# INS402 Interface Description

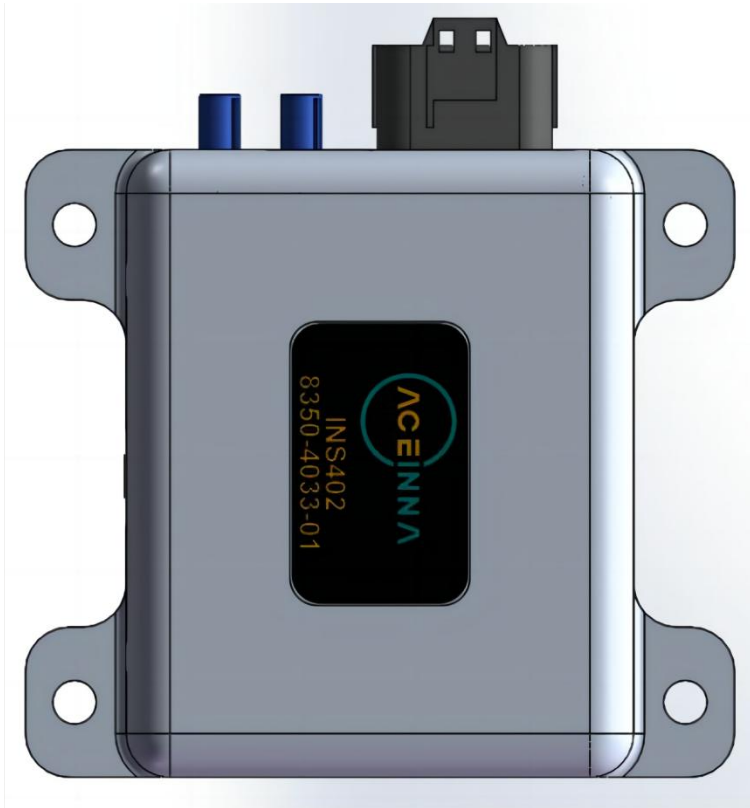
# (Preliminary)

## **Revision** History

| Revision | Date | Author | Change Description |
|---|---|---|---|
| 1.0 | 03/28/2023 | RJW/CEK | First version |
| 2.0 | 04/13/2023 | RJW/CEK | Second version, add bit status in chapter 3.3.1.1 and chapter 3.3.1.6, add Ethernet protocol, update picture of INS402, update DBC file, add CRC reference codes |

Contents

# 1 Introduction




The Aceinna INS402 high-precision navigation system integrates a multi-constellation, multi-frequency Global Navigation Satellite System (GNSS) (supports GPS, GALILEO, GLONASS, Beidou, QZSS, NAVIC and SBAS), an Aceinna MEMS Inertial Measurement Unit (IMU) module IMU330ZA, and a single Cypress multi-core Cortex-M7 micro-controller (MCU) CYT4BFBCHD as the main processor. The INS402 system includes built-in proprietary GNSS RTK positioning engine and a proprietary INS navigation engine

The Aceinna IMU330ZA is a triple-redundant 6-axis (3-axis accelerometer and 3-axis gyro) IMU module that encloses three MEMS IMU sensor chips, and each of the three IMU sensor chips is calibrated individually. The combined IMU sensor data provides 6-DOF (Degrees of Freedom) inertial measurements.

# 2 Scope

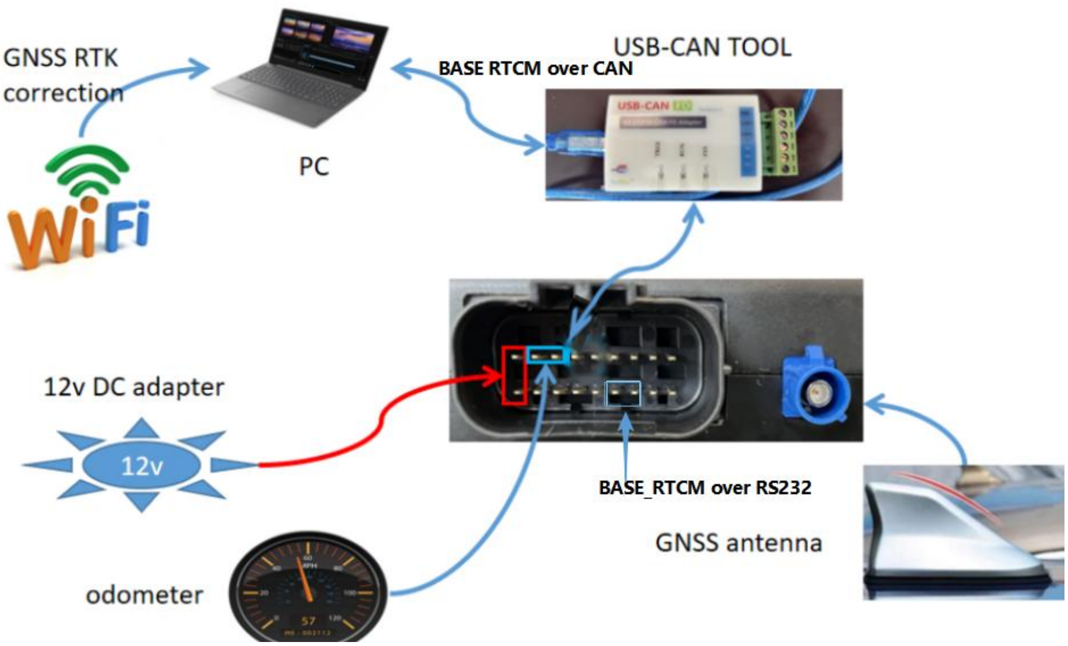
This document provides a description of the CAN, RS-232, and Ethernet interfaces on the INS402.

# 3 Interface Design

## 3.1 Hardware

The INS402 will ultimately support several interfaces including CAN, CANFD, Automotive Ethernet (100Mbps), and RS-232.

### 3.1.1 Connection

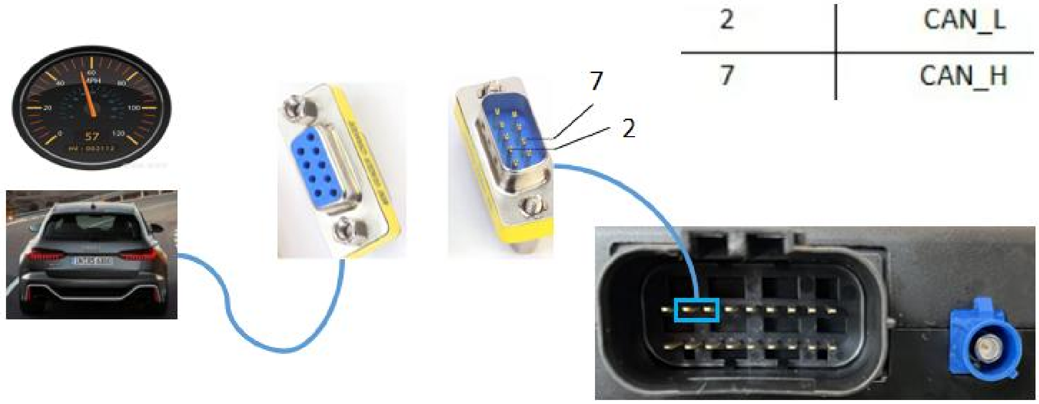



The overview of connections using CAN/RS232 interface with the vehicle is shown in the picture above. The primary interface is CAN J1939, while the RTCM data can be input to the INS402 via either CAN or RS-232 as shown.

There are several combinations of interconnections supported, as shown in table below

| | | | Input Messages Supported | | | Output Messages Supported | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Port | Protocol | Baud Rate | WSS | RTCM | Configuration | GNSS | INS | IMU | Diagnostic | NMEA-msg | RTCM |
| RS-232 | UART | 460.8k | NO | YES | NO | NO | NO | NO | NO | YES | NO |
| CAN-1 | CAN J1939 | 250k | YES(**3) | NO | YES(**1) | NO | YES | YES | YES | NO | NO |
| CAN-2 | future | | | | | | | | | | |
| CAN-3 | future | | | | | | | | | | |
| Ethernet | 100Base-T1 | 100M | YES | YES | YES(**2) | YES | YES | YES | YES | YES | YES |

**1: support configure leaver arms and save configuration. cannot enable/disable some packet.
**2: support configure leaver arms and save configuration. cannot enable/disable some packet.
**3: support 0x0CFF0145/46 odometer info now(see 3.3.2),Aceinna can modify and use new format based on customer input.




