NAV680-AG_Datasheet_产品手册 英文版

FSS-NAV680-AG High Precision Integrated Navigation System User Manual

Forsense (Shanghai) Technology Co., Ltd

Read this user manual carefully before using the product.

Index

1. Product Overview

1.1 Product List

Please confirm the following products in the packaging box when you open it:

Name	Quantity
① NAV680-AG Integrated Navigation System	1
② Satellite Antenna	2
③ Antenna cable Primary Antenna:TNC Male to Fakra Female (C Type) (Blue) Secondary Antenna:TNC Male to Fakra Female (D Type) (Purple)	2
④ Antenna suction cup	2
⑤ Antenna pole	2
⑥ Wire Harness (including 3 RS232 interfaces, 2 CAN interfaces, 1 PPS interface, and 1 (9~24V) power interface).	1

Figure 1 Product Image

If there is any missing, please contact the salesperson in a timely manner.

1.2 Introduction

In meet the demand for precise and continuous positioning needs of outdoor lightly obstructed environments such as low-speed autonomous driving in fields, orchards, scenic spots and so on, Forsense has designed the FSS-NAV680-AG, a multi-sensor integrated navigation product specifically tailored for agricultural applications.

This product combines the Agricultural Single-Antenna module IMU614E-AG with Unicore’s full-system full-frequency high-precision positioning and directional module UM982.

Equipped with our multi-model intelligent position fusion Algorithms optimized for in-vehicle scenarios, FS982-AG provides customers with high-precision attitude and heading information in complex scenarios and enables them to achieve continuous positioning and velocity measurement even in scenarios with signal interference or shielding. The product contributes to the reliable full-scene operation and enhanced end-user experience of Agricultural autopilot equipment.

To meet the needs of measuring the steering angle of the front wheels in agricultural machinery, mining trucks, etc., the NAV680-AG can be connected to our G200 the front wheel steering angle sensor, which can output the steering angle of the front wheels directly.

