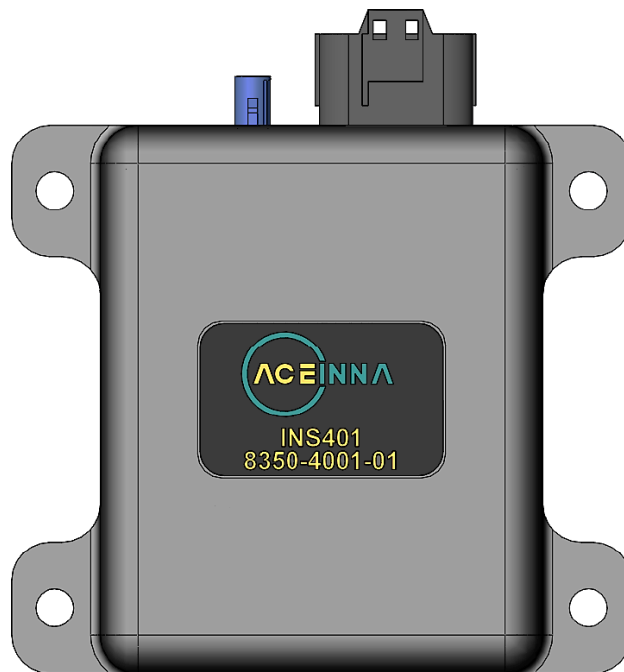




# INS401 USER MANUAL

Document Part Number: 7430-4006-02\_B



ACEINNA, Inc.

email: [info@aceinna.com](mailto:info@aceinna.com), website: [www.aceinna.com](http://www.aceinna.com)



Date	Document Revision	Firmware Applicability	Description	Author
Aug 19, 2021	-D1	28.00.01	Preliminary Release	JN/DW/BL/CL
Oct 01, 2021	-D2	28.00.02 (maybe 28.01.00)	Update figures to the final product look and update content to 28.00.02	CT/KL/DW/CL
Nov 05, 2021	-D3	28.01	Update contents to reflect FW 28.01	DW
Dec 29, 2021	-D4	28.02	Mounting instructions updated to reflect FW 28.02 Add more contents for Firmware Upgrade	AB
Apr 11, 2022	-D5	28.04	Various updates See Appendix D for full revision details	AB
Dec 12, 2022	-D6	28.05/28.06	Various updates See Appendix D for full revision details	JT/CEK/XG/JN
Oct 8, 2023	-D7	28.05/28.06	Various updates See Appendix D for full revision details	JT/JN
Jun 05 2024	-D8	28.05/28.06	Various updates See Appendix D for full revision details	JT/CEK

## **⚠ WARNING**

Aceinna has developed this product exclusively for commercial applications. It has not been tested for, and Aceinna makes no representation or warranty as to conformance with, any military specifications or that the product is appropriate for any military application or end-use. Additionally, any use of this product for nuclear, chemical, biological weapons, or weapons research, or for any use in missiles, rockets, and/or UAV's of 300km or greater range, or any other activity prohibited by the Export Administration Regulations, is expressly prohibited without the written consent of Aceinna and without obtaining appropriate, US export license(s) when required by US law. Diversion contrary to U.S. law is prohibited.



Aceinna is a registered trademark of Aceinna Inc. Other product and trade names are trademarks or registered trademarks of their respective holders.

**Table of Contents**

Table of Contents ..... 1

1 Introduction..... 1

    1.1 INS401 Overview..... 1

2 INS401 Hardware ..... 2

    2.1 Housing and Dimensions..... 2

    2.2 Mechanical Interface ..... 3

        2.2.1 RF Connector..... 3

        2.2.2 Main Connector and Pin Description..... 3

3 INS401 Installation ..... 5

    3.1 Mounting Instructions ..... 5

    3.2 Antenna Selection and Connection ..... 6

    3.3 Power Requirement ..... 7

4 INS401 Operation ..... 8

    4.1 Setting Up Communications ..... 8

    4.2 System Operation and Data Logging ..... 8

    4.3 RTK Corrections ..... 12

    4.4 Odometer ..... 12

    4.5 Vehicle installation code ..... 12

5 Diagnostic Capabilities ..... 13

6 Ethernet Port and Messages ..... 14

    6.1 Ethernet Data Frame Definition ..... 14

    6.2 Aceinna Binary Packet Format..... 14

    6.3 Output Binary Packets..... 15

        6.3.1 GNSS Solution Packet..... 15

        6.3.2 INS navigation solution ..... 16

        6.3.3 Diagnostic Message ..... 17

        6.3.4 Raw IMU Data..... 21

        6.3.5 RTCM Data ..... 21

    6.4 Output ASCII Messages ..... 22

    6.5 Input Binary Packets ..... 23

        6.5.1 RTCM correction data ..... 23

6.5.2 Vehicle speed data .....	24
6.6 User Commands .....	25
6.6.1 Table 19 Get the Product Information .....	25
6.6.2 Table 20 Get User Configuration.....	26
6.6.3 Table 21 Set User Configuration .....	27
6.6.4 Table 22 Save User Configuration.....	27
6.6.5 Table 23 User Configuration Parameters.....	28
6.6.6 Table 24 User sends system reset command .....	29
6.6.7 Table 25 User gets ins save buffer of fixed position of ins power on.....	29
6.6.8 Table 26 Get prestored data of vehicle projects in flash.....	29
6.6.9 Table 27 Get Customer Part Number .....	31
7 Firmware Upgrade .....	32
7.1 Definitions.....	32
7.2 FW Upgrade with AceNav .....	32
7.3 Bootloader Protocol.....	32
7.3.1 Structure of Firmware Bin.....	33
7.3.2 Upgrade RTK/INS Parts.....	34
7.3.3 Upgrade GNSS-Chip Part.....	36
7.3.4 Upgrade IMU Part .....	40
7.4 Statistics of Upgrade Steps.....	41
7.4.1 Steps of RTK/INS Upgradation.....	41
7.4.2 Steps of GNSS-Chip Upgradation .....	41
7.4.3 Steps of IMU Upgradation.....	42
Appendix A: 16-bit CRC Implementation Sample Code.....	43
Appendix B: AceNav CLI Software Usage .....	44
B.1 System requirement .....	44
B.2 GNSS/INS operation user settings.....	44
B.3 Commands .....	44
Appendix C: Firmware 28.01 and Earlier IMU Axis Definition .....	46
Appendix D: Detailed Revision History .....	47
Appendix E: Functions Implementation in GNSS-chip upgradation.....	48

## **About this Manual**

The following annotations have been used to provide additional information.

### **◀ NOTE**

Note provides additional information about the topic.

### **☑ EXAMPLE**

Examples are given throughout the manual to help the reader understand the terminology.

### **🚩 IMPORTANT**

This symbol defines items that have significant meaning to the user

### **⚠ WARNING**

The user should pay particular attention to this symbol. It means there is a chance that physical harm could happen to either the person or the equipment.

# 1 Introduction

## 1.1 INS401 Overview

The Aceinna INS401 is an Inertial Navigation System (INS) with RTK-enabled GNSS positioning engine and triple-redundant IMU sensors. It is designed to work with odometer and RTK corrections for optimal performance. It is a lightweight and compact enclosure that consists of only two interfaces – the RF interface, and the power and data communication interface. It is a single-box solution developed for easy evaluation and flexible deployment for various applications, including automotive applications and ADAS systems.

The Aceinna INS401 supports multiple constellations including GPS, GLONASS, Galileo, Beidou and QZSS. It tracks the following satellite signals as shown in Table 1.

Table 1 INS401 Frequency Plan

Constellation	Satellite Signals
GPS	L1 C/A + L2C
GLONASS	G1
BeiDou	B1I + B2I
Galileo	E1 + E5b
QZSS	L1C/A + L2C

The INS401 provides the following features:

1. Position, velocity, heading and attitude solutions at 100Hz
2. Multi-constellation and multi-frequency RTK algorithm support for centimeter accurate positioning
3. Integrated and calibrated triple-redundant MEMS IMU with range of  $\pm 8$  g and 200 °/s
4. Automotive ethernet interface
5. 1 PPS output
6. Includes functionality for saving navigation data prior to shutdown to allow faster initialization

## 2 INS401 Hardware

As described in the overview, the INS401 is a complete Inertial Navigation System contained in a ruggedized aluminum housing, qualified for use in industrial and automotive applications. It contains a dual-band RTK-capable GNSS receiver, a triple redundant IMU which is fully calibrated for bias, scale factor, linearity, and misalignment and an ethernet interface for external communication. The INS401 is powered directly from 12 V or 24 V battery systems (9 V ~ 32 V input range) and is designed and qualified to withstand the worst-case conditions in terms of overvoltage, reverse voltage and other fault conditions experienced in these vehicles.

### 2.1 Housing and Dimensions

The INS401 housing is cast aluminum for ruggedness and low manufacturing cost as shown in Figure 1. There are two connectors on the front panel, and they are permanently installed into the INS401 housing:

1. The small circular connector is the FAKRA-J C type connector for RF connection to the GNSS antenna
2. The large connector is the main electrical connector. It is an automotive grade MX23A connector that contains the power, communication interfaces to connect with the vehicle electronic control unit (ECU)



Figure 1 INS401 Housing

The overall dimensions are 130 x 115 x 34.5 mm, as shown in the 2D drawings in Figure 2.



