INS401 USER MANUAL

(Preliminary)

Document Part Number: 7430-4006-02

ACEINNA, Inc.

email: info@aceinna.com, website: 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

⚠ 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.

Table of Contents

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.

The following paragraph heading formatting is used in this manual:

1 Heading 1

1.1 Heading 2

1.1.1 Heading 3

1.1.1.1 Heading 4

Normal

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 powered by RTK330LA – an ASIL B certified product of Aceinna. 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 – GPS, GLONASS, Galileo, Beidou and QZSS. It tracks the following satellite signals 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:

- Position, velocity, heading and attitude solution at 100Hz
- Multi-constellation and multi-frequency RTK algorithm support for centimeter accurate positioning
- Integrated and calibrated triple-redundant MEMS IMU with range of ±8 g and 250 °/s
- Automotive ethernet interface
- 1 PPS output
- 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 rugged 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, plus 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:

- The small circular connector is the FAKRA-J C type connector for RF connection to the GNSS antenna
- 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.

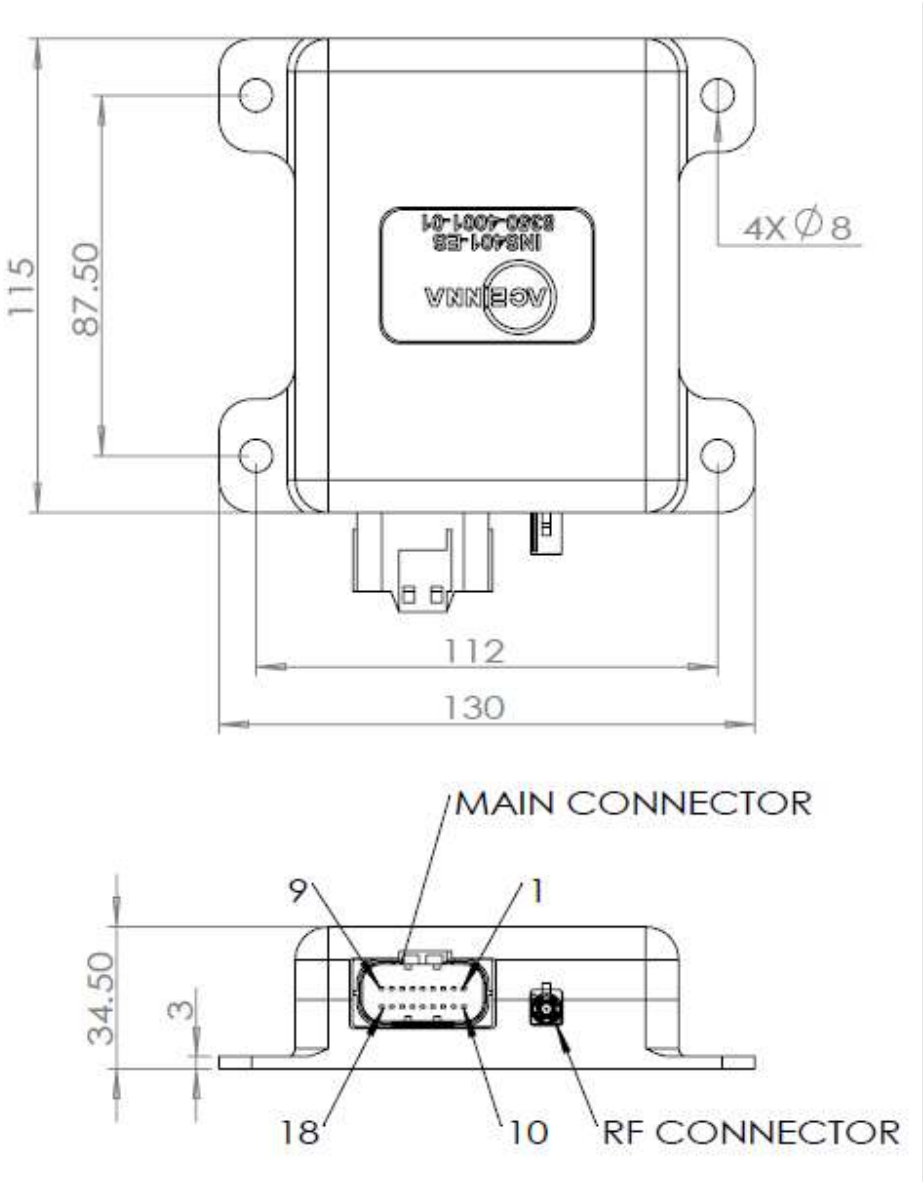


Figure 2 INS401 Hardware Dimensions

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.

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.

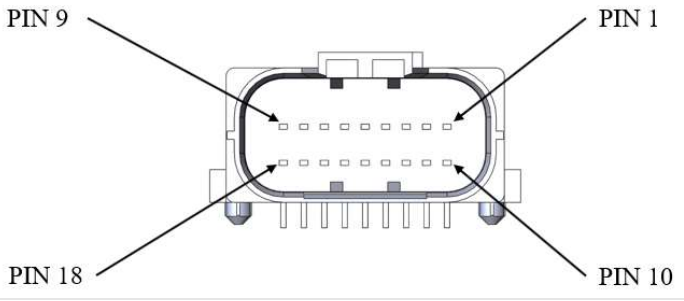


Figure 3 Pin Diagram of the Male End

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 11) 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.