DB9 connector is recommended to connect with odometer signal line.

### 3.1.2 Channels of CAN/CANFD ports

CAN1: CAN-J1939, 250 Kbps

CAN2: reserved(in future release, CAN2 will support either CANFD or standard CAN)

CAN3: reserved(not populated)

### 3.1.3 General settings of RS232

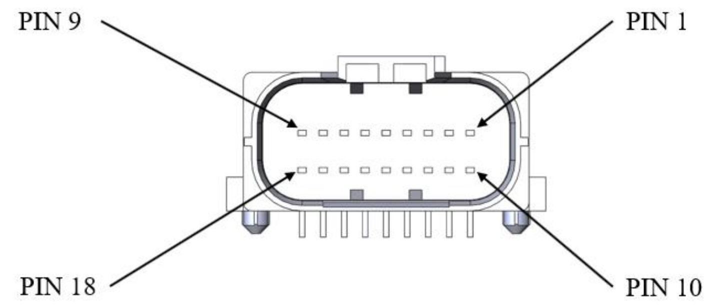
The serial port settings are RS232 with 1 start bit, 8 data bits, no parity bit, 1 stop bit, and no flow control. The serial port is used for RTCM input and ASCII NMEA messages.
Baud rate: 460800

### 3.1.4 RF Connector

An automotive grade FAKRA-J C type connector, manufactured by Molex, is used for the GNSS antenna connection. Its manufacturing part number is 734035112. The mating connector has part number 734036262. The center conductor carries the RF signal into the INS402 receiver and delivers 5 V DC from the INS402 to the external active antenna.
The master antenna should be connected with the RF connector which is near the main connector, the other RF connector for the slave antenna.
Installation: ensure that the 2 antennas have a clear view of the sky, and configure the lever arms based on accurate measurements of the antenna and INS locations (see detailed description of lever arm definition). For best performance the distance between master and slave antennas should be at least 1m.

### 3.1.5 Main Connector and Pin Description

The main connector carries all the other power and I/O signals to and from the INS402 module. This connector is also of automotive grade and is manufactured by JAE Electronics. The male end which is installed in the INS402 housing has part number MX23A18NF1; the female end, which is attached to the external wiring harness, has part number MX23A18SF1. Figure below illustrates the location of the 18 pins in the male part, as seen facing the connector from outside the module.




**Pin Diagram of the Male End**

This table shows the functional description of the 18 pins in the main connector.

| Pin # | INS402 | Signal Description |
|---|---|---|
| 1 | CAN3_L | reserved |
| 2 | CAN3_H | reserved |
| 3 | ETH_TRX_N | Ethernet Negative |
| 4 | ETH_TRX_P | Ethernet Positive |
| 5 | CAN2_L | CANFD Low |
| 6 | CAN2_H | CANFD High |
| 7 | CAN1_L | CAN Low |
| 8 | CAN1_H | CAN High |
| 9 | VCC_IN | Power Supply Positive, range 9v to 32V |
| 10 | Reserved | Reserved |
| 11 | Reserved | Reserved |
| 12 | RXD | RS-232 RX Pin |
| 13 | TXD | RS-232 TX Pin |
| 14 | GND | Power Supply Negative |
| 15 | GND | Power Supply Negative |
| 16 | PPS | Synchronization Signal |
| 17 | GND | Power Supply Negative |
| 18 | GND | Power Supply Negative |

### 3.1.6 Mounting Instructions

Use four bolts of 1/4-20 UNC socket head cap screw (ASME B18.3) to fix the INS402 system on a flat rigid panel on the vehicle, using the mechanical dimension measures shown in Figure 1. The IMU navigation center and the IMU body frame default coordinate definition is shown in Figure 1. Align the INS402 system x-axis with the forward driving direction of the vehicle, Y is right, and Z is down.

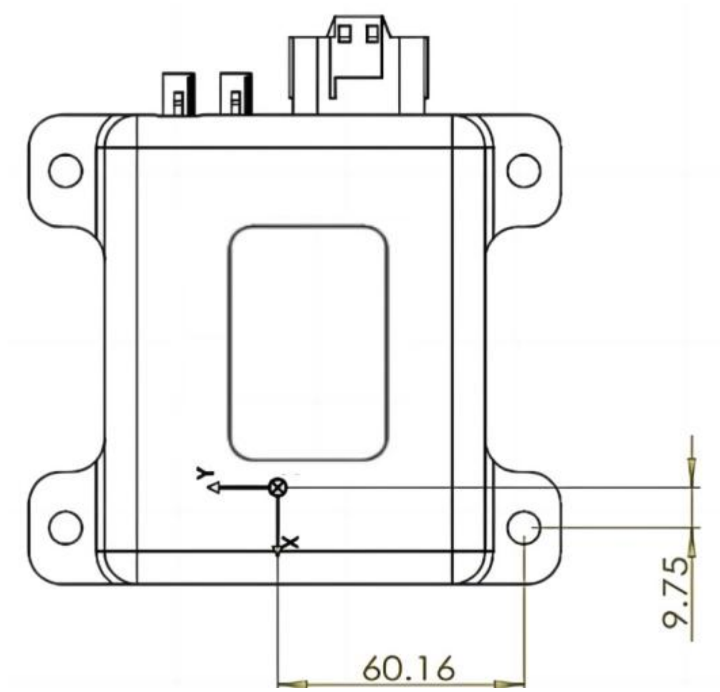



**Figure 1 INS402 Hardware Dimensions (reference point sign)**

By default, the IMU body frame orientation of INS402 is defined as in the figure above, with the X-axis pointing to the opposite direction of the connectors, Z-axis pointing down, and Y-axis completing a right-hand coordinate system. To align with the vehicle frame definition, the INS402 should be mounted on the vehicle with the connectors facing the tail of the vehicle, i.e., the X-axis of the IMU body frame points to the forward driving direction of the vehicle. If a different mounting orientation is used, the INS402 must be configured to re-align +X with the forward direction, +Y to the right, and +Z down. This is accomplished by the User Configuration Message (see Section 3.3.1).

For best performance, it is necessary to configure the lever arms, based on the mounting locations of the INS402 and the two antennas. Each lever arm specifies the translational 3D offset from the IMU navigation center (Figure 1) to the GNSS antenna phase center. For optimal performance, it is required to have the lever arm accuracy of less than 2 cm. For instance, a lever arm measurement of one antenna is shown in Figure 2. The translation offset is measured as 1 m in each direction of x, y, z. The IMU to the antenna lever arm is [x, y, z] = [1.0, -1.0, -1.0] m.

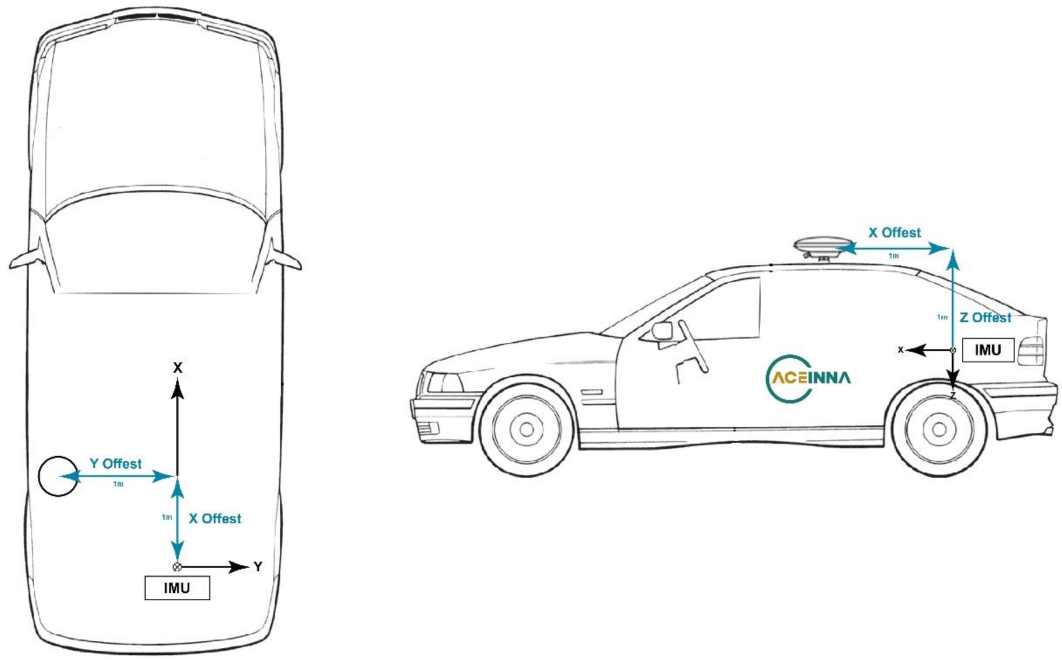



**Figure 2 INS402 IMU to GNSS Antenna Lever Arm Definition and Measurement Demonstration**

## 3.2 RS232 Messages Definition

ASCII packets are transited over RS232 interface.