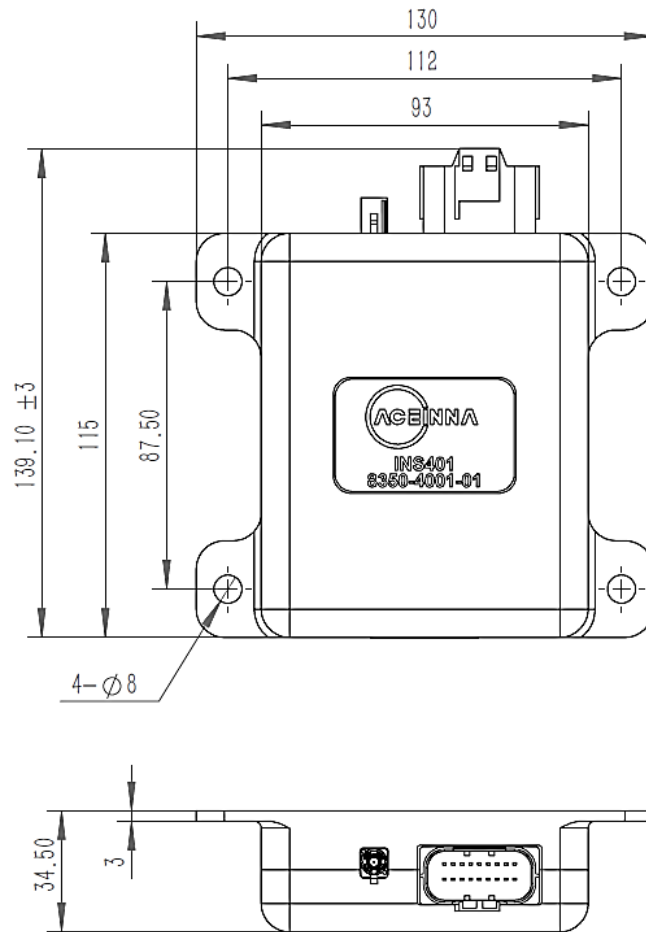


Figure 2 INS401 Hardware Dimensions

## 2.2 Mechanical Interface

### 2.2.1 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 INS401 receiver and delivers 5 V DC from the INS401 to the external active antenna.

If building your own RF cable please use Fakra Part No. 734036262.

### 2.2.2 Main Connector and Pin Description

The main connector carries all the other power and I/O signals to and from the INS401 module. This connector is also of automotive grade and is manufactured by JAE Electronics. The male end which is installed in the INS401 housing has part number MX23A18NF1; the female end, which is attached to the external wiring harness, has part number MX23A18SF1. Figure 3 illustrates the location of the 18 pins in the male part, as seen facing the connector from outside the module.

If building your own wiring harness please use part number MX23A18SF1.

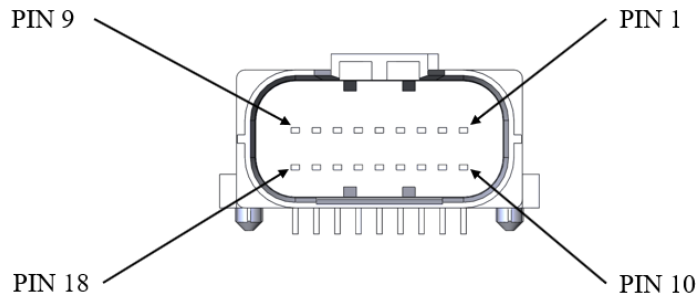


Figure 3 Pin Diagram of the Male End

Table 2 shows the functional description of the 18 pins in the main connector.

Table 2. Pin Description of the Main Connector

Pin Number	Type	Pin Name	Pin Function
1	Reserved	N/A	Reserved
2	Reserved	N/A	Reserved
3	I/O	ETH_TRX_N	Ethernet (negative)
4	I/O	ETH_TRX_P	Ethernet (positive)
5	Reserved	N/A	Reserved
6	Reserved	N/A	Reserved
7	Reserved	N/A	Reserved
8	Reserved	N/A	Reserved
9	Power	VCC_IN	9V ~ 32V DC power input
10	Reserved	N/A	Reserved
11	Reserved	N/A	Reserved
12	Reserved	N/A	Reserved
13	Reserved	N/A	Reserved
14	Power	GND	Negative power supply input
15	Power	GND	Negative power supply input
16	O	PPS	1 Pulse per Second output, synchronized to GNSS
17	Power	GND	Negative power supply input
18	Power	GND	Negative power supply input

## 3 INS401 Installation

### 3.1 Mounting Instructions

Use four bolts of 1/4-20 UNC socket head cap screw (ASME B18.3) to fix the INS401 system on a flat rigid panel on the vehicle, using the mechanical dimension measures shown in Figure 2. The IMU navigation center and the IMU body frame default coordinate definition is shown in Figure 4. Align the INS401 system X-axis with the forward driving direction of the vehicle.

By default, the IMU body frame orientation of INS401 is defined as in the figure below, 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. (Note in the firmware versions 28.01 and earlier, the default IMU body frame orientation was different with the X-axis pointing to the same direction as the connectors, Z-axis pointing down, and Y-axis completing a right-hand coordinate system). To align with the vehicle frame definition, the INS401 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.

The “Set User Configuration” command (see Table 21) should be used to re-align the coordinate system so that X is facing forward, Y is right, and Z is down if the INS401 is mounted in a different orientation.

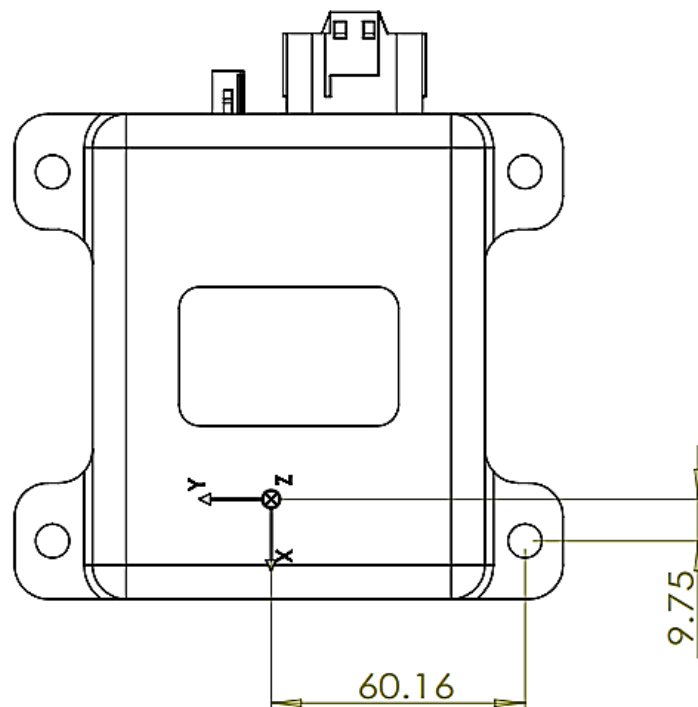


Figure 4 IMU Axis Definition and Navigation Center Location

After the INS401 is mounted to the vehicle, measure the IMU to the GNSS antenna lever arm (translational offset) from the IMU navigation center to the GNSS antenna phase center. The GNSS antenna is typically installed on top of the vehicle roof. For optimal performance, it is

required to have the lever arm accuracy of less than 2 cm. For instance, a lever arm measurement is shown in Figure 5. The translation offset is measured as 1 m in each direction of x, y, z. The IMU to the GNSS antenna lever arm is  $[x, y, z] = [1.0, -1.0, -1.0]$  m. The “Set User Configuration” command in Table 21 should be used to configure the INS401 with the correct lever arm.



Figure 5 INS401 IMU to GNSS Antenna Lever Arm Definition and Measurement Demonstration

### 3.2 Antenna Selection and Connection

The INS401 works with a customized external wiring harness to connect to the antenna connector and the main connector.

1. Connect antenna cable to FAKRA-J C type RF terminal
2. Connect power supply, Ethernet, PPS signal to main connector MX23A18NF1
3. The INS401 will supply power (+5Vdc) to the antenna via the antenna cable

Based on your application, select an GNSS L1&L2&L5 active antenna. A typical antenna LNA gain between 30 dB and 40 dB is recommended.

When installing the GNSS antenna:

1. Choose an antenna location with a clear view of the sky so each satellite above the horizon can be tracked without obstruction
2. Mount the antenna on a secure, stable structure capable of safe operation in the specific environment

- 
3. Ensure antenna is mounted to a rigid surface to minimize any movement due to vehicle dynamics.

Table 3 Recommended GNSS Antenna Specification

Item	SPEC
GNSS Frequency	L1 + L2 +L5
Impedance	50Ω
Polarization	RHCP
Axial Ratio	3dB (max)
Gain @ Zenith (90°)	3dBi (min)
LNA Gain	30dB - 40dB
Noise Figure	2dB (max)
VSWR	2 (max)
Operation Voltage	DC 3.3V ~ DC 5V

### 3.3 Power Requirement

Operating voltage: 12V DC Nominal (9V-32V DC is acceptable range)

Current requirement: The INS401 consumes 4 W (typical value). It's recommended to make sure the supply is capable of at least 600mA from 12V.