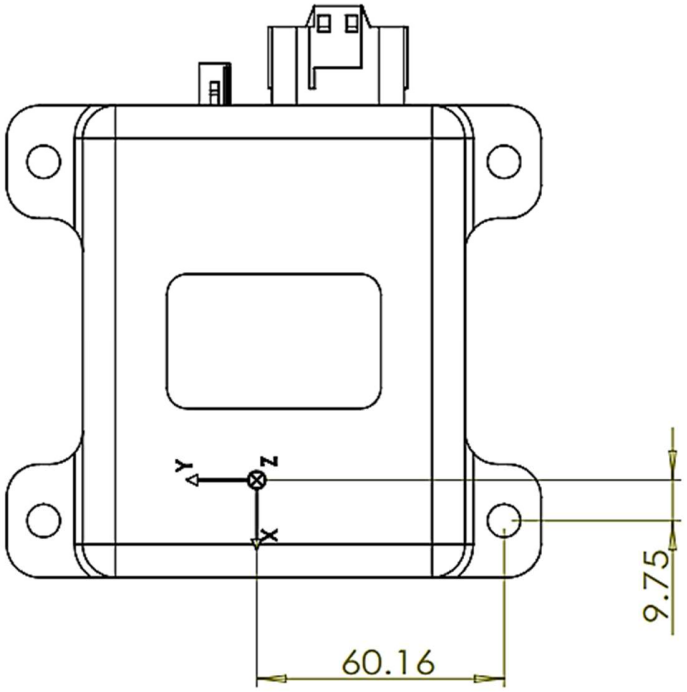


Figure 4 IMU Axis Definition and Navigation Center Location

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 11 should be used to configure the INS401 with the correct lever arm.

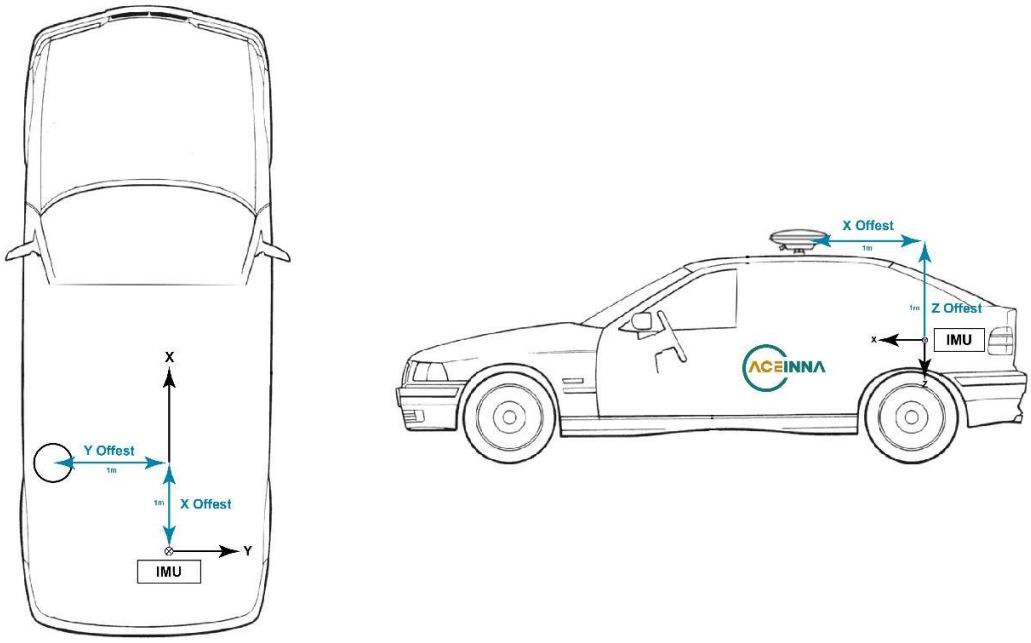


Figure 5 INS401 IMU to GNSS Antenna Lever Arm Definition and Measurement Demonstration

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.

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

Based on your application, select an GNSS L1&L2 active antenna. A typical antenna LNA gain between 30 dB and 45 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.

3.3 Power Requirement

Operating voltage: 12V DC

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:

- Active multi-constellation multi-band GNSS antenna
- 12 V power adapter (battery)
- 100Base-TX (RJ-45 connector) to 100Base-T1 converter
- Testing PC (Windows 10 or Ubuntu 18.04) with internet access
- External harness with MX23A automotive connector and cables
- 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 key 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.

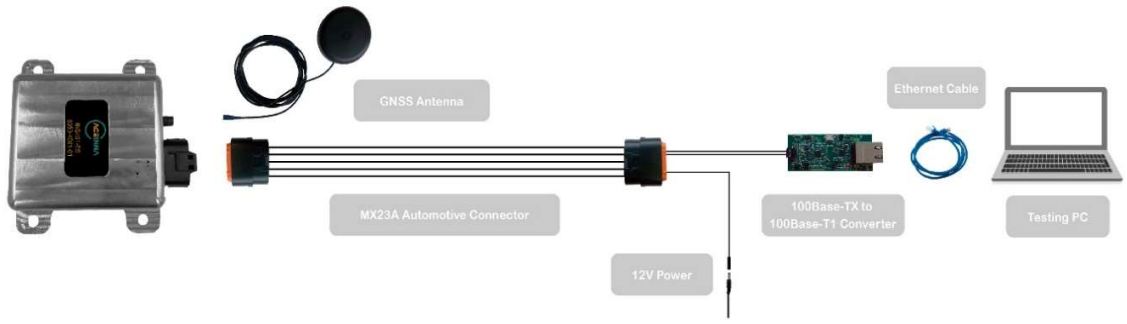


Figure 6 Typical INS401 Configuration

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 (INS401 EVK will be available Q1-2022 which include: INS401, antenna, cables, mating connectors etc.).