1.3 Technical Specifications

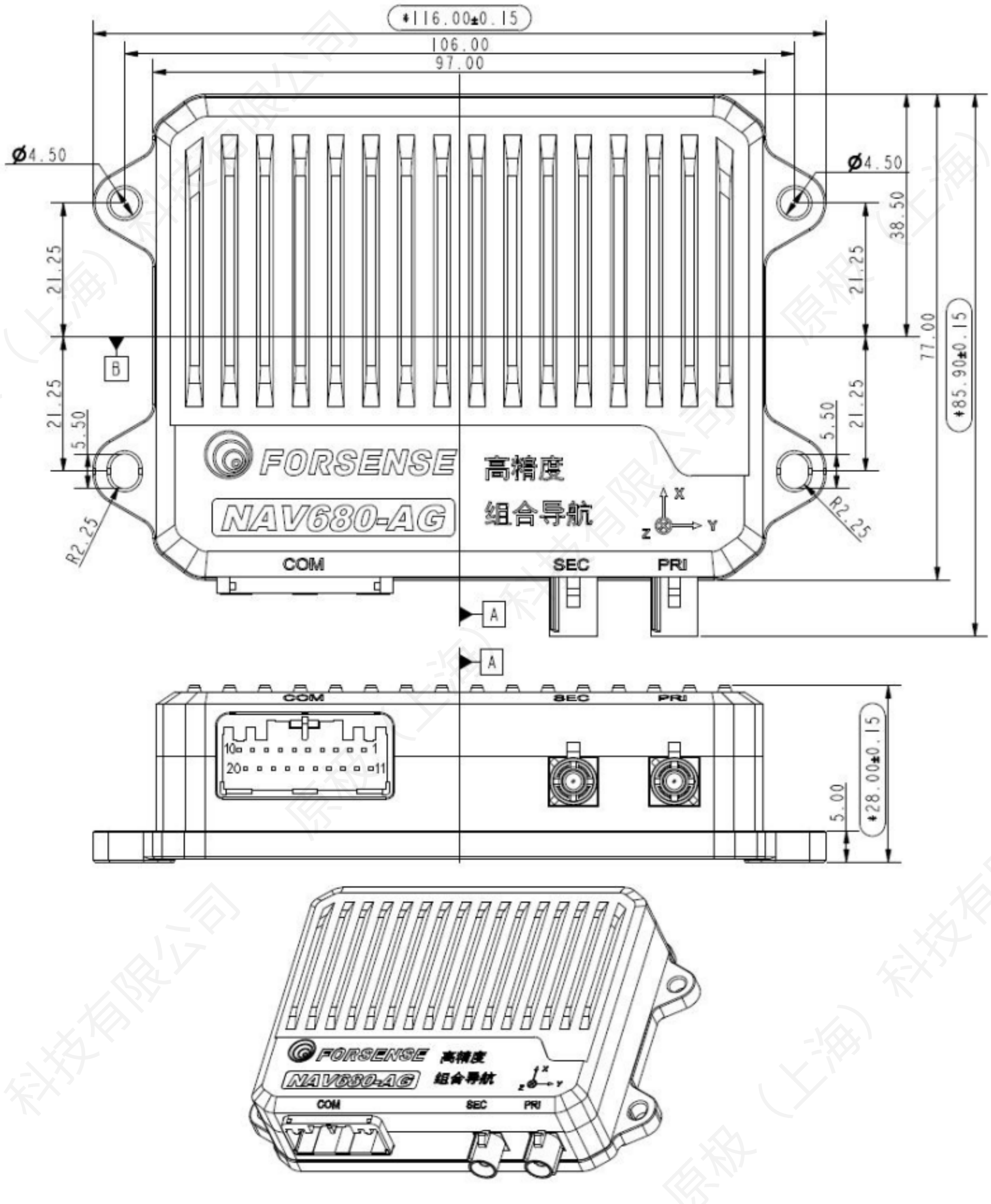
Table 1

Attitude Accuracy	Roll/Pitch :<0.2°rms Heading:<0.3°rms (When the vehicle speed exceeds 1 km/h, the Ackermann steering mechanism is engaged in the vehicle's on-road scenario.)	
Update Rate	100Hz	
Gyroscope Measurement Range	±500°/s	
Gyroscope Bias Instability	4deg/h @1σ	
Accelerometer Measurement Range	±6g	
Accelerometer Bias Instability	0.04mg @1σ	
Positioning Accuracy	<2% @1σ （in-vehicle scenario, loss of GPS signal for 30s, no combination of wheel speedometer）	
RTK Indicators	Positioning Accuracy (RMS)	Single point positioning: 1.5m RTK: Horizontal: 1cm+1ppm Vertical: 2cm+1ppm
	Directional accuracy (RMS)	0.1°/1m Baseline
	Velocity accuracy (RMS)	0.03m/s
	PPS accuracy (RMS)	20 ns
	Update Rate	20 Hz
	RTK initialization time	<5s
Interface	3×RS232 2xCAN 2×GNSS Antenna Interface 1×Power interface 1×PPS interface	