## 4 INS401 Operation

The following equipment and items are needed for a testing setup of the INS401 to demo its operation:

1. Active multi-constellation multi-band GNSS antenna
2. 12 V power adapter (battery or other power module in range 9V-32V)
3. 100Base-TX (RJ-45 connector) to 100Base-T1 converter
4. Testing PC (Windows 10 or Ubuntu 18.04) with internet access
5. External harness with MX23A automotive connector and cables
6. GNSS RTK correction service account (NTRIP protocol settings)

### 4.1 Setting Up Communications

A typical configuration of the INS401 is shown in Figure 6. The circular RF connector connects with an active GNSS antenna, and the main MX23A connector contains the connection with the external power, the ethernet data cable and the other communication port cable if needed. The primary data port is the automotive grade ethernet port to connect to a vehicle ECU directly or to a testing computer via a converter. The connection to a vehicle ECU is described in the next subchapter for system integration. Figure 6 shows a test setup only with a testing PC, which requires a INS401 unit and the 100Base-TX (with RJ-45 connector) to 100Base-T1 converter, to connect an automotive-grade ethernet device with an industrial ethernet device typically seen on a computer.

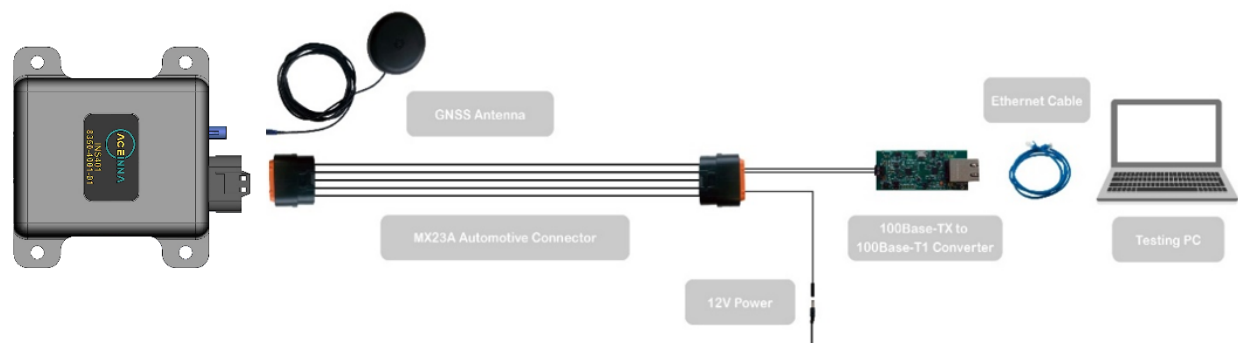


Figure 6 Typical INS401 Configuration

#### NOTE

INS401 does not support power over ethernet.

When ordering the INS401, only the physical module with its integrated connectors is included. The other components shown in Figure 6 (antenna, cables, mating connectors etc.) must be purchased separately.

### 4.2 System Operation and Data Logging

On the testing computer, only the “AceNav” Command Line Interface (CLI) software is needed for the INS401 system to work and to log all the data output from the INS401 system. This CLI

command line software is open sourced on Aceinna/acenav-cli (github.com). It is recommended for the user to click here to download the executables of the CLI software. Follow the steps below to operate the system:

1. Connect with GNSS antenna
2. Connect the main connector with a MX23A female connector (associate cables to the pins defined in Table 2)
3. Use an ethernet converter to connect with a testing PC via the RJ-45 jack
4. Power on the system
5. Go to the “AceNav” CLI folder, modify the “ins401.json” JSON file inside the “settings/INS401” subfolder, input the correct NTRIP account information as shown below:

```
"ntrip":[
  {
    "name": "ip",
    "value": "58.215.20.43"
  },
  {
    "name": "port",
    "value": 2201
  },
  {
    "name": "mountPoint",
    "value": "WX02"
  },
  {
    "name": "username",
    "value": "Aceinna"
  },
  {
    "name": "password",
    "value": "Aceinna888888"
  }
]
```

6. Input the lever arm values in the field of “gnss lever arm x”, “gnss lever arm y” and “gnss lever arm z” (if installed in a vehicle). Note the lever arm is in meters

```
"userParameters": [
  {
    "paramId": 1,
    "name": "gnss lever arm x",
    "value": 1.19
  },
  {
    "paramId": 2,
    "name": "gnss lever arm y",
    "value": -0.24
  },
  {
    "paramId": 3,
    "name": "gnss lever arm z",
    "value": -0.9
  },
],
```

7. After running the data logging command below, the information about INS401, the PC connection, and the NTRIP connection status will be displayed on the console. The NTRIP status shows “ok” of both [connect] and [request], then the system has connected to a valid NTRIP server for the GNSS RTK operation

```
PS C:\Aceinna\bin\acenav> .\acenav.exe
[Info] Aceinna Navigation CLI, version 2.6.2
[NetworkCard] Ethernet 2 MAC: 00:e0:4c:78:7c:fc
# Connected INS401 with ethernet #
Device: INS401 5020-4007-01 2179000023 Hardware v1.0
Firmware: RTK_INS App v28.01 Bootloader v01.01 IMU330ZA FW v27.00.07 STA9100 FW v5.8.12
NTRIP:[connect] 58.215.20.43:2201 start...
NTRIP:[connect] ok
NTRIP:[request] ok
```





8. Check the output bin files. Every session the “AceNav” CLI has started, a data log subfolder with the time tag in the folder name is created under “acenav/data”. The data log subfolder for each data logging session contains four files:

configuration.json: INS401 configuration file read from the device flash

- 1) rtm\_base\_<time\_tag>.bin: GNSS RTK correction data in RTCM format
- 2) rtm\_rover\_<time\_tag>.bin: INS401 GNSS raw data in RTCM format
- 3) user\_<time\_tag>.bin: high precision positioning solution data, including GNSS solution, INS solution, and other related variance/status information



(C:) > Aceinna > bin > acenav > data > ins401\_log\_20210726\_175001

Name	Date modified	Type
 configuration.json	8/19/2021 7:34 PM	JSON File
 rtc_base_2021_07_26_17_50_01.bin	7/26/2021 5:59 PM	BIN File
 rtc_rover_2021_07_26_17_50_01.bin	7/26/2021 5:50 PM	BIN File
 user_2021_07_26_17_50_01.bin	7/26/2021 6:00 PM	BIN File









### 9. Run the data parsing command

```
PS C:\Aceinna\bin\acenav> .\acenav.exe parse -t ins401 -p .\data\ins401_log_20210726_175001\
[Info] Aceinna Navigation CLI, version 2.6.0
Process : 6.3 %
```

### 10. Check the decoded files for results representation and analysis

- 1) user\_<time\_tag>\_dm.csv: INS401 system diagnostic messages
- 2) user\_<time\_tag>\_gnss.csv: GNSS solution at 1 Hz
- 3) user\_<time\_tag>\_imu.csv: raw IMU data at 100 Hz
- 4) user\_<time\_tag>\_ins.csv: INS solution at 100 Hz
- 5) user\_<time\_tag>\_gnss.kml: GNSS solution trajectory in kml format for Google Earth presentation
- 6) user\_<time\_tag>\_ins.kml: INS solution trajectory in kml format for Google Earth presentation
- 7) user\_<time\_tag>\_nmea.txt: GNSS solution in NMEA 0183 ASCII strings
- 8) user\_<time\_tag>\_odo.txt: vehicle odometer speed data received by INS401

(C:) > Aceinna > bin > acenav > data > user\_2021\_07\_26\_17\_50\_01\_p

Name	Date modified	Type
 user_2021_07_26_17_50_01_dm.csv	7/26/2021 6:09 PM	CSV File
 user_2021_07_26_17_50_01_gnss.csv	7/26/2021 6:09 PM	CSV File
 user_2021_07_26_17_50_01_imu.csv	7/26/2021 6:09 PM	CSV File
 user_2021_07_26_17_50_01_ins.csv	7/26/2021 6:09 PM	CSV File
 user_2021_07_26_17_50_01-gnss.kml	7/26/2021 6:06 PM	KML File
 user_2021_07_26_17_50_01-ins.kml	7/26/2021 6:09 PM	KML File
 user_2021_07_26_17_50_01-nmea.txt	7/26/2021 6:09 PM	TXT File
 user_2021_07_26_17_50_01-odo.txt	7/26/2021 6:06 PM	TXT File

Refer to AceNav CLI Software Usage for a detailed description and the usage of the “AceNav” CLI.

---

### 4.3 RTK Corrections

As stated in the overview, 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. Refer to chapter 6 Ethernet Port and Messages section 6.5 Input Binary Packets for more information about RTCM messages and Table 15 for base RTCM data packet.

### 4.4 Odometer

As stated in the overview, for optimal performance, it is necessary to provide the odometer input to the INS401. Refer to chapter 6 Ethernet Port and Messages section 6.5 Input Binary Packets for more information about the vehicle reference point and Table 17 for vehicle speed data packet.

For optimal performance, it is recommended for the odometer to meet the following requirements:

1. Sampling period of  $10 \pm 3$  ms
2. Error standard deviation smaller than 0.1 m/s
3. Normalized error autocorrelation at 0.1 s below 0.75 (mean removed before autocorrelation)
4. Normalized error autocorrelation at 1 s below 0.70 (mean removed before autocorrelation)
5. Error probability density function must be symmetrical (small deviations accepted)
6. Maximum delay of 0.1 s
7. The vehicular speed must report 0 m/s when the vehicle is not moving
8. The vehicular speed must not report 0 m/s when the vehicle is moving
9. The vehicular speed must be valid for the entire range of operation of the vehicle, including negative speeds

### 4.5 Vehicle installation code

The INS401 has the capability to store, at the time of manufacturing, lever arm and orientation settings specific to a certain vehicle. The purpose of this is to enable configuration after installation by a single user command. This is useful in case the INS401 is used by multiple vehicles (enables procurement of a single part number, and final configuration in the vehicle manufacturing line). Contact the factory for more information about this option.



