds, and the steering angle or another heading information of the vehicle in the vehicle frame. The algorithms inside of the sensor fusion software calculate from the different inputs the current velocity and heading information in the body frame of the vehicle for the positioning algorithm.

To adjust the CAN-Interface to user specific CAN messages, a dynamic CAN decoder is part of the MS-RTK module as command-line tool. The Dynamic CAN Decoder allows you to use the MS-RTK system with CAN signals from your .DBC file without sharing the file with third parties or ANavS. The DBC-generation on the MS-RTK module is described in detail in chapter 8.

2.8.3. Wizard Step-3

The Figure 20 shows Step 3 of the ANavS® Wizard application. The user needs to define the number of used GNSS antennas mounted on the rover respectively screwed on the MSRTK module.
The impact of user’s selection:

**1-Dimensional:** The attitude is estimated with movement and coupled with the IMU-gyroscope and accelerometer (one GNSS antenna connected). With this setup, you need some rover dynamic (curves, circles) to get an accurate attitude information of your rover.

**2-Dimensional:** The attitude (yaw and pitch angle) is already determined without movement using fixed carrier-phase GNSS measurements and coupled with the IMU-gyroscope (two GNSS antennas connected).

**3-Dimensional:** The attitude (yaw, pitch, and roll angles) is already determined without movement using carrier-phase GNSS measurements and coupled with the IMU-gyroscopes (three GNSS antennas connected).

Figure 20: Step-3, Select Attitude-Dimensionality

2.8.4. Wizard Step-4

The next step proves the GNSS antenna connection and if the count of founded GNSS antennas match with the previous selected Attitude dimensionality (Step-3). The user can only go forwards if it matches in a proper way. To pass this step, the GNSS antennas need to be
correctly connected and set up in an environment where GNSS signals can be received (not indoor).

The following circumstances would prevent a successful GNSS-antenna connection:

- **Failure in Antenna-Status**: The positioning system need only some more boot-time for the OS and some time for gather satellite information (30 – 90sec).
- **Failure in Antenna-Status**: If the antenna-Status problem stays the same after some retries, too long antenna cables could lead to a missing recognition of the antenna from the GNSS-receiver side. The text field “Antenna-Status” shows **ERROR** or **SHORT** in that case.
- **No Data received/ No Satellites visible**: Please check this step with GNSS antennas outdoor.

![Figure 21: Step-4, Proving Antenna-Connections of MS-RTK module](image)

2.8.5. **Wizard Step-5**

The following Figure 22 shows the placement input of the users GNSS antennas in the body-frame of your rover (top-view). Each Text field needs input, before proceeding with the next steps.
Please pay attention on the following:

- The x-direction shown in graph/coordinate-frame defines the movement-direction of your rover
- Please measure the distances as accurate as possible. **The measurement-error should be below 1 cm**
- Defining the z-Axis corresponds to the right-hand-rule. The thumb shows in movement-direction
- For setups with three GNSS Antennas, please use a right-angle measurement-tool

![Figure 22: Step-5, GNSS-Antenna Dimension in rover body-frame](image)

In addition, the lever arms of the other sensors can also be configured in this step. Please open the corresponding window “Lever arm” shown in the bottom of this step. Figure 23 shows the input-options.
2.8.6. Wizard Step-6

Step 6 is used for defining the input-stream for RTK correction data. The source could be your own ANavS RTK Reference Station, third-party reference stations or external service provider like SAPOS or Axio-Net. The nomenclature for a suitable RTK input-format notation is given in the Wizard-Window. To facilitate the user input, the last three streams are cached in a selectable history.
2.8.7. Wizard Step-7

The ANavS® Sensor Fusion Framework is customizable with some specific a priori information.

- For using a priori information about the reference station, an input-option is given in this step. The user can fill the dimension in the NED (North-East-Down)-Frame format for using this mask. Additionally, the accuracy of the measurement can be set and needs to be activated with the appropriate button. **This input is NOT mandatory:** Use this input only if the information is exactly known and the user is introduced in the NED-frame and how one can set this parameter in a correct manner.
- The additional setting “**Up-Velocity**” is for constraining the velocity mainly in the horizontal plane. It’s recommended to activate this option for applications like automotive or robotics. Please deactivate this option for UAV-applications or something similar.
- The additional setting “**Heading<->Velocity**” is for constraining the velocity mainly to the heading direction. It’s recommended to activate this option for applications like
automotive or robotics. Please deactivate this option for UAV-applications or something similar.

![Image of reference guide page](image_url)

**Figure 25: Step-7, Settings for a priori information**

### 2.8.8. Wizard Step-8

In the last step of the Wizard, the user can start the ANavS® sensor fusion with the green start-button. Before starting, please check further settings:

- The checkbox “Enable Save Solution File” activates recording the Realtime sensor fusion solution in a file. The format of the file depends on the selected type of solution output. Furthermore, the user gets a converted KML-file, which gives the user the opportunity to see the most interesting information (position, attitude, velocity, and acceleration) in a Google-Earth plot. The files are saved in `<UserPath>/Documents/AnavsWizardAppData/output_data`. With the button “Output” on the bottom of the Wizard, the user can directly step to the generated output-folders. Each recorded dataset is saved in one own folder with the naming “YYYYMMDD_HHMM_UTC”.

- The checkbox “Enable Save Raw Data Files” activates recording of all processed sensor data (GNSS, IMU, Barometer, Odometry, Visual-Odometry, …) with highly
accurate timestamps to use the data for postprocessing. The files are saved in
\texttt{<UserPath>/Documents/AnavsWizardAppData/output\_data}.
With the button “Output” on the bottom of the Wizard, the user can directly step to
the generated output-folders. Each recorded dataset is saved in one own folder with
the naming “\texttt{YYYYMMDD\_HHMM\_UTC}”.

- The checkbox “Enable Odometry (via USB)” is used for enabling the Odometry-input
  via USB-Typ-A. Only with activated checkbox, the tightly coupled sensor fusion is
  using this input sensor data.

- The checkbox “Enable Odometry (via CAN)” is used for enabling the Odometry-Input
  via CAN. Only with activated checkbox, the tightly coupled sensor fusion is using this
  input sensor data.

- The checkbox “Enable Autostart of Navigation-Service” effects an automated start
  of the sensor fusion after switching-on the MS-RTK module with the last user-
  configuration. To see the solution in the ANavS® Visualizer, start the program
  \texttt{ANavS\_GUI.exe} in the folder \texttt{<Installation-Path>/ANavS\_Wizard/bin}.

- The checkbox “Enable Low-Latency-Mode” can be used for very time-critical
  applications. The software limits the used GNSS-signals and pushes the solution
  output in a strict manner.

- The checkbox “Enable Dynamic Attitude Fix” forces an Attitude-Fix with GNSS phase
  measurements not only in stand-still mode but also in movement. Applications with
  less stand-still phases should activate the checkbox.

- The option “Type of Solution-Output” is explained in chapters about \texttt{ANavS® Binary
  Solution Output Format} and \texttt{NMEA Solution Output Format} in XX and XX.

- The option “Customer-Code” is only for experts with appropriate support-level. The
  default-value is 0.

- The option “Output-Rate” is selectable with Low (5Hz), Medium(55-65Hz), and
  High(105-125Hz). The exact rate depends on the used sensors within the tightly
  coupled sensor fusion.

After pushing the green start-button for triggering the ANavS® sensor fusion on the MS-RTK
module, the settings are stored on the module, the ANavS® Visualizer pops-up and the sensor
fusion navigation service starts to run.

\textbf{HINT:} In case of an already running Navigation-Mode, the Wizard and Visualizer is only
attaching to this process without restarting the Navigation-Service. Please stop and start again
the Navigation-Service to perform with the newest settings.
2.9. The ANavS®-Wizard to Update the MS-RTK Module

The ANavS® Wizard can also be used for updating the MS-RTK module. Trying this, the user must provide internet access to the MS-RTK module. The default way would be through the mobile network or through Ethernet with a connection to a router. The first step in the Wizard is selecting the option “Update your ANavS-Module (…)” on the starting page.

The next window is showed in Figure 27. The user has two options with the **Online-Updater**. Selecting the button “Stable Update”, the user updates the MS-RTK (and also the RTK Reference Station) with the latest stable version on the ANavS repository. It is well tested but is not including all recent minor updates and sensor fusion improvements. Selecting the button “Beta Update”, the user updates the MS-RTK (and also the RTK Reference Station) with the latest Beta version on the ANavS repository. It is less tested but is including all recent minor updates and sensor fusion improvements.

In case of problems with getting internet access to the MS-RTK module, ANavS is also providing an **Offline-Updater**. Using this option, please contact the support to get the recent
Update-Zip, load it in the Wizard (do NOT unzip this file) with the SEARCH button and click the update-button.

![Image](image-url)

**Figure 27: Update of the MS-RTK module with the ANavS Wizard**

### 2.10. The ANavS®-Wizard for Postprocessing recorded sensor raw data

The ANavS® Wizard provides a powerful **tightly coupled Postprocessing Engine** for recorded datasets of the MS-RTK modules. The following section describes the subsequent steps.

#### 2.10.1. Wizard Step-1

Figure 28 shows Step-1 of the ANavS® Wizard postprocessing application. Please select the fourth option “Postprocess ANavS sensor raw data”.
2.10.2. Wizard Step-2

Figure 29 shows Step-2 of the ANavS® Wizard postprocessing application. The user needs to define the number of used GNSS antennas mounted on the rover respectively screwed on the MSRTK module for the recorded dataset.

The impact of user’s selection:

**1-Dimensional:** The attitude is estimated with movement and coupled with the IMU-gyroscope and accelerometer (one GNSS antenna connected). With this setup, you need some rover dynamic (curves, circles) to get an accurate attitude information of your rover.

**2-Dimensional:** The attitude (yaw and pitch angle) is already determined without movement using fixed carrier-phase GNSS measurements and coupled with the IMU-gyroscope (two GNSS antennas connected).
3-Dimensional: The attitude (yaw, pitch, and roll angles) is already determined without movement using carrier-phase GNSS measurements and coupled with the IMU-gyroscopes (three GNSS antennas connected).

Figure 29: Step-3, Select Attitude-Dimensionality

2.10.3. Wizard Step-3

Figure 30 shows Step-3 of the ANavS® Wizard postprocessing application. The user needs to define all the recorded raw data files and the proper data format.

The names of the corresponding raw data files have typically the following naming and format:

<table>
<thead>
<tr>
<th>Datastreams</th>
<th>Files and Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>GNSS receiver 1, 2 and 3</td>
<td>Files: LOGrover0.ubx (including also the IMU data), LOGrover1.ubx, LOGrover2.ubx</td>
</tr>
<tr>
<td></td>
<td>Format:</td>
</tr>
</tbody>
</table>
The format depends on the purchased MS-RTK System:
- The triple frequency system has for LOGrover0.ubx and LOGrover2.ubx the U-Blox Format and for LOGrover1.ubx the Septentrio Format.
- The dual frequency system has for LOGrover0.ubx, LOGrover1.ubx and LOGrover2.ubx the U-Blox Format.

<table>
<thead>
<tr>
<th>Odometry data</th>
<th>File: LOGcan.ubx</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTK corrections</td>
<td>File: LOGvrs.osr</td>
</tr>
</tbody>
</table>

Format:
The standard format with recorded datasets with the MS-RTK system is RTCM 3.X.
Using afterwards downloaded RINEX-Obs file is also possible to be used in the postprocessing engine.
2.10.4. Wizard Step-4

Step 4 needs the placement input of the users GNSS antennas in the body-frame of your rover (top-view). Each Text field needs input, before proceeding with the next steps. Please refer to the section 2.8.5 for a detailed description.

2.10.5. Wizard Step-5

The ANavS® Sensor Fusion Framework is customizable with some specific a priori information. Please refer to the section 2.8.7 for a detailed description.
2.10.6. Wizard Step-6

In the last step of the Wizard, the user can start the ANavS® sensor fusion with the green start-button. Please refer to the section 2.8.8 for a detailed description.
3. Getting Started with the RTK/RTCM Reference Station

This guide is intended for first time RTK/RTCM Reference Station users and provides an overview of how to handle with the required software, connect to and configure the Reference Station.

Figure 31: RTK/RTCM Reference Station in industrial casing with touch panel

3.1. Powering the RTK/RTCM Reference Station

The RTK/RTCM Reference Station can be powered with standard 230V AC-Voltage plug or with standard USB Type-C (e.g., USB Powerbank with PD (Power-Delivery)).

3.2. The Setup for the RTK/RTCM Reference Station

Figure 32 shows the standard connections for the typical RTK/RTCM Reference Station setup:

- Please connect the GNSS antenna with the marked SMA connector to the Reference Station. The GNSS antenna must stay outside with open-sky environment (see Figure 33).
- Please connect the Wi-Fi antenna with the marked SMA connector to the Reference Station.
- Provide internet-access to the Reference Station for transmitting RTCM 3.2 messages. For this, please connect the station to a router or screw on the mobile network antenna.
Recommendation: The GNSS antenna should be fixed mounted on position outside with best possible satellite visibility (see Figure 33). A recommended place would be the roof of a high building or similar places. The RTK correction data of the Reference Station are highly accurate for rovers within 15 km distance from rover (MS-RTK module) to Reference Station. A common way is to fix the Reference Station on a location where the user can calibrate the position once a time and don’t change the position again.

Figure 32: RTK/RTCM Reference Station connections

Figure 33: Reference Station placement guideline
3.3. The ANavS®-Wizard to configure the RTK/RTCM Reference Station

In the previous section, you became familiar with the hardware and the setup. This section describes the use of the ANavS® Wizard software. To be able to configure and receive data of the module, please connect with the Wi-Fi Access-Point “ANAVS_RTCM_AP” (Password: anavsrtk) of your switched-on Reference Station. Please note that the boot time of the Linux-OS is approximately 1-2 min. As described in chapter 1.5, the default static IP of the Reference Station is:

- 192.168.42.1 or
- 192.168.43.1 (on older hardware)

3.3.1. Wizard Step-1

For a precise absolute/relative position solution the Reference Station needs a one-time calibration before using it for the new position. Figure 34 shows Step-1 of the ANavS® Wizard application. Please select the second option “Calibrate ANavS RTCM Basestation”.

![Figure 34: Step-1, Select the desired modus "Calibrate ANavS RTCM Base Station"](attachment:figure34.png)
3.3.2. Wizard Step-2

Figure 35 shows Step-2 of the ANavS® Wizard application. In this step, the Wizard is proving the communication between the users Laptop/PC and the MS-RTK module. If the error-dialog as shown in Figure 17 appears, please check again your Wi-Fi/Ethernet connection settings with the button in the bottom left corner (“Settings”).

On this settings-window one can adjust the settings for the mobile network module, the Wi-Fi module, and the Ethernet module.

Mobile Network Settings:

ANavS delivers all Positioning Systems with already equipped SIM-cards, which are provider-independent and dial-in into the best available mobile network at users’ location. The region (Europe, North America, Asia, ...) is selectable by the customer. The fee of the SIM-cards is part of a service-contract or directly bookable with ANavS. The following settings are needed for the SIMs:

- **APN:** em
- **Dial-Up Number:** T*99#
Own SIM-cards can also be used. With the 3D printed casing, the user can exchange the SIM-card without any help and extra cost. With the industrial casing, the user must send the SIM card to ANavS before delivering the Positioning System. If it has to be done afterwards, ANavS reserves the right to charge for the replacement.

In case of using an own SIM-card, please change the settings here accordingly and click afterwards to the button “SEND CONFIG” to transmit the new settings to the MS-RTK module. In the next step, click “(RE-)START 4G” button. If all provided information for the SIM-card are correct and LTE-reception is available at your position, the field “4G is Inactive” becomes “4G is Active” and the color turns from red to green.

The configuration is saved to enable an automatic re-connection with the mobile network after each reboot.

**Wi-Fi Settings (obsolete):**

The left section is showing the current Wi-Fi settings. The right section is for the Wi-Fi configuration. The Wi-Fi-module can be used in four different Modis.

- **Access-Point (AP-STATIC Profile):**
  The standard/default way is the Access-Point (AP) profile. This means that the MS-RTK module is creating its own Wi-Fi-network with the SSID “ANAVS_MSRTK_AP” with password “anavsrtk” and static IP 192.168.42.1. Please use this mode for your first steps to get familiar with the system.

- **Client DHCP-Mode (CLI-DHCP Profile):**
  With this selected profile, the MS-RTK module tries to connect to an existing Wi-Fi network with a running DHCP-server running on it. To give the information to the MSRTK module, please use the “SCAN” Button and select the suitable SSID from the dropdown-list. After this, please define the Password in the dialog-window, activate the “ADD” radio button and click “APPLY NEW CONFIGURATION”. It is possible to define a lot of known Wi-Fi-networks. But take care, you must know the assigned IP-address to your MSRTK module in your network to be able to communicate with the ANavS® Wizard to the MSRTK module. The new IP must be signed in the message box by clicking on the “Settings”-button in the bottom left corner. A detailed description of how to find the IP-address of your MSRTK module is described in the ANavS knowledge base “Using the IP scanning-tool NMAP” 4. Another knowledge base

---

article is explaining how to broadcast RTK-Data from your ANavS RTK reference station directly via Wi-Fi to your MS-RTK module.

- **Client Static-Mode (CLI-STATIC Profile):**
  With this selected profile, the MS-RTK module tries to connect to an existing Wi-Fi network without a running DCHP-server running on it. The procedure for configuration is the same as for the CLI-DHCP profile.

- **Powered-Off Wi-Fi module (OFF Profile):**
  Use this mode to save power consumption or to reduce traffic in the Wi-Fi frequency range. BUT NOT RECOMMENDED.

**Ethernet Settings:**

The left section is showing the current Ethernet settings. The right section is for the Ethernet configuration. The Ethernet module can be used in three different Modis.

- **DHCP-Mode (DHCP Profile):**
  With this default setting, the Ethernet port is waiting to get an IP address of a DHCP server (e.g., from a router). After connecting the MS-RTK module via Ethernet cable with a router/DHCP-server, you can identify the MSRTK modules IP-address via the Ethernet information section in the Wizard window, the dashboard of your router or scanning the network as explained in the knowledge base article “Using the IP scanning-tool NMAP”.

- **Static-Mode (STATIC Profile):**
  Another option is to set a static IP for the MSRTK module and directly connect it with another device or laptop with same static IP range address.

- **Powered-Off Ethernet port (OFF Profile):**
  Use this mode to save power consumption. BUT NOT RECOMMENDED.

**3.3.3. Wizard Step-3**

The next step proves the GNSS antenna connection. The user can only go forwards if it matches in a proper way. To pass this step, the GNSS antennas need to be correctly connected and set up in an environment where GNSS signals can be received (not indoor).

The following circumstances would prevent a successful GNSS-antenna connection:

- **Failure in Antenna-Status:** The positioning system need only some more boot-time for the OS and some time for gather satellite information (30 – 90sec).

---

5 Broadcasting RTCM-Data for MSRTK Modules: [https://anavs.com/knowledgebase/publishing-rtcm-data-for-msrtk-modules/](https://anavs.com/knowledgebase/publishing-rtcm-data-for-msrtk-modules/)
- **Failure in Antenna-Status**: If the antenna-Status problem stays the same after some retries, too long antenna cables could lead to a missing recognition of the antenna from the GNSS-receiver side. The text field “Antenna-Status” shows **ERROR** or **SHORT** in that case.
- **No Data received/ No Satellites visible**: Please check this step with GNSS antennas outdoor.

![Proving Data-Transmission](image)

Figure 36: Step-3, Proving GNSS antenna connection

### 3.3.4. Wizard Step-4

The user has two options for calibrating the stationary position of the RTK/RTCM Reference Station:

- **External RTK correction data**: Use this option to calibrate your Reference Station with centimeter-accurate position. To do this, the user needs an external RTK correction input-stream from an appropriate correction-data service-provider with centimeter accuracy (e.g., SAPOS, Axio-Net). Depending on your service-level, ANavS® can do this also for you. The advantage of this mode is a precise relative and absolute position estimation of your rover equipped with MS-RTK module and its
tightly coupled ANavS® sensor fusion framework.

**NOTE:** To stream the correction data from external service provider, your RTK Reference Station needs internet access (e.g., through the integrated mobile network module or ethernet connection to a router).

- **Filtered Least-Squares Position-Solution:** This is the standard way to calibrate a rough position estimation for the Reference Station. The user can do this without the need of additional services. It has no influence on the accuracy of the relative position-solution between rover (MS-RTK module) and Reference Station, but the absolute position includes a position-offset depending on the accuracy of the position of the RTCM base station.

![Figure 37: Step-4, select the mode for RTK/RTCM Reference Station position calibration](image)

**3.3.5. Wizard Step-5**

Figure 38 shows Step-5 of the ANavS® Wizard for Reference Station calibration. Hereby, the user can define the error-bound which must be reached and the time for averaging after being below this defined error-bound. After attaining the limits, the calibration stops, and the position is stored permanently on the Reference Station (also after reboot of the system).
3.4. Receiving RTCM-Messages from the RTK/RTCM Reference Station

The Reference Station provides an NTRIP-Server which converts the GNSS data into the standardized RTCM 3.2 Format and connects either to a local NTRIP-caster (on the Reference Station itself) or to a cloud-based NTRIP-caster, hosted by ANavS (without fee). To receive this RTK correction data from your Reference Station one could use the own local network or the ANavS cloud-based correction data service via internet-connection.

An example of a defined RTK correction data stream with the ANavS cloud-based correction data service is:

Username: RTCM_User  
Password: anavs123  
IP/URL: customer-specific; info is included in the system delivery  
Port: customer-specific; info is included in the system delivery  
Mountpoint: ANAVS

This results in the following NTRIP input stream for your MS-RTK module or ISP:

RTCM_User:anavs123@<customer-specific-IP>:<customer-specific-Port>/ANAVS
The default setting of the Reference Station is using the cloud-based NTRIP-Caster, which ensures the usability independent of any company’s network policy (e.g., port-forwarding). The local NTRIP-caster setup is activatable via the global settings file, described in the following subsections.

The used RTCM 3.2 messages are the following:

<table>
<thead>
<tr>
<th>Message-Type</th>
<th>RTCM 3.2</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1005</td>
<td>10s</td>
<td>ARP-Station coordinates, ECEF XYZ</td>
</tr>
<tr>
<td>1008</td>
<td>10s</td>
<td>Antenna-Type of the RTK Reference Station</td>
</tr>
<tr>
<td>1033</td>
<td>10s</td>
<td>Receiver- und Antenna-Type</td>
</tr>
<tr>
<td>1077</td>
<td>1s</td>
<td>GPS Observation data (MSM7)</td>
</tr>
<tr>
<td>1085</td>
<td>1s</td>
<td>GLONASS Observation data (MSM7)</td>
</tr>
<tr>
<td>1095</td>
<td>1s</td>
<td>Galileo Observation data (MSM7)</td>
</tr>
<tr>
<td>1125</td>
<td>1s</td>
<td>BeiDou Observation data (MSM7)</td>
</tr>
<tr>
<td>1230</td>
<td>10s</td>
<td>GLONASS L1 and L2 Code-Phase Biases</td>
</tr>
</tbody>
</table>

Other RTCM messages or other time-intervals are adjustable via the global configuration file `<home/user>/device/settings` on the Reference Station.

The Range of the ANavS Reference Stations are only limited by (mobile) network connection and the useful distance of RTK corrections from Reference Station to the rover (MS-RTK module or ISP), which is ~20 km.