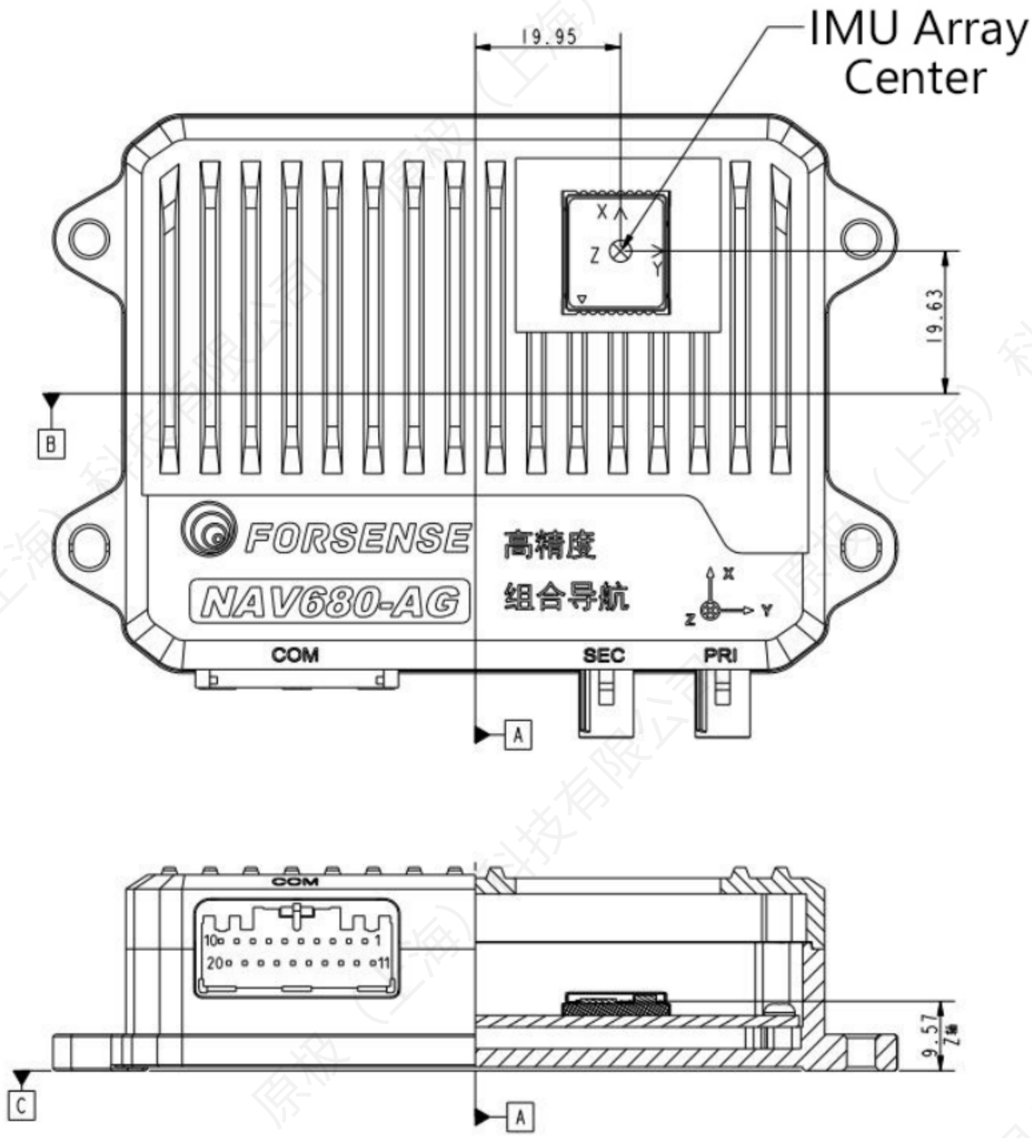
2. Hardware Structure

2.1 Electrical Properties

Fig. 1 Mechanical Dimensions (unit: mm)



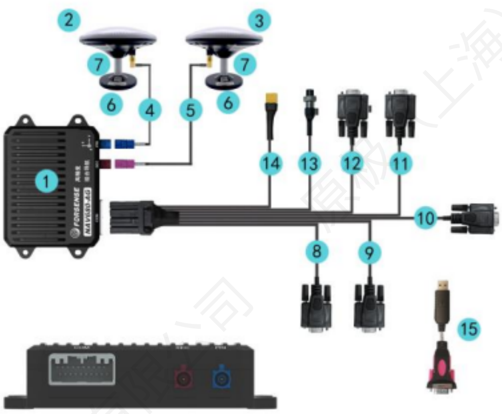
FSS-NAV680-AG Dimensional Specifications



FSS-NAV680-AG Position and Dimensional Specifications

2.2 User Interface

The NAV680-AG includes COM bus interfaces, leading power and signal cables, PRI primary Antenna and SEC secondary Antenna.



	Name	Quantity
1	FSS-NAV680-AG Integrated Navigation System	1
2	Primary antenna (Positioning Antenna)	1
3	Secondary antenna (Directional Antenna)	1
4	Primary antenna Cable	1
5	Primary antenna Cable	1
6	Antenna suction Cup	2
7	Antenna Pole	2
8	Wire Harness - Power Supply Interface	1
9	Wire Harness - RS232-A Interface	1
10	Wire Harness - RS232-B Interface	1
11	Wire Harness - CAN-A Interface	1
12	Wire Harness - CAN-B Interface	1
13	Wire Harness - RS232-RTK Interface	1
14	Wire Harness - IO Interface	1
	Accessory Name	**Quantity**
15	RS232 to USB Serial Cable	1

2.3 Data Cable Interface Definition

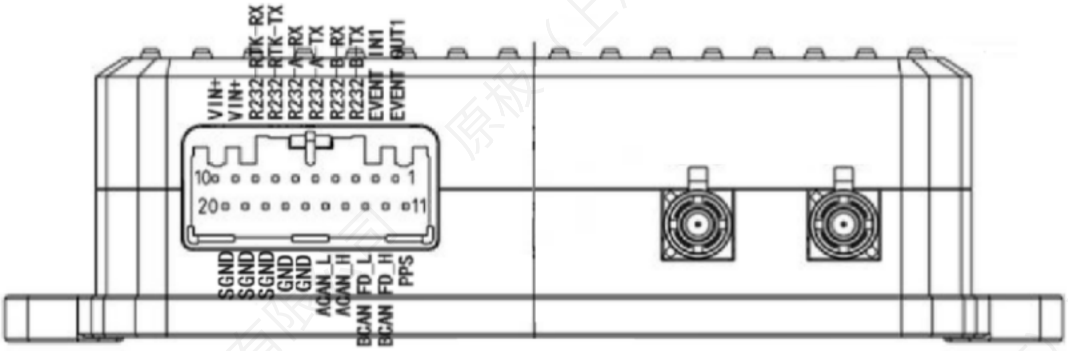


Figure 2 NAV680-AG Front Panel

The corresponding pin definitions are as follows:

No.	Definition	port
1	EVENT OUT1	IO pin (reserved)
2	EVENT IN1	
3	MCU TX3 232	R232-B
4	MCU RX3 232	
5	MCU TX2 232	R232-A
6	MCU RX2 232	
7	RTK TX1 232	R232-RTK
8	RTK RX1 232	
9	VIN+	Power Positive 9-24V
10	VIN+	Power Positive 9-24V
11	PPS	RTK second pulse signal
12	BCAN FD_H	CAN_FD B （reserved）
13	BCAN FD_L	
14	ACAN_H	CAN_ A （The default baud rate is set to 500k）
15	ACAN_L	
16	GND	power ground (power negative)
17	GND	
18	SGND	signal ground
19	SGND	
20	SGND	

Description: RS232-A serial port A, RS232-B serial port B, RS232-RTK serial port differential import, CAN A, CAN B (undefined), XT60H-M power supply, IO (undefined).

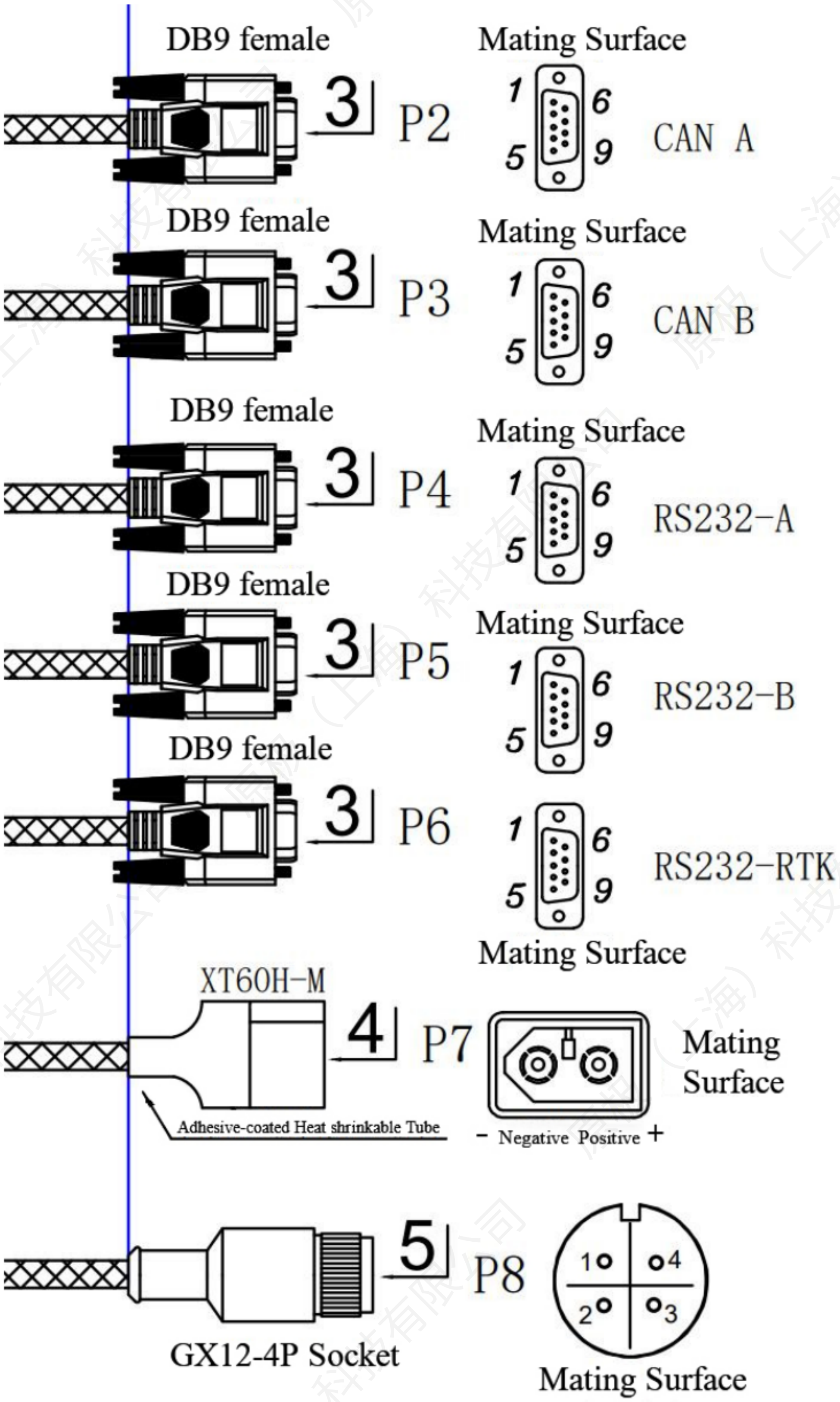
2.4 Electrical Properties

Table 2

Parameters	Symbol	Min.	Typ.	Max.	Unit
Power Input	VIN	9	12	24	V
Ground	GND				
Power Consumption	P		1.8		W
Operating Temperature	T	-40	\	85	℃
Storage Temperature	T	-40	\	95	℃
Waterproof	95% Non-condensing				
Dimension	116mm * 85.9mm * 28mm				
Weight	203g				