### 3.2.1 ASCII Messages

The output ASCII Messages are the NMEA 0183 messages based on the NMEA 0183 version 4.10 standard.
Refer to 3.4.4 Output ASCII Messages

### 3.2.2 BASE_RTCM Message Input Over RS232

For optimal performance, it is necessary to enable RTK corrections. The performance is related to the distance between the rover and the base station (ideally 20 km or less). The error increases according to the factor of 1 ppm times the baseline length.

In order to perform GNSS RTK, the INS402 device needs RTK correction data (RTCM messages) input from user device (vehicle). The user device can send the RTCM messages over RS232 to the INS402, and refer to General settings of RS232.

| | |
|---|---|
| Message | GNSS RTK correction RTCM data from base station, periodic input at 1 Hz |

| | |
|---|---|
| Protocol | RTCM 3.2 |
| Length | Depending on the various RTCM message length |
| Payload | RTCM 3.2 protocol messages |

**Table 1 Base RTCM Data**

Input GNSS RTK correction data is in RTCM format based on the version 3.x protocol, including a series of RTCM messages, e.g., GPS observation (type ID 1077), GLONASS observation (type ID 1087), GPS ephemeris (type ID 1019), and so on. The different type of RTCM messages can be accommodated in one Aceinna binary packet. If the length of a full epoch of RTCM messages is less than 1490 (1500-10) bytes, they can fit in one ethernet data frame. Note that the 10 bytes are the overhead of one Aceinna binary packet.

## 3.3 CAN1 port and Messages Definition

The CAN1 port of the INS402 supports standard CAN J1939 messages, including position, velocity, attitude, heading, and IMU data. RTCM data, odometer data, and lever arm configurations can also be input over the CAN1 port. INS402 source address: 0x64(manufacturer defined, can modify if need)

| CAN ID | PGN | Priority | Cycle | Source to Destination | Description |
|---|---|---|---|---|---|
| 0x0CFF0264 | 0xFF02 | 3 | 50hz | INS to other node | INS_Acc |
| 0x0CFF0364 | 0xFF03 | 3 | 50hz | INS to other node | INS_GYRO |
| 0x0CFF0464 | 0xFF04 | 3 | 50hz | INS to other node | INS_HeadingPitchRoll |
| 0x0CFF0564 | 0xFF05 | 3 | 50hz | INS to other node | INS_HeightAndTime |
| 0x0CFF0664 | 0xFF06 | 3 | 50hz | INS to other node | INS_LatitudeLongitude |
| 0x0CFF0764 | 0xFF07 | 3 | 50hz | INS to other node | INS_Speed |
| 0x0CFF0864 | 0xFF08 | 3 | 50hz | INS to other node | INS_DataInfo |
| 0x0CFF0964 | 0xFF09 | 3 | 50hz | INS to other node | INS_Std |
| 0x18FF50XX (XX is client address in this file) | 0xFF50 | 6 | event | other node to INS | Set user configuration |
| 0x18EAFFXX | 0xEA00 | 6 | event | other node to INS | Request configuration |
| 0x18FF5064 | 0xFF50 | 6 | event | INS to other node | Configuration feedback |
| 0x18FF51XX | 0xFF51 | 6 | event | other node to INS | Save configuration |
| **Device information below** | | | | | |
| 0x18FF4064 | 0xFF40 | 6 | event | INS to other node | Product name |
| 0x18FF4164 | 0xFF41 | 6 | event | INS to other node | Serial No |
| 0x18FF4264 | 0xFF42 | 6 | event | INS to other node | App name |

| | | | | | |
|---|---|---|---|---|---|
| 0x18FF4364 | 0xFF43 | 6 | event | INS to other node | App version |
| 0x18FF4464 | 0xFF44 | 6 | event | INS to other node | Boot loader version |
| 0x18FF4564 | 0xFF45 | 6 | event | INS to other node | Hardware version |
| 0x18FF4664 | 0xFF46 | 6 | event | INS to other node | IMU name |
| 0x18FF4764 | 0xFF47 | 6 | event | INS to other node | IMU version |
| 0x18FF4864 | 0xFF48 | 6 | event | INS to other node | IMU serial No |
| 0x18FF4964 | 0xFF49 | 6 | event | INS to other node | RTK name |
| 0x18FF4A64 | 0xFF4A | 6 | event | INS to other node | RTK version |

**Table 2 All CAN messages**

**Note**: XX is client address in this file.

### 3.3.1 Output Binary Packets

The CAN messages include the input and output data packets, and the input packets are subject to user change. CAN1 output data frames are defined in the following table:

#### 3.3.1.1 INS_Acc And_IMU_BitStatus(ID: 0x0CFF0264, from CAN1, 50hz)

**PGN: 0xFF02**

**Direction: INS to other node**

| Name | Start bit | Length Bit | Value Type | Byte Order | Range | factor | Offset | Unit | Conversion | Description |
|---|---|---|---|---|---|---|---|---|---|---|
| ACC_X | 0 | 16 | Unsigned | Intel | [-4,4] | 0.0001220722 | -4 | g | E=N*0.0001220722-4 | Accelerometer: X axis |
| ACC_Y | 16 | 16 | Unsigned | Intel | [-4,4] | 0.0001220722 | -4 | g | E=N*0.0001220722-4 | Accelerometer: Y axis |
| ACC_Z | 32 | 16 | Unsigned | Intel | [-4,4] | 0.0001220722 | -4 | g | E=N*0.0001220722-4 | Accelerometer: Z axis |
| BitStatus_IMU_Master_Fail | 48 | 1 | Unsigned | Intel | [0,1] | 1 | 0 | - | E=N | |
| BitStatus_IMU_Hw_Err | 49 | 1 | Unsigned | Intel | [0,1] | 1 | 0 | - | E=N | |
| BitStatus_IMU_Sw_Err | 50 | 1 | Unsigned | Intel | [0,1] | 1 | 0 | - | E=N | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| BitStatus_IMU_Config_Err | 51 | 1 | Unsigned | Intel | [0,1] | 1 | 0 | - | E=N | |
| BitStatus_IMU_Calib_Err | 52 | 1 | Unsigned | Intel | [0,1] | 1 | 0 | - | E=N | |
| BitStatus_IMU_Accel_Degradation | 53 | 1 | Unsigned | Intel | [0,1] | 1 | 0 | - | E=N | |
| BitStatus_IMU_Gyro_Degradation | 54 | 1 | Unsigned | Intel | [0,1] | 1 | 0 | - | E=N | |
| BitStatus_IMU_Foreced_Restart | 55 | 1 | Unsigned | Intel | [0,1] | 1 | 0 | - | E=N | |
| BitStatus_IMU_Crc_Err | 56 | 1 | Unsigned | Intel | [0,1] | 1 | 0 | - | E=N | |
| BitStatus_IMU_Tx_Overflow_Err | 57 | 1 | Unsigned | Intel | [0,1] | 1 | 0 | - | E=N | |

#### 3.3.1.2 INS_GYRO (0x0CFF0364, from CAN1, 50hz)

**PGN: 0xFF03**

**Direction: INS to other node**

| Name | Start bit | Length Bit | Value Type | Byte Order | Range | factor | offset | Unit | Conversion | paraphrase |
|---|---|---|---|---|---|---|---|---|---|---|
| GYRO_X | 0 | 16 | Unsigned | Intel | [-250,250] | 0.00762951 | -250 | deg/s | E=N*0.00762951-250 | Gyro: Y axis |
| GYRO_Y | 16 | 16 | Unsigned | Intel | [-250,250] | 0.00762951 | -250 | deg/s | E=N*0.00762951-250 | Gyro: X axis |
| GYRO_Z | 32 | 16 | Unsigned | Intel | [-250,250] | 0.00762951 | -250 | deg/s | E=N*0.00762951-250 | Gyro: Z axis |