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"
    }
]
```

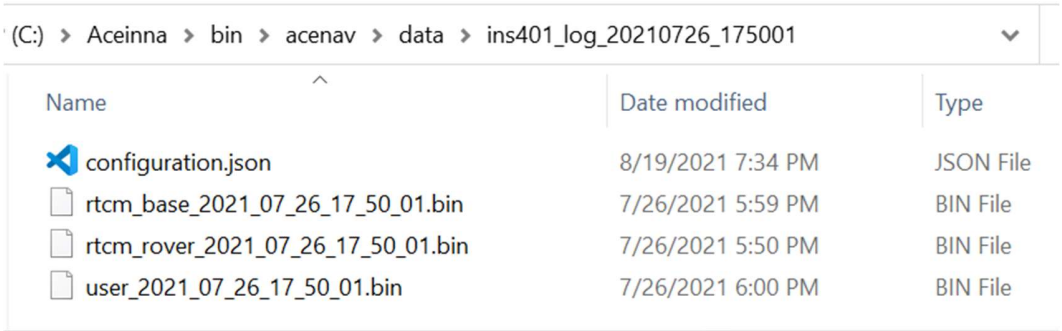
6. Input the lever arm values in the field of "gnss lever arm x", "gnss lever arm y" and "gnss lever arm z" (if installed in a vehicle). Note the lever arm is in meters