3.4.1. Broadcasting RTCM-Data via NTRIP-Caster hosted by ANavS cloud service

It’s the most comfortable way to broadcast and reach correction data over the internet (Ethernet, Wi-Fi, or mobile network) and the recommendation to bring the users RTK setup-up working. The RTK correction data is transmitted over internet-access to the ANavS cloud service with a running NTRIP-Caster and broadcasted on a specific IP-address and port-number.

The individual information, IP-address and port-number, is included in the delivered package and defined in the global customer settings file `/home/pi/device/settings` as follows:

---

6 The Radio Technical Commission for Maritime Services (RTCM) is a US organization. Among other things, it pursues the goal of realizing internationally standardized data formats for the transmission of corrections for GNSS applications and making correction data available in real time.
RTCM.caster.host=<customer-specific-IP-Address>
RTCM.port=<customer-specific-Port>

To edit this file, please connect per SSH to the Reference Station. For the SSH-credentials, please ask the support team.

3.4.2. Broadcasting RTCM-Data via local NTRIP-Caster

The following description shows broadcasting RTK correction data in a local network or via internet-access via a local NTRIP caster, running on the Reference Station. For receiving the RTK correction data, which are mandatory for RTK-Positioning, the RTCM station must be reachable via TCP/IP for the MS-RTK module or ISP.

Step 1:
Comment out (#) the following line in the global settings file /home/pi/device/settings:
#RTCM.caster.host  =<customer-specific-IP-Address>
#RTCM.port  =<customer-specific-Port>

To edit this file, please connect per SSH to the Reference Station. For the SSH-credentials, please ask the support.

Step 2:
Connect the Reference Station with a router via ethernet cable or Wi-Fi. A DHCP-server running on the router is mandatory. To get correction data outside or inside of this network, the user must follow the next mandatory steps.

Step 3:
First of all, the user needs the IP address of the Reference Station in the local network of the router. The user has the following options: either discover the IP address by connecting to the router (usually by typing http://192.168.1.1 or http://192.168.0.1 in the web browser), then look for a device named “ANavS XXX”, or you use network discover software like Nmap. More details on this can be found in the article Using the IP scanning-tool NMAP.

Step 4: In case of using the Reference Station from outside of the local network, the IP-address of the router is needed. The easiest way to do this is to call the page https://www.outsideopen.com/ip/ within user’s local network. The shown numbers are your public IP address. This public IP address is managed from the users Internet Service Providers. Some providers do not provide with a static IP address, but rather change it usually in a 24h time window. If this is the case, the user will need to set up a Dynamic DNS service that keeps track of the external IP changes. Please refer to your network administrator to know the policy regarding the external IP address.
Step 5: Assuming to have an external IP address set up correctly, the user must configure the router to enable port forwarding. Port forwarding maps requests coming from outside the network at a specific port number to a local IP address from within the local network at a second port number. Therefore, in other words, it redirects the traffic from outside to a specific device inside the local network.

The following example shows the rerouting of correction-data of the Reference Station to an arbitrary Port of the router:

<table>
<thead>
<tr>
<th>External IP address of the router</th>
<th>external port No.</th>
<th>Local IP address of the ANavS RTCM Station</th>
<th>Internal port No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>82.135.2.37</td>
<td>2101</td>
<td>192.168.1.101</td>
<td>2101</td>
</tr>
</tbody>
</table>

In this example, the user configures the port forwarding to reroute the traffic coming from outside (landing so at the external IP of the router, namely 82.135.2.37) at port 2101 to the local IP of the Reference Station (192.168.1.101) at port 2101. Note that choosing external port 2101 is arbitrary (is always best practice not to choose a port that is a well known port). Having the router’s port forwarding set up as in the above example, to connect to the Reference Station from outside the local network just configure the RTCM stream on the MS-RTK module or ISP with IP 82.135.2.37 and with port 2101. The traffic will be automatically rerouted to 192.168.1.101 at port 2101 in your local network.

**IMPORTANT**
The internal port number is not changeable and always 2101.

In case of a local network without port-forwarding, this results in the following NTRIP input stream for the MS-RTK module or ISP:

RTCM_User: anavs123@<Local-IP-of-RTCM-station>:2101/ANAVS

In case of port forwarding, the NTRIP input stream for the MS-RTK module or ISP needs the external port number as configured in the router-settings and the external IP-Address of the router. Following an example for this:

RTCM_User: anavs123@<External-IP-address-of-the-router>:<External-Port-No>/ANAVS
3.5. The ANavS®-Wizard to Update the RTK/RTCM Reference Station

The ANavS® Wizard can also be used for updating the Reference Station. Trying this, the user must provide internet access to the station. The default way would be through the mobile network or through Ethernet with a connection to a router. The first step in the Wizard is selecting the option “Update your ANavS-Module (…)” on the starting page.

The next window is showed in Figure 39. The user has two options with the Online-Updater. Selecting the button “Stable Update”, the user updates the Reference Station (and also the MS-RTK module) with the latest stable version on the ANavS repository. It is well tested but is not including all recent minor updates and sensor fusion improvements. Selecting the button “Beta Update”, the user updates the Reference Station (and also the MS-RTK module) with the latest Beta version on the ANavS repository. It is less tested but is including all recent minor updates and sensor fusion improvements.

In case of problems with getting internet access to the Reference Station, ANavS is also providing an Offline-Updater. Using this option, please contact the support to get the recent Update-Zip, load it in the Wizard (do NOT unzip this file) with the SEARCH button and click the update-button.

Figure 39: Update of the RTK/RTCM Reference Station with the ANavS Wizard
4. Getting Started with the Integrated-Sensor-Platform (ISP)

This guide is intended for first time ISP users and provides an overview of how to handle with the required software, connect to and configure the ISP.

![Image of the Integrated-Sensor-Platform (ISP)]

Figure 40: The Integrated-Sensor-Platform (ISP)

4.1. General

The Integrated Sensor Platform (ISP) is a hardware basis for easy integration of a large variety of sensors without any effort. It comes with a standard configuration of three GNSS receivers and integrated antennas, an industrial-grade inertial measurement unit (IMU), a CAN interface for wheel odometry data and a barometer. On top of the standard sensors a fully integrated computer vision module is equipped, that can be flexibly configured with two cameras and/or a 3D-LiDAR.

The camera only configuration includes a high-quality, high frame-rate global shutter monocular HD camera and a global shutter stereo camera with either fisheye objective or depth sensing capabilities.

The LiDAR only configuration includes a high-quality 3D-LiDAR for 360° sensing. On the one hand, both the camera sensors and the LiDAR sensor provide an additional odometry or positioning information through visual-inertial odometry or Simultaneous Localization and Mapping (SLAM). On the other hand, 2D and 3D maps of the environment or road can be generated. Furthermore, semantic segmentation enabled by deep learning algorithms allows to add semantic information to the maps and further enhances the SLAM performance.

The additional positioning information obtained from the computer vision module is coupled with GNSS, IMU and wheel odometry information in the ANavS® Sensor Fusion. This leads to an improved positioning accuracy to overcome also the most challenging environments.
4.2. The ISP Setup

The ISP consists of the following modules and components:

**Multi-Sensor Fusion System (Integrated ANavS® MS-RTK module):**

- 3x Multi-GNSS, Multi-Frequency Receivers
- 3x High-Class, Survey-Grade, Triple-Frequency GNSS Antennas
- High-grade MEMS IMU

**Computer Vision Module**

- Camera sensors:
  - High-quality RGB camera with global shutter ([FLIR Grasshopper3](#))
  - Stereo camera with global shutter and integrated IMU ([Intel Realsense D435i](#))
- LiDAR sensors:
  - 3D mechanical 360° LiDAR ([Velodyne Puck/ VLP-16](#))
- Embedded computing platforms (ECPs):
  - Embedded GPU AI compute module ([NVIDIA Jetson Module](#), esp. Jetson Xavier NX and Jetson Orin NX), in the following referred to as: Vision Processing Unit (VPU)
  - NVIDIA Jetson compatible Carrier Board
  - Integrated data storage (M.2 NVMe SSD)
  - 1-2 embedded compute platforms can be integrated

**Interfaces**

- Ethernet, WLAN, CAN, USB, LTE

The ISP is ready to use out-of-the-box. With its vacuum cups (suction cups, Figure 41) it can safely be mounted for any kind of automotive dynamics. Pump the plunger a few times until the suction cup is completely fixed with the surface of mounting. When the red line has faded out, the vacuum is sufficient for lifting. A firm push or pull on the tab on the edge of the cup can loosen the holder. A suction cup protector prevents damage.
4.3. The Basic-Configuration: GNSS, IMU and Odometry Sensor Fusion

The basic configuration with a tightly coupled sensor fusion of three GNSS receivers, an IMU and Odometry sensor data on the ISP corresponds to the description and handling of the MS-RTK module. Please follow the instructions in chapter 2.

A major advantage of the ISP is the fixed lever arms of the system and of each sensor. Corresponding to this, please use the following dimensions in Step-5 in the ANavS® Wizard Software (see section 2.8.5).

<table>
<thead>
<tr>
<th>Sensor Type</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>GNSS-Antenna-1 [m]</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>GNSS-Antenna-2 [m]</td>
<td>0.708</td>
<td>-0.39</td>
<td>0.0</td>
</tr>
<tr>
<td>GNSS-Antenna-3 [m]</td>
<td>0.708</td>
<td>0.39</td>
<td>0.0</td>
</tr>
<tr>
<td>IMU [m]</td>
<td>0.493</td>
<td>0.017</td>
<td>0.106</td>
</tr>
</tbody>
</table>

All dimensions refer to the body frame. The movement-direction is assigned to the field-of-view of the cameras. The GNSS-Antenna-1 is the rear one, GNSS-Antenna-2 is the left one and GNSS-Antenna-3 the right one.

An extrinsic calibration of all ISP sensors including cameras and LiDAR is given in the Appendix 7.
4.3.1. The ANavS®-Wizard to Update the ISP Basic-Configuration

Please follow the instructions in section 2.9 for updating the ISP regarding the GNSS/IMU/Odometry sensor fusion.

4.4. The Computer-Vision Operation-Modes

The Computer Vision module offers a basic operation mode for data acquisition and can be extended by customer or application specific algorithms that run in real-time on the Vision Processing Unit, such as object detection or Simultaneous Localization and Mapping (SLAM).

4.4.1. Prerequisites

For basic operation, such as starting and stopping data acquisition or other application modules, any operating system with installed terminal application supporting SSH is suitable.

For basic visualization the ANavS GUI can be used (no ROS requirement). For extended visualization of sensor data or application output a client operating system supporting ROS is required, a Linux Operating System for example (such as Ubuntu version 18.04, 20.04, etc.) and a ROS environment that supports ROS1 or ROS2 (such as ROS1 Melodic, Noetic, etc.). To use the ROS visualizations tools a ROS “Desktop” or “Desktop-Full” install is required.

4.4.2. Setup

Connecting to the ISP and login to the VPU:

You can connect via Ethernet, or WLAN. For transferring huge data or visualization of live camera or LiDAR data on a client an Ethernet connection is recommended. Ethernet provides a direct Gigabit connection to the VPU. WLAN only provides indirect access to the VPU, but it is sufficient for executing commands.

The VPU may distribute processes onto multiple embedded computing platforms (ECPs). In a system with two ECPs these are termed Master (IP address: 192.168.1.101) and Slave (IP address: 192.168.1.102). To operate the VPU, you only need to connect to the Master.

1. Ethernet

Connect your host to the ISP via Ethernet. You need to set a static IP in the domain 192.168.1.xxx, e.g.: 192.168.1.210. (Do not use these reserved IP addresses: 192.168.1.100, 192.168.1.101, 192.168.1.201, for a system with two ECPs: 192.168.1.102)

Add the following line to /etc/hosts:

192.168.1.101 <Master-host-name>
The Master host name is usually set to:  
master-orin, or ubuntu

Directly login to the VPU using SSH (Password: anavsisp):

```
ssh anavs@192.168.1.101
```

2. WLAN

Connect to the ISP WiFi Access Point ANAVS_ISP_XXX (Password: anavsrpk). Indirectly login to the VPU using SSH:

1. `ssh pi@192.168.42.1` Login to the Multi-Sensor Fusion System
2. `ssh anavs@192.168.1.101` Login to the Master

Before proceeding with the following operation modes make sure the ISP has good satellite view, LTE reception is available and the system is powered on already for some minutes, to ensure completion of MS-RTK system startup including services and internal time-synchronization. Furthermore, configure and start the sensor fusion (described in section 4.3). Starting the sensor fusion is optional if only camera and LiDAR sensor data shall be acquired.

4.4.3. Basic Operation Mode with Data Acquisition

The basic operation mode with data acquisition runs the sensor modules (camera and LiDAR sensors), provides the sensor data ROS messages and stores sensor data in rosbag files. A list of modules and sensor data ROS topics is provided in the tables below. The live sensor data can be visualized on an external host (e.g. notebook) using ROS. The Multi-Sensor Fusion System (MS-RTK) solution (as displayed in the ANavS GUI) is converted by a ROS wrapper and provided in ROS. All ROS messages are originally provided in ROS1. A ROS bridge is used to provide all ROS1 messages in ROS2 as well.

1. **Log in to the VPU using SSH** (as described above under Setup):

2. **Start the VPU modules:**

   ```
   start_vpu
   --help
   --no_ntp_sync
   ```

   Enable specific features:

   ```
   --record
   --enable_ros_bridge
   ```

   Starts default modules for data acquisition mode

   Overview of command line arguments

   Disable check for time-synchronization offset

   Recording of ROS topics to bag files

   Bridging all ROS1 topics to ROS2
Enable application modules:

```
--object_detection  Enable Object Detection and Tracking
--lidar_slam        Enable LiDAR SLAM
```

Start specific VPU sensor or application modules, for example:

```
start_vpu --sensors "d435i gh3"
start_vpu --apps "object_detection ros_bridge"
```

*) Command line arguments only applicable for systems delivered before 01.10.2023.

An overview of all VPU modules and ROS topics available is provided below in this section.

3. **Show log messages** of VPU modules (for information or debugging)

On a system with a single embedded compute platform execute the following command from the docker-compose root dir, i.e.: `~/vpu`

```
docker compose logs -f
```

For previous versions:

```
docker-compose logs -f
```

On a system with two embedded compute platforms:

Log messages of modules running on the Master:

```
vpu_master_logs
```

Log messages of modules running on the Slave:

```
vpu_slave_logs
```

4. **Docker contexts** on a system with two ECPs (does not apply for single ECP systems)

Two Docker contexts exist, i.e. a different Docker node is running on each platform.

To list all available Docker contexts run:

```
docker context ls
```

An asterisk marks the current context.

Available Docker contexts and corresponding embedded computing platform:

```
default       Master
```
slave_nx Slave
To change the context, use the following command:

docker use <context_name>

5. **Visualization of sensor data** on an external host (notebook)

   a) **Basic visualization in the ANavS GUI**

   For basic visualization of sensor data (camera, LiDAR) and application output (e.g., object detection) the ANavS GUI can be used. ROS is not required in this case. The ANavS GUI is part of the ANavS Wizard which is available for Unix and Windows and can be [downloaded from the ANavS Website](#).

   b) **Extended visualization in ROS**

   For extended visualization available ROS tools can be used. The following system and ROS settings are required for live data visualization on your host:

   - **Linux system settings (/etc/hosts):**

     Add the following line to the system file /etc/hosts:

     ```
     192.168.1.101 <master-host-name>
     ```

     For example:

     ```
     192.168.1.101 master-orin
     ```

   - **ROS1 settings (ROS master URI and ROS IP):**

     ```
     export ROS_MASTER_URI=http://192.168.1.101:11311
     export ROS_IP=<your-host-IP>
     ```

   Following ROS tools are available for visualization, for example:

   - Displaying images: `rqt_image_view`
   - 3D visualization tool: `rosrun rviz rviz`

6. **Data recording in ROS**

   If data recording is enabled data is stored in rosbag files to:

   ```
   ~/vpu/data/
   ```

   The file naming convention is:
Rosbag files are automatically split after 30 minutes of recording time to prevent dealing with huge files. Further, LZ4 compression is used, to reduce file sizes.

7. Systems time synchronization and ROS message time stamping

Recorded ROS messages are timestamped using the embedded computing platforms system time, which is synchronized to the Multi-Sensor Fusion System (MS-RTK) system time. To achieve global time synchronization, either GNSS satellite signal reception must be available, or an LTE mobile connection. The current time-offset to the MS-RTK system (NTP-server) can be queries using:

```
chronyc tracking
ntpq -p
```

For systems delivered before 01.10.2023:

Per default at VPU start the time-offset is required to fall below a defined threshold (default: 3 ms). Checking the time-offset can be disabled using the flag `--no_ntp_sync`. The threshold can set using the argument `--max_ntp_offset`.

Time-synchronization offsets are regularly logged to the file: `data/logs/ntp_log.txt`. In case a second computing platforms is used its time-offset is separately logged to the file: `data/logs/ntp_log_slave.txt`.

8. Stop all VPU modules

```
stop_vpu
```

9. Copy recorded data

Files can be copied to an external host using an FTP-client, or using SCP for example. An Ethernet connection and using an SFTP-client is recommended. For copying using SCP, for example use:

```
scp anavs@192.168.1.101:/home/anavs/vpu/data/<file-to-copy> <destination-folder>
```

Available VPU modules are listed below:
<table>
<thead>
<tr>
<th>Basic Modules</th>
<th>Sensors/ Algorithms</th>
<th>Module Name (docker-compose)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROS Master</td>
<td>ROS1 master</td>
<td>ros_master</td>
</tr>
<tr>
<td><strong>Sensors:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>High-quality RGB Camera ROS Wrapper</td>
<td>FLIR Grasshopper3 USB3</td>
<td>gh3*</td>
</tr>
<tr>
<td>Stereo Camera ROS Wrapper</td>
<td>Intel Realsense D435i*</td>
<td>d435i*</td>
</tr>
<tr>
<td>LiDAR Sensor ROS Wrapper</td>
<td>Velodyne Puck/ VLP16*</td>
<td>vlp16*</td>
</tr>
<tr>
<td><strong>Tools:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multi-Sensor Fusion System Solution ROS Wrapper 7</td>
<td>ROS wrapper for multi-sensor fusion system solution</td>
<td>pad2ros*</td>
</tr>
<tr>
<td>Bridge for ROS Images</td>
<td>Bridge for ROS images to ANavS GUI</td>
<td>roswgui*</td>
</tr>
<tr>
<td>Conversion of ROS LiDAR scans</td>
<td>Conversion of ROS LiDAR scans to ROS images</td>
<td>bev*</td>
</tr>
<tr>
<td>Rosbag Recorder</td>
<td>Recording of ROS messages to rosbag file</td>
<td>ros_recorder</td>
</tr>
<tr>
<td>ROS Bridge</td>
<td>Bridge between ROS1 and ROS2 messages.</td>
<td>rosw_bridge</td>
</tr>
</tbody>
</table>

**Applications:**

- **Object Detection and Tracking**
  - 2D real-time object detection and tracking
    - object_detection

- **LiDAR Simultaneous Localization and Mapping (SLAM)**
  - 3D real-time LiDAR Simultaneous Localization and Mapping (SLAM)
    - lidar_slam

By default the modules marked with an asterisk (*) in the table above are launched when executing the `start_vpu` script.

**Available ROS topics** are listed below:

<table>
<thead>
<tr>
<th>ROS Topic</th>
<th>Description</th>
<th>ROS Message Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sensors:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>/camera/image_raw</td>
<td>Raw camera image of the high-quality RGB camera.</td>
<td>sensor_msgs/Image</td>
</tr>
<tr>
<td>/d435i/infra1/image_rect_raw</td>
<td>Raw camera image 1 of the stereo camera.</td>
<td>sensor_msgs/Image</td>
</tr>
<tr>
<td>/d435i/infra2/image_rect_raw</td>
<td>Raw camera image 2 of the stereo camera.</td>
<td>sensor_msgs/Image</td>
</tr>
</tbody>
</table>

7 The VPU provides a ROS Wrapper for the Multi-Sensor Fusion System solution. Running the module pad2ros will publish an available solution in ROS.
Reference Guide

Multi-Sensor RTK Module, RTK Reference Station and ISP

<table>
<thead>
<tr>
<th>Topic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/d435i/imu</td>
<td>IMU data from stereo camera.</td>
</tr>
<tr>
<td>/velodyne_packets</td>
<td>LiDAR raw data</td>
</tr>
<tr>
<td>sensor_msgs/Imu</td>
<td></td>
</tr>
<tr>
<td>velodyne_msgs/VelodyneScan</td>
<td></td>
</tr>
</tbody>
</table>

**Solutions:**

<table>
<thead>
<tr>
<th>Topic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/anavs/solution/*</td>
<td>Multi-Sensor Fusion System Solution, provided by the ROS Wrapper. More details in Chapter 10, ANavS ROS-Ethernet-Adapter (REA).</td>
</tr>
</tbody>
</table>

**Applications:**

Object Detection and Tracking ROS topics, see separate section below.

LiDAR Simultaneous Localization and Mapping (SLAM) ROS topics, see separate section below.

If data recording is enabled, all topics listed above are recorded by default. In addition, more ROS topics are available that are advertised by the different modules, including application modules. Application modules are described in the following sections.

### 4.4.4. Sensor Data Visualization in the ANavS GUI and in ROS

The ANavS GUI provides a basic visualization of sensor data from camera and LiDAR. This does not require a ROS installation. Sensor data can be further visualized in ROS using available ROS tools, this requires a ROS installation on your host PC.

**Note:** Visualization in the ANavS GUI and in ROS requires an Ethernet connection to the ISP.

**Visualization in the ANavS GUI**

The ANavS GUI provides live visualization of camera images and LiDAR scans in the Tab “Sensors” and the sub-categories “Camera” and “LiDAR”, as depicted in Figure 42 and in Figure 43. LiDAR scans are visualized in a bird’s-eye view. LiDAR SLAM visualization is only available if the LiDAR SLAM application has been enabled and is explained separately.
Figure 42: ANavS GUI with live visualization of camera images (Tab “Sensors/Camera”).

Figure 43: ANavS GUI with live visualization of LiDAR scans in bird’s-eye view (Tab “Sensors/LiDAR”).

Visualization data is streamed from the VPU. The individual streams source IP and port are specified in the configuration file located in the ANavS Wizard installation folder at “bin/vision_streams.txt”. Figure 44 shows the default configuration. Table 2 summarizes the
support of the ANavS GUI visualization features with regard to the version of the ANavS Wizard and the ANavS ISP. The corresponding IP and port settings for vision streams are also given.

**Note that, for most recent ISPs the IP settings have to be changed to IP: 192.168.1.102 to enable GUI visualizations.**

![Configuration file for vision streams](image)

**Figure 44: Configuration file for vision streams.**

**Table 2: ANavS GUI visualization feature support and settings.**

<table>
<thead>
<tr>
<th>Version</th>
<th>ANavS GUI Visualization Feature</th>
<th>Vision Streams Config File</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Camera</td>
<td>LiDAR</td>
</tr>
<tr>
<td>ANavS Wizard</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt; v7.4.1</td>
<td>✓</td>
<td>x</td>
</tr>
<tr>
<td>&gt;= v7.4.1</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ISP Delivery Date</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt; 01.11.2023</td>
<td>✓ 192.168.1.101:52712</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**Visualization in ROS**

Live sensor data can be visualized in ROS using available ROS tools, such as RViz for 3D visualization, or rqt_image_view for displaying images. Figure 45 shows an example for the visualization of all camera data and 3D LiDAR data from the ISP sensors in RViz.
4.4.5. 2D Object Detection and Tracking

The object detection and tracking module performs real-time 2D object detection and tracking using live camera data from the monocular RGB camera. The output is tracked 2D bounding boxes with detection confidences. Object detection output is provided in ROS and in the ANavS GUI and described in the following.