---

## 5 Diagnostic Capabilities

Please refer to Table 6 for details on the diagnostic messages.

## 6 Ethernet Port and Messages

The INS401 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 6.1. 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

### 6.1 Ethernet Data Frame Definition

The INS401 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 INS401 LSB for input message to INS401.
- 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

### 6.2 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.

## 6.3 Output Binary Packets

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

### 6.3.1 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:
2	uint32	gps_millisecs	ms	GPS week and seconds in week
6	uint8	position_type		0: INVALID 1: Single-point positioning (SPP) 2: Real time differential GNSS (RTD) 3: INS_PROPOGATED 4: Real time kinematic (RTK), ambiguity fixed (RTK_FIXED) 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

### 6.3.2 INS navigation solution

After following items are satisfied, then INS algorithm will be initialized:

- With GNSS solution
- There is no significant steering action of the vehicle (Z-gyro rate  $< \pm 5$  dps)
- Speed  $> 10$ km/h
- More than 8s(SPP), 3s is needed if RTK solution is available.

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: GPS week and seconds in week
2	uint32	gps_milliseecs	ms	
6	uint8	ins_status		0: INVALID 1: INS_ALIGNING 2: INS_HIGH_VARIANCE 3: INS_SOLUTION_GOOD 4: INS_SOLUTION_FREE 5: INS_ALIGNMENT_COMPLETE
7	uint8	ins_position_type		0: INVALID 1: SPP/INS 2: RTD/INS 3: INS_PROPAGATE 4: RTK_FIXED/INS 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: ID_NONE = -2, ID_ERROR = -1, ID_UNKNOWN = 0, ID_ASIA = 1, ID_EUROPE = 2, ID_OCEANIA = 3, ID_AFRICA = 4, ID_NORTHAMERICA = 5, ID_SOUTHAMERICA = 6, ID_ANTARCTICA = 7

### 6.3.3 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: 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	°C	Temperature of the IMU
14	float32	MCU temperature	°C	Temperature of the MCU
18	float32	GNSS-chip temperature	°C	Temperature of the GNSS-CHIP chipset (To be implemented in FW28.05)

Table 7 Typedef struct--Device status bit field

	Bit	Description	Value
IMU	0	Master fail	0: normal 1: fatal error occurred
	1	HW error	0: normal 1: hardware exception detected
	2	SW error	0: normal 1: software exception detected
	3	Config error	0: normal 1: config error detected by periodic self-test
	4	Calibration error	0: normal 1: calibration data corrupted
	5	Accel degradation	0: normal 1: accel data degradation due to sensor exception
	6	Gyro degradation	0: normal 1: gyro data degradation due to sensor exception
	7	Forced restart	0: normal 1: forced restart
	8	Application CRC Error	0: normal 1: CRC error detected Will be sent from Bootloader periodically with interval of 1 second
	9	Tx overflow error	0: normal





			1: Tx Overflow occurred 10 consecutive cycles
GNSS	10	PPS status	0: normal 1: 1PPS pulse exception
	11	GNSS data status	0: normal 1: GNSS-chipset has NO data output
	12	GNSS signal status	0: normal 1: GNSS-chipset has data output, but no valid signal detected
Operation	13	Power	0: normal 1: any component has no power
	14	MCU status	0: normal 1: MCU failure MCU and peripherals HW self-test status when power on
	15	Temperature under MCU flag	0: normal 1: under temperature
	16	Temperature under STA flag	0: normal 1: under temperature
	17	Temperature under IMU flag	0: normal 1: under temperature
	18	Temperature over MCU flag	0: normal 1: under temperature
	19	Temperature over STA flag	0: normal 1: under temperature
	20	Temperature over IMU flag	0: normal 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).

### 6.3.3.1 GNSS-chipset Diagnostic

Table 8 GNSS-CHIP Diagnostic Packet

<b>Message</b>	<b>GNSS-CHIP diagnostic data, periodic output at 1 Hz</b>
Message ID	0x6664
Length	5
Payload Description:	

Byte Offset	Type	Name	Unit	Description
0	uint8	gnss_err_out_pin		0: EOUT is normal; 1: EOUT error.
1	uint8	gnss_rf_err_pin		0: RF_ERR is normal; 1: RF_ERR error.
2	uint8	gnss_ant_err_pin		0: ANT_ERR is normal; 1: ANT_ERR error.
3	uint8	gnss_handshake_flag		0: Handshake between MCU and GNSS-CHIP fail; 1: Handshake between MCU and GNSS-CHIP OK.
4	uint8	gnss_reset_pin		0: GNSS-CHIP reset; 1: GNSS-CHIP works normally.

### 6.3.3.2 DM-Ext1

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 voltage monitor	0: normal 1: wave of voltage out of range Bit0: ADC_IN Bit1: ADC_CORE Bit2: ADC_1V2 Bit3: ADC_GND
	4	ExtOsc external oscillator	0: normal 1: oscillator exception detected
	5	Power chip	0: normal 1: power exception detected



6:8	RAM	0: normal 1: ram error detected by MCU Bit0: SRAM0 Bit1: SRAM1 Bit2: SRAM2
9	Watch-dog	0: normal 1: watch-dog error detected
10:11	Flash	0: normal 1: flash error detected Bit0: cFlash, Bit1: wFlash
12	clkCSV Core oscillator	0: normal 1: core oscillator error detected by MCU
13:31	19bits Reserved	

### 6.3.4 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 <sup>2</sup>	accel x axis measurement
10	float	accel_y	m/s <sup>2</sup>	accel y axis measurement
14	float	accel_z	m/s <sup>2</sup>	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

### 6.3.5 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

<b>Message</b>	<b>RTCM data, periodic output at 10 Hz</b>
Message ID	0x0a06
Length	1~1024bytes
Payload	

INS401 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.

**6.4 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: 0: GNSS position Fix not valid 1: GNSS position fix (SPP) 2: Differential GNSS position fix (RTD)

	4: Real-Time Kinematic, fixed integers 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 *

## 6.5 Input Binary Packets

### 6.5.1 RTCM correction data

In order to perform GNSS RTK, the INS401 device needs RTK correction data (RTCM messages) input from user device (vehicle). And the user device must send the RTCM messages over ethernet to the ethernet port of the INS401. 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.

INS401 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	INS401 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		

**6.5.2 Vehicle speed data**

In addition, another user input of the INS401 is defined as below, to enhance the dead reckoning (DR) performance of the IMU based INS solution. The accurate vehicle speed from the vehicle reference point provides accurate speed constraint to the along-track direction. This speed is decoded from CAN odometer messages, and usually is the average of the two rear wheels odometer





<b>Request Command:</b>	
Command	0xcc01
Length	0
Payload	N/A
<b>Response Message:</b>	
Message ID	0xcc01
Length	N/A
Payload	ASCII message (separated by space), e.g. INS401 5020-4007-01 21790xxxxx Hardware v1.0 IMU_SN 21791xxxxx (Product) (P/N) (S/N) (HW Version) (IMU S/N) RTK_INS App v28.xx.xx Bootloader v01.xx.xx (FW Version) (Bootloader Version) IMU330NL FW v27.xx.xx GNSS-CHIP FW v5.XX.XX (IMU FW version) (GNSS-CHIP FW version)

\*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.

**☑ EXAMPLE**

Example in FW28.03.11: INS401 5020-4007-01 2109000083 Hardware v1.0 IMU\_SN 2179100041 RTK\_INS App v28.03.11 Bootloader v01.01 IMU330NL FW v27.01.01 GNSS-CHIP FW v5.8.12 UUID 12312421432

**6.6.2 Table 20 Get User Configuration**

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



		each defined in Table 23		
4		Parameter value		

### 6.6.3 Table 21 Set User Configuration

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

\*When new vehicle code is set to ID14 (one vehicle code prestored in flash before), INS algorithm will use prestored lever arms and orientation. For example, when vehicle code (AB35) is configured in ID-14, prestored parameters (ID1-12) of AB35 will be used in algorithm, whatever values of ID1-12 will not be used;

If you want use values of ID1-12 in INS algorithm, please confirm 0 in ID14, it is 0 by default;

### 6.6.4 Table 22 Save User Configuration

<b>Description</b>		Save user configuration parameters (defined in Table 23) permanently in device flash		
<b>Request Command:</b>				
Command	0xcc04			
Length	0			
Payload		N/A		
<b>Response Message:</b>				

Message ID	0xcc04			
Length	4			
Payload				
<b>offset</b>	<b>variable</b>	<b>name</b>	<b>unit</b>	<b>desc</b>
0	int32	result		0: success -1: fail