### 3.3.1.3 INS_PitchRollHeading (0x0CFF0464, from CAN1, 50hz)

**PGN: 0xFF04**

**Direction: INS to other node**

| Name | Start bit | Length Bit | Value Type | Byte Order | Range | factor | offset | Unit | Conversion | paraphrase |
|---|---|---|---|---|---|---|---|---|---|---|
| INS_PitchAngle | 0 | 16 | Unsigned | Intel | [-360,360] | 0.0109865 | -360 | deg | E=N*0.0109865-360 | Vehicle coordinate: pitch |
| INS_RollAngle | 16 | 16 | Unsigned | Intel | [-360,360] | 0.0109865 | -360 | deg | E=N*0.0109865-360 | Vehicle coordinate: roll |
| INS_HeadingAngle | 32 | 16 | Unsigned | Intel | [-360,360] | 0.0109865 | -360 | deg | E=N*0.0109865-360 | Vehicle coordinate: heading |

### 3.3.1.4 INS_HeightAndTime (0x0CFF0564, from CAN1, 50hz)

**PGN: 0xFF05**

**Direction: INS to other node**

| Name | Start bit | Length Bit | Value Type | Byte Order | Range | factor | offset | Unit | Conversion | paraphrase |
|---|---|---|---|---|---|---|---|---|---|---|
| INS_Height | 0 | 32 | Unsigned | Intel | [-10000,10000] | 0.001 | -10000 | m | E=N*0.001-10000 | altitude |
| INS_TimeOfWeek | 32 | 32 | Unsigned | Intel | [0,460800000] | 1 | 0 | ms | E=N | second in week |

### 3.3.1.5 INS_LatitudeLongitude (0x0CFF0664, from CAN1, 50hz)

**PGN: 0xFF06**

**Direction: INS to other node**

| Name | Start bit | Length Bit | Value Type | Byte Order | Range | factor | offset | Unit | Conversion | paraphrase |
|---|---|---|---|---|---|---|---|---|---|---|
| INS_ Latitude | 0 | 32 | Unsigned | Intel | [-180,180] | 0.0000001 | -180 | deg | E=N*1e-7-180 | latitude |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| INS_Longitude | 32 | 32 | Unsigned | Intel | [-180,180] | 0.0000001 | -180 | deg | E=N*1e-7-180 | longitude |

### 3.3.1.6 INS_Speed_And_BitStatus (0x0CFF0764, from CAN1, 50hz)

**PGN: 0xFF07**

**Direction: INS to other node**

| Name | Start bit | Length Bit | Value Type | Byte Order | Range | factor | offset | Unit | Conversion | paraphrase |
|---|---|---|---|---|---|---|---|---|---|---|
| INS_NorthSpeed | 0 | 16 | Unsigned | Intel | [-100,100] | 0.0030518 | -100 | m/s | E=N*0.0030518-100 | north speed |
| INS_EastSpeed | 16 | 16 | Unsigned | Intel | [-100,100] | 0.0030518 | -100 | m/s | E=N*0.0030518-100 | east speed |
| INS_ToGroundSpeed | 32 | 16 | Unsigned | Intel | [-100,100] | 0.0030518 | -100 | m/s | E=N*0.0030518-100 | vertex speed |
| BitStatus_PPS_Status | 48 | 1 | Unsigned | Intel | [0, 1] | 1 | 0 | - | E=N | |
| BitStatus_GNSS_Data_Status | 49 | 1 | Unsigned | Intel | [0, 1] | 1 | 0 | - | E=N | |
| BitStatus_GNSS2_Data_Status | 50 | 1 | Unsigned | Intel | [0, 1] | 1 | 0 | - | E=N | |
| BitStatus_GNSS_Signal_Status | 51 | 1 | Unsigned | Intel | [0, 1] | 1 | 0 | - | E=N | |
| BitStatus_Power | 52 | 1 | Unsigned | Intel | [0, 1] | 1 | 0 | - | E=N | |
| BitStatus_MCU_Status | 53 | 1 | Unsigned | Intel | [0, 1] | 1 | 0 | - | E=N | |
| BitStatus_Temperature_Under_MCU | 54 | 1 | Unsigned | Intel | [0, 1] | 1 | 0 | - | E=N | |
| BitStatus_Temperature_Under_RTK | 55 | 1 | Unsigned | Intel | [0, 1] | 1 | 0 | - | E=N | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| BitStatus_Temperature_Under_IMU | 56 | 1 | Unsigned | Intel | [0, 1] | 1 | 0 | - | E=N | |
| BitStatus_Temperature_Over_MCU | 57 | 1 | Unsigned | Intel | [0, 1] | 1 | 0 | - | E=N | |
| BitStatus_Temperature_Over_RTK | 58 | 1 | Unsigned | Intel | [0, 1] | 1 | 0 | - | E=N | |
| BitStatus_Temperature_Over_IMU | 59 | 1 | Unsigned | Intel | [0, 1] | 1 | 0 | - | E=N | |

#### 3.3.1.7 INS_DataInfo (0x0CFF0864, from CAN1, 50hz)

**PGN: 0xFF08**

**Direction: INS to other node**

| Name | Start bit | Length Bit | Value Type | Byte Order | factor | offset | Convertion | paraphrase |
|---|---|---|---|---|---|---|---|---|
| INS_GpsFlag_Pos | 0 | 8 | Unsigned | Intel | 1 | 0 | E=N | gps solution<br>0: INVALID<br>1: Single-point positioning (SPP)<br>2: Real time differential GNSS (RTD)<br>4: Real time kinematic (RTK), ambiguity fixed (RTK_FIXED)<br>5: RTK with ambiguity float (RTK_FLOAT) |
| INS_NumS_Used | 8 | 8 | Unsigned | Intel | 1 | 0 | E=N | GPS used satellites |
| INS_PosType | 16 | 8 | Unsigned | Intel | 1 | 0 | E=N | ins solution<br>0: INVALID<br>1: SPP/INS<br>2: RTD/INS<br>3: INS_PROPAGATE<br>4: RTK_FIXED/INS<br>5: RTK_FLOAT/INS |
| INS_Gps_Age | 24 | 8 | Unsigned | Intel | 1 | 0 | E=N | age |
| INS_Car_Status | 32 | 8 | Unsigned | Intel | 1 | 0 | E=N | car speed status<br>0: car speed used<br>1: car speed unused |
| INS_Status | 40 | 8 | Unsigned | Intel | 1 | 0 | E=N | Ins status<br>(INS initialization need Base_RTCM data input)<br>0: INVALID<br>1: INS_ALIGNING<br>2: INS_HIGH_VARIANCE |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | 3: INS_SOLUTION_GOOD<br>4: INS_SOLUTION_FREE<br>5: INS_ALIGNMENT_COMPLETE |
| INS_Week | 48 | 16 | Unsigned | Intel | 1 | 0 | E=N | Gps week number |

#### 3.3.1.8 INS_Std (0x0CFF0964, from CAN1, 50hz)

**PGN: 0xFF09**

**Direction: INS to other node**

| Name | Start bit | Length Bit | Value Type | Byte Order | Range | factor | offset | Unit | paraphrase |
|---|---|---|---|---|---|---|---|---|---|
| INS_Latitude_Std | 0 | 16 | Unsigned | Intel | [0,65.535] | 0.001 | 0 | E=N*0.001 | Standard deviation of latitude |
| INS_Longitude_Std | 16 | 16 | Unsigned | Intel | [0,65.535] | 0.001 | 0 | E=N*0.001 | Standard deviation of longitude |
| INS_LocatHeightStd_ | 32 | 16 | Unsigned | Intel | [0,65.535] | 0.001 | 0 | E=N*0.001 | Standard deviation of height |
| INS_Heading_Std | 48 | 16 | Unsigned | Intel | [0,65.535] | 0.001 | 0 | E=N*0.001 | Standard deviation of heading |

### 3.3.2 Input Binary Packets

#### 3.3.2.1 Set user configuration (0x18FF50XX, CAN1)

"Set user configuration" is used to configure data value based on configurations index, if you want to save it in flash, use "save configuration" command, refer to 3.3.2.6