**Note:** Visualization in the ANavS GUI requires an Ethernet connection to the ISP.

**Object detection model**

An initial pretrained model for the detection of the following object classes is available:

<table>
<thead>
<tr>
<th>Object Class</th>
<th>Class ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Car</td>
<td>0</td>
</tr>
<tr>
<td>Truck</td>
<td>1</td>
</tr>
<tr>
<td>Bus</td>
<td>2</td>
</tr>
<tr>
<td>Pedestrian</td>
<td>3</td>
</tr>
<tr>
<td>Motorcycle</td>
<td>4</td>
</tr>
<tr>
<td>Bicycle</td>
<td>5</td>
</tr>
<tr>
<td>Motorcyclist</td>
<td>6</td>
</tr>
<tr>
<td>Bicyclist</td>
<td>7</td>
</tr>
</tbody>
</table>

Object detection can be tailored to fit customer specific needs.
ROS specification

Object detection and tracking results are provided as:

- Object list
- Camera image with detected objects bounding boxes
- Camera image with detected objects trajectories

<table>
<thead>
<tr>
<th>ROS Topic</th>
<th>Description</th>
<th>ROS Message Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>/bounding_boxes/array</td>
<td>Object detections list (camera image coordinates)</td>
<td>derived_object_msgs/ObjectArray</td>
</tr>
<tr>
<td>/bounding_boxes/compressed</td>
<td>Camera image with annotated object detection bounding boxes (and predicted class label, track ID, classification certainty)</td>
<td>sensor_msgs/CompressedImage</td>
</tr>
<tr>
<td>/trajectory/compressed</td>
<td>Camera image with annotated object trajectories</td>
<td>sensor_msgs/CompressedImage</td>
</tr>
</tbody>
</table>

**Object list** (Topic name: /bounding_boxes/array)

Provides for each camera image frame a list of objects detected (message type: derived_object_msgs/ObjectArray). Single object detections (message type: derived_object_msgs/Object) contain the image coordinates of the 2D bounding box.

/bounding_boxes/array (derived_object_msgs/ObjectArray):

```plaintext
Header header
    uint32 seq:
    time stamp:
    string frame_id:
Object[] objects
    Header header
    uint32 seq:
    image
```

Sequence ID
ROS timestamp
“camera”
Sequence ID of the associated camera
Camera image with detected objects bounding boxes (Topic Name: /bounding_boxes/compressed)

The camera image with annotated object detection bounding boxes, including predicted class label, track ID and classification certainty is provided as ROS message, see Figure 46.

![Camera image with detected objects bounding boxes](image)

Figure 46: Camera image with detected objects bounding boxes.

Camera image with detected objects trajectories (Topic Name: /trajectory/compressed)

The camera image with annotated object trajectories is provided as ROS message, see Figure 47.

```
<table>
<thead>
<tr>
<th>time stamp:</th>
<th>ROS timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td>string frame_id:</td>
<td>“camera”</td>
</tr>
<tr>
<td>uint32 id</td>
<td>Object ID</td>
</tr>
<tr>
<td>geometry_msgs/Polygon polygon</td>
<td>Bounding box coordinates are stored as 4-point polygon in polygon/points[0-3]</td>
</tr>
<tr>
<td>uint8 classification</td>
<td>Object class ID (list of class IDs see above)</td>
</tr>
<tr>
<td>uint8 classification_certainty</td>
<td>Certainty of classification (0-255)</td>
</tr>
<tr>
<td>uint32 classification_age</td>
<td>Number of frames this object has been tracked and classified with the current class label</td>
</tr>
</tbody>
</table>
```
Figure 47: Camera image with detected objects trajectories.

Visualization in the ANavS GUI

Object detections are visualized in the Tab “Sensors/Camera”. Bounding boxes including class label, track ID and classification certainty are visualized in the live camera image, see Figure 48.
Figure 48: ANavS GUI with live visualization of 2D object detections (Tab “Sensors/Camera”).

Module configuration

The modules settings can be adjusted in the configuration file stored at:

```
~/vpu/config.yaml
```

Figure 49 shows the configuration file and the available parameters, that are sole related to the visualization of bounding boxes and trajectories.

**Note:** The visualization of trajectories is experimental! It is recommended to disable trajectory visualization. Trajectory visualization can be disabled by setting:

```
drawTrajectories: false.
```

Issues with high and increasing memory utilization may potentially occur when setting a larger number of frames using parameter `trajectoriesMaxNumFrames`. 
4.4.6. 3D LiDAR Simultaneous Localization and Mapping (SLAM)

The LiDAR SLAM module performs real-time 3D LiDAR-Inertial Simultaneous Localization and Mapping (SLAM) using the integrated 3D LiDAR sensor and the IMU sensor. Localization estimates the current 3D pose while mapping generates a 3D point cloud map of the environment at the same time. Furthermore, GNSS-based pose information is fused to improve localization accuracy and robustness in areas with sufficient satellite visibility. GNSS-fusion further allows absolute global positioning and the generation of geo-references maps. The output of localization is the current estimated 3D pose in the local map frame. Localization and mapping output is provided in ROS and in the ANavS GUI and described in the following.

**Note:** Visualization in the ANavS GUI and in ROS requires an Ethernet connection to the ISP.

**Starting the ANavS Sensor Fusion and specific settings in the ANavS Wizard**

GNSS-assisted LiDAR SLAM requires a GNSS-based positioning solution from the ANavS Sensor Fusion. Starting the ANavS Sensor Fusion is thus required. Specific settings that are required for LiDAR SLAM are depicted and explained in the following.

The LiDAR SLAM solution is computed relative to the LiDAR sensor position. To provide a compatible GNSS-based position for fusion set the ANavS Sensor Fusion origin to the LiDAR sensor position by applying the antenna positions depicted in Figure 50.

Furthermore, the ANavS Sensor Fusion output rate is required to be set to “Medium”, as depicted in Figure 51, to provide a sufficient rate for GNSS-based pose as input to LiDAR SLAM. Setting a higher rate is not required. Output rate “Low” is however not sufficient.

```yaml
object_detection:
  drawBoundingBoxes: true
  drawTrajectories: true
  trajectoriesMaxNumFrames: 5
  trajectoriesMaxNumIDs: 100
  vis_u_scale: 0.3333
  vis_h_scale: 0.4
```

Figure 49: Configuration file for 2D object detection visualization.
Figure 50: ANavS Wizard GNSS-antenna positions and sensor fusion coordinate frame. Setting the origin to the LiDAR sensor position.

Figure 51: ANavS Wizard starting sensor fusion with output rate set to Medium (55-65Hz) or High (105-125Hz) using the slider for Output-Rate. Low (5Hz) rate is not sufficient as input for LiDAR-Inertial SLAM.
LiDAR SLAM Initialization Outdoor with GNSS RTK-Fix

For initializing GNSS-assisted LiDAR SLAM starting outdoor with good GNSS-reception and obtaining a GNSS RTK-Fix is required. A GNSS RTK-Fix is achieved as soon as the “RTK” indicator at the top of the ANavS GUI turns green. An Attitude-fix is not required. For initial synchronization between the LiDAR-frame and the GNSS-based frame an initial motion over a certain distance with valid RTK-Fix is required. After initialization phase continuous GNSS-reception is not required, but GNSS-based pose will be fused when available and improves accuracy and robustness.

**ROS specification**

Localization and mapping provide this output:

- 3D pose
- 3D point cloud maps (local and global map)

<table>
<thead>
<tr>
<th>ROS Topic</th>
<th>Description</th>
<th>ROS Message Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>/lio_sam_6axis/mapping/odometry</td>
<td>3D pose in globally optimized map frame</td>
<td>nav_msgs/Odometry</td>
</tr>
<tr>
<td>/lio_sam_6axis/mapping/map_local</td>
<td>3D point cloud map (local map)</td>
<td>Sensor_msgs/PointCloud2</td>
</tr>
<tr>
<td>/lio_sam_6axis/mapping/map_global</td>
<td>3D point cloud map (globally optimized map)</td>
<td>Sensor_msgs/PointCloud2</td>
</tr>
</tbody>
</table>

**3D pose and point cloud maps** (Topic Names: /lio_sam_6axis/mapping/odometry, /lio_sam_6axis/mapping/map_local, /lio_sam_6axis/mapping/map_global)

The optimized pose is tracked and the corresponding path followed by the vehicle is visualized in RViz as shown in Figure 52. After receiving a few GNSS-keyframes, SLAM poses are being aligned with GNSS. Aligned GNSS-keyframes along the trajectory are depicted in Figure 53. The instantaneous local map is a denser point cloud that is evaluated in the frontend, and the global map is a coarser point cloud generated post optimization. The maps and the corresponding trajectory can be visualized in RViz, such as shown in Figure 52.

Setup RViz for visualization of Lidar SLAM output on your host in ROS:
• An RViz config file is provided on the ISP. You can copy the config file from the Master onto your host using:
  
  `scp anavs@192.168.1.101:/home/anavs/vpu/apps/liosam/lio-sam-6axis/launch/include/config/vlp.rviz <destination-folder>`

• On your host, run RViz using the config file:
  
  `rviz -d vlp.rviz`

The visualization of the globally optimized point cloud is disabled in our config file, because it leads to high load on RViz visualization especially for longer sessions with larger maps, which makes RViz visualization unstable. You can still enable it in RViz, you may set a decay time to restrict the time range for which the map is visualized.

![Figure 52: LiDAR SLAM point cloud map and the corresponding trajectory (cyan) visualized in RViz.](image-url)
Figure 53: Synchronization with GNSS: SLAM trajectory (cyan) and GNSS-keyframes (purple) after synchronization.

Visualization in the ANavS GUI

The local point cloud map is visualized in the Tab “Sensors/LiDAR SLAM” in bird’s-eye view.

Figure 54: ANavS GUI live visualization of the LiDAR map in bird’s-eye view (Tab “Sensors/LiDAR SLAM”).

4.4.7. Known Issues and Troubleshooting

General
• **Issues with the Intel Realsense camera device:**
The camera device resource may still be blocked by previous access. Reboot the embedded computing platform, or power off and power on the ISP to release the camera device resource.

For systems delivered before 01.10.2023:

• **No internet connection** on the embedded computing platforms:
The embedded compute platforms have no access to the internet by default. If you require an internet connection, you may switch from static to DHCP network settings an connect the ISP to an external router via Ethernet. To do so modify the following line in the Linux system file and **reboot**: /etc/network/interfaces

  ```
  from: source interfaces.d/eth0_static
  to: source interfaces.d/eth0
  ```

**Warning:** Do this with caution, a wrong network configuration will deny system access after reboot! To prevent network issues in your local network, when using an ethernet connection to the ISP, it is recommended to disable the Velodyne LiDAR via the Velodyne webinterface (192.168.1.201) by setting “Laser” to “Off” and “Motor RPM” to “0” and storing the settings. Otherwise the LiDAR UDP broadcast may yield network issues.

• **NTP-offset**
  - **is not available** (Error: Command ‘./scripts/get_ntp_offset.sh’ returned non-zero exit status 1) The NTP-offset may not be retrieved if the NTP-daemon on the MS-RTK module is not active, which will result in a failure of the start_vpu script.

  To resolve this issue:
  - make sure satellites are visible or LTE reception is available and wait some minutes after system power on before starting the VPU
  - reboot the embedded computing platform
  - **does not decrease** after waiting some time:
    - increase the NTP-offset threshold when starting the vpu_start script using the parameter `--max_ntp_offset`
    - restart the ntp service: `sudo service ntp restart`

**Module: 2D Object Detection and Tracking**

• **System performance may slow down** after longer continuous operation (more than 3-4 days). To prevent this reboot the system regularly.
• **High and increasing memory utilization:** Enabling trajectory visualization may result in memory utilization issues (see module configuration).

**Module: 3D LiDAR SLAM**

• LiDAR SLAM may fail
  - *for high velocity and strong dynamic motions* including sharp turns
  - *for very dynamic environments*
  - *for very small indoor environments*, such as long passages or small rooms.

• **RViz visualization might start lagging** when running it over a longer period of time, for example caused by large messages, such as global point clouds, that are continuously increasing in size.
5. The Command Line API Reference Guide

All ANavS Positioning Systems provide an API used by command line. The API is directly used on the modules through the SSH-access or via an external client on your Laptop/PC using a TCP/IP connection. Please ask the support-team for access to this functionality. The following sections describes the most important commands for customers. An autocomplete is available.

A detailed description of each command is also available via command-line:

**MSRTKF help -search <command>**

### 5.1. Navigation commands

This section describes the commands regarding the navigation software. With these commands, the user can control the ANavS Systems without using the Wizard Software. Each API command needs the preamble `MSRTKF` (Example: `MSRTKF Navigation.attach`).

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Navigation.attach</td>
<td>Prints the sensor fusion software trace into the command line.</td>
</tr>
<tr>
<td>Navigation.configNtripPolicy</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Get the currently configured policy.</td>
</tr>
<tr>
<td></td>
<td><strong>Navigation.configNtripPolicy</strong> -set SET</td>
</tr>
<tr>
<td></td>
<td>Set the policy.</td>
</tr>
<tr>
<td></td>
<td><strong>SET</strong> in OFF mode, NTRIP is deactivated. In AUTO mode, NTRIP is automatically enabled when the navigation software starts and disabled when it stops. <strong>-set SET</strong> is mandatory. Allowed values for <strong>SET</strong> are AUTO, OFF.</td>
</tr>
<tr>
<td>Navigation.getLatency</td>
<td>Prints the current solution latency</td>
</tr>
<tr>
<td>Navigation.getMode</td>
<td>Prints the current Navigation mode</td>
</tr>
<tr>
<td>Navigation.setMode</td>
<td><strong>Navigation.setMode</strong> -mode MODE -temp TEMP</td>
</tr>
<tr>
<td></td>
<td>Set navigation mode.</td>
</tr>
<tr>
<td></td>
<td><strong>MODE</strong> Navigation mode to set. <strong>-mode MODE</strong> is mandatory. Allowed values for <strong>MODE</strong> are: UNMANAGED, PAD_1D_RTCM, PAD_2D_RTCM, PAD_3D_RTCM, RTCM_NTRIP, RTCM_MOSAIC, PPP_RT.</td>
</tr>
<tr>
<td></td>
<td>The modes PAD_1D_RTCM, PAD_2D_RTCM, PAD_3D_RTCM launches the sensor fusion software with 1-Antenna/2-Antenna/3-Antenna RTCM mode. Automatic start and restart are enabled. The mode UNMANAGED deactivates the sensor fusion again. The modes RTCM_NTRIP, RTCM_MOSAIC launches the RTCM NTRIP-server/caster software. Automatic start and restart enabled.</td>
</tr>
<tr>
<td></td>
<td><strong>TEMP</strong> Set mode only temporary. The change is not saved persistent. <strong>-temp TEMP</strong> is optional. Allowed values for <strong>TEMP</strong> are: true, false.</td>
</tr>
</tbody>
</table>
Enables/Disables the navigation mode.

The navigation software is monitored by the firmware of the positioning systems. The user gets one of these responses:

**UNMANAGED:** Binaries are not currently running and are not scheduled to start.

**DELAY:** Selected binaries are scheduled to start soon (boot delay).

**STARTUP_WAIT:** Selected binaries have been started but have not been verified to be functional (startup period).

**RECOVER_WAIT:** The selected binaries are up and have already been verified to be functional, however currently there is a problem. We have started a timer for the problem to disappear before we enforce a module-restart.

**NOMINAL:** The selected binaries are up and have been verified to be functional, moreover there is currently no indication of a problem with the navigation binaries.

**UNKNOWN:** There is currently no information on the state of the navigation software.

Restart the currently active navigation software.

Description follows in next version.

Description follows in next version.

Upload a PAD configuration resource.

**Navigation.PAD.properties**

Because of the large amount of text, please have a look for the description of this command via command line: `MSRTKF help -search Navigation.PAD.properties`

Set parameters for NTRIP-caster in an RTCM station.

**Navigation.RTCM.config**

Set values for MODE are: `acm`, `devices`.

Wait for a given time until the specified device becomes available.

**SYS.assertDevice**

**SYS.assertDevice -name NAME -t T**

**SYS.assertDevice -name NAME -t T** is mandatory.

**SYS.assertDevice -name NAME -t T** is optional.

**SYS.assertDevice -name NAME -t T** is optional.

This section describes the system commands. Each API command needs the preamble `MSRTKF` (Example: `MSRTKF SYS.ping`).

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYS.acmInfo</td>
<td><strong>SYS.acmInfo -mode MODE</strong> Print information of connected usb ACM devices. <strong>MODE</strong> Select a mode to list data. <strong>-mode MODE</strong> is optional. Allowed values for <strong>MODE</strong> are: <code>acm</code>, <code>devices</code>.</td>
</tr>
<tr>
<td>SYS.assertDevice</td>
<td><strong>SYS.assertDevice -name NAME -t T</strong> Wait for a given time until the specified device becomes available. <strong>NAME Name</strong> of the device. <strong>-name NAME</strong> is mandatory.</td>
</tr>
</tbody>
</table>
## Reference Guide

**Multi-Sensor RTK Module, RTK Reference Station and ISP**

### SYS.assertInterface

SYS.assertInterface -name NAME -t T -syslog SYSLOG

- **NAME** Name of the interface. **-name NAME** is mandatory.
- **T** is the maximum number of milliseconds to wait. **-t T** is mandatory.
- **SYSLOG** is to optionally search the syslog for a message that the chatscript has failed. **-syslog SYSLOG** is optional. Allowed values for **SYSLOG** are: true, false.

### SYS.assertIp

SYS.assertIp

Description follows in next version.

### SYS.assertUsbDevice

SYS.assertUsbDevice -vendor VENDOR -prodid PRODID -t T

- **VENDOR** Hex encoded vendorId of the device. **-vendor VENDOR** is mandatory.
- **PRODID** Hex encoded productId of the device. **-prodid PRODID** is mandatory.
- **T** is the maximum number of milliseconds to wait before aborting. **-t T** is mandatory.

### SYS.checklist

SYS.checklist -index INDEX -verbose VERBOSE

- **INDEX** Index of the checklist item to be displayed. **-index INDEX** is optional.
- **VERBOSE** Set true to print the result. **-verbose VERBOSE** is optional. Allowed values for **VERBOSE** are: true, false.

### SYS.initNtp

SYS.initNtp

Description follows in next version.

### SYS.Memory.block

SYS.Memory.block

Description follows in next version.

### SYS.Memory.info

SYS.Memory.info

Description follows in next version.

### SYS.ping

SYS.ping

Proving the current connection state.

### SYS.productName

SYS.productName -verbose VERBOSE

- **VERBOSE** Print product name. **-verbose VERBOSE** is optional. Allowed values for **VERBOSE** are: true, false.

### SYS.restart

SYS.restart -mode MODE

- **MODE** Name of the restart-mode. **-mode MODE** is mandatory. Allowed values for **MODE** are: COMMAND_SERVER, DRIVER, POSITIONING_APPLICATION, MCU_REBOOT, MCU_RESET, SYSTEM_REBOOT.

### SYS.serial

SYS.serial

- **STREAM** Read the system serial number.
- **STREAM** Read the serial number of a stream.
### 5.3. Role commands

This section describes the role commands. A role defines a specific customer or internal setup of the MS-RTK-System on a system level. Each API command needs the preamble `MSRTKF` (Example: `MSRTKF Role.get`).

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Role.get</td>
<td><strong>Role.get</strong> -silent SILENT</td>
</tr>
<tr>
<td></td>
<td>Gets the currently fulfilled role of the system.</td>
</tr>
<tr>
<td></td>
<td>SILENT Suppress readable output. -silent SILENT is optional. Allowed</td>
</tr>
<tr>
<td></td>
<td>values for SILENT are: true, false.</td>
</tr>
<tr>
<td></td>
<td>To get the current role of your MS-RTK module. The default role is</td>
</tr>
<tr>
<td></td>
<td>MSRTK.WIZARD</td>
</tr>
<tr>
<td>Role.install</td>
<td><strong>Role.install</strong> -name NAME</td>
</tr>
<tr>
<td></td>
<td>Sets the currently fulfilled role of the system and stores role-specific</td>
</tr>
<tr>
<td></td>
<td>information to /home/pi/device/settings.</td>
</tr>
</tbody>
</table>
5.4. CAN commands

This section describes the commands regarding the CAN-interface. The commands are described in more detail in chapter 8. Each API command needs the preamble `MSRTKF` (Example: `MSRTKF CAN.dbcc`).

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAN.config.reset</td>
<td>Description follows in next version.</td>
</tr>
<tr>
<td>CAN.config.solution</td>
<td>Because of the large amount of text, please have a look for the description</td>
</tr>
<tr>
<td></td>
<td>of this command via command line:</td>
</tr>
<tr>
<td></td>
<td><code>MSRTKF help -search CAN.config.solution</code></td>
</tr>
<tr>
<td>CAN.config.solution.list</td>
<td><code>CAN.config.solution.list -format FORMAT -onlyenabled ONLYENABLED -file FILE</code></td>
</tr>
<tr>
<td></td>
<td>Display list of currently enabled and disabled CAN output signals with</td>
</tr>
<tr>
<td></td>
<td>their current CAN-Ids.</td>
</tr>
<tr>
<td></td>
<td>FORMAT Allows to select a plain list or dbc. -format FORMAT is</td>
</tr>
<tr>
<td></td>
<td>mandatory. Allowed values for FORMAT are: LIST, DBC.</td>
</tr>
<tr>
<td></td>
<td>ONLYENABLED If set to true, only enabled messages are shown. -</td>
</tr>
<tr>
<td></td>
<td>onlyenabled ONLYENABLED is optional. Allowed values for</td>
</tr>
<tr>
<td></td>
<td>ONLYENABLED are: true, false.</td>
</tr>
<tr>
<td></td>
<td>FILE Allows to select a plain list or dbc. -file FILE is mandatory.</td>
</tr>
<tr>
<td>CAN.dbcc</td>
<td><code>CAN.dbcc -source SOURCE</code></td>
</tr>
<tr>
<td></td>
<td>Analyze the loaded .dbc file with dbcc</td>
</tr>
<tr>
<td></td>
<td>SOURCE Path of the .dbc file. Defaults to the currently loaded file.</td>
</tr>
<tr>
<td></td>
<td>-source SOURCE is optional.</td>
</tr>
<tr>
<td>CAN.generateDecoder</td>
<td><code>CAN.generateDecoder</code></td>
</tr>
<tr>
<td></td>
<td>Generate a dynamic decoder binary</td>
</tr>
<tr>
<td>CAN.getStatus</td>
<td><code>CAN.getStatus</code></td>
</tr>
<tr>
<td></td>
<td>Get a description of the CAN status</td>
</tr>
<tr>
<td></td>
<td><code>CAN.getStatus -item ITEM</code></td>
</tr>
<tr>
<td></td>
<td>Show CAN odometry rates.</td>
</tr>
<tr>
<td></td>
<td>ITEM Item to be displayed. -item ITEM is mandatory. Allowed values for</td>
</tr>
<tr>
<td></td>
<td>ITEM are: CAN_MESSAGE_RATE, DRIVER_SIGNAL_RATE,</td>
</tr>
<tr>
<td></td>
<td>DRIVER_MEASUREMENT_RATE, PAD_PROCESSING_RATE, ALL.</td>
</tr>
<tr>
<td>CAN.hardwareFilter.config</td>
<td><code>CAN.hardwareFilter.config</code></td>
</tr>
<tr>
<td></td>
<td>Automatically generate optimal CAN reception hardware filter settings.</td>
</tr>
<tr>
<td></td>
<td><code>CAN.hardwareFilter.config -clear CLEAR</code></td>
</tr>
</tbody>
</table>
### Disable CAN reception hardware filter.