### 6.6.5 Table 23 User Configuration Parameters

Sequence ID	Type	Name	Unit	Description
1	float	gnssLeverArmBx	m	IMU to GNSS antenna phase center lever arm
2	float	gnssLeverArmBy	m	
3	float	gnssLeverArmBz	m	
4	float	vrpLeverArmBx	m	IMU to vehicle reference point lever arm
5	float	vrpLeverArmBy	m	
6	float	vrpLeverArmBz	m	
7	float	userLeverArmBx	m	The offset between IMU to the user point of interest
8	float	userLeverArmBy	m	
9	float	userLeverArmBz	m	
10	float	rotationRbvX	deg	Rotation angles to align IMU body frame to vehicle frame, in order Z->Y->X
11	float	rotationRbvY	deg	
12	float	rotationRbvZ	deg	
13	uint32_t	Odometer enable		Default: odometer enable
14	Char*4	ASCII format of Vehicle codes Example: AB35: (0x41, 0x42, 0x33, 0x35)		0000: default from plant, no vehicle codes now, and above current ID1-12 values are used; AB35: prestored parameters (ID1-12) of AB35 in flash will be configured and used;

**\*Logic of ID14: please refer to comments in 6.6.3**

**\*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. Refer to Table 22 Save User Configuration command.



**6.6.6 Table 24 User sends system reset command**

Table 24

<b>Description</b>	User sends system reset command
<b>Request Command:</b>	
Command	0xcc06
Length	0
Payload	N/A
<b>Response Message:</b>	
Message ID	0xcc06
Length	N/A
Payload	N/A

**6.6.7 Table 25 User gets ins save buffer of fixed position of ins power on**

Table 25

<b>Description</b>	User gets ins save buffer of fixed position of ins power on			
<b>Request Command:</b>				
Command	0x0a09			
Length	0			
Payload	N/A			
<b>Response Message:</b>				
Message ID	0x0a09			
Length	156 bytes			
Payload	Ins save buffer			
<b>offset</b>	<b>variable</b>	<b>name</b>	<b>unit</b>	<b>desc</b>
0	uint8_t	Ins save buffer [156]		Ins save buffer

**6.6.8 Table 26 Get prestored data of vehicle projects in flash**

Table 26

<b>Description</b>	Get prestored data (lever arm configuration of projects) of vehicle projects in flash, version and detailed data (ID1-12 of User Configuration Parameters in 6.6.5)
<b>Request Command:</b>	
Command	0xcc07
Length	0



Payload	N/A			
<b>Response Message:</b>				
Message ID	0xcc07			
Length	132 bytes			
Payload	Vehicle code related info struct			
offset	variable	name	unit	desc
0	uint8_t	vechicle_codes_valid	-	0-No prestored vehicle projects matrix data 1-Prestored vehicle projects matrix data
1	uint8_t	vc_table_valid	-	0- ID14 vehicle code not set 1- ID14 vehicle code set
2	uint16_t	version	-	Aceinna internal version, detailed data of prestored vehicle projects codes are different in different version
4	char*4	vcode	-	ASCII format of Vehicle codes
8	float	pri_lever_arm_x	m	ID1-12 of User Configuration Parameters in 6.6.5
12	float	pri_lever_arm_y	m	
16	float	pri_lever_arm_z	m	
20	float	vrp_lever_arm_x	m	
24	float	vrp_lever_arm_y	m	
28	float	vrp_lever_arm_z	m	
32	float	user_lever_arm_x	m	
36	float	user_lever_arm_y	m	
40	float	user_lever_arm_z	m	
44	float	rotation_rbv_x	deg	
48	float	rotation_rbv_y	deg	
52	float	rotation_rbv_z	deg	
53-131	-	reserved	-	

---

### 6.6.9 Table 27 Get Customer Part Number

Table 27

<b>Description</b>	<b>Get the product device information, defined by customer</b>
<b>Request Command:</b>	
Command	0xdf01
Length	0
Payload	N/A
<b>Response Message:</b>	
Message ID	0xdf01
Length	N/A
Payload	ASCII message (separated by space), e.g. (Product Name) (Product P/N)/(REVISION) (SW Name) (SW P/N)/(SW REVISION) (HW Name) (HW P/N) (HW REVISION)

#### **☑ EXAMPLE**

Example in FW28.07: INS\_GPS\_IMU EEP30008064/05 INS\_FW\_SW SOW30008064/02  
INS\_IMU\_HW EEP30008127/01

---

## 7 Firmware Upgrade

### 7.1 Definitions

APP: Application software program;

Write App: Write APP data to flash;

Jl: Jump from APP to bootloader;

CS: Set the core of the burning program;

WA: Abbreviation of Write app;

JA: Jump from Bootloader to APP;

JS: Application software enters SDK burning mode;

WP: Write first 5120 bytes to flash.

WS: Write firmware payload to flash.

JG: Application software returns to GNSS mode from SDK burning mode.

IMU\_JA: Jump from IMU APP to Bootloader.

IMU\_Jl: Jump from Bootloader to IMU APP.

#### ◀ NOTE

1. INS401 only supports Ethernet port upgrade;
2. INS401 only supports the upgrade of APP, ST9100 and IMU, and does not support the online upgrade of Bootloader;

### 7.2 FW Upgrade with AceNav

Refer to AceNav CLI Software Usage for AceNav operation to upgrade INS401 firmware.

### 7.3 Bootloader Protocol

Bootloader communication packets are payload in Ethernet packet frame. For the specific format of the protocol, refer to section 6.1 Ethernet Data Frame Definition and section.

As shown in Figure 7, firmware of RTK/INS/GNSS-Chip/IMU can be upgraded in INS401.

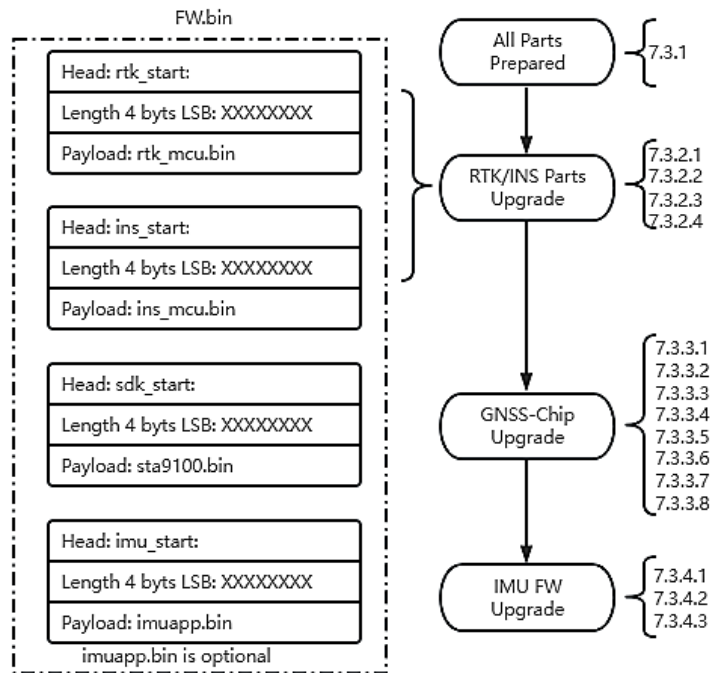


Figure 7 INS401 Upgrade Flowchart

### 7.3.1 Structure of Firmware Bin

The FW bin file can consist of four parts: the RTK part, the INS part, the GNSS-chip part and the IMU part. The starting address of each part in the FW bin file is indicated by a start string as shown in table 17. One can search for the start string of each part in the FW file (ASCII code in FW.bin) to get the corresponding part.

There can be 2 types of FW bin files.

1. The first type consists of three parts: RTK, INS, GNSS-chip
2. The second type consist of four parts: RTK, INS, GNSS-chip and IMU

Table 17 All Parts of Firmware Prepared

Table 28

Parts	Start string of each part in FW file if included
RTK part	rtk_start:
INS part	ins_start:
GNSS-chip part	sdk_start:
IMU part	imu_start:

Take RTK part as example, one can get the RTK part in the FW bin file following the steps below:

1. Search for the start string: 72 74 6B 5F 73 74 61 72 74 3A (rtk\_start:)

2. Get the length of RTK part (4 bytes following the start string, LSB): A0 A1 04 00 (303520 bytes)
3. Get the payload of RTK part: 303520 bytes after the above length bytes.

We can prepare length and payload for all parts which need to be upgraded.

### 7.3.2 Upgrade RTK/INS Parts

#### 7.3.2.1 Enter Upgradation Mode of RTK/INS

Send “JI” command to enter MCU into Bootloader mode.

Timeout: 1.5s, Retry times: 3

Send:

Header	Msg_Type	Length	CRC
0x5555	0xaa01	0	2 bytes

Receive:

Header	Msg_Type	Length	CRC
0x5555	0xaa01	0	2 bytes

#### 7.3.2.2 Upgrade RTK

Send “CS” command to initiate the upgradation of the RTK part.

Timeout: 2.1s, Retry times: 3

Send:

Header	Msg_Type	Length	Update Core 2 bytes ('C0')	Part Size 4 bytes MSB	App Info 3Bytes (‘rtk’)	CRC 2 bytes
0x5555	0xaa04	9	0x43, 0x30	Prepared in 7.3.1	0x72, 0x74, 0x6B	

Receive:

Header	Msg_Type	Length	Update Core 2 bytes ('C0')	Update Result 1bytes MSB	CRC 2bytes
0x5555	0xaa04 or 0x1515	3	0x43, 0x30	0x53 – Success 0x46 – Failed	

0x1515 is NAK when wrong bytes in Update Core.