**PGN: 0xFF50**

**Direction: Other node to INS (set user configuration), or INS to Other node (feedback)**

| CAN ID | Byte0 | Byte1 | Byte2 | Byte3 |
|---|---|---|---|---|
| 0x18FF50XX | Destination address | Index of configuration | Unsigned Data-LSB | Unsigned Data-MSB |

*scale factor of Data, pls refer to Table 3 Index of Configurations

**Example of ID and payload:**

0x18FF5000 payload: 64 01 44 7C: 0x7C44 = 31812 * 0.001 + (-30) = 1.812 m

| Index | Configuration | unit | Factor | Offset | Comments |
|---|---|---|---|---|---|
| 1 | Pri_lever_arm_x | m | 0.001 | -30(m) | Master Antenna position under NED coordinate system (forward-right-down vehicle frame), reference point of INS402 is origin point (0,0,0). Same coordinate system is used for below points. |
| 2 | Pri_lever_arm_y | m | 0.001 | -30(m) | |
| 3 | pri_lever_arm_z | m | 0.001 | -30(m) | |
| 4 | vrp_lever_arm_x | m | 0.001 | -30(m) | Vehicle reference point position.<br>vrp is the position where odometer speed come from, which would be the center position of 2 rear wheels if your speed comes from 2 rear wheels. |
| 5 | vrp_lever_arm_y | m | 0.001 | -30(m) | |
| 6 | vrp_lever_arm_z | m | 0.001 | -30(m) | |
| 7 | user_lever_arm_x | m | 0.001 | -30(m) | Interesting point position, it could be the position of one antenna or any position you want. INS will send out the specified position in output packets. |
| 8 | user_lever_arm_y | m | 0.001 | -30(m) | |
| 9 | user_lever_arm_z | m | 0.001 | -30(m) | |
| 10 | Rotation_rbv_x | deg | 0.01 | -180(deg) | Rotation angles to align IMU body frame to vehicle frame, in order Z->Y->X |
| 11 | Rotation_rbv_y | deg | 0.01 | -180(deg) | |
| 12 | Rotation_rbv_z | deg | 0.01 | -180(deg) | |
| 13 | Second_lever_arm_x | m | 0.001 | -30(m) | Slave Antenna position under NED coordinate system(forward-right-down), reference point of INS402 is origin point (0,0,0). |
| 14 | Second_lever_arm_y | m | 0.001 | -30(m) | |
| 15 | Second_lever_arm_z | m | 0.001 | -30(m) | |
| 22 | Dual ant switch | - | 1 | 0 | 0-Disable heading calibration<br>1-Default. Enable heading calibration |
| 23 | Baseline length | m | 0.001 | -30(m) | Distance between the 2 antennas (prefer > 1m) |
| 24 | Heading calibration | deg | 0.01 | -180(deg) | Compensate the small angle (from master to slave) bias caused by lever arm measurement accuracy.<br>Take example:<br>The calibration angle should be -0.5deg, when actual heading(89.5deg) different with calculated heading(90.0deg) based on current lever arms |

**Table 3 Index of Configurations**

***Reminder for new ID value configuration:**

The new value is valid only in current power cycle after set and save command is needed for permanent configuration.

#### 3.3.2.2 Get user configuration (0x18EAFFXX, CAN1)

Request user configuration has feedback.

**Request:**

**PGN: 0xEA00**

**Direction: other node to INS**

| CAN ID | Byte0 | Byte1 | Byte2 | Byte3 |
|---|---|---|---|---|
| 0x18EAFFXX | 0x50 | 0xFF | 0x00 | Index you required |

**Feedback:**

**Direction: INS to other node**

| CAN ID | Byte0 | Byte1 | Byte2 | Byte3 |
|---|---|---|---|---|
| 0x18FF5064 | 0x64(INS address) | Index you required | unsigned Data-LSB | unsigned Data-MSB |

*scale factor of Data, please refer to Table 3 Index of Configurations

#### 3.3.2.3 Get Serial Number (0x18FF4164, CAN1)

**Request:**

**PGN: 0xEA00**

**Direction: other node to INS**

| CAN ID | Byte0 | Byte1 | Byte2 |
|---|---|---|---|
| 0x18EAFFXX | 0x41 | 0xFF | 0x00 |

**Feedback:**

**Direction: INS to other node**

| CAN ID | Byte0 | Byte1 | Byte2 | Byte3(MSB) |
|---|---|---|---|---|
| 0x18FF4164 | 0xDC(LSB) | 0xD2 | 0xD5 | 0x8D(MSB) |

*example: 0x8DD5D2DC(decimal: 2379600604)

#### 3.3.2.4 Get App version (0x18FF4364, CAN1)

**Request:**

**PGN: 0xEA00**

**Direction: other node to INS**

| CAN ID | Byte0 | Byte1 | Byte2 |
|---|---|---|---|
| 0x18EAFFXX | 0x43 | 0xFF | 0x00 |

**Feedback:**

**Direction: INS to other node**

| CAN ID | Byte0 | Byte1 | Byte2 | Byte3 | Byte4 | Byte5 | Byte6 | Byte7 |
|---|---|---|---|---|---|---|---|---|
| 0x18FF4364 | ‘3’ | ‘1’ | ‘.’ | ‘0’ | ‘0’ | ‘.’ | ‘0’ | ‘3’ |

*example: v31.00.03

### 3.3.2.5 Wheel Speed Input (0x0CFF0145/46, to CAN1)

The speed of the left and right rear wheels needs to be input, 0x45(SA) is left and 0x46(SA) is right.

**PGN: 0xFF01**

| Name | Start bit | Length Bit | Value Type | Byte Order | Range | factor | Offset | Unit | Conversion | Description |
|---|---|---|---|---|---|---|---|---|---|---|
| Speed | 40 | 16 | signed | Intel | [-21.53, 21.53] | 0.000657203 | 0 | m/s | E=N*0.000657203 | TC_Motor_RPM |

TC_Motor_RPM (bytes 5-6) is signed:

- Position value: moving forward.
- Negative value: move backward.

**Examples:**

Speed calculation based on TC_Motor_RPM:

- Moving forward 10m/s:
  - Right and left both: Byte5-0x70, byte6-0x3B
  - Rev/min = (TC_Motor_RPM_L+TC_Motor_RPM_R)/2 = 0x3B70 = 15216
  - Speed = 15216 * 0.000657203 = +10.0 m/s
- Moving backward -10m/s:
  - Right and left both: Byte-0x90, byte6-0xC4
  - Rev/min = (TC_Motor_RPM_L+TC_Motor_RPM_R)/2 = 0xC490= -15216 (complement of signed 16-bit value)
  - Speed = -15216 * 0.000657203 = -10.0 m/s

### 3.3.2.6 Save configuration (0x0CFF51XX, CAN1)

To save user configuration in flash. has no feedback

**PGN: 0xFF51**

**Direction: other node to INS**

| CAN ID | Byte0 |
|---|---|
| 0x18FF51XX | 0x64(address of INS) |

### 3.3.3 DBC FILE

Pls download DBC from link:

https://navview.blob.core.windows.net/forum/upload/ins402_mosaic_20230413-lhfzn1yn.dbc

## 3.4 Ethernet Port and Messages

The INS402 ethernet interface is defined to have the following functions:

- Data I/O to communicate with the external system control unit
- The firmware upgrade interface via the Bootloader

The communication protocol data are defined in section 3.4.1. An ethernet data frame is first defined, with two types of messages – Binary packets and ASCII messages described below to be embedded in the ethernet data frame as effective payload:

- Binary packets: Aceinna proprietary binary data format
- ASCII messages: standard NMEA0183 messages, including GNGGA which is mandatory to be sent to the NTRIP server to report location and fetch GNSS correction data from nearby base station

### 3.4.1 Ethernet Data Frame Definition

The INS402 ethernet TX/RX data frame conforms to IEEE802.3, the format is shown below. The input/output binary packets and ASCII messages should be filled in the Ethernet data frame payload. Note it is also designed to contain two packets/messages in the same one data frame. If one packet/message is more than 1500 bytes, it is split into two continuous data frames.

| Destination Address | Source Address | Length | Payload | Data Frame Checksum |
|---|---|---|---|---|
| 6 bytes | 6 bytes | 2 bytes | 46 to 1500 bytes | 4 bytes |

- Destination Address: destination MAC address (FF:FF:FF:FF:FF:FF or 04:00:00:00:00:04 can be used when the MAC address is unknown)
- Source Address: source MAC address
- Length: length of user data (in the range of 46~1500 bytes), MSB for output message from INS402 LSB for input message to INS402.
- Payload: Min/Max payload length is 46/1500. Zero bytes will be filled if the payload is less than 46 bytes
- Data Frame Checksum: CRC checksum bytes

### 3.4.2 Aceinna Binary Packet Format

The Aceinna proprietary binary packet format is defined as below.

| Header (uint16) | Message ID (uint16) | Length (uint32) | Payload | Checksum (uint16) |
|---|---|---|---|---|

- Header: packet starts with header bytes 0x5555
- Message ID: the ID of the current message, LSB-first
- Length: the number of bytes in the payload, LSB-first
- Payload: user data contents, LSB-first
- Checksum: CRC16 check. Bytes from the beginning of the "Message ID" to the end of the "payload" are included in the checksum calculation, and a sample of the checksum algorithm C code is shown in Appendix A:16-bit CRC Implementation Sample Code. Binary data of CRC is in Little Endian format.