```
"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
```

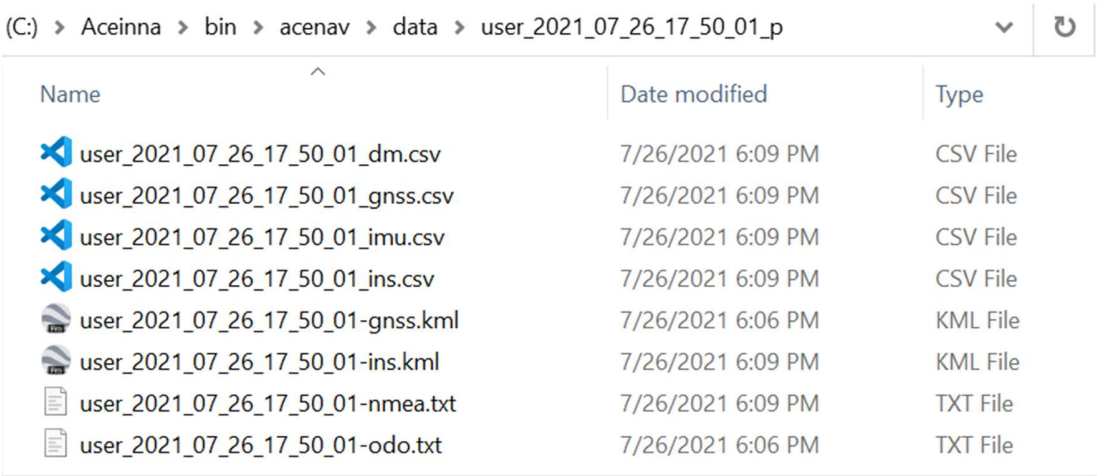
8. Check the output bin files. Every session the "AceNav" CLI has started, a data log subfolder with the time tag in the folder name is created under "acenav/data". The data log subfolder for each data logging session contains four files:
    - *configuration.json*: INS401 configuration file read from the device flash
    - *rtcm_base_<time_tag>.bin*: GNSS RTK correction data in RTCM format
    - *rtcm_rover_<time_tag>.bin*: INS401 GNSS raw data in RTCM format
    - *user_<time_tag>.bin*: high precision positioning solution data, including GNSS solution, INS solution, and other related variance/status information



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
    - *user_<time_tag>_dm.csv*: INS401 system diagnostic messages
    - *user_<time_tag>_gnss.csv*: GNSS solution at 1 Hz
    - *user_<time_tag>_imu.csv*: raw IMU data at 100 Hz
    - *user_<time_tag>_ins.csv*: INS solution at 100 Hz
    - *user_<time_tag>_gnss.kml*: GNSS solution trajectory in kml format for Google Earth presentation
    - *user_<time_tag>_ins.kml*: INS solution trajectory in kml format for Google Earth presentation
    - *user_<time_tag>_nmea.text*: GNSS solution in NMEA 0183 ASCII strings
    - *user_<time_tag>_odo.text*: vehicle odometer speed data received by INS401



Refer to Appendix B: 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 7 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 8 for vehicle speed data packet.

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

- Sampling period of 10±3 ms

- Error standard deviation smaller than 0.1 m/s
- Normalized error autocorrelation at 0.1 s below 0.75 (mean removed before autocorrelation)
- Normalized error autocorrelation at 1 s below 0.70 (mean removed before autocorrelation)
- Error probability density function must be symmetrical (small deviations accepted)
- Maximum delay of 0.1 s
- The vehicular speed must report 0 m/s when the vehicle is not moving
- The vehicular speed must not report 0 m/s when the vehicle is moving
- The vehicular speed must be valid for the entire range of operation of the vehicle, including negative speeds

5 Diagnostic Capabilities

Future section.
Please refer to Table 5 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 binary packets and ASCII messages should be filled in user data. 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	User Data	Data Frame Checksum
6 bytes	6 bytes	2 bytes	46 to 1500 bytes	4 bytes

- Destination Address: destination MAC address
- Source Address: source MAC address
- Length: length of user data (0~1500), MSB for output message from INS401 to ECU or the 100Base-TX/100Base-T1 converter, LSB for input message.
- User Data: Min/Max data length is 46/1500. Zero bytes will be filled if 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 "Data content" 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 3 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: GPS week and seconds in week
2	uint32	gps_millisecs	ms	
6	uint8	position_type		0: INVALID 1: Single-point positioning (SPP) 2: Real time differential GNSS (RTD) 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

Table 4 INS Solution Packet

Message	INS navigation solution, periodic output at 100 Hz			
Message ID	**0x0a03**			
Length	108			
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	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 5 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		Definition in the struct below*
10	float32	IMU temperature	ºC	Temperature of the IMU
14	float32	MCU temperature	ºC	Temperature of the MCU
18	float32	GNSS chip temperature	ºC	Temperature of the STA9100 chipset

Definition of the device status bit field is as below. Each status bit has two states :0 is valid and 1 is invalid.

Table 6 Typedef struct

	Byte	Type	Description	Value
IMU	0	Uint32	Master fail	0: normal 1: fatal error occured
	1	Uint32	HW error	0: normal 1: hardware exception detected
	2	Uint32	SW error	0: normal 1: software exception detected