After receiving the success response of CS command, send ‘WA’ command to write payload of RTK part which is prepared in 7.3.1. Generally, the payload needs to be divided into several packets and sent out via sequential ‘WA’ command packets. For each ‘WA’ command packet, it is recommended that 960 bytes are transferred each time, and the actual number of bytes remaining are transferred in the last packet.



Send:

Header	Msg_Type	Length	Start_Addr 4bytes MSB	Numbytes 4bytes MSB	Bytes Data	CRC 2bytes
0x5555	0xaa03	Numbytes + 8				

Receive:

Header	Msg_Type	Length	CRC 2 bytes
0x5555	0xaa03	0	

**\*Notes:** After the first frame is written, one needs to wait for the MCU to erase the data in flash (both in RTK and INS parts), the waiting time depends on the firmware size. 8s per 100kB.

### 7.3.2.3 Upgrade INS

Send “CS” command to initiate the upgradation of the INS part.

Timeout: 2.1s, Retry times: 3

Send:

Header	Msg_Type	Length	Update Core 2 bytes ('C1')	Part Size 4 bytes MSB	App Info 3Bytes (‘ins’)	CRC 2 bytes
0x5555	0xaa04	9	0x43, 0x31	Prepared in 7.3.1	0x69, 0x6E, 0x73	

Receive:

Header	Msg_Type	Length	Update Core 2 bytes ('C1')	Update Result 1bytes MSB	CRC 2bytes
0x5555	0xaa04 or 0x1515	3	0x43, 0x31	0x53 – Success 0x46 – Failed	

0x1515 is NAK when wrong bytes in Update Core.

After success feedback of ‘CS’ command, send ‘WA’ command to write payload of INS part which is prepared in 7.3.1, following the same steps as the upgradation of the RTK part.

### 7.3.2.4 Complete Upgradation

After RTK/INS upgradation is done, one needs to send ‘JA’ command to complete the upgradation.

Timeout: 1.5s, Retry times: 3

Send:

Header	Msg_Type	Length	rtkCRC_L 1 byte	rtkCRC_H 1 byte	insCRC_L 1 byte	insCRC_H 1 byte	CRC 2 bytes



0x5555	0xaa02	4						
--------	--------	---	--	--	--	--	--	--

Receive:

Header	Msg_Type	Length	Data 6bytes						CRC 2 bytes
0x5555	0xaa02	6	0x43	0x30	0x53-succ 0x46-Fail	0x43	0x31	0x53-succ 0x46-Fail	

CRC values (calculate by Appendix A:) of RTK part payload and INS part payload (prepared in 7.3.1) are needed in 'JA' command. After the MCU receives the CRC values, the MCU will calculate the CRC of received RTK part and INS part. When calculated CRC values are consistent with CRC values in 'JA' command, it will jump to the APP, otherwise it will stay in Bootloader.

### 7.3.3 Upgrade GNSS-Chip Part

#### 7.3.3.1 Enter Upgradation Mode of GNSS-Chip

Send 'JS' command to let the GNSS-chip enter bootloader mode.

Timeout: 1.5s, Retry times: 3

Send:

Header	Msg_Type	Length	CRC 2bytes
0x5555	0xaa05	0	

Receive:

Header	Msg_Type	Length	CRC 2bytes
0x5555	0xaa05	0	

**\*Notes:** This step is not the begin of GNSS-chip upgrade, it is used to set GNSS-chip to Bootloader mode through MCU

#### 7.3.3.2 Synchronization

Use 'WS' command to Send the synchronization code to synchronize with GNSS-chip.

Timeout: 0.1s, Retry times: 100

Send:

Header	Msg_Type	Length	Synchronization Code 4bytes	CRC 2bytes
0x5555	0xaa07	4	0xFD 0xC6 0x49 0x28	

Receive:

Header	Msg_Type	Length	Synchronization Code 4bytes	CRC 2bytes
0x5555	0xaa07	4	0x3A 0x54 0x2C 0xA6	

### 7.3.3.3 Send 1st 5120 bytes

The 5120 bytes (0-5119) at the beginning of the GNSS-chip part needs to be first transferred by 5 consecutive 'WP' command packets.

Send:

Header	Msg_Type	Length	Data 1024bytes	CRC 2bytes
0x5555	0xaa08	1024		

\*Notes: no feedback for the 'WP' command, just wait 100ms.

### 7.3.3.4 Confirm GNSS-chip is ready

Timeout: 1s, Retry times: 5

Send:

Header	Msg_Type	Length	Ready check code	CRC 2bytes
0x5555	0xaa07	1	0x5A	

Receive:

Header	Msg_Type	Length	Response 1byte	CRC 2bytes
0x5555	0xaa07	4	0xCC	

### 7.3.3.5 Send boot program

5 steps are needed to send boot program\_v1.0 by 'WS' command, boot program\_v1.0 can be downloaded from [link](#). Timeout after below 5 steps finished: 3.5s, Retry times: 0

Step1: Send preamble

Header	Msg_Type	Length	Preamble 16bytes	CRC 2bytes
0x5555	0xaa07	16	0xf4,0x01,0xd5,0xbc,0x73,0x40,0x98,0x83, 0x04,0x01,0xff,0x00,0x00,0x00,0x00,0x00	

Step2: Send CRC

Header	Msg_Type	Length	Boot program CRC 4bytes	CRC 2bytes
0x5555	0xaa07	4	0x03,0xD3,0x26,0xDD	

\*Notes: CRC is calculated based on boot program v1.0, function code please refer to Appendix E: Functions Implementation in GNSS-chip upgradation.

Step3: Send boot size

Header	Msg_Type	Length	Boot size 4bytes LSB	CRC 2bytes
0x5555	0xaa07	4	0xA0,0x2A,0x00,0x00 (10912 bytes)	

Step4: Send Entry message

Header	Msg_Type	Length	Entry 4bytes	CRC 2bytes
0x5555	0xaa07	4	0x00, 0x00, 0x00, 0x00	

Step5: Send Boot program

The boot program needs to be divided into three parts, the range of each part is 0-5120/5120-10240/10240-10912 bytes. The three parts are transferred by three packets.

Header	Msg_Type	Length	Each Part of Boot	CRC 2bytes
0x5555	0xaa07			

After the above five steps, the response message '0xCC' is received.

Receive:

Header	Msg_Type	Length	Response 1byte	CRC 2bytes
0x5555	0xaa07	4	0xCC	

### 7.3.3.6 Write Flash

Before writing flash, one needs to send write flash command to GNSS-Chip.

Timeout: 1s, Retry times: 3

Send:

Header	Msg_Type	Length	Write_Flash_Command 1bytes	CRC 2bytes
0x5555	0xaa07	1	0x4A	

Receive:

Header	Msg_Type	Length	Response 1byte	CRC 2bytes
0x5555	0xaa07	1	0xCC	

Then info bytes of GNSS-chip part firmware prepared in 7.3.1 is sent out. Refer to Appendix E: for the info generation function.

Timeout: 9s, Retry times: 3

Send:

Header	Msg_Type	Length	GNSS-chip part firmware info	CRC 2bytes
0x5555	0xaa07	56		

Receive:

Header	Msg_Type	Length	Response 1byte	CRC 2bytes
0x5555	0xaa07	1	0xCC	

After these two steps, one needs to wait for the device to initialize. The response '0xCC' is received when device initialization is done.

Receive:

Header	Msg_Type	Length	Response 1byte	CRC 2bytes
0x5555	0xaa07	1	0xCC	

And then the data in flash will be erased, after erase done response '0xCC' will be received.

Receive:

Header	Msg_Type	Length	Response 1byte	CRC 2bytes
0x5555	0xaa07	1	0xCC	

Then needs to write the remaining bytes (5120-end) of GNSS-chip part payload to flash, which is after the initial 5120 bytes (0-5119).

The remaining bytes are split into frames of 5120 bytes, and each frame is then split into 5 packets of 1024 bytes, and each packet is sent out by 'WS' command. After each frame is sent successfully, the GNSS-chip will respond with '0xCC'.

Timeout: 3.5s, Retry times: 0

Send:

Header	Msg_Type	Length	Firmware 1024 bytes	CRC 2bytes
0x5555	0xaa07	5120		

Receive:

Header	Msg_Type	Length	Response 1byte	CRC 2bytes
0x5555	0xaa07	1	0xCC	

### 7.3.3.7 Check flash CRC Feedback

When the GNSS-chip part is transferred, "0xCC" response will be received after CRC check is passed in GNSS-chip.

Timeout: 3s, Retry times: 0

Receive:

Header	Msg_Type	Length	Response 1byte	CRC 2bytes
0x5555	0xaa07	1	0xCC	

### 7.3.3.8 Complete Upgradation

Send 'JS' command to let the GNSS-chip get back to the GNSS mode.

Timeout: 1.5s, Retry times: 3

Send:

Header	Msg_Type	Length	CRC 2bytes
0x5555	0xaa06	0	2 bytes

Receive:

Header	Msg_Type	Length	CRC 2bytes
0x5555	0xaa06	0	2 bytes

### 7.3.4 Upgrade IMU Part

#### 7.3.4.1 Enter Upgradation Mode of IMU

Send 'IMU\_JI' command to let IMU enter the bootloader mode.

Timeout: 1.5s, Retry times: 3

Send:

Heade	Msg_Type	Length	CRC 2bytes
0x5555	0x4a49	0	

Receive:

Heade	Msg_Type	Length	CRC 2bytes
0x5555	0x4a49	0	

#### 7.3.4.2 Write IMU Firmware

Use 'IMU\_WA' command write IMU part firmware prepared in 7.3.1 to flash.