## 3.4.3 Output Binary Packets

Three types of output data binary packets are defined: GNSS solution, INS solution and diagnostic message.

### 3.4.3.1 GNSS Solution Packet

**Table 4 GNSS Solution Packet**

| **Message** | **GNSS positioning solution data, periodic output at 1 Hz** | | | |
|---|---|---|---|---|
| Message ID | 0x0a02 | | | |
| Length | 77 | | | |
| Payload Description: | | | | |
| **Byte Offset** | **Type** | **Name** | **Unit** | **Description** |
| 0 | uint16 | gps_week | | GPS time:<br>GPS week and seconds in week |
| 2 | uint32 | gps_millisecs | ms | |
| 6 | uint8 | position_type | | 0: INVALID<br>1: Single-point positioning (SPP)<br>2: Real time differential GNSS (RTD)<br>4: Real time kinematic (RTK), ambiguity fixed (RTK_FIXED)<br>5: RTK with ambiguity float (RTK_FLOAT) |
| 7 | double | latitude | deg | Geodetic latitude |
| 15 | double | longitude | deg | Geodetic longitude |
| 23 | double | height | m | Height above ellipsoid |
| 31 | float | latitude_std | m | Latitudinal position accuracy |
| 35 | float | longitude_std | m | Longitudinal position accuracy |
| 39 | float | height_std | m | Vertical position accuracy |
| 43 | uint8 | numberOfSVs | | Number of satellites |

| | | | | |
|---|---|---|---|---|
| 44 | uint8 | numberOfSVs_in_solution | | Number of satellites in solution |
| 45 | float | Hdop | | Horizontal Dilution of Precision |
| 49 | float | Diffage | s | Age of differential GNSS correction |
| 53 | float | north_vel | m/s | North velocity |
| 57 | float | east_vel | m/s | East velocity |
| 61 | float | up_vel | m/s | Up velocity |
| 65 | float | north_vel_std | m/s | North velocity accuracy |
| 69 | float | east_vel_std | m/s | East velocity accuracy |
| 73 | float | up_vel_std | m/s | Up velocity accuracy |

### 3.4.3.2 INS navigation solution

**Table 5 INS Solution Packet**

| **Message** | **INS navigation solution, periodic output at 100 Hz** | | | |
|---|---|---|---|---|
| Message ID | 0x0a03 | | | |
| Length | 110 | | | |
| Payload Description: | | | | |
| **Byte Offset** | **Type** | **Name** | **Unit** | **Description** |
| 0 | uint16 | gps_week | | GPS time:<br>GPS week and seconds in week |
| 2 | uint32 | gps_millisecs | ms | |
| 6 | uint8 | ins_status | | 0: INVALID<br>1: INS_ALIGNING<br>2: INS_HIGH_VARIANCE<br>3: INS_SOLUTION_GOOD<br>4: INS_SOLUTION_FREE<br>5: INS_ALIGNMENT_COMPLETE |
| 7 | uint8 | ins_position_type | | 0: INVALID<br>1: SPP/INS<br>2: RTD/INS<br>3: INS_PROPAGATE<br>4: RTK_FIXED/INS<br>5: RTK_FLOAT/INS |
| 8 | double | latitude | deg | Geodetic latitude |
| 16 | double | longitude | deg | Geodetic longitude |
| 24 | double | height | m | Height above ellipsoid |
| 32 | float | north_velocity | m/s | North velocity in navigation ENU frame |
| 36 | float | east_velocity | m/s | East velocity in navigation ENU frame |
| 40 | float | up_velocity | m/s | Up velocity in navigation ENU frame |
| 44 | float | longitudinal_velocity | m/s | Forward velocity in vehicle frame |

| | | | | |
|---|---|---|---|---|
| 48 | float | lateral_velocity | m/s | Lateral velocity in vehicle frame |
| 52 | float | roll | deg | Vehicle roll |
| 56 | float | pitch | deg | Vehicle pitch |
| 60 | float | heading | deg | Vehicle heading |
| 64 | float | latitude_std | m | Latitudinal position accuracy |
| 68 | float | longitude_std | m | Longitudinal position accuracy |
| 72 | float | height_std | m | Vertical position accuracy |
| 76 | float | north_velocity_std | m/s | North velocity accuracy |
| 80 | float | east_velocity_std | m/s | East velocity accuracy |
| 84 | float | up_velocity_std | m/s | Up velocity accuracy |
| 88 | float | long_vel_std | m/s | Longitudinal velocity accuracy |
| 92 | float | lat_vel_std | m/s | Lateral velocity accuracy |
| 96 | float | roll_std | deg | Vehicle roll accuracy |
| 100 | float | pitch_std | deg | Vehicle pitch accuracy |
| 104 | float | heading_std | deg | Vehicle heading accuracy |
| 108 | int16 | Continent ID | | Continent ID:<br>ID_NONE = -2,<br>ID_ERROR = -1,<br>ID_UNKNOWN = 0,<br>ID_ASIA = 1,<br>ID_EUROPE = 2,<br>ID_OCEANIA = 3,<br>ID_AFRICA = 4,<br>ID_NORTHAMERICA = 5,<br>ID_SOUTHAMERICA = 6,<br>ID_ANTARCTICA = 7 |

#### 3.4.3.3 Diagnostic Message

**Table 6 Diagnostic Message**

| **Message** | **Device diagnostic message, periodic output at 1 Hz** | | | |
|---|---|---|---|---|
| Message ID | 0x0a05 | | | |
| Length | 22 | | | |
| Payload Description: | | | | |
| **Byte Offset** | **Type** | **Name** | **Unit** | **Description** |
| 0 | uint16 | gps_week | | GPS time:<br>GPS week and seconds in week |
| 2 | uint32 | gps_millisecs | ms | |
| 6 | uint32 | Device status bit field | | The Definition is listed in Table 6. Each status bit has two values:0 is valid and 1 is invalid. |
| 10 | float32 | IMU temperature | ℃ | Temperature of the IMU |

| | | | | |
|---|---|---|---|---|
| 14 | float32 | MCU temperature | ℃ | Temperature of the MCU |
| 18 | float32 | GNSS-chip temperature | ℃ | Temperature of the GNSS-CHIP chipset |

**Table 7 Typedef struct--Device status bit field**

| | Bit | Description | Value |
|---|---|---|---|
| IMU | 0 | Master fail | 0: normal<br>1: fatal error occurred |
| | 1 | HW error | 0: normal<br>1: hardware exception detected |
| | 2 | SW error | 0: normal<br>1: software exception detected |
| | 3 | Config error | 0: normal<br>1: config error detected by periodic self-test |
| | 4 | Calibration error | 0: normal<br>1: calibration data corrupted |
| | 5 | Accel degradation | 0: normal<br>1: accel data degradation due to sensor exception |
| | 6 | Gyro degradation | 0: normal<br>1: gyro data degradation due to sensor exception |
| | 7 | Forced restart | 0: normal<br>1: forced restart |
| | 8 | CRC error | 0: normal<br>1: CRC error detected |
| | 9 | Tx overflow error | 0: normal<br>1: Tx Overflow occurred 10 consecutive cycles |
| GNSS | 10 | PPS status | 0: normal<br>1: 1PPS pulse exception |
| | 11 | GNSS data status | 0: normal<br>1: GNSS-chipset has NO data output |
| | 12 | GNSS signal status | 0: normal<br>1: GNSS-chipset has data output, but no valid signal detected |
| Operation | 13 | Power | 0: normal<br>1: any component has no power |
| | 14 | MCU status | 0: normal<br>1: MCU failure<br>MCU and peripherals HW self-test status when power on |
| | 15 | Temperature under MCU flag | 0: normal<br>1: under temperature |

| | | | |
|---|---|---|---|
| | 16 | Temperature under STA flag | 0: normal<br>1: under temperature |
| | 17 | Temperature under IMU flag | 0: normal<br>1: under temperature |
| | 18 | Temperature over MCU flag | 0: normal<br>1: under temperature |
| | 19 | Temperature over STA flag | 0: normal<br>1: under temperature |
| | 20 | Temperature over IMU flag | 0: normal<br>1: under temperature |
| Reserved | 21:31 | 11bit | |

The 1PPS signal on pin 16 is a 1 Hz periodic signal with 50% duty cycle. The rising edge of the 1PPS has a jitter of 30 ns (typical). The timing of the PPS signal is valid only when, prior to the rising edge of the 1PPS output, the pps_status bit is set to 0 in the device diagnostic message (message ID 0x0a05).