	3	Uint32	Config error	0: normal 1: config error detected by periodic self-test
	4	Uint32	Calibration error	0: normal 1: calibration data corrupted
	5	Uint32	Accel degradation	0: normal 1: accel data degradation due to sensor exception
	6	Uint32	Gyro degradation	0: normal

				1: gyro data degradation due to sensor exception
	7	Uint32	Forced restart	0: normal 1: forced restart
	8	Uint32	CRC error	0: normal 1: CRC error detected
	9	Uint32	Tx overflow error	0: normal 1: Tx Overflow occurred 10 consecutive cycles
GNSS	10	Uint32	PPS status	0: normal 1: 1PPS pulse exception
	11	Uint32	GNSS data status	0: normal 1: GNSS chipset has NO data output
	12	Uint32	GNSS signal status	0: normal 1: GNSS chipset has data output but no valid signal detected
Operation	13	Uint32	Power	0: normal 1: any component has no power
	14	Uint32	MCU status	0: normal 1: MCU failure MCU and peripherals HW self-test status when power on
	15	Uint32	Temperature under MCU flag	0: normal 1: under temperature
	16	Uint32	Temperature under STA flag	0: normal 1: under temperature
	17	Uint32	Temperature under IMU flag	0: normal 1: under temperature
	18	Uint32	Temperature over MCU flag	0: normal 1: under temperature
	19	Uint32	Temperature over STA flag	0: normal 1: under temperature
	20	Uint32	Temperature over IMU flag	0: normal 1: under temperature
Reserved		Uint32	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 1 in the device diagnostic message (message ID 0x0a05).

6.3.4 Raw IMU Data

Table 6 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

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 7 RTCM Data

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

◂ 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

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

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

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

Field Sequence	Description
0	Message ID $GPZDA
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

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 7, followed by a detailed example.

Table 7 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.

☑ EXAMPLE

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

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		

In addition, another user input of the INS401 is defined as below, in order 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 speed. In such case, the vehicle reference point (VRP) is the middle point of the two rear wheels in the vehicle frame.

Table 8 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, negative if backward moving, zero if stationary.

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

6.6 User Commands

Several user commands are defined on the ethernet interface for the user system (e.g. a vehicle) to poll information from INS401 and to set user parameters in INS401. The user request

commands and the INS401 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.

Table 9 Get the Product Information

Description	Get the product device information, as per user poll
Request Command:	
Command	**0xcc01**
Length	0
Payload	None
Response Message:	
Message ID	**0xcc01**
Length	N/A
Payload	ASCII message (separated by space), e.g. INS401 8550-4006-01 21090xxxxx Hardware v1.0 *（Product） (P/N) (S/N) (HW Version)* *RTK_INS Firmware v28.XX.XX* Bootloader v01.xx.xx *(FW Version) (Bootloader Version)* IMU FW v27.00.XX STA9100 FW v5.XX.XX (IMU FW version) (STA9100 FW version)

Table 10 Get User Configuration

Description	Get user configuration on the parameters (defined in Table 13)			
Request Command:				
Command	**0xcc02**			
Length	4			
Payload	Sequence ID as defined in Table 13			
Response Message:				
Message ID	**0xcc02**			
Length	4 bytes (Sequence ID) + length of parameter value (each defined in Table 13)			
content				
offset	variable	name	unit	desc
0	uint32	Sequence ID		refer to Table 13
4		parameter value		

Table 11 Set User Configuration

Description	Set user configuration on the parameters (defined in Table 13)			
Request Command:				
Command	**0xcc03**			