**CLEAR** instructs the command to disable the filter. `-clear CLEAR` is mandatory. Allowed values for `CLEAR` are: `true`.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAN.loadDbc</td>
<td><strong>CAN.loadDbc</strong> <code>-source RSOURCE -encoding ENCODING</code></td>
</tr>
<tr>
<td></td>
<td>Loads a <code>.dbc</code> file and tries to detect in a best-effort way if it really is a <code>.dbc</code> file.</td>
</tr>
<tr>
<td></td>
<td><code>RSOURCE</code> <code>SOURCE</code> is a resource name used to provide the file.</td>
</tr>
<tr>
<td></td>
<td><code>-source RSOURCE</code> is mandatory.</td>
</tr>
<tr>
<td></td>
<td><code>ENCODING</code> <code>NAME</code> of the encoding. If the encoding is UTF-8 or ISO-8859-1</td>
</tr>
<tr>
<td></td>
<td>this can be omitted. <code>-encoding ENCODING</code> is optional.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAN.signal</td>
<td><strong>CAN.signal</strong> <code>-get GET -sup SUP</code></td>
</tr>
<tr>
<td></td>
<td>Returns the mapping for given sensor.</td>
</tr>
<tr>
<td></td>
<td><code>GET</code> Sensor to be retrieved. <code>-get GET</code> is mandatory. Allowed values for</td>
</tr>
<tr>
<td></td>
<td><code>GET</code> are: FL, FR, RL, RR, STEER, ALL.</td>
</tr>
<tr>
<td></td>
<td><code>SUP</code> Suppress warning that signal cache will be rebuilt. <code>-sup SUP</code> is</td>
</tr>
<tr>
<td></td>
<td>mandatory. Allowed values for <code>SUP</code> are: <code>true</code>, <code>false</code>.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAN.signal</td>
<td><strong>CAN.signal</strong> <code>-clear CLEAR</code></td>
</tr>
<tr>
<td></td>
<td>Disable CAN reception hardware filter.</td>
</tr>
<tr>
<td></td>
<td>CLEAR instructs the command to disable the filter. <code>-clear CLEAR</code> is</td>
</tr>
<tr>
<td></td>
<td>mandatory. Allowed values for <code>CLEAR</code> are: <code>true</code>.</td>
</tr>
</tbody>
</table>
5.5. Driver commands

This section describes the driver commands. Each API command needs the preamble `MSRTKF` (Example: `MSRTKF Driver.config`).

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
</table>
| Driver.checkInstallation | Driver.checkInstallation -source SOURCE  
  Compare driver installation to an archive file.  
  SOURCE Path of the installation archive. -source SOURCE is optional. |
| Driver.config         | Description follows in next version.                                         |
| Driver.createArchive  | Description follows in next version.                                         |
| Driver.deployArchive  | Driver.deployArchive -source SOURCE  
  Deploy driver files from archive  
  SOURCE Path to a local update archive. -source SOURCE is optional. |
| Driver.installUpdate  | Driver.installUpdate  
  Install the last downloaded update archive. Most users will prefer  
  SYS.update instead. |
| Driver.listInterfaces | Description follows in next version.                                         |
| Driver.listMappings   | Description follows in next version.                                         |
| Driver.listPlugins    | Description follows in next version.                                         |
| Driver.reinit         | Driver.reinit -reason REASON -dump DUMP  
  Request to send a SIGINT to the driver  
  REASON Log message. -reason REASON is optional.  
  DUMP Request to dump a stack trace. In some situations, a stack trace  
  will be generated anyway. -dump DUMP is optional. Allowed values for  
  DUMP are: true, false. |
| Driver.setNtripEnabled | Driver.setNtripEnabled -enabled ENABLED -pushconfig PUSHCONFIG  
  Connect to driver to set state of NTRIP plugin.  
  ENABLED True to enable NTRIP. -enabled ENABLED is mandatory.  
  Allowed values for ENABLED are: true, false.  
  PUSHCONFIG True to also push NTRIP configuration. -pushconfig  
  PUSHCONFIG is mandatory. Allowed values for PUSHCONFIG are: true,  
  false. |
| Driver.start          | Driver.start  
  Start driver. |
| Driver.stop           | Driver.stop  
  Stop driver and guard. |
| Driver.version        | Driver.version -silent SILENT -cached CACHED  
  Show version of driver.  
  SILENT Don’t print the result. -silent SILENT is optional. Allowed values for  
  SILENT are: true, false.  
  CACHED Don’t query the driver, use last response. -cached CACHED is  
  optional. Allowed values for CACHED are: true, false. |
## 5.6. GNSS commands

This section describes the commands related to the GNSS receivers. Each API command needs the preamble **MSRTKF** (Example: **MSRTKF GNSS.config**).

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
</table>
| GNSS.chat        | GNSS.chat -path PATH -message MESSAGE  
  Exchange data with a serial device. The command expects some answer data and prints it.  
  **PATH** Path of the device. For example /dev/ttyACM0. **-path PATH** is mandatory.  
  **MESSAGE** Message to send. \r and \n will be changed to newline and carriage return. **-message MESSAGE** is mandatory. |
| GNSS.config      | GNSS.config  
  Detect GNSS receiver presence and device. Send correct configuration to all detected devices. |
| GNSS.config.Mosaic | GNSS.config.Mosaic -highrate HIGHRATE  
  Detect Mosaic receivers and write the configuration  
  **HIGHRATE** Set all messages to 100Hz if true, 5Hz otherwise. **-highrate HIGHRATE** is optional. Allowed values for **HIGHRATE** are: true, false. |
| GNSS.leapSeconds | Description follows in next version. |
| GNSS.listMessageTypes | Description follows in next version. |
| GNSS.reset       | GNSS.reset -num NUM -mode MODE  
  Reset a GNSS module on one of the three RF ports.  
  **NUM** Index of the GNSS slot. **-num NUM** is mandatory. Allowed values for **NUM** are: 1, 2, 3.  
  **MODE** Chooses what to do with the reset pin. **-mode MODE** is optional. Allowed values for **MODE** are: CYCLE, LOW, HIGH, RELEASE. |
## 5.7. LTE commands

This section describes the commands related to the mobile network module and the SIM-card settings. Each API command needs the preamble `MSRTKF` (Example: `MSRTKF LTE.config`).

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LTE.block</td>
<td>LTE.block Retrieves the current blocking date.</td>
</tr>
</tbody>
</table>
| LTE.block -set SET | Set a blocking date (or null to disable blocking)  
|                  | SET Time period in the form 10s or 22m or 1h to set the new blocking date. null to clear the blocking date. -set SET is mandatory.                 |
| LTE.chat         | LTE.chat -message MESSAGE Send a command to the AT interface of the LTE modem. MESSAGE The AT command. -message MESSAGE is mandatory.                    |
| LTE.config       | LTE.config -apn APN -dial DIAL -user USER -isppw ISPPW -pin PIN Set the configuration for the LTE modem. APN that is required to use your SIM card. Contact your internet service provider for that information. -apn APN is mandatory. DIAL Dial up number that is required to use your SIM card. Contact your internet service provider for that information. -dial DIAL is mandatory. USER Username that is required to use your SIM card. Contact your internet service provider for that information. -user USER is optional. ISPPW Password that is required to use your SIM card. Contact your internet service provider for that information. -isppw ISPPW is optional. PIN Pin that is required to use your SIM card. Contact your internet service provider for that information. -pin PIN is optional. |
| LTE.config.read  | LTE.config.read -get GET Retrieve a single configuration item of your LTE configuration. GET Name of the item to get. -get GET is mandatory. Allowed values for GET are: apn, dial, user, isppw, pin. |
| LTE.deviceName   | Description follows in next version.                                                                                                                                                                           |
| LTE.getIccid     | Description follows in next version.                                                                                                                                                                           |
| LTE.mobileEquipmentStatus | LTE.mobileEquipmentStatus Query the modem for the mobile equipment status.                                                                                                                                          |
| LTE.receivedSignal | LTE.receivedSignal -dev DEV Show the received signal quality. DEV tty device of the modem control interface. Defaults to the correct value. -dev DEV is optional. |
| LTE.reception    | LTE.reception Show the received signal quality in unit of 'bars'.                                                                                                                                              |
| LTE.restart      | LTE.restart Restart LTE modem.                                                                                                                                                                                 |
| LTE.service.reception | Description follows in next version.                                                                                                                                                                           |
| LTE.service.state | LTE.service.state -verbose VERBOSE Return the state of the LTE modem without blocking.                                                                                                                       |
### 5.8. Network commands

This section describes the commands related to the network settings. Each API command needs the preamble `MSRTKF` (Example: `MSRTKF Network.config`).

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network.config</td>
<td><strong>Network.config -interface INTERFACE -get GET</strong>  &lt;br&gt;Get a config item.  &lt;br&gt;<strong>INTERFACE</strong> Name of the interface to configure (= ETH). <strong>-interface</strong> INTERFACE is mandatory. Allowed values for INTERFACE are: ETH.  &lt;br&gt;<strong>GET</strong> Name of the item to retrieve. <strong>-get GET</strong> is mandatory. Allowed values for GET are: profile, static.address, static.netmask, linkLocalAddress, current.address, current.netmask, current.profile.</td>
</tr>
<tr>
<td></td>
<td><strong>Network.config -interface INTERFACE -set SET -value VALUE</strong>  &lt;br&gt;Set a config item.  &lt;br&gt;<strong>INTERFACE</strong> Name of the interface to configure (= ETH). <strong>-interface</strong> INTERFACE is mandatory. Allowed values for INTERFACE are: ETH.  &lt;br&gt;<strong>SET</strong> Name of the item to change. <strong>-set SET</strong> is mandatory. Allowed values for SET are: profile, static.address, static.netmask.  &lt;br&gt;<strong>VALUE</strong> New value of the changed item. <strong>-value VALUE</strong> is mandatory.</td>
</tr>
<tr>
<td></td>
<td><strong>Network.config -interface INTERFACE -get GET</strong>  &lt;br&gt;Get a config item.  &lt;br&gt;<strong>INTERFACE</strong> Name of the interface to configure (= WIFI). <strong>-interface</strong> INTERFACE is mandatory. Allowed values for INTERFACE are: WIFI.  &lt;br&gt;<strong>GET</strong> Name of the item to retrieve. <strong>-get GET</strong> is mandatory. Allowed values for GET are: profile, static.address, static.netmask, linkLocalAddress, current.address, current.netmask, current.ssid, current.quality, current.profile, ap.ssid, ap.psk, client.ssid, client.psk.</td>
</tr>
<tr>
<td></td>
<td><strong>Network.config -interface INTERFACE -set SET -value VALUE</strong>  &lt;br&gt;Set a config item.  &lt;br&gt;<strong>INTERFACE</strong> Name of the interface to configure (= WIFI). <strong>-interface</strong> INTERFACE is mandatory. Allowed values for INTERFACE are: WIFI.  &lt;br&gt;<strong>SET</strong> Name of the item to change. <strong>-set SET</strong> is mandatory. Allowed values for SET are: profile, static.address, static.netmask, ap.ssid, ap.psk, client.ssid, client.psk.  &lt;br&gt;<strong>VALUE</strong> New value of the changed item. <strong>-value VALUE</strong> is mandatory.</td>
</tr>
<tr>
<td>Network.config.apply</td>
<td><strong>Network.config.apply</strong>  &lt;br&gt;Apply pending network configuration.</td>
</tr>
<tr>
<td>Network.config.reset</td>
<td>Description follows in next version.</td>
</tr>
</tbody>
</table>
5.9. Record commands

This section describes the commands related to recording sensor raw data, the sensor fusion solution and the corresponding logs. Each API command needs the preamble `MSRTKF` (Example: `MSRTKF Record.setMode`).

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Record.deleteDataset</td>
<td><code>Record.deleteDataset -name NAME</code> Delete a recorded dataset.</td>
</tr>
<tr>
<td></td>
<td><code>NAME</code> Name of the dataset. <em>-name NAME</em> is mandatory. Allowed values for</td>
</tr>
<tr>
<td></td>
<td><code>NAME</code> are: <code>20210409_1149_UTC</code>, <code>20210803_1222_UTC</code> (examples)</td>
</tr>
<tr>
<td>Record.extractStream</td>
<td><code>Record.extractStream -name NAME -stream STREAM</code> Extract a stream from a</td>
</tr>
<tr>
<td></td>
<td>dataset.</td>
</tr>
<tr>
<td></td>
<td><code>NAME</code> Name of the dataset. <em>-name NAME</em> is mandatory. Allowed values for</td>
</tr>
<tr>
<td></td>
<td><code>NAME</code> are: <code>20210409_1149_UTC</code>, <code>20210803_1222_UTC</code> (examples)</td>
</tr>
</tbody>
</table>
5.10. Server commands

This section describes the server commands. Each API command needs the preamble **MSRTKF** (Example: **MSRTKF Server.installUpdate**).

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server.installUpdate</td>
<td>Description follows in next version.</td>
</tr>
<tr>
<td>Server.mirrorStream</td>
<td>Description follows in next version.</td>
</tr>
</tbody>
</table>
| Server.performanceLog         | **Server.performanceLog -sort SORT**  
|                               | Print performance log.  
|                               | SORT Sort by. -sort SORT is optional. Allowed values for SORT are:  
|                               | SELF_TIME, TOTAL_DURATION, INVOCATIONS, AVG_DURATION.                     |
| Server.printTrace             | **Server.printTrace**  
|                               | Prints the complete trace log of this command server.                     |
| Server.remotePerformanceLog   | **Server.remotePerformanceLog -sort SORT -summed SUMMED**  
|                               | Print performance log.  
|                               | SORT Sort by. -sort SORT is optional. Allowed values for SORT are:  
|                               | TERMINATION_TIME, ISSUE_TIME, SERVING_LATENCY,  
|                               | EXECUTION_LATENCY, DURATION, WASTED_TIME, WAITED_TIME,  
|                               | BLOCKED_TIME, CPU_TIME.  
|                               | SUMMED Accumulate over all entries of the same operation.  
|                               | -summed SUMMED is optional. Allowed values for SUMMED are: true, false.  |
| Server.traceRetention         | **Server.traceRetention**  
|                               | Get current trace retention mode.                                          |
|                               | **Server.traceRetention -set SET**  
|                               | Set current trace retention mode. For safety-reasons, this setting can  
|                               | not be made persistent and will go back to SOFT after a restart.  
|                               | SET New mode. -set SET is mandatory. Allowed values for SET are:  
|                               | SOLID, SOFT, WEAK.                                                        |
| Server.updateMaintenanceTool  | **Server.updateMaintenanceTool -rbin RBIN**  
|                               | Write binary to temporary location and use atomic file system  
|                               | operation to replace the old binary with the new binary.                 |
5.11. Time commands

This section describes the time commands related to NTP. The commands are described in more detail in chapter 9. Each API command needs the preamble MSRTKF (Example: MSRTKF Time.config).

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time.adjust</td>
<td>Description follows in next version.</td>
</tr>
<tr>
<td>Time.config</td>
<td>Time.config -set SET -value VALUE</td>
</tr>
<tr>
<td></td>
<td>Set a time configuration parameter.</td>
</tr>
<tr>
<td></td>
<td>SET Item to be configured. -set SET is mandatory. Allowed values for</td>
</tr>
<tr>
<td></td>
<td>SET are: timePolicy, timeServer.</td>
</tr>
<tr>
<td></td>
<td>VALUE Time sync policy. -value VALUE is mandatory. Allowed values for</td>
</tr>
<tr>
<td></td>
<td>VALUE are: SYNC_OFF, SYNC_GNSS, SYNC_NETWORK, AUTO, time.nist.gov.</td>
</tr>
<tr>
<td></td>
<td>Time.config -get GET</td>
</tr>
<tr>
<td></td>
<td>Get a time configuration parameter.</td>
</tr>
<tr>
<td></td>
<td>GET Item to be retrieved. -get GET is mandatory. Allowed values for</td>
</tr>
<tr>
<td></td>
<td>GET are: timePolicy, timeServer.</td>
</tr>
<tr>
<td>Time.error</td>
<td>Description follows in next version.</td>
</tr>
<tr>
<td>Time.NTP.sync</td>
<td>Description follows in next version.</td>
</tr>
<tr>
<td>Time.service.status</td>
<td>Description follows in next version.</td>
</tr>
</tbody>
</table>
6. The ANavS Binary Solution Output Format

The ANavS® sensor fusion solution provides two different output-streams for the position, attitude, velocity and many more states of solution and quality of solution, the standardized NMEA-Format and the proprietary binary protocol. In the following, the proprietary ANavS® binary protocol is described.

By default, it is possible to configure the system simultaneously to write the solution into a file and broadcast the solution via TCP/IP on port 6001. To stream the solution via Wi-Fi, Ethernet or mobile network, the user needs additionally to the defined port-number the IP-address of the module, which is by default 192.168.42.1 in case you are connected with Wi-Fi Access-Point “ANavS_MSRTK_XXX”.

6.1. The Standard Binary Solution Message

The Binary Protocol is defined as follows:

<table>
<thead>
<tr>
<th>Structure:</th>
<th>Sync Char 1</th>
<th>Sync Char 2</th>
<th>Class</th>
<th>ID</th>
<th>Length</th>
<th>Payload</th>
<th>ChecksumA</th>
<th>ChecksumB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bytes:</td>
<td>1 Byte</td>
<td>1 Byte</td>
<td>1 Byte</td>
<td>1 Byte</td>
<td>2 Byte</td>
<td>Length-Byte</td>
<td>1 Byte</td>
<td>1 Byte</td>
</tr>
</tbody>
</table>

The endianness of Length and the following Payload is little endian. The payload is of Length. The checksum is calculated with the 16-Bit Fletcher algorithm over Class, ID, Length and Payload with modulo 256. The ANavS® binary protocol message can be identified by Sync Char 1 = 0xB5 (dec 181), Sync Char 2 = 0x62 (dec 98), Class = 0x02 (dec 2) and Id = 0xE0 (dec 224).

In the following, the type of double has 8 bytes. If a variable in the Payload is given in the NED frame, the variable has three components (hence 3 times the data type, e.g. 3*double), where the components are north, east and down. This also holds for the body frame, which is the fixed frame of the rover, and where the components are given in x, y and z.

6.1.1. The Payload

The Payload is given as:

<table>
<thead>
<tr>
<th>Size</th>
<th>Scaling</th>
<th>Name</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint8</td>
<td>–</td>
<td>id</td>
<td>–</td>
<td>Identifier of the system/the ANavS Position and Attitude Determination (PAD) solution.</td>
</tr>
<tr>
<td>Data Type</td>
<td>Field</td>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>-------</td>
<td>-------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>uint16</td>
<td>resCode</td>
<td>Result code bitfield, which keeps the system status and information.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>uint16</td>
<td>week</td>
<td>Week number of the current epoch (epoch means Kalman filter state-update with GNSS, IMU or another sensor data).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>double</td>
<td>tow</td>
<td>Time of Week of the current epoch.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>uint16</td>
<td>weekInit</td>
<td>Week number of the epoch when the system was started.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>double</td>
<td>towInit</td>
<td>Time of Week of the epoch when the system was started.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>int16</td>
<td></td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>double</td>
<td>lat</td>
<td>Latitude in WGS84.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>double</td>
<td>lon</td>
<td>Longitude in WGS84.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>double</td>
<td>height</td>
<td>Height in WGS84 with geoid EGM96.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>double</td>
<td>ECEF-X</td>
<td>X-position in ECEF-coordinate frame (WGS84).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>double</td>
<td>ECEF-Y</td>
<td>Y-position in ECEF-coordinate frame (WGS84).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>double</td>
<td>ECEF-Z</td>
<td>Z-position in ECEF-coordinate frame (WGS84).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3*double</td>
<td>b</td>
<td>Baseline in NED frame spanned by the position given by lat, lon and height, and by the position of the reference station.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3*double</td>
<td>bStdDev</td>
<td>Standard deviation of the baseline.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3*double</td>
<td>vel</td>
<td>Velocity in NED frame.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Reference Guide

#### Multi-Sensor RTK Module, RTK Reference Station and ISP

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>3*double</td>
<td>velStdDev</td>
<td>m/s</td>
</tr>
<tr>
<td>3*double</td>
<td>acc</td>
<td>m/s²</td>
</tr>
<tr>
<td>3*double</td>
<td>accStdDev</td>
<td>m/s²</td>
</tr>
<tr>
<td>3*double</td>
<td>att</td>
<td>deg</td>
</tr>
<tr>
<td>3*double</td>
<td>attStdDev</td>
<td>deg</td>
</tr>
<tr>
<td>double</td>
<td>accuracy</td>
<td>m</td>
</tr>
<tr>
<td>double</td>
<td>systemTimeSolOut</td>
<td>s</td>
</tr>
<tr>
<td>5*double</td>
<td>timingInfo</td>
<td>s</td>
</tr>
<tr>
<td>double</td>
<td>systemTimePrevPAD</td>
<td>s</td>
</tr>
<tr>
<td>double</td>
<td>systemTimePostPAD</td>
<td>s</td>
</tr>
<tr>
<td>3*double</td>
<td>accNed</td>
<td>m/s²</td>
</tr>
<tr>
<td>1*double</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>uint8</td>
<td>numSats</td>
<td>--</td>
</tr>
</tbody>
</table>

**Standard deviation of the velocity.**

**Acceleration in body frame.**

**Standard deviation of the acceleration.**

**Attitude/Euler angles (heading, pitch, roll).**

**Standard deviation of the attitude.**

**Estimated accuracy of the baseline.**

**CPU-Load of used sensors.** The sum (green line in the GUI) of elapsed time should not over 1 second (→ to slow processor).