Timeout: 1.5s, Retry times: 3

Send:

Header	Msg_Type	Length 4bytes	Start_Addr 4bytes MSB	Numbytes 1byte	BinData less than 240bytes	CRC 2bytes
0x5555	0x5741	Numbytes + 5				

Receive:

Header	Msg_Type	Length 4bytes	Start_Addr 4bytes MSB	Numbytes 1byte	CRC 2bytes
0x5555	0x5741	5			

Start\_Addr is the starting address of each frame of data writing, with the high byte first.

#### EXAMPLE

The starting address of the first frame of data writing is 0, and 192 bytes are written. Then the starting address of the second frame of data should be 192.

When receiving the first frame of the write command, IMU will erase the flash first. Please wait until the erase is complete.

At the same time, ensure that Length = Numbytes + 5, and the written BinData is a multiple of 16, otherwise IMU will return the NAK message shown below.

Message NAK:

Header	Msg_Type	Length	CRC 2 bytes
0x5555	0x1515	2	

### 7.3.4.3 Complete Upgradation

After IMU upgradation is done, one needs to send 'IMU\_JA' command to return to APP mode of IMU.

Timeout: 1.5s, Retry times: 3

Send:

Header	Msg_Type	Length	CRC 2bytes
0x5555	0x4a41	0	

Receive:

Header	Msg_Type	Length	CRC 2bytes
0x5555	0x4a41	0	

## 7.4 Statistics of Upgrade Steps

### 7.4.1 Steps of RTK/INS Upgradation

Step Name	Error Code	Percentile (%)	Command (type, times)
<a href="#">Jump to bootloader</a>	Error 1	0-5	JI (0xaa01) *1
<a href="#">Core select for RTK part</a>	Error 2	5-10	CS (0xaa04) *1
<a href="#">Write flash</a>	Error 3	10-25	WA (0xaa03) *N
<a href="#">Core select for INS part</a>	Error 4	25-30	CS (0xaa04) *1
<a href="#">Write flash</a>	Error 5	30-35	WA (0xaa03) *N
<a href="#">Jump to APP</a>	Error 6	35-40	JA (0xaa02) *1

### 7.4.2 Steps of GNSS-Chip Upgradation

Step Name	Error Code	Percentile (%)	Command (type, times)
<a href="#">GNSS-Chip jump to bootloader</a>	Error 7	40-45	JS (0xaa05) *1

<a href="#">GNSS-Chip Synchronization</a>	Error 8	45-50	WS (0xaa07) *1
<a href="#">Send 1st 5120 bytes</a>	N/A	50-55	WP (0xaa08) *5
<a href="#">Confirm GNSS-chip is ready</a>	Error 9	55-60	WS (0xaa07) *1
<a href="#">Send boot program</a>	Error 10	60-65	WS (0xaa07) *N
<a href="#">Write flash command</a>	Error 11	65-70	WS (0xaa07) *1
<a href="#">Write GNSS-Chip firmware info</a>	Error 12	70-73	WS (0xaa07) *1
<a href="#">Wait device initialize</a>	Error 13	73-74	N/A
<a href="#">Wait flash erased</a>	Error 14	74-75	N/A
<a href="#">Write flash</a>	Error 15	75-80	WS (0xaa07) *N
<a href="#">Check flash CRC feedback</a>	Error 16	80-82	N/A
<a href="#">Jump to GNSS mode</a>	Error 17	82-85	JS (0xaa06) *1

#### 7.4.3 Steps of IMU Upgradation

Step Name	Error Code	Percentile (%)	Command (type, times)
<a href="#">IMU jump to bootloader</a>	Error 18	85-90	IMU_JI (0x4a49) *1
<a href="#">Write flash</a>	Error 19	90-95	IMU_WA (0x5741) *N
<a href="#">Jump to IMU APP mode</a>	Error 20	95-100	IMU_JA (0x4a41) *1



---

## 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);
}
```

---

## Appendix B: AceNav CLI Software Usage

The “AceNav” is a Python based CLI software running on user computers, to connect and interact with Aceinna GNSS/INS navigation devices for user parameter settings, GNSS RTK correction streaming from a NTRIP server, and output data logging and parsing.

### B.1 System requirement

- Ubuntu 18.04:
  - Python 3.7
  - Lib: tcpdump, <https://scapy.readthedocs.io/en/latest/installation.html#debian-ubuntu-fedora>
- Windows 10:
  - Python 3.7
  - Lib: npcap, <https://scapy.readthedocs.io/en/latest/installation.html#windows>

### B.2 GNSS/INS operation user settings

Run the CLI software and connect with the INS401 system for the first time, there is a “setting” folder generated with subfolders named by Aceinna GNSS/INS navigation devices. Each type of device has a JSON format of configuration file, all the user settings including GNSS RTK correction NTRIP account information, IMU to GNSS antenna lever arm and so on. User must modify the related configurations/settings to achieve effective GNSS/INS operation. For example, user should change the NTRIP settings (IP address, port, mount point, account, and password) to accessible GNSS correction service.

### B.3 Commands

Run the following command to log all data output from Ethernet port to binary files, and streaming GNSS RTK correction data through Ethernet port to INS401 (e.g., on Ubuntu)

```
./acenav -i 100base-t1
```

A “data” subfolder will be created for the first time, and every session of data logging will be stored in a subfolder inside the “data” folder.

Run the following command to parse the logged data into text or csv files,

```
./acenav parse -t ins401 -p <path to data folder/session data subfolder>
```

If the user changes the GNSS/INS user settings in the “ins401.json” file, to make it effective, run the data logging command with “-s” option as below, and the changed user settings will be saved to flash.

```
./acenav -i 100base-t1 -s
```

INS401 supports In-Application Programming (BOOTLOADER) through the Ethernet interface. Run the executable with the CLI option to prompt for user input as below.



---

```
./acenav -i 100base-t1 --cli  
>>upgrade <INS401 FW file path>
```

After successful FW upgrade, the INS401 system will restart and log data automatically.

---

## Appendix C: Firmware 28.01 and Earlier IMU Axis Definition

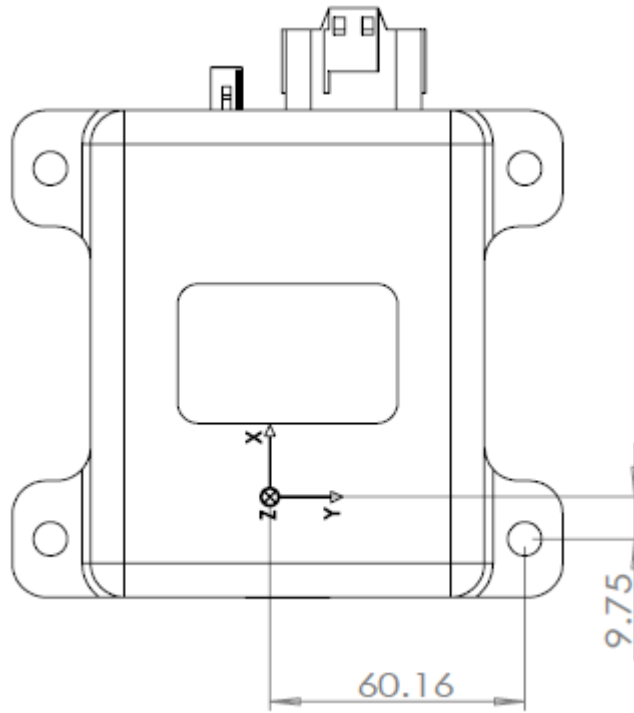


Figure 8 IMU Axis Definition and Navigation Center Location

## Appendix D: Detailed Revision History

Date	Document Revision	Firmware Applicability	Author
Aug 19, 2021	-D1	28.00.01	JN/DW/BL/CL
Details	<ul style="list-style-type: none"> <li>Preliminary Release</li> </ul>		
Oct 01, 2021	-D2	28.00.02 (maybe 28.01.00)	CT/KL/DW/CL
Details	<ul style="list-style-type: none"> <li>Update figures to the final product look and update content to 28.00.02</li> </ul>		
Nov 05, 2021	-D3	28.01	DW
Details	<ul style="list-style-type: none"> <li>Update contents to reflect FW 28.01</li> </ul>		
Dec 29, 2021	-D4	28.02	AB
Details	<ul style="list-style-type: none"> <li>Mounting instructions updated to reflect FW 28.02</li> <li>Add more contents for Firmware Upgrade</li> </ul>		
Apr 11, 2022	-D5	28.04	AB
Details	<ul style="list-style-type: none"> <li>Sdk9100 upgrade baud rate is the default baud rate 115200;</li> <li>Update GPZDA to GNZDA;</li> <li>Get product info(0xCC01) packet had been updated to new format;</li> <li>Length of ethernet packet will use 46 directly when valid user data length is smaller than 46;</li> </ul>		
Dec 12 2022	-D6	28.05/28.06	JT/CEK/XG/JN
Details	<ul style="list-style-type: none"> <li>Support vehicle code set command based on prestored projects</li> <li>Update chapter of Firmware Upgrade</li> <li>Add DM-Ext1 packet</li> </ul>		
Oct 8 2023	-D7	28.05/28.06	JT/JN
	<ul style="list-style-type: none"> <li>Add the conditions required for the INS initialization</li> <li>Add limitation for timeout and retry times</li> <li>Add chapter 7.4 to show percentiles of each step</li> <li>Update Figure 7 INS401 Upgrade Flowchart and downgrade 7.3.4 to 7.3.3.3(update table of contents)</li> </ul>		
Jun 05 2024	-D8	28.05/28.06	JT/CEK
	<ul style="list-style-type: none"> <li>Delete FuSa related contents</li> </ul>		

## Appendix E: Functions Implementation in GNSS-chip upgradation

The following is the CRC sample code used in GNSS-chip upgradation.