Length	4 bytes (Sequence ID) + length of parameter value			
Payload				
offset	variable	name	unit	desc

0	uint32	Sequence ID		refer to Table 13
4		Parameter value		
Response Message:				
Message ID	**0xcc03**			
Length	4			
Payload				
offset	variable	name	unit	desc
0	int32	result		0：success -1：invalid parameter number -2：invalid parameter value

Table 12 Save User Configuration

Description	Save user configuration parameters (defined in Table 13) permanently in device flash			
Request Command:				
Command	**0xcc04**			
Length	N/A			
Payload	None			
Response Message:				
Message ID	**0xcc04**			
Length	4 bytes			
Payload				
offset	variable	name	unit	desc
0	int32	result		0: success -1: fail

Table 13 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

Table 15 User sends system reset command

Description	User sends system reset command
Request Command:	
Command	**0xcc06**
Length	N/A
Payload	None
Response Message:	
Message ID	**0xcc06**
Length	N/A
Payload	None

Table 16 User gets ins save buffer of fixed position of ins power on

Description	User gets ins save buffer of fixed position of ins power on			
Request Command:				
Command	**0x0a09**			
Length	N/A			
Payload	None			
Response Message:				
Message ID	**0x0a09**			
Length	4 bytes			
Payload	Ins save buffer			
offset	variable	name	unit	desc
0	uint8_t	Ins save buffer[156]		Ins save buffer

7 Firmware Upgrade

Definitions:

IAP : BootLoader;

APP : Application software program;

Write App: Write APP data to flash;

JI: Jump from application software 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;

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

Notice:

1) INS401 only supports Eth port upgrade;
2) INS401 only supports the upgrade of APP, ST9100 and IMU, and does not support the online upgrade of BootLoader;

7.1 FW Upgrade with AceNav

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

7.2 BootLoader Design

BootLoader communication follows the protocol of the Ethernet interface. For the specific content of the protocol, refer to section 6.1 Ethernet Data Frame Definition.

The status switching of the INS401, IMU and ST9100 upgrade process is shown in Figure 7.

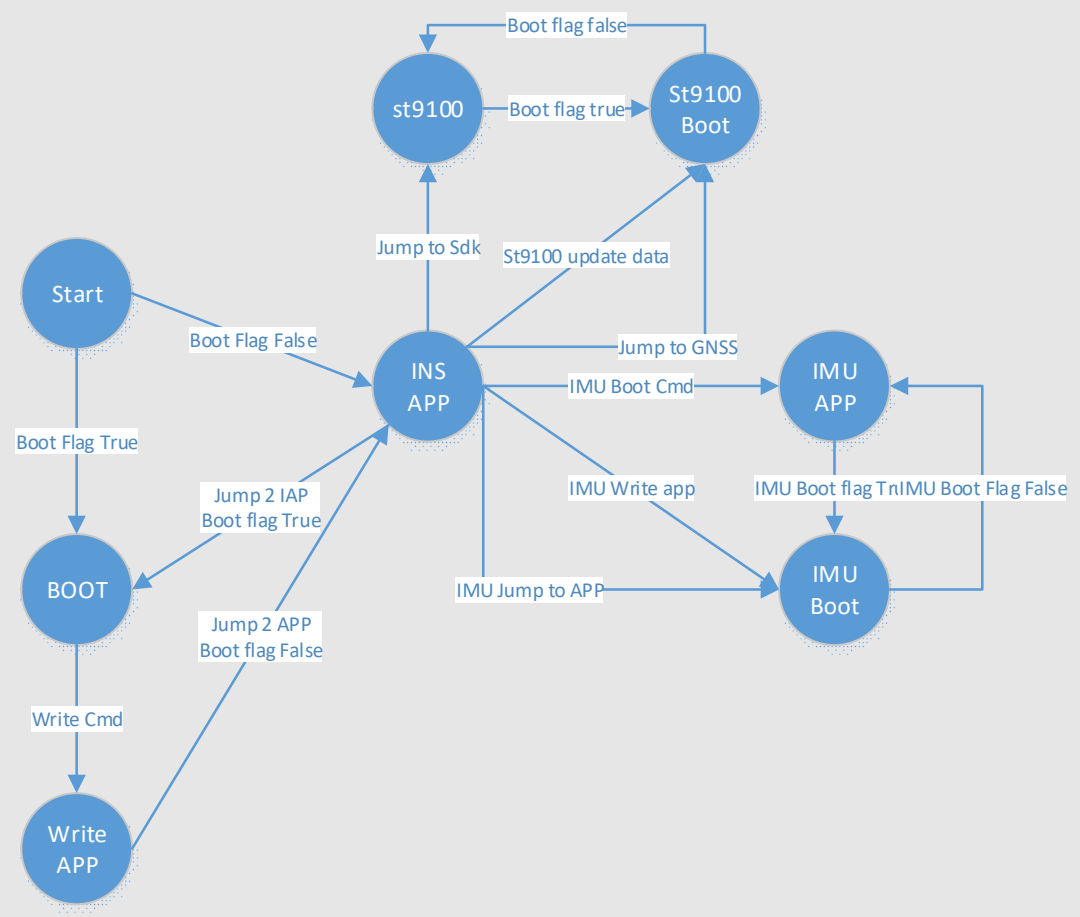


Figure 7 Relationship between BootLoader, IMU APP, INS APP and ST9100

7.2.1 INS401 APP Upgrade

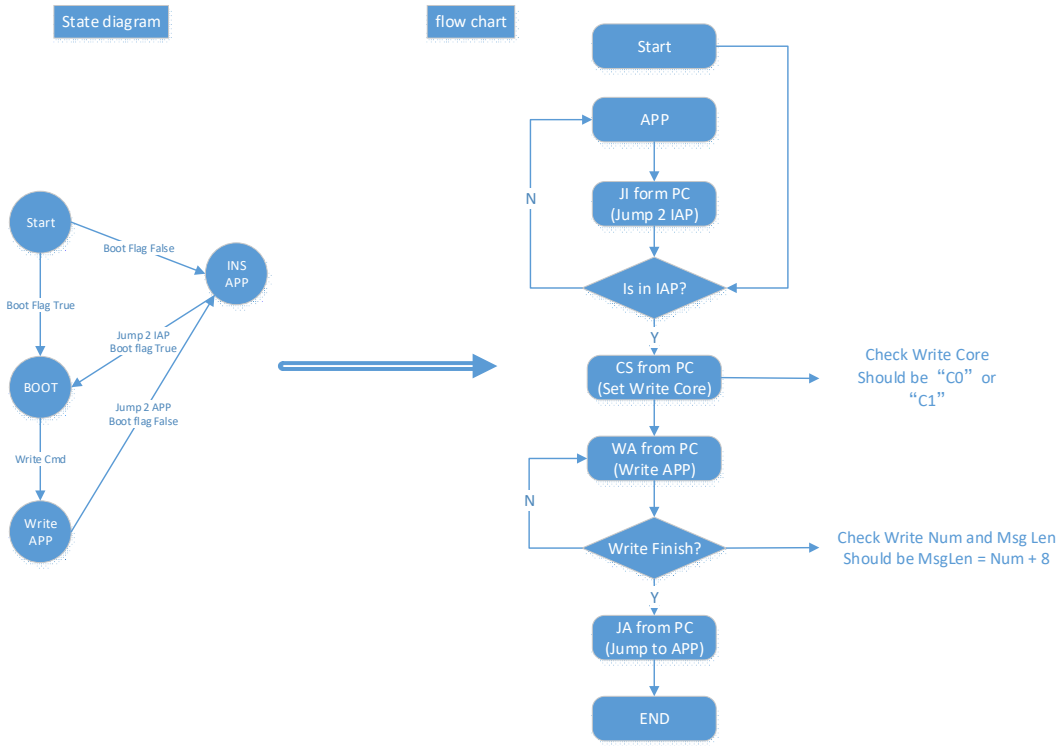


Table 14 Jump to IAP (JI)

Send:

Header	Msg_Type	Length	Data	CRC
0x5555	0xcc01	0	--	2 bytes

Receive:

Header	Msg_Type	Length	Data	CRC
0x5555	0xcc01		PRODUCT_NAME_STRING, unit_version, PART_NUMBER_STRING, serialNum	2 bytes

OR

Send:

Header	Msg_Type	Length	Data	CRC
0x5555	0xcc0b	0	--	2 bytes

Receive:

Header	Msg_Type	Length	Data	CRC
0x5555	0xcc0b		PRODUCT_NAME_STRING, unit_version, PART_NUMBER_STRING, serialNum	2 bytes

Table 15 Jump to IAP (JI)

Send:

Header	Msg_Type	Length	Data	CRC
0x5555	0xaa01	0	--	2 bytes

Receive:

This command is used to jump to BootLoader. This command can be sent regardless of whether it is currently in the APP or BootLoader.

Table 16 Jump to APP (JA)

Send:

Header	Msg_Type	Length	Data				CRC
0x5555	0xaa02	4	rtkCRC_L 1 byte	rtkCRC_H 1 byte	insCRC_L 1 byte	insCRC_H 1 byte	2 bytes

Receive:

Header		Msg_Type	Length	Data						CRC
0x5555		0xaa02	6	'C'	'0'	'S'-succ 'F'-Fail	'C'	'1'	'S'-succ 'F'-Fail	2 bytes

The sent JA command needs to include the CRC values of rtk.bin and ins.bin. After receiving the JA command, the MCU will calculate the CRC of the written rtk.bin and ins.bin. Only when the calculated CRC values are correct , It will jump to the APP, otherwise it will stay under BootLoader.

Table 18 Set Update Core and Bin Size (CS)

Send:

Header	Msg_Type	Length	Data			CRC
0x5555	0xaa04	9	Update core 2 bytes	Bin size 4 bytes	AppInfo 3Bytes	2 bytes

Receive:

Header	Msg_Type	Length	Data		CRC
0x5555	0xaa04	2	Update core 2 bytes	'S' – Set Succ 'F' -- Set Failed	2 bytes

Update core = "C0" or "C1". "C0" corresponds to rtk_app, and "C1" corresponds to ins_app.

Bin size is the length of the actual rtk or ins bin file. This value should be sent with the high order first.