**Table 8 GNSS-CHIP Diagnostic Packet**

| **Message** | **GNSS-CHIP diagnostic data, periodic output at 1 Hz** | | | |
|---|---|---|---|---|
| Message ID | 0x6664 | | | |
| Length | 5 | | | |
| Payload Description: | | | | |
| **Byte Offset** | **Type** | **Name** | **Unit** | **Description** |
| 0 | uint8 | gnss_err_out_pin | | 0: EOUT is normal;<br>1: EOUT error. |
| 1 | uint8 | gnss_rf_err_pin | | 0: RF_ERR is normal;<br>1: RF_ERR error. |
| 2 | uint8 | gnss_ant_err_pin | | 0: ANT_ERR is normal;<br>1: ANT_ERR error. |
| 3 | uint8 | gnss_handshake_flag | | 0: Handshake between MCU and GNSS-CHIP fail;<br>1: Handshake between MCU and GNSS-CHIP OK. |
| 4 | uint8 | gnss_reset_pin | | 0: GNSS-CHIP reset;<br>1: GNSS-CHIP works normally. |

**Table 9 DM Extent Packet1**

| **Message** | **Extent message for diagnostics, periodic output at 1 Hz** | | | |
|---|---|---|---|---|
| Message ID | 0x4D44 | | | |
| Length | 8 | | | |
| Payload Description: | | | | |
| **Byte Offset** | **Type** | **Name** | **Unit** | **Description** |

| | | | | |
|---|---|---|---|---|
| 0 | uint32 | StFdHw | | Feedback of HW self-testing |
| 4 | Reserved | - | | |

**Table 10 Typedef struct--StFdHw**

| | Bit | Description | Value |
|---|---|---|---|
| StFdHw | 0:3 | Adc<br>voltage monitor | 0: normal<br>1: wave of voltage out of range<br>Bit0: ADC_IN<br>Bit1: ADC_CORE<br>Bit2: ADC_1V2<br>Bit3: ADC_GND |
| | 4 | ExtOsc<br>external oscillator | 0: normal<br>1: oscillator exception detected |
| | 5 | Power chip | 0: normal<br>1: power exception detected |
| | 6:8 | RAM | 0: normal<br>1: ram error detected by MCU<br>Bit0: SRAM0<br>Bit1: SRAM1<br>Bit2: SRAM2 |
| | 9 | Watch-dog | 0: normal<br>1: watch-dog error detected |
| | 10:11 | Flash | 0: normal<br>1: flash error detected<br>Bit0: cFlash,<br>Bit1: wFlash |
| | 12 | clkCSV<br>Core oscillator | 0: normal<br>1: core oscillator error detected by MCU |
| | 13:31 | 19bits Reserved | |

#### 3.4.3.4 Raw IMU Data

**Table 11 Raw IMU Data**

| **Message** | **Raw IMU data, periodic output at 100 Hz** | | | |
|---|---|---|---|---|
| Message ID | 0x0a01 | | | |
| Length | 30 | | | |
| Payload Description: | | | | |
| **Byte Offset** | **Type** | **Name** | **Unit** | **Description** |
| 0 | uint16 | gps_week | | GPS week |

| | | | | |
|---|---|---|---|---|
| 2 | uint32 | gps_millisecs | ms | GPS time of week |
| 6 | float | accel_x | m/s^2 | accel x axis measurement |
| 10 | float | accel_y | m/s^2 | accel y axis measurement |
| 14 | float | accel_z | m/s^2 | accel z axis measurement |
| 18 | float | gyro_x | deg/s | gyro x axis measurement |
| 22 | float | gyro_y | deg/s | gyro y axis measurement |
| 26 | float | gyro_z | deg/s | gyro z axis measurement |

#### 3.4.3.5 RTCM Data

The RTCM data contains raw data in RTCM format for the GNSS observables, such as satellite carrier phase, pseudo range and Doppler.

**Table 12 RTCM Data**

| **Message** | **RTCM data, periodic output at 10 Hz** |
|---|---|
| Message ID | 0x0a06 |
| Length | 1~1024bytes |
| Payload | |

INS402 output following RTCM information as rover

1. Ephemeris (standard RTCM): 1019(GPS), 1020(GLONASS), 1044(QZSS), 1045(Galileo F/NAV), 1046(Galileo I/NAV), 63/1042 (Beidou)
2. MSM (standard RTCM): 1077(GPS), 1087(GLONASS), 1097(GAL), 1117(QZSS), 1127(BDS)
3. receiver position (standard RTCM): 1005, 1006
4. ST defined RTCM: 999 (RF, measurement & PVT info)

## NOTE

Raw IMU Data will output immediately after the unit boots up from a power cycle.

### 3.4.4 Output ASCII Messages

The output ASCII Messages are the NMEA 0183 messages based on the NMEA 0183 version 4.10 standard. Refer to http://www.nmea.org/ for more information.

The tables below describe the brief meaning of each field that is delimited by comma in the NMEA messages.

## ☑ EXAMPLE of GNGGA

$GNGGA, 172814.10, 3723.46587704, N, 12202.26957864, W, 2, 6, 1.2, 18.893, M, -25.669, M, 2.0, 0031*4F

**Table 13 GNGGA Packet**

| **Field Sequence** | **Description** |
|---|---|

| | |
|---|---|
| 0 | Message ID $GNGGA |
| 1 | UTC time of position fix |
| 2 | Latitude |
| 3 | Direction of latitude: N – North; S - South |
| 4 | Longitude |
| 5 | Direction of longitude: E - East; W - West |
| 6 | GNSS Positioning Quality indicator:<br>0: GNSS position Fix not valid<br>1: GNSS position fix (SPP)<br>2: Differential GNSS position fix (RTD)<br>4: Real-Time Kinematic, fixed integers<br>5: Real-Time Kinematic, float integers |
| 7 | Number of SVs in use, range from 00 through to 40 |
| 8 | HDOP |
| 9 | Orthometric height (MSL reference) |
| 10 | M: unit of measure for orthometric height is meters |
| 11 | Geoid separation |
| 12 | M: geoid separation measured in meters |
| 13 | Age of differential GNSS correction data. Null field when differential GNSS is not used. |
| 14 | Reference station ID, range 0000-4095. A null field when no reference station ID is available, and no corrections are received. |
| 15 | The checksum data, always begins with * |

## ☑ EXAMPLE

$GNZDA,172809.40,12,07,1996,00,00*45

**Table 14 GNZDA Packet**

| Field Sequence | Description |
|---|---|
| 0 | Message ID $GNZDA |
| 1 | UTC |
| 2 | Day, ranging between 01 and 31 |
| 3 | Month, ranging between 01 and 12 |
| 4 | Year |
| 5 | Local time zone offset from GMT, ranging from 00 through ±13 hours |
| 6 | Local time zone offset from GMT, ranging from 00 through 59 minutes |
| 7 | The checksum data, always begins with * |

### 3.4.5 Input Binary Packets

#### 3.4.5.1 RTCM correction data

In order to perform GNSS RTK, the INS402 device needs RTK correction data (RTCM messages) input from user device (vehicle). These RTCM messages can be sent over ethernet or RS232; the INS402 will automatically detect data on either port. The input binary packets for this function are defined in Table 15, followed by a detailed example.