- First double: Elapsed time GNSS;
- Second double: Elapsed time IMU;
- Third double: Elapsed time Baro;
- Fourth double: Elapsed time Odometry;
- Fifth double: Overall elapsed time

**System time before executing sensor data**

**System time after executing sensor data**

**Acceleration in NED frame.**

**Number of satellites.**
| Loop | uint8 | -- | gnssId | -- | Identifier of the GNSS (GPS = 1, SBAS = 2, GLONASS = 4, Galileo = 8).
| uint8 | -- | svId | -- | Identifier of the satellite (“PRN”).
| double | -- | elev | deg | Elevation angle of the satellite.
| double | -- | azim | deg | Azimuth angle of the satellite.
| uint8 | -- | numRcv | -- | Number of GNSS receivers.
| Loop | uint8 | -- | rcvId | -- | Identifier of the receiver (unique within this system).
| bool | -- | isRefStation | -- | Is true if this receiver is a reference station (stationary).
| uint16 | -- | week | -- | Week number of the current epoch for this receiver. If the following tow is NaN, the week number is not valid!
| double | -- | tow | s | Time of week of the current epoch for this receiver. Can be NaN, if there are no measurements at this epoch (e.g. commonly for reference station).
| double | -- | lat | deg | Latitude of this receiver. Only given if the receiver is a reference station, otherwise it is NaN.
### Multi-Sensor RTK Module, RTK Reference Station and ISP

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>double</td>
<td>lon</td>
<td>Longitude of this receiver. Only given if the receiver is a reference station, otherwise it is NaN.</td>
</tr>
<tr>
<td>double</td>
<td>height</td>
<td>Height of this receiver. Only given if the receiver is a reference station, otherwise it is NaN.</td>
</tr>
<tr>
<td>3*double</td>
<td>bodyPos</td>
<td>Receiver position in x, y and z (body frame).</td>
</tr>
<tr>
<td>3*double</td>
<td>bodyMisalign</td>
<td>Misalignment of the receiver in x, y and z (body frame).</td>
</tr>
<tr>
<td>uint8</td>
<td></td>
<td>Reserved.</td>
</tr>
<tr>
<td>uint8</td>
<td>numSatsMeas</td>
<td>Number of satellites for which measurements are available for this receiver.</td>
</tr>
<tr>
<td>uint8</td>
<td>gnssId</td>
<td>Identifier of the GNSS.</td>
</tr>
<tr>
<td>uint8</td>
<td>svId</td>
<td>Identifier of the satellite.</td>
</tr>
<tr>
<td>uint8</td>
<td>freq</td>
<td>Frequency band (currently that is 1 L1).</td>
</tr>
<tr>
<td>uint16</td>
<td>locktime</td>
<td>Carrier phase locktime counter (maximum 64500ms).</td>
</tr>
<tr>
<td>uint8</td>
<td>cno</td>
<td>Carrier-to-noise density ratio (signal strength) [dB-Hz].</td>
</tr>
<tr>
<td>Field</td>
<td>Type</td>
<td>Value</td>
</tr>
<tr>
<td>--------------</td>
<td>----------</td>
<td>------------------------</td>
</tr>
<tr>
<td>prStdDev</td>
<td>uint8</td>
<td>0.01*2^n</td>
</tr>
<tr>
<td>cpStdDev</td>
<td>uint8</td>
<td>0.004</td>
</tr>
<tr>
<td>doStdDev</td>
<td>uint8</td>
<td>0.002*2^n</td>
</tr>
<tr>
<td>trkStat</td>
<td>uint8</td>
<td>–</td>
</tr>
<tr>
<td>numBl</td>
<td>uint8</td>
<td>–</td>
</tr>
<tr>
<td>rcvId1</td>
<td>uint8</td>
<td>–</td>
</tr>
<tr>
<td>rcvId2</td>
<td>uint8</td>
<td>–</td>
</tr>
<tr>
<td>isFixed</td>
<td>bool</td>
<td>–</td>
</tr>
<tr>
<td>gnssIdJointRefSat</td>
<td>uint8</td>
<td>–</td>
</tr>
<tr>
<td>svIdJointRefSat</td>
<td>uint8</td>
<td>–</td>
</tr>
<tr>
<td>svIdGloRefSat</td>
<td>uint8</td>
<td>–</td>
</tr>
<tr>
<td>svIdUnlSat</td>
<td>uint8</td>
<td>–</td>
</tr>
<tr>
<td>aprioriBl</td>
<td>3*double</td>
<td>–</td>
</tr>
<tr>
<td>Data Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>double</td>
<td>A priori baseline length.</td>
<td></td>
</tr>
<tr>
<td>double</td>
<td>Standard deviation of the a priori baseline length.</td>
<td></td>
</tr>
<tr>
<td>uint8</td>
<td>Number of filters.</td>
<td></td>
</tr>
<tr>
<td>char[namelen]</td>
<td>Filter name.</td>
<td></td>
</tr>
<tr>
<td>uint32</td>
<td>Bitfield of parameters (states, residuals), which are transmitted by the filter.</td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td>Is true if the filter is active.</td>
<td></td>
</tr>
<tr>
<td>double</td>
<td>Absolute position in latitude, longitude (both in deg) and height (in m) (first 3 doubles) and its standard deviation (latter 3 doubles).</td>
<td></td>
</tr>
<tr>
<td>double</td>
<td>Receiver clock error (first double) and its standard deviation (latter double).</td>
<td></td>
</tr>
<tr>
<td>uint8</td>
<td>Identifier of the receiver the baseline is pointing to.</td>
<td></td>
</tr>
<tr>
<td>uint8</td>
<td>Identifier of the receiver the baseline is pointing from.</td>
<td></td>
</tr>
</tbody>
</table>

### Additional Notes:
- Bit 2:
  - `uint8` `rcvId1` - Identifier of the receiver the baseline is pointing to.
  - `uint8` `rcvId2` - Identifier of the receiver the baseline is pointing from.
### Reference Guide

**Multi-Sensor RTK Module, RTK Reference Station and ISP**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Data Type</th>
<th>Value</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>6*double</td>
<td>baseline</td>
<td>m</td>
<td>Baseline in NED frame (first 3 doubles) and its standard deviation (latter 3 doubles).</td>
</tr>
<tr>
<td>1</td>
<td>6*double</td>
<td>vel</td>
<td>m/s</td>
<td>Velocity in body frame (first 3 doubles) and its standard deviation (latter 3 doubles).</td>
</tr>
<tr>
<td>2</td>
<td>6*double</td>
<td>acc</td>
<td>m/s²</td>
<td>Acceleration in body frame (first 3 doubles) and its standard deviation (latter 3 doubles).</td>
</tr>
<tr>
<td>3</td>
<td>6*double</td>
<td>accBias</td>
<td>m/s²</td>
<td>Accelerometer bias in body frame (first 3 doubles) and its standard deviation (latter 3 doubles).</td>
</tr>
<tr>
<td>4</td>
<td>6*double</td>
<td>eulerAng</td>
<td>deg</td>
<td>Euler angles in heading, pitch and roll (first 3 doubles) and its standard deviation (latter 3 doubles).</td>
</tr>
<tr>
<td>5</td>
<td>6*double</td>
<td>angRate</td>
<td>deg/s</td>
<td>Angular rate in heading, pitch and roll (first 3 doubles) and its standard deviation (latter 3 doubles).</td>
</tr>
<tr>
<td>6</td>
<td>6*double</td>
<td>gyroBias</td>
<td>deg/s</td>
<td>Gyroscope bias (first 3 doubles) and its standard deviation (latter 3 doubles).</td>
</tr>
<tr>
<td>7</td>
<td>2*double</td>
<td>tropoZenDel</td>
<td>m</td>
<td>Tropospheric zenith delay (first double) and its standard deviation (latter double). Currently not in use (Bit 9 is always 0).</td>
</tr>
</tbody>
</table>
### Reference Guide

**Multi-Sensor RTK Module, RTK Reference Station and ISP**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Data Type</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>2*double</td>
<td>Accuracy m</td>
<td>Accuracy of the baseline (first double) and its standard deviation (latter double). Currently not in use (Bit 10 is always 0).</td>
</tr>
<tr>
<td>11-17</td>
<td>7*double</td>
<td>Reserved</td>
<td>Currently not in use (Bit 11-17 is always 0).</td>
</tr>
<tr>
<td></td>
<td>uint8</td>
<td><strong>numPhase</strong></td>
<td>Number of phase measurements.</td>
</tr>
<tr>
<td>Loop</td>
<td>uint8</td>
<td>gnssId</td>
<td>Identifier of the GNSS.</td>
</tr>
<tr>
<td></td>
<td>uint8</td>
<td>svId</td>
<td>Identifier of the satellite.</td>
</tr>
<tr>
<td></td>
<td>uint8</td>
<td>freq</td>
<td>Frequency band (currently that is 1 (L1)).</td>
</tr>
<tr>
<td>18</td>
<td>2*double</td>
<td>ambiguities cycles</td>
<td>Double difference ambiguities and std. dev.</td>
</tr>
<tr>
<td>19</td>
<td>2*double</td>
<td>phaseMp m</td>
<td>Phase multipath and std. dev.</td>
</tr>
<tr>
<td>20</td>
<td>2*double</td>
<td>phaseRes m</td>
<td>Phase residuals and std. dev.</td>
</tr>
<tr>
<td></td>
<td>uint8</td>
<td><strong>numCode</strong></td>
<td>Number of code measurements.</td>
</tr>
<tr>
<td>Loop</td>
<td>uint8</td>
<td>gnssId</td>
<td>Identifier of the GNSS.</td>
</tr>
<tr>
<td></td>
<td>uint8</td>
<td>svId</td>
<td>Identifier of the satellite.</td>
</tr>
<tr>
<td></td>
<td>uint8</td>
<td>freq</td>
<td>Frequency band (currently that is 1 (L1)).</td>
</tr>
<tr>
<td>21</td>
<td>2*double</td>
<td>codeMp m</td>
<td>First double is code multipath, second its standard deviation.</td>
</tr>
<tr>
<td>22</td>
<td>2*double</td>
<td>codeRes m</td>
<td>First double is code residual, second its standard deviation.</td>
</tr>
<tr>
<td></td>
<td>uint8</td>
<td><strong>numDoppler</strong></td>
<td>Number of Doppler measurements.</td>
</tr>
<tr>
<td>Loop</td>
<td>uint8</td>
<td>gnssId</td>
<td>Identifier of the GNSS.</td>
</tr>
<tr>
<td></td>
<td>uint8</td>
<td>svId</td>
<td>Identifier of the satellite.</td>
</tr>
</tbody>
</table>
The payload contains loops and conditions. The number of loop iterations and the conditions depends on other payload data. For example, the number of satellites determines, how often the following color-coded data is repeated. For numSats = 2 and p being a pointer to numSats, the following data is given as:

<table>
<thead>
<tr>
<th>Address</th>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>p+1</td>
<td>uint8</td>
<td>gnssId1</td>
</tr>
<tr>
<td>p+2</td>
<td>uint8</td>
<td>svId1</td>
</tr>
<tr>
<td>p+3</td>
<td>double</td>
<td>elev1</td>
</tr>
<tr>
<td>p+11</td>
<td>double</td>
<td>azim1</td>
</tr>
<tr>
<td>p+19</td>
<td>uint8</td>
<td>gnssId2</td>
</tr>
<tr>
<td>p+20</td>
<td>uint8</td>
<td>svId2</td>
</tr>
<tr>
<td>p+21</td>
<td>double</td>
<td>elev2</td>
</tr>
<tr>
<td>p+29</td>
<td>double</td>
<td>azim2</td>
</tr>
<tr>
<td>p+37</td>
<td>uint8</td>
<td>numRcv</td>
</tr>
</tbody>
</table>

For numSats = 0, no satellite position data is transmitted, and the next variable would be numRcv. If a condition is fulfilled, the following color-coded data is transmitted. For example, if isActive is true, then at least the variables numPhase, numCode and numDoppler are transmitted. If isActive is false, then the end of the binary message is reached. The bitfield params is also a condition, whose bits determine, whether the following data is transmitted. **Example:** Let p being a pointer to isActive with isActive = true, Bit 0 of params is false and Bit 1 is true. Then, p+1 points to the receiver clock error clkErr and not to the absolute position absPos since Bit 0 is false.

### 6.1.2. The Result-Code

The resCode bitfield in the solution message is defined as:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Is set for <strong>GNSS</strong> update-epochs processed in the PAD-Software</td>
</tr>
<tr>
<td>1</td>
<td>Is set for <strong>IMU</strong> update-epochs processed in the PAD-Software</td>
</tr>
<tr>
<td>2</td>
<td>Is set for <strong>Barometer</strong> update-epochs processed in the PAD-Software</td>
</tr>
<tr>
<td>3</td>
<td>Is set for <strong>Odometry</strong> update-epochs processed in the PAD-Software</td>
</tr>
<tr>
<td>4</td>
<td>Is set for <strong>Camera</strong> update-epochs processed in the PAD-Software</td>
</tr>
</tbody>
</table>
5  Is set for **Steering** update-epochs processed in the PAD-Software
6  Is set if the IMU is initialized/calibrated
7  Is set if the Multi-Sensor RTK-Filter is reset in the current epoch
8  Is set if RTK correction were received initially
9 to 10  State-definition for the Attitude Kalman-Filter:
   00 -> No solution
   01 -> Least-Squares solution
   10 -> Float solution
   11 -> Fixed solution
11 to 12  State-definition for the RTK-Position Kalman-Filter:
   00 -> No solution
   01 -> Least-Squares solution
   10 -> Float solution
   11 -> Fixed solution
13  Is set if the RTK-Position Kalman-Filter has only a float-solution accuracy but enough satellites in view to try a RTK-refix (the vehicle must be stationary for this try!)
14  Is set if state update is only a prediction step (without sensor data) in our sensor fusion filter. In this case, Bit 0-5 is set to zero.
15  Reserved

### 6.2. The Extended Integrity Information Message

To extend the existing binary solution message format with more integrity information, ANavS provides since version 5.1.72 an additional message with the following header:

`Sync Char 1 = 0xB5 (dec 181), Sync Char 2 = 0x62 (dec 98), Class = 0x02 (dec 2) and Id = 0xE5 (dec 229).`

The Message is activated via **customer-code 10**.

#### 6.2.1. The Payload

The Payload is given as:

<table>
<thead>
<tr>
<th>Size</th>
<th>Scaling</th>
<th>Name</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint16</td>
<td>–</td>
<td>week</td>
<td>–</td>
<td>Week number of the current epoch</td>
</tr>
<tr>
<td>double</td>
<td>–</td>
<td>tow</td>
<td>s</td>
<td>Time of Week of the current epoch</td>
</tr>
<tr>
<td>double</td>
<td>–</td>
<td>cov_b_xx</td>
<td>–</td>
<td>Covariance matrix of RTK solution (value xx)</td>
</tr>
<tr>
<td>double</td>
<td>–</td>
<td>cov_b_xy</td>
<td>–</td>
<td>Covariance matrix of RTK solution (value xy)</td>
</tr>
<tr>
<td>double</td>
<td>–</td>
<td>cov_b_xz</td>
<td>–</td>
<td>Covariance matrix of RTK solution (value xz)</td>
</tr>
<tr>
<td>double</td>
<td>–</td>
<td>cov_b_yx</td>
<td>–</td>
<td>Covariance matrix of RTK solution (value yx)</td>
</tr>
<tr>
<td>double</td>
<td>–</td>
<td>cov_b_yy</td>
<td>–</td>
<td>Covariance matrix of RTK solution (value yy)</td>
</tr>
<tr>
<td>double</td>
<td>–</td>
<td>cov_b_yz</td>
<td>–</td>
<td>Covariance matrix of RTK solution (value yz)</td>
</tr>
<tr>
<td>double</td>
<td>–</td>
<td>cov_b_zx</td>
<td>–</td>
<td>Covariance matrix of RTK solution (value zx)</td>
</tr>
<tr>
<td>double</td>
<td>–</td>
<td>cov_b_zy</td>
<td>–</td>
<td>Covariance matrix of RTK solution (value zy)</td>
</tr>
<tr>
<td>double</td>
<td>–</td>
<td>cov_b_zz</td>
<td>–</td>
<td>Covariance matrix of RTK solution (value zz)</td>
</tr>
<tr>
<td>double</td>
<td>–</td>
<td>GDOP</td>
<td>–</td>
<td>Dilution of Precision: GDOP</td>
</tr>
</tbody>
</table>
### Reference Guide

**Multi-Sensor RTK Module, RTK Reference Station and ISP**

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>double</td>
<td>PDOP</td>
<td>Dilution of Precision: PDOP</td>
</tr>
<tr>
<td>double</td>
<td>HDOP</td>
<td>Dilution of Precision: HDOP</td>
</tr>
<tr>
<td>double</td>
<td>VDOP</td>
<td>Dilution of Precision: VDOP</td>
</tr>
<tr>
<td>double</td>
<td>TDOP</td>
<td>Dilution of Precision: TDOP</td>
</tr>
<tr>
<td>bool</td>
<td>Flag: No Movement</td>
<td><strong>True</strong>: Rover stand still, no movement detected with IMU -&gt; RTK- and Attitude Fix is triggered, bias estimation of IMU also. <strong>False</strong>: Rover is moving -&gt; NO RTK- and Attitude Fix is triggered, NO bias estimation of IMU</td>
</tr>
<tr>
<td>double</td>
<td>RTK-Fix Validation</td>
<td>Difference of successfully fixed RTK-baseline to the next best candidate. Only if “Flag No Movement” is true, this value is relevant for integrity information of RTK baseline.</td>
</tr>
<tr>
<td>double</td>
<td>Correction-Outage</td>
<td>Time since last RTK correction-data arrived.</td>
</tr>
</tbody>
</table>
7. The NMEA Solution Output Format

The ANavS® sensor fusion solution provides two different output-streams for the position, attitude, velocity and many more states of solution and quality of solution, the standardized NMEA-Format and the proprietary binary protocol. In the following, the NMEA-format is described.

By default, it is possible to configure the system simultaneously to write the solution into a file and broadcast the NMEA solution via TCP/IP on port 6002. To stream the solution via Wi-Fi, Ethernet or mobile network, the user needs additionally to the defined port-number the IP-address of the module, which is by default 192.168.42.1 in case you are connected with Wi-Fi Access-Point “ANavS_MSRTK_XXX”.

7.1. The NMEA-Format

This protocol is based on the international standard for maritime navigation and radio communication, equipment and systems and digital interfaces (IEC 61161-1)\(^8\). This standard adopted the de-facto standards for interfacing marine electronic devices, known as NMEA 0183. The data is transmitted in sentences of variable length with a specified sentence structure.

7.1.1. Sentence Structure

- Address field
- Data fields
- Checksum field
- Terminating field
- All sentences contain only ASCII characters
- The maximum length of a sentence is 82 characters
- All fields are separated by delimiters

7.1.2. Address field

The address field starts with “$” followed by the talker ID and a sentence identifier. The used talker IDs are:

- GP for GPS only solutions
- GL for GLONASS only solutions

---

The used sentence identifiers are:

- **GGA** – Global Positioning System Fix Data
- **VTG** – Course over Ground and Ground Speed
- **GSA** – GNSS DOP and Active Satellites
- **GSV** – GNSS Satellites in View
- **RMC** – Recommended Minimum Specific GNSS Data
- **ZDA** – Time and Date
- **PASHR** – Attitude Data
- **ROT** – Rate of Turn Data
- **THS** – True Heading State Data

### 7.1.3. Data fields

Data fields must always be separated by “,”. They can contain alpha, numeric, and alphanumeric values all coded in ASCII characters. The length of a data field can be constant, variable or can contain a fixed and variable portion. This differs for each sentence.

### 7.1.4. Checksum field

The Checksum field starts with “*” followed by the checksum of the sentence. The Checksum is generated with a bitwise exclusive OR of all fields including the “,” delimiters, between but not including the “$” and the “*” characters. The hexadecimal value of the checksum is then converted to two ASCII characters.

### 7.1.5. Terminating field

The terminating sequence contains the two ASCII characters <CR> and <LF> without any delimiter.

### 7.1.6. Satellite Numbering

- **GPS**: 1-32
- **GLONASS**: 33-96
- **GALILEO**: 301-336

\[ ^9 \text{Currently no standard way to number Galileo satellites.} \]
7.2. Sentence specification

7.2.1. GGA – Global positioning system (GPS) fix data

<table>
<thead>
<tr>
<th>TALKER ID</th>
<th>XX</th>
<th>All talker IDs usable</th>
</tr>
</thead>
<tbody>
<tr>
<td>SENTENCE ID</td>
<td>GGA</td>
<td></td>
</tr>
<tr>
<td>UTC of position</td>
<td>hhmms.s</td>
<td>Fixed length 2 digits after dot</td>
</tr>
<tr>
<td>Latitude</td>
<td>llllllll</td>
<td>Fixed length 4 digits before and 7 after dot</td>
</tr>
<tr>
<td>Hemisphere of latitude</td>
<td>N/S</td>
<td>N if value of latitude is positive</td>
</tr>
<tr>
<td>Longitude</td>
<td>llllllll</td>
<td>Fixed length 5 digits before and 7 after dot</td>
</tr>
<tr>
<td>Hemisphere of longitude</td>
<td>E/W</td>
<td>E if value of longitude is positive</td>
</tr>
<tr>
<td>GPS quality indicator</td>
<td>X</td>
<td>0: GNSS fix not available 1: GNSS fix valid 4: RTK fixed ambiguities 5: RTK float ambiguities</td>
</tr>
<tr>
<td>Number of satellites used for positioning</td>
<td>XX</td>
<td>Fixed length 01 for single digits</td>
</tr>
<tr>
<td>HDOP</td>
<td>XX.X</td>
<td>Variable/fixed length 1 digit after dot, variable before</td>
</tr>
<tr>
<td>Altitude geoid height</td>
<td>(-)X.XX</td>
<td>Variable/fixed length 2 digits after dot, variable before</td>
</tr>
<tr>
<td>Unit of altitude</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>Geoidal separation</td>
<td>(-)X.XX</td>
<td>Variable/fixed length 2 digits after dot, variable before</td>
</tr>
<tr>
<td>Unit of geoidal separation</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>Age of differential data</td>
<td>XX.X</td>
<td>Age of RTK corrections. Empty if RTK corrections never received. Variable/fixed length: 1 digit after dot, variable before</td>
</tr>
<tr>
<td>Differential reference station ID</td>
<td>Empty field</td>
<td></td>
</tr>
</tbody>
</table>

Example:
$GNGGA,185833.80,4808.7402397,N,01133.9325039,E,5,15,1.1,470.50,M,45.65,M,,*75$

7.2.2. VTG – Course over ground and ground speed

<table>
<thead>
<tr>
<th>TALKER ID</th>
<th>XX</th>
<th>All talker IDs usable</th>
</tr>
</thead>
<tbody>
<tr>
<td>SENTENCE ID</td>
<td>VTG</td>
<td></td>
</tr>
<tr>
<td>Course over ground</td>
<td>X.XX</td>
<td>Variable/fixed length 2 digits after dot, variable before Values from 0 to 359.99</td>
</tr>
<tr>
<td>Degrees</td>
<td>T</td>
<td>True course</td>
</tr>
<tr>
<td>Course over ground</td>
<td>X.XX</td>
<td>Variable/fixed length 2 digits after dot, variable before Values from 0 to 359.99</td>
</tr>
<tr>
<td>Degrees</td>
<td>M</td>
<td>Magnetic course</td>
</tr>
<tr>
<td>Speed over ground</td>
<td>X.XX</td>
<td>Variable/fixed length 2 digits after dot, variable before</td>
</tr>
<tr>
<td>Unit</td>
<td>N</td>
<td>knots</td>
</tr>
</tbody>
</table>
The VTG sentence is empty ( $GNVTG,,T,,M,,N,,K,A*3D ) until attitude baseline is valid. Course over ground equals heading.

Example:
$GNVTG,112.99,T,109.99,M,0.15,N,0.08,K,A*3B

### 7.2.3. GSA – GNSS DOP and active satellites

<table>
<thead>
<tr>
<th>TALKER ID</th>
<th>XX</th>
<th>All talker IDs usable*</th>
</tr>
</thead>
<tbody>
<tr>
<td>SENTENCE ID</td>
<td>GSA</td>
<td></td>
</tr>
<tr>
<td>MODE</td>
<td>X</td>
<td>1: GNSS fix not available</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3: 3D</td>
</tr>
<tr>
<td>MODE</td>
<td>XX</td>
<td>M: forced to operate in 3D</td>
</tr>
<tr>
<td>ID number</td>
<td>XX</td>
<td>Fixed length: 01 for single digit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Up to 12 satellites per constellation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Empty field if not used for positioning</td>
</tr>
<tr>
<td></td>
<td>XX</td>
<td>GPS: ID is PRN (1-32)</td>
</tr>
<tr>
<td></td>
<td>XX</td>
<td>GLONASS: ID is slot number + 64</td>
</tr>
<tr>
<td></td>
<td>XX</td>
<td>GALILEO: PRN + 300</td>
</tr>
<tr>
<td>PDOP</td>
<td>XX.XX</td>
<td>Fixed length</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 digits before and after dot</td>
</tr>
</tbody>
</table>

*if GN is used for Talker ID a separate sentence must be created for each GNSS constellation all starting with the Talker ID for multi GNSS.