```
*bin_size = (ptrUcbPacket->payload[2] << 24) |
            (ptrUcbPacket->payload[3] << 16) |
             (ptrUcbPacket->payload[4] << 8) |
            (ptrUcbPacket->payload[5] );
```

It should be noted that when the host computer reads All.bin, it confirms the burned bin.

For example, when burning rtk.bin, in addition to identifying the rtkstart information in all.bin, the host computer should also identify the information in Bin to ensure that it is the real rtk.bin. When it is confirmed as rtk.bin, fill in "rtk" in the AppInfo section. When the MCU receives the correct CS command, it returns'S', otherwise, it returns'F'.

At the same time, when the Update core in the CS command is not "C0" or "C1", the MsgType of the returned frame = 0x1515.

Table 19 Write APP Bin (WA)

Send:

Header	Msg_Type	Length	Data			CRC
0x5555	0xaa03	Numbytes + 8	start Addr 4 bytes	Numbytes 4 bytes	BinData	2 bytes

Receive:

Header	Msg_Type	Length	Data	CRC
0x5555	0xaa03	0	--	2 bytes

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 write command, BootLoader will erase the flash first. Please wait until the erase is complete.

At the same time, ensure that Length = Numbytes + 8, and the written BinData is a multiple of 16, otherwise Msg_Type = 0 will show in the message returned by BootLoader.

When writing is successful, Length = 0 will show in the message returned by BootLoader.

7.2.2 INS401 IMU FW Upgrade

When upgrading the IMU, make sure that INS401 is in APP mode, otherwise the upgrade will not be possible.

The IMU FW upgrade process is shown in Figure 8.

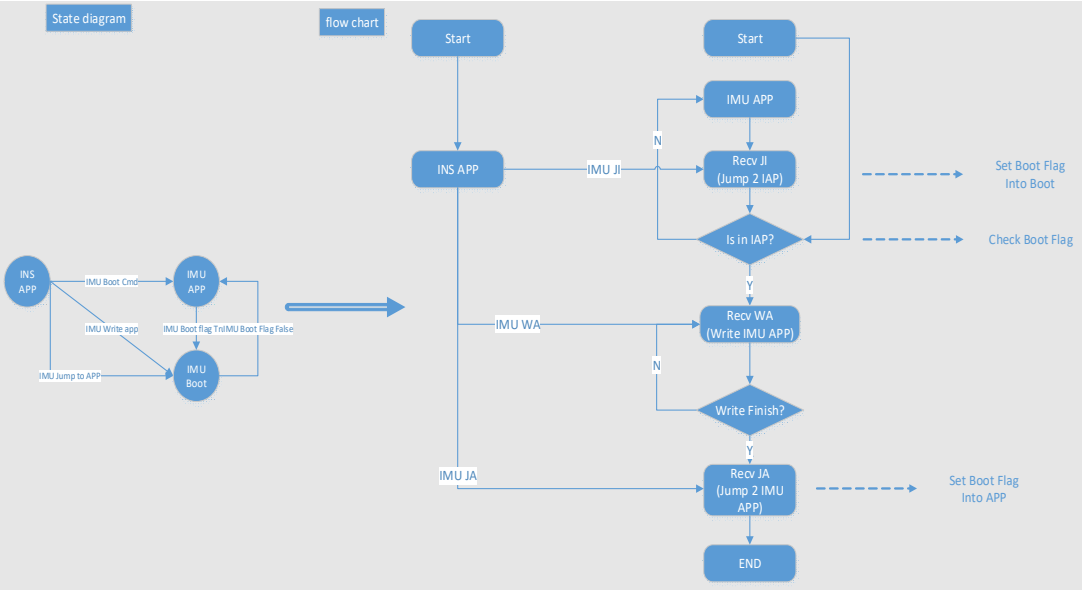


Figure 8 State Diagram and Flowchart of INS401 IMU FW Upgrade

The commands for INS401 IMU FW upgrade are listed below.

Table 20 IMU Jump to IAP (IMU JI)

Send:

Header	Msg_Type	Length	Data	CRC
0x5555	0x4a49	0	--	2 bytes

Receive:

Header	Msg_Type	Length	Data	CRC
0x5555	0x4a49	0	--	2 bytes

This command is used to let IMU jump to IMU BootLoader.

Table 21 Write IMU APP (IMU WA)

Send:

Header	Msg_Type	Length	Data			CRC
0x5555	"WA" (0x5741)	Numbytes + 5	start Addr 4 bytes	Numbytes 1 byte	BinData less than 240 bytes	2 bytes

Receive:

Header	Msg_Type	Length	Data		CRC