3. Hardware Integration Guide

3.1 Hardware Integration Guide



3.2 Antenna

The NAV680-AG leads out the PRI and SEC Primary and Secondary Antenna Interfaces. When no Antenna is connected and a multimeter is used for testing, that is, in the no-load state, the provided voltage is DC 4.8 to 5.4 V. When the RF port of the module is connected to an Antenna and tested at a working current of 30 to 100 mA at room temperature, it can provide an antenna feed of DC 4.6 V ± 0.2 V to the outside. When using an active antenna for the NAV680-AG, pay attention to the 50Ω **Impedance matching** between it and the Antenna.

4. Usage Examples

4.1 Device Installation

1. Installation Requirements

1) Installation of a single antenna. It is necessary to install it on a carrier and connect it to the integrated navigation. The satellite antenna should face the sky without obstruction from the carrier. The antenna is connected to the PRI interface as shown in Figure 7 for positioning. The installation schematic is depicted in Figure 5.

2) Installation of dual antennas. If the dual antenna orientation function is to be utilized, two antennas (main and secondary antennas) need to be installed on the carrier and connected to the integrated navigation. The satellite antennas should face the sky without obstruction from the carrier. The main antenna is connected to the PRI for positioning, and the secondary antenna is connected to the SEC interface as shown in Figure 7 for orientation. The distance between the two antennas, namely the baseline length, should be greater than 0.5 m. The heights of the dual antennas should be kept as consistent as possible. The installation schematic is shown in Figure 20.

3) Keep away from interference sources such as on-board electrical devices or systems that can generate electromagnetic interference signals.

4) Screw the GNSS antenna onto the strong magnetic suction cup and fix it at the center of the test carrier. Position it at the highest point of the test carrier as much as possible to ensure a good GNSS signal reception.

5) To achieve the best performance, the distance between the GNSS main antenna and the equipment host, especially the horizontal distance, should be minimized as much as possible.

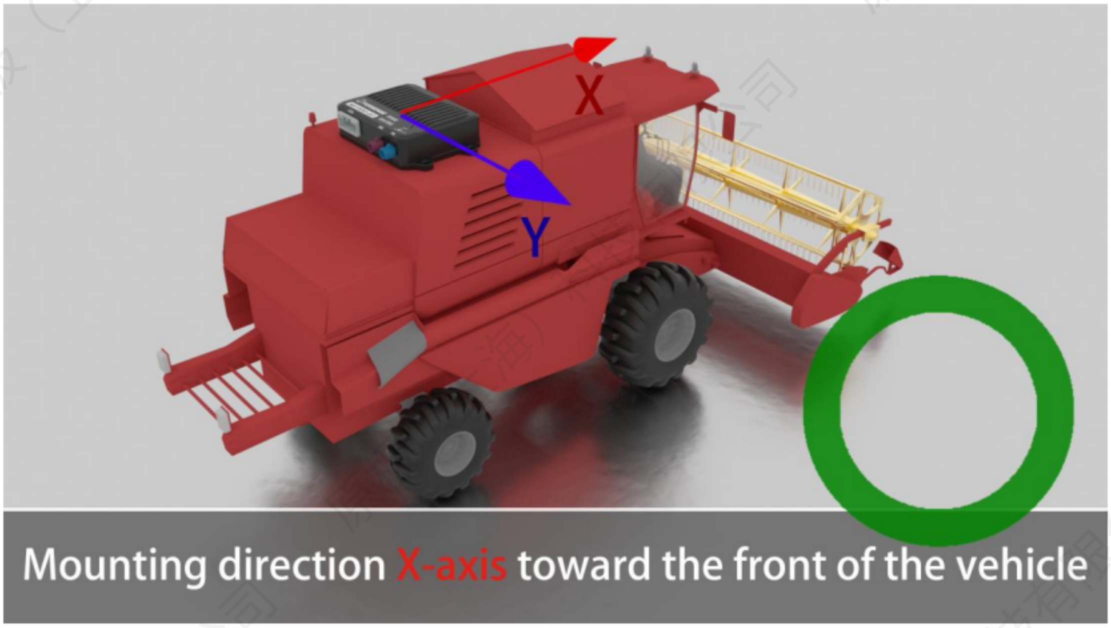
6) It should be firmly fixed on a rigid plane and avoid being installed at positions with significant vibration.