```
static tUInt crc32_tab[] =
{
    0x00000000, 0x77073096, 0xee0e612c, 0x990951ba, 0x076dc419,
    0x706af48f, 0xe963a535, 0x9e6495a3, 0x0edb8832, 0x79dcb8a4,
    0xe0d5e91e, 0x97d2d988, 0x09b64c2b, 0x7eb17cbd, 0xe7b82d07,
    0x90bf1d91, 0x1db71064, 0x6ab020f2, 0xf3b97148, 0x84be41de,
    0x1adad47d, 0x6ddde4eb, 0xf4d4b551, 0x83d385c7, 0x136c9856,
    0x646ba8c0, 0xfd62f97a, 0x8a65c9ec, 0x14015c4f, 0x63066cd9,
    0xfa0f3d63, 0x8d080df5, 0x3b6e20c8, 0x4c69105e, 0xd56041e4,
    0xa2677172, 0x3c03e4d1, 0x4b04d447, 0xd20d85fd, 0xa50ab56b,
    0x35b5a8fa, 0x42b2986c, 0xdbbbc9d6, 0xacbcf940, 0x32d86ce3,
    0x45df5c75, 0xdcd60dcf, 0xabd13d59, 0x26d930ac, 0x51de003a,
    0xc8d75180, 0xbfd06116, 0x21b4f4b5, 0x56b3c423, 0xcfba9599,
    0xb8bda50f, 0x2802b89e, 0x5f058808, 0xc60cd9b2, 0xb10be924,
    0x2f6f7c87, 0x58684c11, 0xc1611dab, 0xb6662d3d, 0x76dc4190,
    0x01db7106, 0x98d220bc, 0xefd5102a, 0x71b18589, 0x06b6b51f,
    0x9fbfe4a5, 0xe8b8d433, 0x7807c9a2, 0x0f00f934, 0x9609a88e,
    0xe10e9818, 0x7f6a0dbb, 0x086d3d2d, 0x91646c97, 0xe6635c01,
    0x6b6b51f4, 0x1c6c6162, 0x856530d8, 0xf262004e, 0x6c0695ed,
    0x1b01a57b, 0x8208f4c1, 0xf50fc457, 0x65b0d9c6, 0x12b7e950,
    0x8bbeb8ea, 0xfcb9887c, 0x62dd1ddf, 0x15da2d49, 0x8cd37cf3,
    0xfbd44c65, 0x4db26158, 0x3ab551ce, 0xa3bc0074, 0xd4bb30e2,
    0xa4dfa541, 0x3dd895d7, 0xa4d1c46d, 0xd3d6f4fb, 0x4369e96a,
    0x346ed9fc, 0xad678846, 0xda60b8d0, 0x44042d73, 0x33031de5,
    0xaa0a4c5f, 0xdd0d7cc9, 0x5005713c, 0x270241aa, 0xbe0b1010,
    0xc90c2086, 0x5768b525, 0x206f85b3, 0xb966d409, 0xce61e49f,
    0x5edef90e, 0x29d9c998, 0xb0d09822, 0xc7d7a8b4, 0x59b33d17,
    0x2eb40d81, 0xb7b7d5c3, 0xc0ba6cad, 0xedb88320, 0x9abfb3b6,
    0x03b6e20c, 0x74b1d29a, 0xeada54739, 0x9dd277af, 0x04db2615,
    0x73dc1683, 0xe3630b12, 0x94643b84, 0x0d6d6a3e, 0x7a6a5aa8,
    0xe40ecf0b, 0x9309ff9d, 0x0a00ae27, 0x7d079eb1, 0xf00f9344,
    0x8708a3d2, 0x1e01f268, 0x6906c2fe, 0xf762575d, 0x806567cb,
    0x196c3671, 0x6e6b06e7, 0xfed41b76, 0x89d32be0, 0x10da7a5a,
    0x67dd4acc, 0xf9b9df6f, 0x8ebeeff9, 0x17b7be43, 0x60b08ed5,
    0xd6d6a3e8, 0xa1d1937e, 0x38d8c2c4, 0x4fdff252, 0xd1bb67f1,
    0xa6bc5767, 0x3fb506dd, 0x48b2364b, 0xd80d2bda, 0xaf0a1b4c,
    0x36034af6, 0x41047a60, 0xdf60efc3, 0xa867df55, 0x316e8eef,
    0x4669be79, 0xcb61b38c, 0xbc66831a, 0x256fd2a0, 0x5268e236,
    0xcc0c7795, 0xbb0b4703, 0x220216b9, 0x5505262f, 0xc5ba3bbe,
    0xb2bd0b28, 0x2bb45a92, 0x5cb36a04, 0xc2d7ffa7, 0xb5d0cf31,
    0x2cd99e8b, 0x5bdeae1d, 0x9b64c2b0, 0xec63f226, 0x756aa39c,
    0x026d930a, 0x9c0906a9, 0xeb0e363f, 0x72076785, 0x05005713,
    0x95bf4a82, 0xe2b87a14, 0x7bb12bae, 0x0cb61b38, 0x92d28e9b,
    0xe5d5be0d, 0x7cdcefb7, 0x0bdbdf21, 0x86d3d2d4, 0xf1d4e242,
    0x68ddb3f8, 0x1fda836e, 0x81be16cd, 0xf6b9265b, 0x6fb077e1,
    0x18b74777, 0x88085ae6, 0xff0f6a70, 0x66063bca, 0x11010b5c,
```

```
0x8f659eff, 0xf862ae69, 0x616bfffd, 0x166ccf45, 0xa00ae278,
0xd70dd2ee, 0x4e048354, 0x3903b3c2, 0xa7672661, 0xd06016f7,
0x4969474d, 0x3e6e77db, 0xaed16a4a, 0xd9d65adc, 0x40df0b66,
0x37d83bf0, 0xa9bcae53, 0xdebb9ec5, 0x47b2cf7f, 0x30b5ffe9,
0xbdbdf21c, 0xcabac28a, 0x53b39330, 0x24b4a3a6, 0xbad03605,
0xcdd70693, 0x54de5729, 0x23d967bf, 0xb3667a2e, 0xc4614ab8,
0x5d681b02, 0x2a6f2b94, 0xb40bbe37, 0xc30c8ea1, 0x5a05df1b,
0x2d02ef8d
};
```

```
tUInt crc32_eval(tUInt crc32val, const tVoid *void_ptr, const tSize len)
{
    tUInt i,j;
    tPU32 buf_ptr = (tPU32)void_ptr;
    tU32 word_data;
    tPU8 byte_ptr = (tPU8)&word_data;

    if (NULL == buf_ptr)
    {
        return 0;
    }

    crc32val = crc32val ^ 0xffffffff;

    for (i = 0; i < (len / 4); i++)
    {
        word_data = buf_ptr[i];
        for (j = 0; j < 4; j++)
        {
            crc32val = crc32_tab[(crc32val ^ byte_ptr[j]) & 0xFF] ^ (crc32val >> 8);
        }
    }

    return crc32val ^ 0xffffffff;
}
```

The following is sample code of how to calculate bytes of GNSS-chip part firmware info.

```
typedef struct send_bin_info_demo_s
{
    tInt fileSize;
    tU32 bootMode;
    tU32 crc32;
    tInt destinationAddress;
    tInt entryPoint;
    tInt baudRate;
    tU8 eraseNVM;
    tU8 eraseOnly;
    tU8 programOnly;
    tU8 subSector;
    tU8 sta8090fg;
```

```
tU8 res1;
tU8 res2;
tU8 res3;
tU8 debug;
tInt nvmOffset;
tInt nvmEraseSize;
tU32 debug;
tInt nvmOffset;
tInt nvmEraseSize;
tU32 debugAction;
tU32 debugAddress;
tInt debugSize;
tU32 debugData;
}send_bin_info_demo_t;

int binInfoVal_set(binData, binDataLen)
{
    tUInt crc32val;
    tInt fileSize;

    send_bin_info_demo_t info;
    fileSize = info.fileSize;

    crc32val = 0;
    crc32val = crc32_eval(crc32, fileSize, 4);
    crc32val = crc32_eval(crc32, binData, binDataLen);

    info.bootMode = 0x01;
    info.crc32 = crc32val;
    info.destinationAddress = 0x10000000;
    info.entryPoint = 0;
    info.eraseNVM = 3;
    info.eraseOnly = 0;
    info.programOnly = 0;
    info.subSector = 0;
    info.sta8090fg = 0;
    info.res1 = 0;
    info.res2 = 0;
    info.res3 = 0;
    info.debug = 0;
    info.nvmOffset = 0;
    info.nvmEraseSize = 0;
    info.debugAction = 0;
    info.debugAddress = 0;
    info.debugSize = 0;
    info.debugData = 0;
}
```