Example:
$GNGSA,2,M,06,12,15,17,19,24,25,32,1.34,0.96,0.93*1D
$GNGSA,2,M,70,71,79,80,81,82,88,1.34,0.96,0.93*3A

### 7.2.4. GSV – GNSS satellites in view

<table>
<thead>
<tr>
<th>TALKER ID</th>
<th>XX</th>
<th>GN must not be used*</th>
</tr>
</thead>
<tbody>
<tr>
<td>SENTENCE ID</td>
<td>GSV</td>
<td></td>
</tr>
<tr>
<td>Total number of messages</td>
<td>X</td>
<td>1-9</td>
</tr>
<tr>
<td>Message number</td>
<td>X</td>
<td>1-9</td>
</tr>
<tr>
<td>Total number of satellites in view</td>
<td>XX</td>
<td>Fixed length: 01 for single digit</td>
</tr>
</tbody>
</table>
Satellite ID number | XX | Fixed length: 01 for single digit Empty field if not used for positioning GPS: ID is PRN (1-32) GLONASS: ID is slot number + 64 GALILEO: PRN + 300

Elevation | XX | Fixed length: 00 for 0° elevation Values from 0 to 90 Empty if not used

Azimuth | XXX | Fixed length: 000 for 0° azimuth Values from 000 to 360 Empty if not used

SNR | XX | Fixed length: 05 for 5 db/Hz Values from 0-99 Empty if not used

*if multi GNSS is used a separate GSV sentence must be created for each constellation starting with the constellation specific talker ID.

This block is repeated 4 times per sentence in total. For more than multiples of 4 a new sentence is started each time. Blocks are left empty if the number of satellites in view is lower than a multiple of 4.

Example:

$GPGSV,6,1,10,02,3.6,133.2,26,10,06,11.7,100.7,39,10,09,6.2,281.5,35,10,12,63.1,256.5,46*58$
$GPGSV,6,2,10,15,26.5,186.0,43,10,17,30.5,48.7,42,10,19,43.9,65.3,46,10,24,86.5,103.6,46*5E$
$GPGSV,6,3,10,25,21.6,250.8,43,10,32,21.7,316.0,41,,,,,,,,,,,*5E$
$GLGSV,6,4,09,69,7.0,215.9,30,09,70,30.8,267.4,44,09,71,23,0,324.4,46,09,73,13.0,286.8,33*72$
$GLGSV,6,5,09,79,47.8,70.6,43,09,80,54.9,314.5,38,09,81,48.6,86.8,43,09,82,28.4,150.8,46*49$
$GLGSV,6,6,09,88,21.3,280.4,40,,,,,,,,,,,*4E$

7.2.5. RMC – Recommended minimum specific GNSS data

<table>
<thead>
<tr>
<th>TALKER ID</th>
<th>XX</th>
<th>All talker IDs usable</th>
</tr>
</thead>
<tbody>
<tr>
<td>SENTENCE ID</td>
<td>RMC</td>
<td></td>
</tr>
<tr>
<td>UTC time</td>
<td>hhmms.s</td>
<td>Fixed length</td>
</tr>
<tr>
<td>Status</td>
<td>X</td>
<td>A: data valid</td>
</tr>
<tr>
<td>Latitude</td>
<td>lllllllll1</td>
<td>Fixed length 4 digits before and 7 after dot</td>
</tr>
<tr>
<td>Hemisphere of latitude</td>
<td>N/S</td>
<td>N if value of latitude is positive</td>
</tr>
</tbody>
</table>
### Longitude

| Longitude | Fixed length  
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ll.lllll</td>
<td>5 digits before and 7 after dot</td>
</tr>
</tbody>
</table>

### Hemisphere of longitude

<table>
<thead>
<tr>
<th>Hemisphere of longitude</th>
<th>E/W</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>E if value of longitude is positive</td>
</tr>
</tbody>
</table>

### Speed over ground

<table>
<thead>
<tr>
<th>Speed over ground</th>
<th>Variable/fixed length</th>
</tr>
</thead>
<tbody>
<tr>
<td>X.XX</td>
<td>2 digits after dot, variable before</td>
</tr>
</tbody>
</table>

### Course over ground

| Course over ground | Variable/fixed length  
|--------------------|------------------------|
| X.XX               | 2 digits after dot, variable before  
|                    | Values from 0 to 359.99 |

### Date

<table>
<thead>
<tr>
<th>Date</th>
<th>dmmmyy</th>
</tr>
</thead>
</table>

### Magnetic variation

| Magnetic variation | Variable/fixed length  
|--------------------|------------------------|
| X.XX               | 2 digits after dot, variable before  
|                    | E if variation positive |

### Mode indicator

| Mode indicator | A: Autonomous  
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>D: Differential</td>
</tr>
</tbody>
</table>

#### Example:

$\text{GNRMC},185823.40,A,4808.7402374,N,01133.9324760,E,0.00,112.64,130117,3.00,E,A*14$

#### 7.2.6. ZDA – Time and date

<table>
<thead>
<tr>
<th>TALKER ID</th>
<th>XX</th>
</tr>
</thead>
<tbody>
<tr>
<td>SENTENCE ID</td>
<td>ZDA</td>
</tr>
<tr>
<td>UTC time</td>
<td>hhmmss.ss</td>
</tr>
<tr>
<td>Day</td>
<td>XX</td>
</tr>
<tr>
<td>Month</td>
<td>XX</td>
</tr>
<tr>
<td>Year</td>
<td>XXXX</td>
</tr>
<tr>
<td>Local zone hours</td>
<td>Empty field</td>
</tr>
<tr>
<td>Local zone minutes</td>
<td>Empty field</td>
</tr>
</tbody>
</table>

#### Example:

$\text{GNZDA},185823.40,13,01,2017,,,*7E$

#### 7.2.7. PASHR – Attitude Data

<table>
<thead>
<tr>
<th>TALKER ID</th>
<th>No talker ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>SENTENCE ID</td>
<td>PASHR</td>
</tr>
<tr>
<td>UTC time</td>
<td>hhmmss.ss</td>
</tr>
</tbody>
</table>

### Issue: 2.19

Date: 30.04.2024

Page 115 of 157

Reference Guide Document
## Reference Guide

### Multi-Sensor RTK Module, RTK Reference Station and ISP

<table>
<thead>
<tr>
<th>Heading*</th>
<th>XXX.XX</th>
<th>Fixed value: 000.00 for 0° 3 digits before dot, 2 after</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roll angle*</td>
<td>(-)XXX.XX</td>
<td>True heading</td>
</tr>
<tr>
<td>Pitch angle*</td>
<td>(-)XXX.XX</td>
<td>Fixed value: 000.00 for 0° 3 digits before dot, 2 after</td>
</tr>
<tr>
<td>Heave</td>
<td>Empty field</td>
<td></td>
</tr>
<tr>
<td>Roll standard deviation*</td>
<td>XX.XXX</td>
<td>Fixed value 2 digits before dot 3 after</td>
</tr>
<tr>
<td>Pitch standard deviation*</td>
<td>XX.XXX</td>
<td>2 digits before dot 3 after</td>
</tr>
<tr>
<td>Heading standard deviation*</td>
<td>XX.XXX</td>
<td></td>
</tr>
<tr>
<td>Quality flag</td>
<td>X</td>
<td>0: No position 1: RTK float position 2: RTK fixed position</td>
</tr>
</tbody>
</table>

* attitude angles and corresponding deviation values are only filled for defined setup (3D, 2D)

Example:

$\$PASHR,190558.56,107.09,T,-0.16,,0.067,0.056,2*34$

### 7.2.8. ROT – Rate of Turn Data

<table>
<thead>
<tr>
<th>TALKER ID</th>
<th>XX</th>
</tr>
</thead>
<tbody>
<tr>
<td>SENTENCE ID</td>
<td>ROT</td>
</tr>
<tr>
<td>Rate of turn</td>
<td>(-)X.X</td>
</tr>
<tr>
<td>Valid/Invalid</td>
<td>X</td>
</tr>
</tbody>
</table>

Example:

$\$GNROT,35.6,A*4E$

### 7.2.9. THS – True Heading State Data

<table>
<thead>
<tr>
<th>TALKER ID</th>
<th>XX</th>
</tr>
</thead>
<tbody>
<tr>
<td>SENTENCE ID</td>
<td>ROT</td>
</tr>
<tr>
<td>True Heading</td>
<td>XXX.XX</td>
</tr>
<tr>
<td>Valid/Invalid</td>
<td>X</td>
</tr>
</tbody>
</table>

Example:

$\$GNTHS,35.6,A*4E$

**Issue:** 2.19  
**Date:** 30.04.2024  
**Page 116 of 157**  
Reference Guide Document
8. CAN-Interface

CAN is a robust and widely spread data bus standard which is among other applications also used in almost all modern vehicles for communication between the various controllers in it.

The ANavS Positioning Systems come with a CAN Interface that allows the user to input wheel odometry values and/or to access the solution data over that interface. When equipped, the Interface can be accessed by the DE-9 connector on the module. The pin assignment is in accordance with CiA® 303-1 and given below:

![Figure 55: pin assignment is in accordance with CiA® 303-1](image)

<table>
<thead>
<tr>
<th>Pin</th>
<th>Pin assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Not connected</td>
</tr>
<tr>
<td>2</td>
<td>CAN-L</td>
</tr>
<tr>
<td>3</td>
<td>GND</td>
</tr>
<tr>
<td>4</td>
<td>Not connected</td>
</tr>
<tr>
<td>5</td>
<td>GND</td>
</tr>
<tr>
<td>6</td>
<td>Not connected</td>
</tr>
<tr>
<td>7</td>
<td>CAN-H</td>
</tr>
<tr>
<td>8</td>
<td>Not connected</td>
</tr>
<tr>
<td>9</td>
<td>Not connected</td>
</tr>
</tbody>
</table>

The CAN Interface can be fully configured in the ANavS Wizard, where the whole configuration is done in the second step of the initialization process. The CAN-settings are divided in three categories: the bus settings, the solution output (CAN-Output) and the odometry input (CAN-Input). Every section has three buttons to get, set and restore the configurations of the corresponding section.

8.1. CAN Bus Settings

In this section, all physical settings of the CAN Interface are configured. These are:

- **Bus-Speed**: Select the Bit rate of the CAN bus. By default, 500 kBit/s are selected.
- **Termination**: CAN requires a 120 Ohm hardware termination at the last transceiver. The MSRTK System has a built in switchable 120 Ohm termination which can be enabled by using this slider.
8.2. CAN-Output: ANavS-Solution

The CAN Interface can be used to output the ANavS solution from the sensor fusion software. The CAN Interface sends for each solution variable a unique message containing the value. The endianness of the values is big-endian. The CAN address corresponds to the ID of the variable. The variables are bundled in groups which can be activated or deactivated by checking the corresponding boxes in the ANavS Wizard. As standard, each packet has a unique ID which is also shown in the table below. To change the IDs of the CAN messages (e.g. to avoid collisions with the information of other devices on the bus), there is the possibility to add an offset to all messages. To use the solution output via CAN, the bus must be configured in the normal mode and the output must be enabled (see section CAN Bus Settings above).

The corresponding DBC file can be found in Appendix 1.

<table>
<thead>
<tr>
<th>ID</th>
<th>Format</th>
<th>Name</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>uint16</td>
<td>resCode</td>
<td>–</td>
<td>Result code bitfield, which keeps the system status and information.</td>
</tr>
<tr>
<td>2</td>
<td>uint16</td>
<td>week</td>
<td>–</td>
<td>Week number of the current epoch.</td>
</tr>
<tr>
<td>3</td>
<td>double</td>
<td>tow</td>
<td>s</td>
<td>Time of Week of the current epoch.</td>
</tr>
<tr>
<td>4</td>
<td>uint16</td>
<td>weekInit</td>
<td>–</td>
<td>Week number of the epoch when the system was started.</td>
</tr>
<tr>
<td>5</td>
<td>double</td>
<td>towInit</td>
<td>s</td>
<td>Time of Week of the epoch when the system was started.</td>
</tr>
<tr>
<td>7</td>
<td>double</td>
<td>lat</td>
<td>deg</td>
<td>Latitude.</td>
</tr>
<tr>
<td>8</td>
<td>double</td>
<td>lon</td>
<td>deg</td>
<td>Longitude.</td>
</tr>
<tr>
<td>No.</td>
<td>Type</td>
<td>Array Length</td>
<td>Units</td>
<td>Description</td>
</tr>
<tr>
<td>-----</td>
<td>--------</td>
<td>--------------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>9</td>
<td>double</td>
<td>height</td>
<td>m</td>
<td>X-position in ECEF-coordinate frame</td>
</tr>
<tr>
<td>10</td>
<td>double</td>
<td>ECEF-X</td>
<td>m</td>
<td>X-position in ECEF-coordinate frame</td>
</tr>
<tr>
<td>11</td>
<td>double</td>
<td>ECEF-Y</td>
<td>m</td>
<td>Y-position in ECEF-coordinate frame</td>
</tr>
<tr>
<td>12</td>
<td>double</td>
<td>ECEF-Z</td>
<td>m</td>
<td>Z-position in ECEF-coordinate frame</td>
</tr>
<tr>
<td>13 – 15</td>
<td>3*double</td>
<td>b</td>
<td>m</td>
<td>Baseline in NED frame spanned by the position given by lat, lon and height, and by the position of the reference station.</td>
</tr>
<tr>
<td>16 – 18</td>
<td>3*double</td>
<td>bStdDev</td>
<td>m</td>
<td>Standard deviation of the baseline.</td>
</tr>
<tr>
<td>19 – 21</td>
<td>3*double</td>
<td>vel</td>
<td>m/s</td>
<td>Velocity in NED frame.</td>
</tr>
<tr>
<td>22 – 24</td>
<td>3*double</td>
<td>velStdDev</td>
<td>m/s</td>
<td>Standard deviation of the velocity.</td>
</tr>
<tr>
<td>25 – 27</td>
<td>3*double</td>
<td>acc</td>
<td>m/s²</td>
<td>Acceleration in body frame.</td>
</tr>
<tr>
<td>28 – 30</td>
<td>3*double</td>
<td>accStdDev</td>
<td>m/s²</td>
<td>Standard deviation of the acceleration.</td>
</tr>
<tr>
<td>31 – 33</td>
<td>3*double</td>
<td>att</td>
<td>deg</td>
<td>Attitude/Euler angles (heading, pitch, roll).</td>
</tr>
<tr>
<td>34 – 36</td>
<td>3*double</td>
<td>attStdDev</td>
<td>deg</td>
<td>Standard deviation of the attitude.</td>
</tr>
<tr>
<td>39 – 43</td>
<td>5*double</td>
<td>timingInfo</td>
<td>s</td>
<td>CPU-Load of used sensors. The sum (green line in the GUI) of elapsed time should not over 1 second (-&gt; to slow processor). First double: Elapsed time GNSS; Second double: Elapsed time IMU; Third double: Elapsed time Baro; Fourth double: Elapsed time Odometry; Fifth double: Reserved</td>
</tr>
</tbody>
</table>
| 49  | double | gnssReception | –     | Scalar, which indicates the GNSS.
8.3.CAN-Input: Dynamic CAN Decoder with .dbc-file

The ANavS sensor fusion can handle different forms of vehicle odometry. Up to 5 variables can therefore be defined. These are either one, two or four independent wheel speeds, and the steering angle or another heading information of the vehicle in the vehicle frame. The algorithms inside of the sensor fusion software calculate from the different inputs the current velocity information in the body frame of the vehicle for the positioning algorithm.

The wheel speed is processed as a signed value. A negative speed means thereby that the wheel is turning backward. It is possible to define the sign on a different CAN message. This could be used if the turning direction of a wheel is encoded in another message or position than the speed. This can also be done with the steering angle if needed.

The implemented Dynamic CAN Decoder allows you to use the ANavS Positioning Systems with CAN signals from your .DBC file without sharing the file with third parties or ANavS. The .dbc file used in this HOW TO is appended to the appendix.

Minimal required software versions:
MAINTENANCE_TOOL=4.1.3059
DRIVER=4.1.1807
(Both released on 23.01.2020 via beta channel)

8.3.1. Overview

The following Figure 56 shows the overall approach for the preparing of the ANavS Positioning System for the DBC-file, already provided e.g. by the car-manufacturer for an easy decoding of the CAN messages.
8.3.2. Copy the DBC-file

Use an FTP/SCP client of your choice to copy your .DBC file to the ANavS Positioning System. To speed up all steps, the user may want to remove unnecessary signals if the DBC-file is very long. Let’s assume the user puts the DBC-file to `/home/pi/golf.dbc` The example file used here is included in the appendix.

8.3.3. Load the file

For every remaining step, the user needs to have access to the ANavS Positioning System via SSH. On the MSRTK-System’s shell, load the file with:

```bash
MSRTKF CAN.loadDbc -source /home/pi/golf.dbc
```

8.3.4. Generate code with DBCC

Now, the user must run `dbcc` to generate the source code required to understand signals from your DBC-file. Though there is no hard limit on the number of signals in your DBC file, the resulting code may get very inefficient, if the DBC-file consists of millions of signals. A few thousand signals should be fine. On the shell, run `dbcc` with:

```bash
MSRTKF CAN.dbcc
```

8.3.5. Select signals for sensors

Now the user needs to map CAN signals to sensors supported by the ANavS Positioning System. Supported sensors are:

- Wheel odometry
  - Front Left (FL)
  - Front Right (FR)
  - Rear Left (RL)
  - Rear Right (RR)
- Steering angle

As of January 2020, the positioning accuracy only benefits from RL and RR. Some cars use unsigned odometry signals and provide turning direction as a separate signal.
To view the current signal map, run on the MSRTK-System’s shell:

**MSRTKF CAN.signal -get ALL**

To view all imported signals, run on the MSRTK-System’s shell:

**MSRTKF CAN.signal**

In this example, we see these:

```shell
>> MSRTKF CAN.signal
can_0x216_WheelDirection.WheelDirRearLeft
can_0x216_WheelDirection.WheelDirRearRight
can_0x217_WheelSpeed.WheelSpeedRearLeft
can_0x217_WheelSpeed.WheelSpeedRearRight
```

To find out a specific signal, run on the shell:

**MSRTKF CAN.signal -find SEARCHTERM**

For example:

```shell
>> MSRTKF CAN.signal -find rearleft
can_0x216_WheelDirection.WheelDirRearLeft
can_0x217_WheelSpeed.WheelSpeedRearLeft
```

To map a signal to a sensor, we use the ‘-map’ parameter. It accepts a signal for the magnitude (this may be signed or unsigned) and an optional ‘sign’ parameter for the turning direction. We run on the shell:

**MSRTKF CAN.signal -map RL -magn speedrearleft -sign dirrearleft,+=0,-=1**

The ‘-magn’ parameter accepts a search term for the magnitude of the signal. It does not need to be the precise identifier, as long as the search turns up only on result. The (optional) ‘-sign’ parameter requires a search term and the Enum values for forward and backward, separated by comma without space. Here, +=1 means that a signal with value ‘1’ announces positive wheel speed. Likewise, -=2 means that a signal with value ‘2’ announces negative wheel speed.

We do the same thing for the rear right (RR) sensor:

**MSRTKF CAN.signal -map RR -magn speedrearright -sign dirrearright,+=0,-=1**

If we made a mistake, we could delete a configuration:

**MSRTKF CAN.signal -clear FL**

Finally, we review our complete setup:
MSRTKF CAN.signal -get ALL

*FL magnitude: null*
*FL signum: null*
*FR magnitude: null*
*FR signum: null*
*RL magnitude:*
  *can_0x217_WheelSpeed.WheelSpeedRearLeft RL signum:*
  *can_0x216_WheelDirection.WheelDirRearLeft,+=0,-=1 RR magnitude:*
  *can_0x217_WheelSpeed.WheelSpeedRearRight RR signum:*
  *can_0x216_WheelDirection.WheelDirRearRight,+=0,-=1*
*STEER magnitude: null*
*STEER signum: null*

*Signals are valid (null counts as disabled)*

Note that you should only map signals that are present on your CAN bus, as the system waits for all mapped signals to arrive for a complete measurement.

### 8.3.6. Compile generated code

In this step, all generated code gets compiled and installed on the system. Once it is installed, it will remain fully configured on the system through reboots and software updates. It will take effect as soon as the process completes.

To have the configuration cross-checked again, build, and install the decoder, run:

`MSRTKF CAN.generateDecoder`

You may get some warnings depending on your .dbc file and signal map. If everything worked, it would say: “Decoder built […]” in the end. The driver will restart immediately and load the new decoder.

### 8.3.7. Setup CAN hardware filter

To reduce the load on the MSRTK System when there is a high CAN load, unrelated messages can be discarded by a hardware filter. The filter is configured with a *mask* and a *tag* value. Each incoming CAN id is only received if this condition holds:

*(mask & id) == tag*

To generate suitable values for mask and id, run:

`MSRTKF CAN.hardwareFilter.config`

To obtain *tag*, all used ids (*id0, id1, ..., idn*) are combined with the bitwise and operator &:
tag = id0 & id1 & ... & idn

This finds all bit positions that have value 1 in every single used id.  
To obtain mask, all bit positions that are equal for all ids have to be searched:

mask = tag | ~(id0 | id1 |... | idn)

We find all bit positions that have value zero by negating the result of applying the bitwise or operator | on all ids. This combined with the tag, which holds all bit positions which have value 1, is a valid filter-mask.

**Note:** tag can also be used as mask. In this case, all selected ids pass the filter, but it is not as specific as possible. This simplification is used in some earlier software versions.

To disable the filter, use:

**MSRTKF CAN.hardwareFilter.config -clear true**

### 8.3.8. Starting Sensor Fusion with ANavS Wizard

The last step is enabling the CAN-Interface and saving raw data for post-processing. Please see the figure below which checkboxes (signed in red) must be activated in the Wizard.
Figure 57: Start ANavS Sensor Fusion with activated CAN-Odometry input
9. The Network-Time-Protocol (NTP)

The ANavS® Positioning Systems are an accurate GNSS and multi sensor positioning system with Ethernet, WIFI, and LTE networking capabilities. These capabilities make it an ideal accurate network time source. This functionality is implemented using the GNU application “ntpd”. As of September 2020, the Positioning Systems ships with the NTP functions installed through the Debian package “ntp” in the version “1:4.2.8p10+dfsg-3+deb9u2”.

![Network Time Protocol Diagram](image.png)

**Figure 58: Overview of NTP functionality in the ANavS Positioning Systems**

As shown in Figure 58, the Positioning System provides NTP time via ethernet and WIFI, while receiving time data through GNSS and a backup NTP server. The behavior of the NTP system can be configured on the command line of the ANavS Positioning Systems. The configuration is persisted in the file `/home/pi/device/settings`. The ntpd-configuration file will be updated accordingly by the system itself.