7) The installation orientation of the X-axis should maintain the relationship described below with the direction of the vehicle head.

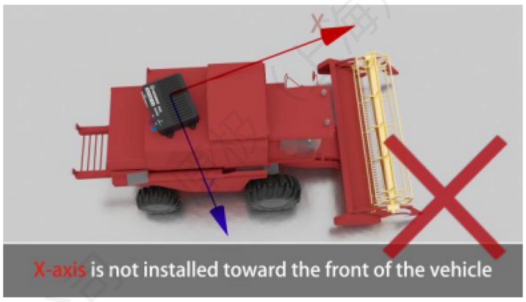
	
Schematic Diagram of Single Antenna Installation	Schematic Diagram of Dual Antenna Installation

The following installation methods are all wrong installation

Fig. 3 Correct Installation Diagram



The following mounting methods are **INCORRECT**:

4.2 RTK Differential Data Import

The user must obtain the GPGGA message of RTK through the RS232-RTK interface, log in to the network CORS account of Chihiro or six Fen and import the differential data, and then make RTK enter the fixed solution state.

4.3 Use Serial Ports to Configure Parameters

4.3.1 Configure Lever Arms

For example, configure the lever arm vector as X=0.5m, Y=-0.6m, Z=-1.0m.

Command: AT+CLUB_VECTOR=0.5,-0.6,-1.0\r\n

Response: GPS_POS_X=0.5,GPS_POS_Y=-0.6,GPS_POS_Z=-1.0/r/n

Note: The lever arm vector is the three dimensional vector (X, Y, Z) representing the position of the RTK main antenna phase center relative to the IMU phase center. (unit: m)

In the Front-Right-Down coordinate system,

the value is positive if the RTK main antenna is in front of the IMU, and it is negative otherwise;

the value is positive if the RTK main antenna is on the right side of the IMU, and it is negative otherwise;

the value is **Negative** if the RTK main antenna is above the IMU, and it is positive otherwise.

The coordinate system diagram is shown in the following figure: (The sticker needs to face upwards, and the IMU needs to be reconfigured in terms of mounting orientation by referring to the section 7.2.2 if it is not installed in accordance with the following.)

Fig 4 Schematic diagram of coordinate system

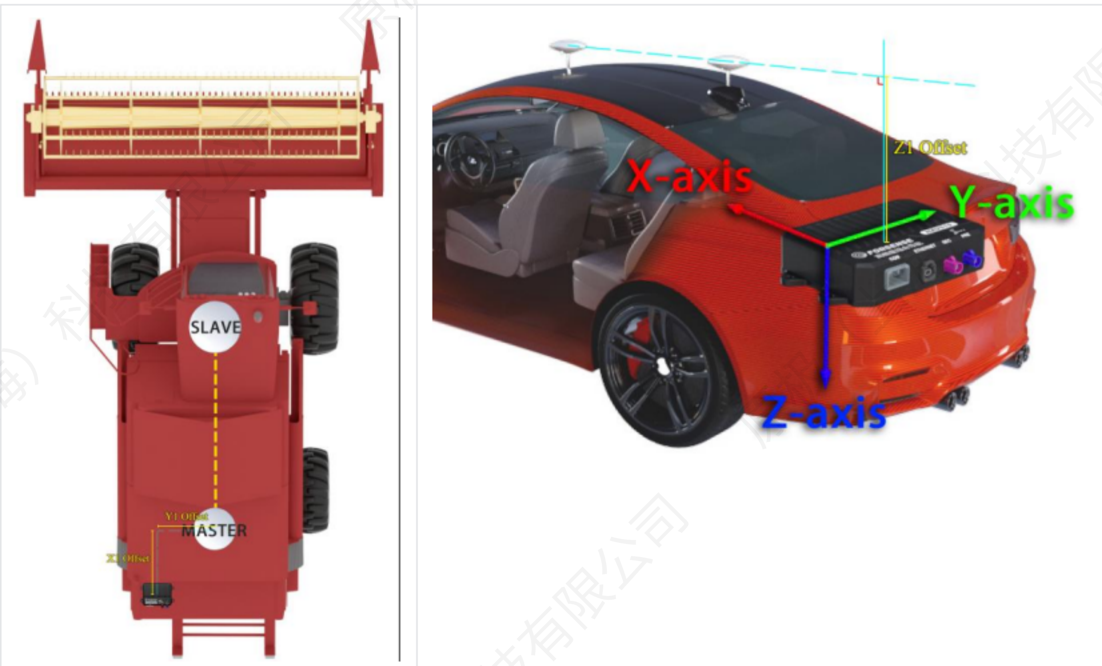


Fig 5 Diagram of the Antenna Arm

4.3.2 Configure Lever Arms

For example, configure the lever arm vector as X=0.5m, Y=-0.6m, Z=1.0m.