**Table 15 Base RTCM Data**

| **Message** | **GNSS RTK correction RTCM data from base station, periodic input at 1 Hz** |
|---|---|
| Message ID | 0x0b02 |
| Length | Depending on the various RTCM message length |
| Payload | RTCM 3.2 protocol messages |

Input GNSS RTK correction data is in RTCM format based on the version 3.x protocol, including a series of RTCM messages, e.g., GPS observation (type ID 1077), GLONASS observation (type ID 1087), GPS ephemeris (type ID 1019), and so on. The different type of RTCM messages can be accommodated in one Aceinna binary packet. If the length of a full epoch of RTCM messages is less than 1490 (1500-10) bytes, they can fit in one ethernet data frame. Note that the 10 bytes are the overhead of one Aceinna binary packet.

INS402 output following RTCM information from base or RTK corrections

1. Ephemeris (standard RTCM): 1019(GPS), 1020(GLONASS), 1044(QZSS), 1045(Galileo F/NAV),1046(Galileo I/NAV), 63/1042 (Beidou)
2. MSM (standard RTCM): 1074(GPS), 1084(GLONASS), 1094(GAL), 1114(QZSS), 1124(BDS)
3. Base station position and station ID (standard RTCM): 1005, 1006

## ☑ EXAMPLE

Assuming that the length of an epoch of RTCM messages is 1024 bytes, the ethernet data frame is described as below:

**Table 16 Example of RTCM Message**

| | | | | |
|---|---|---|---|---|
| Destination Address | 6 bytes | INS402 MAC address, 0x02 0x00 0x00 0x00 0x00 0x02 | | |
| Source Address | 6 bytes | User controller board MAC address, 0x04 0x00 0x00 0x00 0x00 0x04 | | |
| Length | 2 bytes | 1024+10=1034 | | |
| User Data | Base RTCM packet length | Header | 2 bytes | 0x5555 |
| | | Message ID | 2 bytes | 0x0b02 |
| | | Length | 4 bytes | 1024 |

| | | | | |
|---|---|---|---|---|
| | | Payload | 1024 bytes | base RTCM data (1077, 1087, 1019, …) |
| | | Checksum | 2 bytes | crc16 check sum |
| Checksum bytes | 4 bytes | Data frame checksum | | |

#### 3.4.5.2 Vehicle speed data

In addition, another user input of the INS402 is defined as below, to enhance the dead reckoning (DR) performance of the INS solution. The accurate vehicle speed from the vehicle reference point provides accurate speed constraint to the along-track direction. This speed signal is decoded from CAN odometer messages, and usually is the average of the two rear wheels odometer speed. In such case, the vehicle reference point (VRP) is the middle point of the two rear wheels in the vehicle frame.

**Table 17 Vehicle Speed Data**

| **Message** | **Vehicle speed from vehicle reference point, provided by vehicle manufacturer*** | | | |
|---|---|---|---|---|
| Message ID | 0x0b01 | | | |
| Length | 4 | | | |
| Payload Description: | | | | |
| **Byte Offset** | **Type** | **Name** | **Unit** | **Description** |
| 0 | float | speed | km/h | Positive if forward driving,<br>negative if backward moving,<br>zero if stationary. |

*Odometer speed resolution: 0.1 m/s or equivalent to odometer accuracy achievable

## ☑ EXAMPLE

Example of how to Send speed value to INS402 with 100base-T1 protocol:

Example:

**Table 18 Example of Speed Input Packet**

| **Speed values** | **Packets** |
|---|---|
| 39.5km/h | **01 02 03 04 05 06** 06 05 04 03 02 01 00 0e 55 55 01 0b 04 00 00 00 00 00 1e 42 a8 90 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |
| -11.5km/h | **01 02 03 04 05 06** 06 05 04 03 02 01 00 0e 55 55 01 0b 04 00 00 00 00 00 38 c1 a5 3b 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |

*DST_MAC = '01:02:03:04:05:06'

SRC_MAC = '06:05:04:03:02:01'

### 3.4.6 User Commands

Several user commands are defined on the ethernet interface for the user system (e.g., a vehicle) to poll information from INS402 and to set user parameters in INS402. The user request commands and the INS402 response message are defined in pairs, where the request command is a two-byte hex value that is also the ID of the response message.

#### 3.4.6.1 Table 19 Get the Product Information

| **Description** | **Get the product device information, as per user poll** |
|---|---|
| **Request Command:** | |
| Command | 0xcc01 |
| Length | 0 |
| Payload | N/A |
| **Response Message:** | |
| Message ID | 0xcc01 |
| Length | N/A |
| Payload | ASCII message (separated by space), e.g.<br>INS402 5020-4007-01 2379600604 Hardware v2.0 IMU_SN 2179100387 RTK_INS App v31.00.03 Bootloader v01.02 IMU330NL FW v27.01.01 MOSAIC FW v4.8.0 |

*By response message, you will get destination address (DST_MAC) and product information. It means the connection between DST_MAC and source address (SRC_MAC) is built.

#### 3.4.6.2 Table 20 Get User Configuration

| **Description** | **Get user configuration on the parameters (defined in Table 23)** | | | |
|---|---|---|---|---|
| **Request Command:** | | | | |
| Command | 0xcc02 | | | |
| Length | 4 | | | |
| Payload | | | | |
| **offset** | **variable** | **name** | **unit** | **desc** |
| 0 | uint32 | Sequence ID | | refer to Table 23 |
| **Response Message:** | | | | |
| Message ID | 0xcc02 | | | |
| Length | 8 | | | |
| Payload | | | | |
| **offset** | **variable** | **name** | **unit** | **desc** |
| 0 | uint32 | Sequence ID | | refer to Table 23 |

| | | | | |
|---|---|---|---|---|
| | | each defined in Table 23 | | |
| 4 | | Parameter value | | |

### 3.4.6.3 Table 21 Set User Configuration

| **Description** | **Set user configuration on the parameters (defined in Table 23)** | | | |
|---|---|---|---|---|
| **Request Command:** | | | | |
| Command | 0xcc03 | | | |
| Length | 8 | | | |
| Payload | | | | |
| **offset** | **variable** | **name** | **unit** | **desc** |
| 0 | uint32 | Sequence ID | | refer to Table 23 |
| 4 | | Parameter value | | |
| **Response Message:** | | | | |
| Message ID | 0xcc03 | | | |
| Length | 4 | | | |
| Payload | | | | |
| **offset** | **variable** | **name** | **unit** | **desc** |
| 0 | int32 | result | | -0: success<br>-1: invalid parameter number<br>-2: invalid parameter value |

### 3.4.6.4 Table 22 Save User Configuration

| **Description** | **Save user configuration parameters (defined in Table 23) permanently in device flash** | | | |
|---|---|---|---|---|
| **Request Command:** | | | | |
| Command | 0xcc04 | | | |
| Length | 0 | | | |
| Payload | N/A | | | |
| **Response Message:** | | | | |
| Message ID | 0xcc04 | | | |
| Length | 4 | | | |
| Payload | | | | |
| **offset** | **variable** | **name** | **unit** | **desc** |
| 0 | int32 | result | | 0: success<br>-1: fail |

#### 3.4.6.5 Table 23 User Configuration Parameters

For one parameter, same ID number used with CAN interface, pls refer to Index/Configuration/unit columns in Table 3 Index of Configurations.

#### 3.4.6.6 Table 24 User sends system reset command

**Table 24**

| **Description** | **User sends system reset command** |
|---|---|
| **Request Command:** | |
| Command | 0xcc06 |
| Length | 0 |
| Payload | N/A |
| **Response Message:** | |
| Message ID | 0xcc06 |
| Length | N/A |
| Payload | N/A |

# Appendix A: 16-bit CRC Implementation Sample Code

The following is the 16-bit CRC sample code used in most of the code development.

```
uint16_t CalculateCRC (uint8_t *buf, uint16_t length)
{
	uint16_t crc = 0x1D0F;

	for (int i=0; i < length; i++) {
		crc ^= buf[i] << 8;
		for (int j=0; j<8; j++) {
			if (crc & 0x8000) {
				crc = (crc << 1) ^ 0x1021;
			}
			else {
				crc = crc << 1;
			}
		}
	}
	return ((crc << 8) & 0xFF00) | ((crc >> 8) & 0xFF);
}
```