9.1. Time Policy

Change the time policy on the terminal to select the source of the time provided to the user:

```bash
MSRTKF Time.config -set timePolicy -value <PARAMS>
```

With time policy parameter `<SYNC_GNSS>`, only GNSS is selected as input to ntpd on the ANavS Positioning System. If satellites are visible for the system, the provided time accuracy reaches millisecond accuracy quickly after the satellites become visible. After this initialization phase, a delicate clock calibration process begins and ntpd will achieve accuracy better than 10 microseconds usually in less than 15 minutes.
With time policy <SYNCNETWORK>, only the remote NTP server is selected as input to ntpd on the ANavS Positioning System. If it is reachable, the provided time accuracy reaches millisecond accuracy quickly. Due to noise of time measurements over the network, the clock calibration process is limited to an accuracy of 100 microseconds even in ideal conditions. This synchronization is available through Ethernet, Wi-Fi, and mobile network.

With time policy <AUTO>, both NTP and GNSS are selected as input to ntpd on the ANavS Positioning System. Ntpd uses its own algorithm to select which of the two it selects as reference. The selection strategy is not obvious and won’t always select GNSS when available. It even may fail to select GNSS, if the remote NTP server is not reachable.

### 9.2. Time Server

Change the time server that is used as ntpd reference in the Time-Policy <AUTO> and <SYNCNETWORK> modes. The value will default to “time.nist.gov” if not configured. This server is also queried during booting the ANavS Positioning System to initialize the local clock before GNSS has acquired any satellites.

```
MSRTKF Time.config -set timeServer -value time.nist.gov
```

### 9.3. Status Information

The following command will print information on the time policy as well as the last successful clock synchronizations to NTP and GNSS:

```
MSRTKF Time.service.status
```

The “last successful …” fields are queried from ntpd only every few seconds, so an update may have happened in this short window and is not reported when calling.

The following command will print information on the time sources used by ntpd:

```
ntpq -p
```


Perform a precision measurement of the local clock error using the best available time source. This may require restarting ntpd and can thus affect clock accuracy:

```
MSRTKF Time.error
```

Set local time immediately from best available time source, while bypassing ntpd.

```
MSRTKF Time.adjust
```
This command requires restarting ntpd. This method is accurate to 0.5 milliseconds, so it may decrease accuracy if ntpd is already optimally calibrated.

### 9.4. Time-Synchronization with MS-RTK clock

**Prerequisites:**

- Network connection to the MS-RTK module. Ethernet connection in a local network with static IPs configured is recommended. A static Ethernet IP for the MS-RKT can be configured via the ANavS Wizard in the Ethernet-Settings section.
- Ubuntu Linux system (tested with Ubuntu 18.04) to apply the steps described below.

**Install ntp from the Ubuntu sources:**

```bash
>> sudo apt-get install ntp
```

Adapt the NTP configuration file `/etc/ntp.conf` to set the MS-RTK as the only time server. Replace `<MSRTK-IP>` by the configured MS-RTK IP address:

```conf
pool <MSRTK-IP> iburst

# Provide current local time as a default time source if temporarily internet connectivity is lost.
# Uncomment lines below, to enable current local time as a time source. Disabled to prevent observed
# delays when synchronizing time with the MSRTK.
#server 127.127.1.0
#fudge 127.127.1.0 stratum 10

# Logging
# Uncomment lines below, to enable logging of ntp statistics.
#statistics loopstats
#statsdir /var/log/ntp
#filegen peerstats file peers type day link enable
#filegen loopstats file loops type day link enable
```

**Restart NTP Service:**

```bash
>> sudo service ntp restart
```

Check NTP status to see if service is active and running, and the specified time server is used:

```bash
>> service ntp status
```
Monitor the time-synchronization offset (ms) between the clocks of the local client and the time server.

```bash
>> ntpq -p
```

Wait until the offset falls below the desired accuracy, e.g. 1 ms. If the offset is not decreasing, stopping and starting the ntp service may help:

```bash
>> sudo service ntp stop
>> sudo service ntp start
```

**Notes:**

- Within a small local area network and Ethernet connection time-synchronization offset < 1 ms is achievable.
- In case there are transitions from non-GNSS to GNSS reception areas the provided time may jump from local time to global GPS-based time of weeks. Furthermore, provided time accuracy depends to the availability and reception quality of GNSS signals.
- Monitoring and continuously logging the ntp offset and statistics is recommended to assess the quality of time-synchronization.
- Please set the poll interval in a range of some seconds (not days or even weeks).

**References:**

ntpd wiki Ubuntu-Users: [https://wiki.ubuntuusers.de/ntpd/](https://wiki.ubuntuusers.de/ntpd/)
ntpd wiki Windows-Users: [https://www.rz.uni-osnabrueck.de/Dienste/NTP/win10.htm](https://www.rz.uni-osnabrueck.de/Dienste/NTP/win10.htm)
10. ANavS ROS-Ethernet-Adapter (REA)

The ROS-Ethernet Adapter is a command line tool for Linux that provides an adapter between ANavS TCP/IP data streams and ROS. The REA client allows to publish the ANavS sensor fusion solution in ROS (mode 'padsolution2ros').

Please contact the support-team to receive this software package.

10.1. System requirements and dependencies

- OS: Linux/Ubuntu (tested with Ubuntu 18.04, and Ubuntu 20.04)
- ROS base version installed (tested with ROS Melodic/Ubuntu 18.04, and ROS Noetic/Ubuntu 20.04)
  - Follow [http://wiki.ros.org/ROS/Installation](http://wiki.ros.org/ROS/Installation) and install "ROS-Base: (Bare Bones)" version
- Additional Ubuntu packages:
  
  ```bash
  sudo apt install ros-$ROS_VERSION-tf2 ros-$ROS_VERSION-tf2-ros ros-tf2-msgs libtf2-msgs-devros-$ROS_VERSION-tf2-geometry-msgs libtf2-dev ros-$ROS_VERSION-tf libxmlrpcpp-dev librosconsole-dev libactionlib-dev
  ```

10.2. REA Client

10.2.1. Prerequisites

- Linux host with installed REA package.
- ANavS MS-RTK module or Integrated Sensor Platform (ISP)
  - GNSS antennas connected (with sufficient GNSS reception)
  - Device powered on
- Ethernet or WLAN connection between Linux host and ANavS system (Ethernet recommended)
  - Ethernet and WLAN settings can be configured in the ANavS Wizard
- Start ANavS Wizard and configure the ANavS system
- Start sensor fusion
  - A positioning solution should be provided (see 'Solution' tab in the ANavS GUI)

For mode 'padsolution2ros' if a specific customer code is used:

Copy the ANAVS.conf file into the REA subfolder/configs. The file is usually stored on the MS-RTK module at: /share/PAD/configs/ANAVS.conf.
To publish the ANavS sensor data only steps 1-3 are required.

### 10.2.2. Setup your ROS environment

Use the following commands in the command-line:

```
ROS_VERSION=melodic
source /opt/ros/${ROS_VERSION}/setup.bash
roscore&
```

### 10.2.3. Sensor fusion solution (mode: padsolution2ros)

Publishes ANavS sensor fusion solution in ROS. Assumes ANavS sensor fusion is running (eg. on either the ANavS MS-RTK or ISP (remote host)) and TCP/IP data streams are available, eg. via Ethernet connection (default settings IP: 127.0.0.1, port: 6001). Specify the remote host’s IP for your setup (--ip <host-ip>), and ANavS sensor fusion solution port if deviating from the default (--port <port>).

#### Usage

```
./client/bin/Release/anavs_ros_ethernet_client padsolution2ros
[--ip <host-ip>] [--port <port>] [options] # specify the MSRTK host IP
```

#### Use help to print all options

```
./client/bin/Release/anavs_ros_ethernet_client --help
```

#### Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-h --help</td>
<td>Show help.</td>
</tr>
<tr>
<td>--ip &lt;host-ip&gt;</td>
<td>IP address of remote host [default: 127.0.0.1].</td>
</tr>
<tr>
<td>--port &lt;host-port&gt;</td>
<td>TCP/IP data streams port. ANavS sensor fusion solution [default: 6001], sensor data [default: 4001].</td>
</tr>
<tr>
<td>padsolution2ros:</td>
<td></td>
</tr>
<tr>
<td>--topic_prefix &lt;topic-prefix&gt;</td>
<td>Prefix for all ROS topics published [default: /anavs/solution].</td>
</tr>
<tr>
<td>--use_original_timestamps</td>
<td>Use original ANavS solution data timestamps (GPS tow) as ROS message timestamp [default: local host ROS time].</td>
</tr>
</tbody>
</table>
--enable_odom

Enables publishing odometry (topic: /odom; type: nav_msgs::Odometry) and angular rate (topic: /ang_rate; type: geometry_msgs/Vector3Stamped).

--enable_pose

Enables publishing pose (topic: /pose_enu; type: geometry_msgs::PoseWithCovarianceStamped).

--odom_only

Publish only the odometry message '/odom'.

--pose_only

Publish only the pose message '/pose_enu'.

--publisher_queue_size <queue-size>

Sets ROS publisher queue sizes for all messages published. [default: 1]

--pose_enu_frame <frame-id>

Specifies frame_id of published pose (topic: /pose_enu) [default: map].

--remove_pos_offset

Enables removing initial position offset for publishing pose 'pose_enu', and odometry 'odom'.

--publish_to_tf [--published_frame <child-frame-id>]

Enables publishing transformation to ROS tf. Optional: Specify child_frame_id [default: base_link].

--publish_path

Enables publishing path (topic: /path; type: nav_msgs::Path) created from poses (topic: /pose_enu).

--publish_ref_station

Enables publishing of reference station coordinates (topic: /pos_reference_station; type: geometry_msgs/PointStamped).

**Remarks**

**publisher_queue_size**

- Default: 1. Recommended for real-time settings. Old messages are dropped if they cannot be published fast enough. Prevents messages queueing and delaying the publishing of current messages.
- Increase `publisher_queue_size (> 1)` to avoid message drop. Recommended for post-processing settings, where potential delays are not critical but all messages shall be published by priority.
- For more details refer to [Choosing a good queue size](#)

**Advertised ROS topics**

The standard ROS topics provided correspond to the fields of the “Standard Binary Solution Message” documented in Section 6.1. However angular measurements are provided in radians rather than in degrees in the ROS messages.

<table>
<thead>
<tr>
<th>ROS Topic</th>
<th>Description</th>
<th>ROS Message Type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>/anavs/solution/id</code></td>
<td>ID</td>
<td>std_msgs/UInt8</td>
</tr>
<tr>
<td>----------------------</td>
<td>----</td>
<td>----------------</td>
</tr>
<tr>
<td><code>/anavs/solution/week</code></td>
<td>Week of current epoch</td>
<td>std_msgs/UInt16</td>
</tr>
<tr>
<td><code>/anavs/solution/tow</code></td>
<td>Time of Week (sec)</td>
<td>sensor_msgs/TimeReference</td>
</tr>
<tr>
<td><code>/anavs/solution/pos</code></td>
<td>Position (NED* in meters)</td>
<td>geometry_msgs/PointStamped</td>
</tr>
<tr>
<td><code>/anavs/solution/pos_llh</code></td>
<td>Position (lat,lon,height in deg,deg,meters)</td>
<td>geometry_msgs/PointStamped</td>
</tr>
<tr>
<td><code>/anavs/solution/pos_xyz</code></td>
<td>Position (ECEF in meters)</td>
<td>geometry_msgs/PointStamped</td>
</tr>
<tr>
<td><code>/anavs/solution/att_euler</code></td>
<td>Attitude/ Euler angles (heading, pitch, roll in rad)</td>
<td>geometry_msgs/PointStamped</td>
</tr>
<tr>
<td><code>/anavs/solution/att_state</code></td>
<td>Attitude filter state</td>
<td>std_msgs/UInt8</td>
</tr>
<tr>
<td><code>/anavs/solution/rtk_state</code></td>
<td>RTK filter state</td>
<td>std_msgs/UInt8</td>
</tr>
<tr>
<td><code>/anavs/solution/num_sats</code></td>
<td>Number of satellites</td>
<td>std_msgs/UInt8</td>
</tr>
<tr>
<td><code>/anavs/solution/imu_calibrated</code></td>
<td>IMU initialized/calibrated</td>
<td>std_msgs/Bool</td>
</tr>
<tr>
<td><code>/anavs/solution/vel</code></td>
<td>Velocity (NED* in m/s)</td>
<td>geometry_msgs/Vector3Stamped</td>
</tr>
<tr>
<td><code>/anavs/solution/acc</code></td>
<td>Acceleration (body frame in m/s^2)</td>
<td>geometry_msgs/Vector3Stamped</td>
</tr>
<tr>
<td><code>/anavs/solution/pos_stddev</code></td>
<td>Std Dev of position (NED* in meters)</td>
<td>geometry_msgs/PointStamped</td>
</tr>
<tr>
<td><code>/anavs/solution/vel_stddev</code></td>
<td>Std Dev of velocity (NED* in m/s)</td>
<td>geometry_msgs/PointStamped</td>
</tr>
<tr>
<td><code>/anavs/solution/att_euler_stddev</code></td>
<td>Std Dev of attitude/ Euler angles (in rad)</td>
<td>geometry_msgs/PointStamped</td>
</tr>
</tbody>
</table>
### Topics description

<table>
<thead>
<tr>
<th>ROS Topic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Identifier of the ANavS Position and Attitude Determination (PAD) solution.</td>
</tr>
<tr>
<td>week</td>
<td>Week number of the current epoch.</td>
</tr>
<tr>
<td>tow</td>
<td>Time reference message containing the ROS time and the GNSS Time of Week (TOW) in seconds.</td>
</tr>
<tr>
<td>pos</td>
<td>The position in local coordinate frame (NED*) in meters. Baseline in local coordinate frame spanned by the position given by pos_llh in lat, lon and height, and by the position of the reference station.</td>
</tr>
<tr>
<td>pos_llh</td>
<td>The position in geographic coordinates (lat, lon, height) in degree, degree, meters.</td>
</tr>
<tr>
<td>pos_xyz</td>
<td>The position in ECEF coordinates (X, Y, Z) in meters.</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>att_euler</td>
<td>The attitude in Euler angles (heading, pitch, roll) in rad.</td>
</tr>
<tr>
<td>att_state</td>
<td>The State-definition for the Attitude Kalman-filter: 0,1,2,3 - 0: No solution, 1: Least-Squares solution, 2: Float solution (orange light in ANavS GUI), 3: Fixed solution (green light in ANavS GUI)</td>
</tr>
<tr>
<td>rtk_state</td>
<td>The State-definition for the RTK-Position Kalman-filter: 0,1,2,3 - 0: No solution, 1: Least-Squares solution, 2: Float solution (orange light in ANavS GUI), 3: Fixed solution (green light in ANavS GUI)</td>
</tr>
<tr>
<td>num_sats</td>
<td>Number of satellites (May be larger than the number of USED satellites, indicated in the ANavS GUI).</td>
</tr>
<tr>
<td>imu_calibrated</td>
<td>Is set if the IMU is initialized/ calibrated.</td>
</tr>
<tr>
<td>vel</td>
<td>The velocity in local coordinate frame (NED*) in m/s.</td>
</tr>
<tr>
<td>acc</td>
<td>The acceleration in body frame in m/s^2.</td>
</tr>
<tr>
<td>pos_stddev</td>
<td>The standard deviation of the position (NED*) in meters.</td>
</tr>
<tr>
<td>vel_stddev</td>
<td>The standard deviation of the velocity (NED*) in m/s.</td>
</tr>
<tr>
<td>att_euler_stddev</td>
<td>The standard deviation of the attitude in Euler angles in rad.</td>
</tr>
<tr>
<td>accuracy</td>
<td>Estimated accuracy of the position in local coordinate frame in meters. The scalar accuracy value is stored in the component point.x. Components point.y and point.z are set to NaN.)</td>
</tr>
</tbody>
</table>

Optional:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pose_enu</td>
<td>The pose in ENU frame in meters. The geographic north direction is aligned with ROS frame y-axis. Heading angle 0° corresponds to north direction and results in ROS in an orientation where body-frame x-axis (red-axis in rviz) points into ROS frame y-direction. The 6x6 covariance matrix contains the variances of position X, Y, Z and rotation around X, Y, and Z on the diagonal. The covariances are set to zero currently. In case position variances contain NaN/Inf values, the covariance matrix left-upper 3x3 part is set to zeros. In case rotation variances contain NaN/Inf values, the covariance matrix right-lower 3x3 part is set zeros.</td>
</tr>
<tr>
<td>odom</td>
<td>Pose (ENU) in meters, and twist (body frame), velocity in m/s and angular rate in rad/s. Pose covariance matrix, as described for pose_enu. Twist covariance matrix is set zero (not available).</td>
</tr>
<tr>
<td>ang_rate</td>
<td>Angular rate (NED*) in rad/s.</td>
</tr>
<tr>
<td>path</td>
<td>Path for visualization in RViz.</td>
</tr>
</tbody>
</table>
Remarks

Please note that the positioning estimates (position, attitude, velocity and acceleration) potentially may jump, i.e. they are not guaranteed to be continuous all the time. The nav_msgs/Odometry pose frame is thus set to "map" rather than "odom", to express that measurements are not guaranteed to be continuous.

Visualization in RViz

Run the REA client with enabled options to publish pose and odom topics to publish to tf and to publish a path of the position trajectory. To start at zero position, use the flag --remove_pos_offset to remove the initial position offset from the published positions in pose_enu:

```
./client/bin/Release/anavs_ros_ethernet_client padsolution2ros [--ip <host-ip>] [--port <port>] --enable_pose --enable_odom --remove_pos_offset --publish_to_tf --publish_path [options]
```

Run RViz using the provided config file:

```
rviz -d configs/anavs.rviz
```

Following items are displayed:

- Pose with covariance: /anavs/solution/pose_enu
- Odometry: /anavs/solution/odom
- Path: /path

11. ANavS® Solution Decoder Tool

The ANavS® Solution-Decoder Tool brings the binary solution output file into manageable file formats. The purpose of this tool is mainly for postprocessing. It can transform the .bin/.txt-file into .csv-format (Comma-Separated Values) for quick value visualization and .kml-format for coordinate visualization in Google-Earth.

11.1. Download

Download the latest ANavS® Solution Decoder Tool for your desired Operating System (Windows 10 or Ubuntu 18.04) from the ANavS Knowledgebase with the following link:
11.2. Some Hints

- The Source code has been compiled for Windows systems (MinGW-Windows) and Linux systems (GNU-Linux).
- The path to the datasets shall not contain any space characters or blank spaces.
- If the path is detected to be incorrect, it is safer to close the console and restart because problems tend to arise when trying to correct the path.
- The column header contains the same names as the first 49 variables of the proprietary binary protocol. The protocol is described in chapter 6.
- The decoder tool decodes the ANavS-Solution file commonly known as “PAD_solution.bin” or “PAD_solution.txt”. However, the solution file can have another name, as long as it does not contain any blank spaces.
- The delimiter of the .csv files is a semicolon “;”
  - The decoder file can be put in the same folder as the file which is desired to be decoded. If that is done, the path to the file can be the file name itself. However, it is always safer to use a complete path than a relative path, e.g., “C:\Users\ANavS\Documents\Test\PAD_solution.bin” instead of “PAD_solution.bin”

11.3. Hints especially for Linux users

- To use the decoder, please open the bash and change your directory to the path where you saved the decoder file.
- Then enter: “./ubxproject”
  - If the program says that you don’t have any access permissions, it probably means that you must change the permission flags/bits of the decoder file
    - Simply enter in the bash: “chmod +x ubxproject”
    - This will give your user the right to access the file which can be seen by the changed flags from “-rw-rw-r--” to “-rwxrwxr-x”
    - Try it again.

11.4. Decode to .csv-format

The .csv-format is a way to visualize the ANavS solution in an editor or any spreadsheet-program such as Excel and LibreOffice.

1. Please open the “ubxprojext.exe” executable.
2. A console window will open, which asks for the path to the desired document. Please introduce the correct path to the ANavS solution-file.
3. If a correct path was introduced, the console will ask which format you wish to receive. You shall enter 0 for a .csv-format.

4. The console will ask a second time for a format. However, this time it is referring to the format of the .csv-file. There are two formats to choose from basic and complete.
   1. The basic format decodes all the important measurements such as time, position, velocity, and acceleration, in a very compact form.
   2. The complete format includes a few more values than the basic one. However, these extra values are mostly valuable for development purposes and do not represent any added value for the customer. In addition, it requires more space. Until now the complete format includes the satellite information of each satellite recorded. Therefore, the console will first ask you which satellite systems you want to consider in the decoding process.

5. For practicality it is advised to use the basic format. Therefore enter 0.

6. If all values were entered correctly, the tool should begin the decoding process immediately. Thereupon the console visualizes the progress of the decoding process.

7. The resulting .csv-file “PADSolution.csv” will appear in the same folder as the “ubxprojext.exe”.

8. After the decoding process you can open the .csv-file in any spreadsheet-program to see the ANavS® Solution-output.

11.5. Decode to .kml-format

The .kml-format helps to visualize the recorded trajectory with services such as Google Earth or Google Maps. Since the regular data output-rate is quite high, it would make the .kml-file too memory-intensive. As a result, .kml-file only includes the most significant portion of the original data points, that means only GNSS-epochs are visualized.

1. Please open the “ubxprojext.exe” executable.

2. A console window will open, which asks for the path to the desired document. Please introduce the correct path to the ANavS solution file.

3. If a correct path was introduced, the console will ask which format you wish to receive. You shall enter 1 for a .kml-format.

4. If all values were entered correctly, the tool should begin the decoding process immediately. Thereupon the console visualizes the progress of the decoding process.

5. The resulting .kml-file “PADSolution.kml” will appear in the same folder as the “ubxprojext.exe”.

6. After decoding you can use that kml-file to visualize your recorded trajectory, e.g., by incorporating it to Google Earth.
Figure 59: Example kml-file where each purple dot is showing the most interesting states like position, velocity, acceleration, and attitude.
12. ANavS® Record Extractor Tool

When using the ANavS Positioning Systems, it records a variety of data from different sources/sensors and stores it all in a single file within a zip-archive. The Record-Extractor command line tool can be used to extract this data into several files, one for each channel of data recorded. This tool is especially needed for using recorded data is further analysis in post-processing.

**NOTE:** The target-platform using the Record Extractor Tool is Java 8.

12.1. Download

Download the latest ANavS® Record Extractor Tool from the ANavS Knowledgebase with the following link:

https://anavs.com/knowledgebase/anavs-record-extractor-tool/

12.2. Usage

From command line:

```
java -jar RecordExtractor.jar -i <inputfilepath> [-o <outputfilepath>]
```

in the directory of the RecordExtractor.jar file

12.2.1. Arguments

- **inputfilepath**: Required. The path of the zip archive containing the data to extract.
- **outputfilepath**: Optional. The path of the directory where the extracted data should be stored. If the directory doesn’t exist, it will be created. If this argument is omitted, the tool will store the extracted data in the same directory as the source zip-file.

12.2.2. Output

Currently the tool will be able to name the output files correctly if the port number of the fragment in the input file is one of the following:

<table>
<thead>
<tr>
<th>Port number</th>
<th>Corresponding filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>4007</td>
<td>“LOGcan.ubx”</td>
</tr>
<tr>
<td>4001</td>
<td>“LOGrover0.ubx”</td>
</tr>
<tr>
<td>4002</td>
<td>“LOGrover1.ubx”</td>
</tr>
<tr>
<td>4003</td>
<td>“LOGrover2.ubx”</td>
</tr>
<tr>
<td>2102</td>
<td>“LOGvrs.osr”</td>
</tr>
<tr>
<td>6001</td>
<td>“PAD_solution.bin”</td>
</tr>
</tbody>
</table>
If the tool comes across any other port number during extraction it will name the file according to the port number and store the contents as a .bin file.