Command: AT+OD_VECTOR=0.5,-0.6,1.0\r\n

Response: OD_POS_X=0.5, OD_POS_Y=-0.6, OD_POS_Z=1.0/r/n

Note: The lever arm vector is the three dimensional vector (X, Y, Z) representing the position of the RTK main antenna phase center relative to the IMU phase center. (unit: m)

In the Front-Right-Down coordinate system,

the value is positive if the RTK main antenna is in front of the IMU, and it is negative otherwise;

the value is positive if the RTK main antenna is on the right side of the IMU, and it is negative otherwise;

the value is **Positive** if the RTK main antenna is above the IMU, and it is positive otherwise.

The coordinate system diagram is shown in the following figure: (The sticker needs to face upwards, and the IMU needs to be reconfigured in terms of mounting orientation by referring to the section 7.2.2 if it is not installed in accordance with the following.)

Figure 6 Coordinate System Diagram

4.3.3 Configure Vehicle Wheelbase

If the vehicle wheelbase is configured to be 2m, the configuration command is:

AT+WHEEL_BASE=2\r\n

Note: The command needs to be saved after configuration.

4.3.4 Configure RTK Dual Antenna Mounting Angle

If the RTK dual antenna mounting angle is configured to be 0 °, the configuration command is:

Command: AT+RTK_ANGLE=0\r\n

Response: ANGLE=0\r\n

Explanation: The mounting angle is the angle between the ray extending from the main antenna to the secondary antenna and the direction of the vehicle's front. Clockwise motion is considered positive, while counterclockwise motion is negative. The angle input range is -180° to 180°.

Note: After saving the configuration commands, it is crucial to power off and restart. The distance between the two antennas should be more than 50cm.

4.3.5 Enable Dual Antenna Fusion

Note: By default, dual antenna fusion is in the off state and is only used for rapid initialization when stationary. If dual antenna heading is to be used throughout the entire process, enable dual antenna fusion by following the instructions below. 1 indicates enabling dual antenna fusion, while other parameters indicate disabling it. Note that the installation error of the dual antennas needs to be calibrated synchronously; otherwise, there will be a fixed error between the vehicle body heading.
Command: AT+DRTK_USE=1\r\n\$

Response: OK

Note: After saving the configuration command, a power-off and restart is required for it to take effect. The separation distance between the dual antennas needs to be greater than 50 cm.

4.3.6 Calibrate Dual Antenna

The dual antenna calibration process is as follows:

1. Connect serial ports and enter AT+SETNO\r\n to stop all data output;

2. Send the command AT+RTK_BIAS_EST=1\r\n to start the calibration process;

3. Vehicles travel straight at the speed of no less than 3km/h, and the calibration is successfully done when the serial port sends the statement RTK_BIASE_CORRECT_DONE;

4. Send the command AT+CONFIG\r\n to view the current configuration, and send the statement RTK_BIAS_FLAG_AND_VALUE=99,XX (°) to confirm the calibration angle.

The installation angle is normally less than 1°; otherwise, reinstallation is a must;

5. Enter AT+SAVE\r\n to save the result and re-power on.

4.3.7 Configure the INS to Output Projection Points of Position and Velocity

If the INS is configured to output projection points, the configuration command is:

Command: AT+PROJ_VECTOR=1.0,2.0,3.0\r\n

Response: PROJ_VECTOR_X=1.0, PROJ_VECTOR_Y=2.0, PROJ_VECTOR_Z=3.0/r/n

Explanation: The INS default output is projection points of the antenna phase center. To output results from other positions, you need to configure the corresponding lever arm vectors. The configuration method is the same as 4.3.1 lever arm configuration.

4.4 Save Parameters

After all commands are configured, send the command AT+SAVE\r\n to save parameters.

5. Coordinate System Definition

Fig. 7 Coordinate System Diagram

This product is equipped with the Front-Right-Down (FRD) coordinate system, and the range of Euler angles is as follows:

Rotate around Z-axis direction: Yaw angle range (0° ~ 360°);

Rotate around X-axis direction: Roll angle range (-180° ~ 180°);

Rotate around Y-axis: Pitch angle range (-90° ~ 90°).