0x5555	"WA" (0x5741)	5	start Addr 4 bytes	Numbytes 1 byte	2 bytes

star 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 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 in Table .

Table 22 Message NAK

Header	Msg_Type	Length	Data	CRC
0x5555	0x1515	2	--	2 bytes

Table 17 Jump IMU APP (IMU JA)

Send:

Header	Msg_Type	Length	Data	CRC
0x5555	"JA" (0x4A41)	--	--	2 bytes

Receive:

Header	Msg_Type	Length	Data	CRC
0x5555	"JA" (0x4A41)	0	--	2 bytes

This command is used to let IMU jump to APP.

7.2.3 INS401 STA9100 FW Upgrade

When upgrading the STA9100, make sure that INS401 is in APP mode, otherwise the upgrade will not be possible. The STA9100 upgrade process is shown in Figure 9.

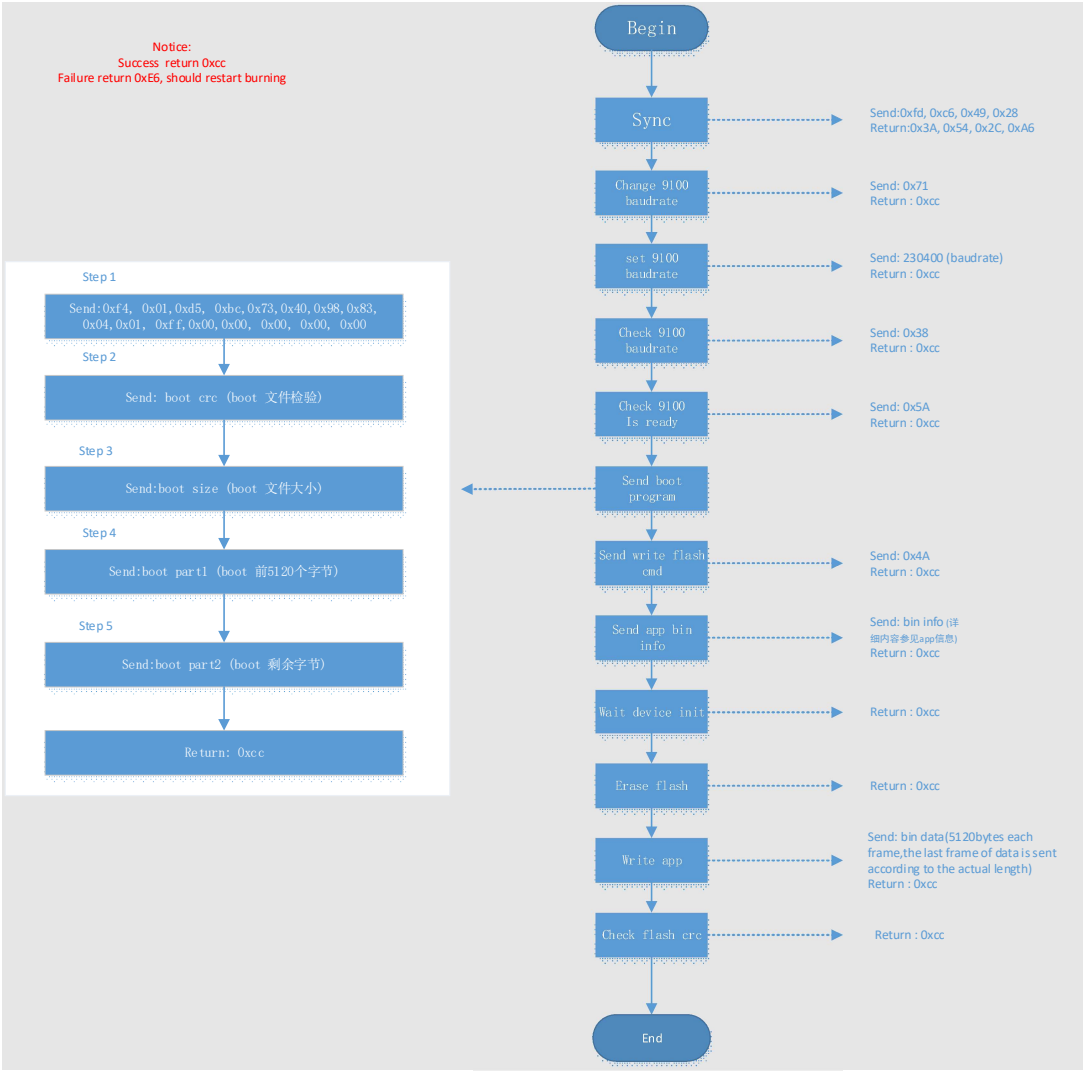


Figure 9 Flowchart of STA9100 Upgrade

The commands for STA9100 upgrade are listed below.

Table 24 Set STA9100 to Boot(JS)

Send:

Header	Msg_Type	Length	Data	CRC
0x5555	0xaa05	0	--	2 bytes

Receive:

Header	Msg_Type	Length	Data	CRC
0x5555	0xaa05	0	--	2 bytes

Table 18 Set STA9100 to GNSS(JG)

Send:

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

Receive:

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

Table 19 Send STA9100 Bin First 5K

Send:

Header	Msg_Type	Length	Data	CRC
0x5555	0xaa08			2 bytes

When STA9100 writes SDK bin, there is a write timeout mechanism. To avoid triggering this mechanism, we will send a 5k bin file first. So separated commands are used to distinguish this.

Table 20 STA9100 Upgrade

Send:

Header	Msg_Type	Length	Data	CRC
0x5555	0xaa07			2 bytes

Receive:

Header	Msg_Type	Length	Data	CRC
0x5555	0xaa07			2 bytes

For the specific upgrade protocol of STA9100, refer to Figure 9.

ST9100 Boot: Sta9100CB.cpp

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 runs 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.

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

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.

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 (IAP) firmware upgrade 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

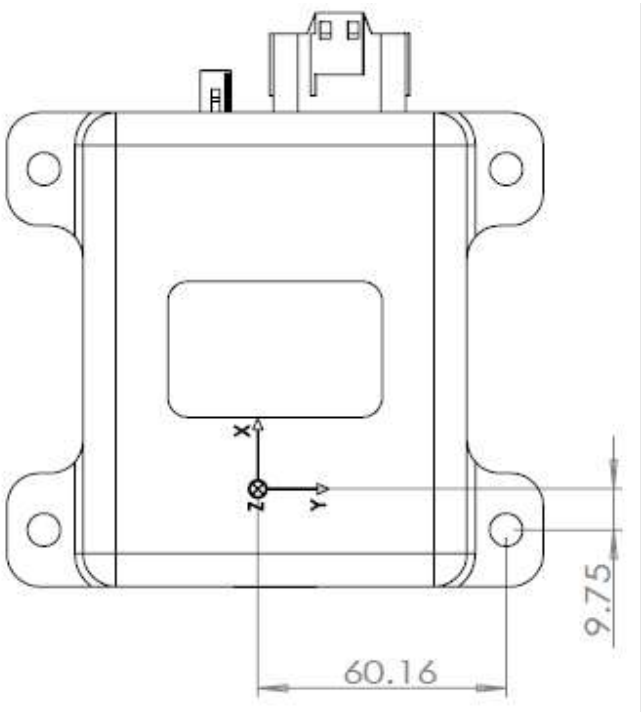


Figure 10 IMU Axis Definition and Navigation Center Location