12.2.3. Integrating into the Windows 11 right-click menu

The ANavS® Record Extractor Tool can be integrated into the Windows 11 right-click menu by:

1. Open the Registry Editor
2. Go to Computer\HKEY_CLASSES_ROOT\*\shell
3. Create a new key with the name of the menu entry, e.g. “Extract with ANavS Extractor”
4. Create a new key in “Extract with ANavS Extractor” with the name “command”. Note, that this name is mandatory.
5. Set the value of the “command” key to the string
   
   "C:\Program Files\java\jre1.8.0_361\bin\javaw.exe" -jar "C:\Development\Software\RecordExtractor_v1\RecordExtractor.jar" -i "%1"

   where the yellow highlighted paths must be adapted accordingly.
6. The successfully created key is shown in Figure 60.

   ![Figure 60: Registry with the new key for the Windows 11 right-click menu of the ANavS® Record Extractor Tool.](image)
13. LTE-VPN connection to the ANavS Positioning Systems

All ANavS Positioning Systems are delivered with already equipped SIM cards. The SIMs aren’t coupled to a fixed service provider but using the best available service provider in its area, a so-called All-Net SIM card. In addition to providing internet access to your module and in case of Wi-Fi connection (through bridging) also to your PC/Laptop, the SIM card provides VPN connection to get SSH remote access and for data streaming through mobile network.

HINT: Please contact the support team to get your customer-specific config-file and Token for the settings described below.

13.1. Setting up OpenVPN for Linux

- Install OpenVPN software (e.g. "apt-get install openvpn")
- Download client.conf (emnify-eu-west-1.conf) and store it on your server in /etc/openvpn
- Create a file credentials.txt in /etc/openvpn with following content to authenticate with your user credentials:

```
7133
YourApplicationToken
```

The corresponding Application-Token is given in the txt-file Application_Token_XXX.txt

- Start openvpn service (e.g. "service openvpn start") and monitor connection in /var/log/syslog

13.2. Setting up the OpenVPN for Windows

- Install OpenVPN software from https://openvpn.net/index.php/open-source/downloads.html
- Download client.opvn (emnify-eu-west-1.ovpn) and store it on your server in C:\Program Files\OpenVPN\config
- Create a file credentials.txt in C:\Program Files\OpenVPN\config with following content to authenticate with your user credentials:
7133

YourApplicationToken

The corresponding Application-Token is given in the txt-file Application_Token_XXX.txt

- Start OpenVPN GUI application and monitor the connection in C:\Program Files\OpenVPN\log\client.txt

Alternatively, you can download OpenVPN clients for Windows, OSX, Android and iOS from https://openvpn.net
APPENDIX-1: EXAMPLE CAN DBC-FILE FOR THE SENSOR DATA

If you are concerned about undisclosed information in your DBC-file, you can remove most traces of it from the MSRTK-System without impeding operation of the CAN functionality. Files that contain information on your CAN signals after the setup process:

<table>
<thead>
<tr>
<th>Path</th>
<th>Description</th>
<th>CAN-related content</th>
<th>Clean up notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>/home/pi/device/can_profile.dbc</td>
<td>Copy of your .dbc file</td>
<td>Your DBC</td>
<td>File may be deleted, not required for operation</td>
</tr>
<tr>
<td>/home/pi/device/settings</td>
<td>Database for MSRTK-device specific settings</td>
<td>Your signal map</td>
<td>Remove CAN related lines manually, not required for operation</td>
</tr>
<tr>
<td>/home/pi/.generator/</td>
<td>Working directory of build process</td>
<td>Generated source code of decoder, binaries</td>
<td>Directory may be deleted, not required for operation</td>
</tr>
<tr>
<td>/share/driver/dbcc/can_profile.dbc</td>
<td>Copy of your .dbc file</td>
<td>Your DBC</td>
<td>File may be deleted, not required for operation</td>
</tr>
<tr>
<td>/share/driver/dbcc/can_profile.c</td>
<td>Source code of generated decoder</td>
<td>Extensive information on your signals</td>
<td>File may be deleted, not required for operation</td>
</tr>
<tr>
<td>/share/driver/dbcc/can_profile.h</td>
<td>Header of generated decoder</td>
<td>Some Information on your signals</td>
<td>File may be deleted, not required for operation</td>
</tr>
<tr>
<td>/home/pi/device/libdynamic-decoder.so</td>
<td>Installed decoder binaries</td>
<td>Executable decoder + names of signals</td>
<td>Required for operation</td>
</tr>
</tbody>
</table>

Example golf.dbc file:

```
VERSION ""

NS_
  NS_DESC_
  CM_
  BA_DEF_
  BA_
  VAL_
  CAT_DEF_
```
CAT_
FILTER
BA_DEF_DEF_
EV_DATA_
ENNVAR_DATA_
SGTYPE_
SGTYPE_VAL_
BA_DEF_SGTYPE_
BA_SGTYPE_
SIG_TYPE_REF_
VAL_TABLE_
SIG_GROUP_
SIG_VALTYPE_
SIGTYPE_VALTYPE_

BS_: 

BU_: 

BO_ 535 WheelSpeed: 8 Car 
SG_ WheelSpeedRearLeft : 33|15@1+ (0.0033,0) [0|0] "" Car 
SG_ WheelSpeedRearRight : 49|15@1+ (0.0033,0) [0|0] "" Car 

BO_ 534 WheelDirection: 8 Car 
SG_ WheelDirRearLeft : 33|1@1+ (1,0) [0|0] "" Car 
SG_ WheelDirRearRight : 39|1@1+ (1,0) [0|0] "" Car
APPENDIX-2: EXAMPLE CAN DBC-FILE FOR THE ANAVS SOLUTION OUTPUT

Example dbc file for all given output parameters according to the Table 1:

VERSION ""

NS_ :

  NS_DESC_
  CM_
  BA_DEF_
  BA_
  VAL_
  CAT_DEF_
  CAT_
  FILTER
  BA_DEF_DEF_
  EV_DATA_
  ENVVAR_DATA_
  SGTYPE_
  SGTYPE_VAL_
  BA_DEF_SGTYPE_
  BA_SGTYPE_
  SIG_TYPE_REF_
  VAL_TABLE_
  SIG_GROUP_
  SIG_VALTYPE_
  SGTYPE_SGTYPE_
  BO_TX_BU_
  BA_DEF_REL_
  BA_REL_
  BA_DEF_DEF_REL_
  BU_SG_REL_
  BU_EV_REL_
  BU_BO_REL_
  SG_MUL_VAL_

BS_ :

BU_ :

BO_ 50 numSats: 8 Vector__XXX
SG_ numSats : 0|8@1+ (1,0) [0|255] "" Vector__XXX
Reference Guide
Multi-Sensor RTK Module, RTK Reference Station and ISP

BO_49 gnssReception: 8 Vector__XXX
SG_ gnssReception : 0|64@1- (1,0) [-1.7E+308|1.7E+308] "" Vector__XXX

BO_42 timingInfo_Odometry: 8 Vector__XXX
SG_ timingInfo_Odometry : 0|64@1- (1,0) [-1.7E+308|1.7E+308] "s" Vector__XXX

BO_41 timingInfo_Baro: 8 Vector__XXX
SG_ timingInfo_Baro : 0|64@1- (1,0) [-1.7E+308|1.7E+308] "s" Vector__XXX

BO_40 timingInfo_IMU: 8 Vector__XXX
SG_ timingInfo_IMU : 0|64@1- (1,0) [-1.7E+308|1.7E+308] "s" Vector__XXX

BO_39 timingInfo_GNSS: 8 Vector__XXX
SG_ timingInfo_GNSS : 0|64@1- (1,0) [-1.7E+308|1.7E+308] "s" Vector__XXX

BO_36 attStdDev_roll: 8 Vector__XXX
SG_ attStdDev_roll : 0|64@1- (1,0) [-1.7E+308|1.7E+308] "deg" Vector__XXX

BO_35 attStdDev_pitch: 8 Vector__XXX
SG_ attStdDev_pitch : 0|64@1- (1,0) [-1.7E+308|1.7E+308] "deg" Vector__XXX

BO_34 attStdDev_heading: 8 Vector__XXX
SG_ attStdDev_heading : 0|64@1- (1,0) [-1.7E+308|1.7E+308] "deg" Vector__XXX

BO_33 att_roll: 8 Vector__XXX
SG_ att_roll : 0|64@1- (1,0) [-1.7E+308|1.7E+308] "deg" Vector__XXX

BO_32 att_pitch: 8 Vector__XXX
SG_ att_pitch : 0|64@1- (1,0) [-1.7E+308|1.7E+308] "deg" Vector__XXX

BO_31 att_heading: 8 Vector__XXX
SG_ att_heading : 0|64@1- (1,0) [-1.7E+308|1.7E+308] "deg" Vector__XXX

BO_30 accStdDev_z: 8 Vector__XXX
SG_ accStdDev_z : 0|64@1- (1,0) [-1.7E+308|1.7E+308] "m/s^2" Vector__XXX

BO_29 accStdDev_y: 8 Vector__XXX
SG_ accStdDev_y : 0|64@1- (1,0) [-1.7E+308|1.7E+308] "m/s^2" Vector__XXX

BO_28 accStdDev_x: 8 Vector__XXX
SG_ accStdDev_x : 0|64@1- (1,0) [-1.7E+308|1.7E+308] "m/s^2" Vector__XXX

BO_27 acc_z: 8 Vector__XXX
SG_ acc_z : 0|64@1- (1,0) [-1.7E+308|1.7E+308] "m/s^2" Vector__XXX
BO_26 acc_y: 8 Vector__XXX
SG_acc_y : 0|64@1- (1,0) [-1.7E+308|1.7E+308] "m/s^2" Vector__XXX

BO_25 acc_x: 8 Vector__XXX
SG_acc_x : 0|64@1- (1,0) [-1.7E+308|1.7E+308] "m/s^2" Vector__XXX

BO_24 velStdDev_D: 8 Vector__XXX
SG_velStdDev_D : 0|64@1- (1,0) [-1.7E+308|1.7E+308] "m/s" Vector__XXX

BO_23 velStdDev_E: 8 Vector__XXX
SG_velStdDev_E : 0|64@1- (1,0) [-1.7E+308|1.7E+308] "m/s" Vector__XXX

BO_22 velStdDev_N: 8 Vector__XXX
SG_velStdDev_N : 0|64@1- (1,0) [-1.7E+308|1.7E+308] "m/s" Vector__XXX

BO_21 vel_D: 8 Vector__XXX
SG_vel_D : 0|64@1- (1,0) [-1.7E+308|1.7E+308] "m/s" Vector__XXX

BO_20 vel_E: 8 Vector__XXX
SG_vel_E : 0|64@1- (1,0) [-1.7E+308|1.7E+308] "m/s" Vector__XXX

BO_19 vel_N: 8 Vector__XXX
SG_vel_N : 0|64@1- (1,0) [-1.7E+308|1.7E+308] "m/s" Vector__XXX

BO_18 bStdDev_D: 8 Vector__XXX
SG_bStdDev_D : 0|64@1- (1,0) [-1.7E+308|1.7E+308] "m" Vector__XXX

BO_17 bStdDev_E: 8 Vector__XXX
SG_bStdDev_E : 0|64@1- (1,0) [-1.7E+308|1.7E+308] "m" Vector__XXX

BO_16 bStdDev_N: 8 Vector__XXX
SG_bStdDev_N : 0|64@1- (1,0) [-1.7E+308|1.7E+308] "m" Vector__XXX

BO_15 b_D: 8 Vector__XXX
SG_b_D : 0|64@1- (1,0) [-1.7E+308|1.7E+308] "m" Vector__XXX

BO_14 b_E: 8 Vector__XXX
SG_b_E : 0|64@1- (1,0) [-1.7E+308|1.7E+308] "m" Vector__XXX

BO_13 b_N: 8 Vector__XXX
SG_b_N : 0|64@1- (1,0) [-1.7E+308|1.7E+308] "m" Vector__XXX

BO_12 ECEF_Z: 8 Vector__XXX
SG_ECEF_Z : 0|64@1- (1,0) [-1.7E+308|1.7E+308] "m" Vector__XXX
BO_ 11 ECEF_Y: 8 Vector__XXX
SG_ ECEF_Y : 0/64@1- (1,0) [-1.7E+308|1.7E+308] "m" Vector__XXX

BO_ 10 ECEF_X: 8 Vector__XXX
SG_ ECEF_X : 0/64@1- (1,0) [-1.7E+308|1.7E+308] "m" Vector__XXX

BO_ 9 height: 8 Vector__XXX
SG_ height : 0/64@1- (1,0) [-1.7E+308|1.7E+308] "m" Vector__XXX

BO_ 8 lon: 8 Vector__XXX
SG_ lon : 0/64@1- (1,0) [-1.7E+308|1.7E+308] "deg" Vector__XXX

BO_ 7 lat: 8 Vector__XXX
SG_ lat : 0/64@1- (1,0) [-1.7E+308|1.7E+308] "deg" Vector__XXX

BO_ 5 towInit: 8 Vector__XXX
SG_ towInit : 0/64@1- (1,0) [-1.7E+308|1.7E+308] "s" Vector__XXX

BO_ 4 weekInit: 8 Vector__XXX
SG_ weekInit : 0/16@1+ (1,0) [0|65535] "" Vector__XXX

BO_ 3 tow: 8 Vector__XXX
SG_ tow : 0/64@1- (1,0) [-1.7E+308|1.7E+308] "s" Vector__XXX

BO_ 2 week: 8 Vector__XXX
SG_ week : 0/16@1+ (1,0) [0|65535] "" Vector__XXX

BO_ 1 resCode: 8 Vector__XXX
SG_ resCode : 0/16@1+ (1,0) [0|65535] "" Vector__XXX

CM_ SG_ 50 numSats "Number of satellites";
CM_ SG_ 49 gnssReception "Scalar, which indicates the GNSS signal reception";
CM_ SG_ 42 timingInfo_Odometry "CPU-Load of used sensor. The sum (green line in the GUI) of elapsed time should not over 1 second (→ to slow processor)";
CM_ SG_ 41 timingInfo_Baro "CPU-Load of used sensor. The sum (green line in the GUI) of elapsed time should not over 1 second (→ to slow processor)";
CM_ SG_ 40 timingInfo_IMU "CPU-Load of used sensor. The sum (green line in the GUI) of elapsed time should not over 1 second (→ to slow processor)";
CM_ SG_ 39 timingInfo_GNSS "CPU-Load of used sensor. The sum (green line in the GUI) of elapsed time should not over 1 second (→ to slow processor)";
CM_ SG_ 36 attStdDev_roll "Standard deviation of the attitude";
CM_ SG_ 35 attStdDev_pitch "Standard deviation of the attitude";
Reference Guide

Multi-Sensor RTK Module, RTK Reference Station and ISP

CM_ SG_ 34 attStdDev_heading "Standard deviation of the attitude"
CM_ SG_ 33 att_roll "Attitude/Euler angles (heading, pitch, roll)"
CM_ SG_ 32 att_pitch "Attitude/Euler angles (heading, pitch, roll)"
CM_ SG_ 31 att_heading "Attitude/Euler angles (heading, pitch, roll)"
CM_ SG_ 30 accStdDev_z "Standard deviation of the acceleration"
CM_ SG_ 29 accStdDev_y "Standard deviation of the acceleration"
CM_ SG_ 28 accStdDev_x "Standard deviation of the acceleration"
CM_ SG_ 27 acc_z "Acceleration in body frame"
CM_ SG_ 26 acc_y "Acceleration in body frame"
CM_ SG_ 25 acc_x "Acceleration in body frame"
CM_ SG_ 24 velStdDev_D "Standard deviation of the velocity"
CM_ SG_ 23 velStdDev_E "Standard deviation of the velocity"
CM_ SG_ 22 velStdDev_N "Standard deviation of the velocity"
CM_ SG_ 21 vel_D "Velocity in NED frame"
CM_ SG_ 20 vel_E "Velocity in NED frame"
CM_ SG_ 19 vel_N "Velocity in NED frame"
CM_ SG_ 18 bStdDev_D "Standard deviation of the baseline"
CM_ SG_ 17 bStdDev_E "Standard deviation of the baseline"
CM_ SG_ 16 bStdDev_N "Standard deviation of the baseline"
CM_ SG_ 15 b_D "Baseline in NED frame spanned by the position given by lat, lon and height, and by the position of the reference station"
CM_ SG_ 14 b_E "Baseline in NED frame spanned by the position given by lat, lon and height, and by the position of the reference station"
CM_ SG_ 13 b_N "Baseline in NED frame spanned by the position given by lat, lon and height, and by the position of the reference station"
CM_ SG_ 12 ECEF_Z "Z-position in ECEF-coordinate frame"
CM_ SG_ 11 ECEF_Y "Y-position in ECEF-coordinate frame"
CM_ SG_ 10 ECEF_X "X-position in ECEF-coordinate frame"
CM_ SG_  9 height "Height"
CM_ SG_  8 lon "Longitude"
CM_ SG_  7 lat "Latitude"
CM_ SG_  5 towInit "Time of Week of the epoch when the system was started"
CM_ SG_  4 weekInit "Week number of the epoch when the system was started"
CM_ SG_  3 tow "Time of Week of the current epoch"
CM_ SG_  2 week "Week number of the current epoch"
CM_ SG_  1 resCode "Result code bitfield, which keeps the system status and information"
BA_DEF_ "BusType" STRING
BA_DEF_DEF_ "BusType" "CAN"
SIG_VALTYPE_ 49 gnssReception : 2;
SIG_VALTYPE_ 42 timingInfo_Odometry : 2;
SIG_VALTYPE_ 41 timingInfo_Baro : 2;
SIG_VALTYPE_ 40 timingInfo_IMU : 2;
SIG_VALTYPE_ 39 timingInfo_GNSS : 2;
SIG_VALTYPE_ 36 attStdDev_roll : 2;
SIG_VALTYPE_ 35 attStdDev_pitch : 2;

Issue: 2.19       Date: 30.04.2024       Page 150 of 157

Reference Guide Document
SIG_VALTYPE_ 34 attStdDev_heading : 2;
SIG_VALTYPE_ 33 att_roll : 2;
SIG_VALTYPE_ 32 att_pitch : 2;
SIG_VALTYPE_ 31 att_heading : 2;
SIG_VALTYPE_ 30 accStdDev_z : 2;
SIG_VALTYPE_ 29 accStdDev_y : 2;
SIG_VALTYPE_ 28 accStdDev_x : 2;
SIG_VALTYPE_ 27 acc_z : 2;
SIG_VALTYPE_ 26 acc_y : 2;
SIG_VALTYPE_ 25 acc_x : 2;
SIG_VALTYPE_ 24 velStdDev_D : 2;
SIG_VALTYPE_ 23 velStdDev_E : 2;
SIG_VALTYPE_ 22 velStdDev_N : 2;
SIG_VALTYPE_ 21 vel_D : 2;
SIG_VALTYPE_ 20 vel_E : 2;
SIG_VALTYPE_ 19 vel_N : 2;
SIG_VALTYPE_ 18 bStdDev_D : 2;
SIG_VALTYPE_ 17 bStdDev_E : 2;
SIG_VALTYPE_ 16 bStdDev_N : 2;
SIG_VALTYPE_ 15 b_D : 2;
SIG_VALTYPE_ 14 b_E : 2;
SIG_VALTYPE_ 13 b_N : 2;
SIG_VALTYPE_ 12 ECEF_Z : 2;
SIG_VALTYPE_ 11 ECEF_Y : 2;
SIG_VALTYPE_ 10 ECEF_X : 2;
SIG_VALTYPE_ 9 height : 2;
SIG_VALTYPE_ 8 lon : 2;
SIG_VALTYPE_ 7 lat : 2;
SIG_VALTYPE_ 5 towInit : 2;
SIG_VALTYPE_ 3 tow : 2;
APPENDIX-3: DRAWING 3D-PRINTED CASING TYPE
Appendix 4: Drawing Industrial Casing Type

Frontpanel with connectors

Note:
This drawing applies to all Anav/Multi-Sensor RTK and RTCA modules in the industrial casing.

2021114
APPENDIX-5: DRAWING HIGH-CLASS GNSS-ANTENNA
APPENDIX-6: DRAWING SURVEY-GRADE GNSS-ANTENNA
APPENDIX-7: EXTRINSIC CALIBRATION FOR THE INTEGRATED SENSOR PLATFORM

This extrinsic calibration provides an initial calibration of the sensor positions and orientations, relative to the rear antenna of the ISP.

- Leverarms from origin of body frame (given in meters, body frame assumed to be on the rear antenna):

\[
\begin{align*}
\text{leverarm\_body\_to\_antenna\_rear} &= [0.0, 0.0, 0.0] \\
\text{leverarm\_body\_to\_antenna\_left} &= [0.708, -0.390, 0.0] \\
\text{leverarm\_body\_to\_antenna\_right} &= [0.708, 0.390, 0.0] \\
\text{leverarm\_body\_to\_msrtk\_imu} &= [0.493, 0.017, 0.106] \\
\text{leverarm\_body\_to\_lidar\_vlp16} &= [0.488, 0.0, -0.071] \\
\text{leverarm\_body\_to\_cam\_d435i} &= [0.566, 0.0, -0.03] \\
\text{leverarm\_body\_to\_cam\_d435i\_left} &= [0.566, -0.0175, -0.03] \\
\text{leverarm\_body\_to\_cam\_d435i\_right} &= [0.566, 0.0325, -0.03] \\
\text{leverarm\_body\_to\_cam\_gh3} &= [0.527, 0.0, -0.059] \\
\end{align*}
\]

1) Vertical and horizontal center of camera, at front of camera cover glass
2) 17.5mm from centerline of 1/4-20 to left imager (Figure 4-6 and Table 4-15 in [1])
3) Stereo baseline distance is 50mm (Figure 10-5, 10-8 in [1])
4) Vertical and horizontal center of camera, at front of lens holder (excluding lens)

- Rotation from body frame (given in Euler angles [deg] roll, pitch, yaw):

\[
\begin{align*}
\text{euler\_body\_to\_antenna\_rear} &= [0.0, 0.0, 0.0] \\
\text{euler\_body\_to\_antenna\_left} &= [0.0, 0.0, 0.0] \\
\text{euler\_body\_to\_antenna\_right} &= [0.0, 0.0, 0.0] \\
\text{euler\_body\_to\_msrtk\_imu} &= [0.0, 0.0, 0.0] \\
\text{euler\_body\_to\_lidar\_vlp16} &= [180.0, 0.0, 0.0] \\
\text{euler\_body\_to\_cam\_d435i} &= [90.0, 0.0, 90.0] \\
\text{euler\_body\_to\_cam\_d435i\_left} &= [90.0, 0.0, 90.0] \\
\text{euler\_body\_to\_cam\_d435i\_right} &= [90.0, 0.0, 90.0] \\
\text{euler\_body\_to\_cam\_gh3} &= [90.0, 0.0, 90.0] \\
\end{align*}
\]

1) When rotation is applied to a point, axis rotations are applied in the order roll, pitch, yaw, using fixed coordinate axis, ie. the coordinate frame is not moving

References:
[1] Intel RealSense D435i (related to “cam\_d435i” above):