6. Integrated Navigation Output Protocol

6.1 Binary Protocol- AG Data Stream

Notes:

- CRC check starts from the frame header covering all bytes of the frame except the CRC check bit itself, seeing the appendix for calculation methods and routines.
- The frame length is the total number of data bytes excluding frame header, frame ID, frame length and parity bits.
- In little-endian mode where the low bytes are sent first.

Table 7 Binary Protocol - AG Data Stream

Content	Type	Relative position
Frame head 1:0 xAA	Uint8	0
Frame head 2:0x55	Uint8	1
Frame ID: 0x0156	Uint16	2
Frame length: 0x0032	Uint16	4
Seconds in GPS cycle (ms)	Uint32	6
Roll Angle (degrees)	Float	10
Pitch Angle (degree)	Float	14
Heading Angle (degrees)	Float	18
Reserved	Float	22
Triaxial angular speed (deg/s, front bottom right)	Float*3	26
Triaxial acceleration (g, front bottom right)	Float*3	38
RTK positioning state	Uint8	50
Attitude significant bit Bit0:1 indicates that attitude is valid, 0 is not	Uint8	51

Bit1:1 means the front wheel Angle is valid and 0 is not		
Status bits: bit0:1 indicates that the RTK data is valid and 0 indicates that it is invalid Bit1:1 indicates that the PPS signal is valid and 0 indicates that it is invalid Bit2:1 means the integrated navigation is initialized and 0 means it is not initialized (Single antenna initialization condition: RTK positioning state 4+ speed reaches 0.5m/S or more + straight line driving with acceleration and deceleration,Dual antenna initialization condition: RTK positioning state 4+RTK orientation state 50) Bit3:1 indicates that the front wheel Angle is valid and 0 is not Bit4:1 indicates that the combined navigation has converged and 0 has not Bit5:1 means front wheel gyro data is valid, 0 is not (no external gyro can be ignored) Bit6:1 means the steering wheel motor data is valid and 0 is not Bit7 & Bit8: 01 for vehicle forward (Bit7=1, Bit8=0) 10 means the vehicle is moving backwards (Bit7=0, Bit8=1) 00 means invalid (Bit7=0, Bit8=0)	Uint32	52
CRC check	Uint32	56

6.2 Binary Protocol - Combined Navigation Data Stream

Note:
•CRC check is calculated from the frame header, excluding the CRC check byte itself, for the CRC check of all bytes in the frame, including the calculation method and procedure in the appendix.
•Frame length is the total number of data bytes excluding the frame header, frame ID, frame length, and CRC check byte.
•Little-endian mode, low byte transmitted first.
•Configuration method is described in Section 7.1.9, command: AT+SETNAV\r\n
•Cannot output NMEA data stream simultaneously with binary protocol data stream. Outputting binary protocol data stream disables output of NMEA data stream. Before switching to the binary protocol data stream, the current data stream output must be stopped by following the instruction AT+SETNO\r\nin Section 7.1.11.

Content	Type	Relative position
Frame head 1:0 xAA	Uint8	0
Frame head 2:0x55	Uint8	1
Frame ID: 0x0166	Uint16	2
Frame length: 0x005E	Uint16	4
GPS seconds per week (ms)	Uint32	6
GPS week count	Uint16	10
Latitude (degrees x 10000000)	Int32	12
Longitude (degrees x 10000000)	Int32	16
Height (mm)	Int32	20
Northbound speed (m/s)	Float	24
Eastbound velocity (m/s)	Float	28
Ground velocity (m/s)	Float	32
Roll Angle (degree)	Float	36

Pitch Angle (degrees)	Float	40
Heading Angle (degrees)	Float	44
Single antenna case: AHRS course (degree) (no reference value) Dual antenna case: RTK dual antenna course (degree) With G200 front gyro connected: Front gyro Z-axis angular speed (degrees per second)	Float	48
Reserved (displayed as front wheel Angle when G200 is connected)	Float	52
Accelerometer X axis (g)	Float	56
Accelerometer Y-axis (g)	Float	60
Accelerometer Z-axis (g)	Float	64
Gyroscope X-axis (deg/s)	Float	68
Gyro Y-axis (deg/s)	Float	72
Gyroscope Z-axis (deg/s)	Float	76
IMU temperature (° C)	Float	80
RTK positioning state (same as GGA positioning state) 0: unpositioned 1: single point positioning 2: pseudo-distance differential positioning 4: fixed solution 5: floating point solution	Uint8	84
Number of satellites	Uint8	85
Differential time delay	Uint8	86
Dual antenna directional state 50 indicates directional Others indicate unoriented	Uint8	87
Position precision factor (cm)	Uint16	88

Compositional navigation is valid after initialization		
Status bits: bit0:1 indicates that the RTK data is valid and 0 indicates that it is invalid Bit1:1 indicates that the PPS signal is valid and 0 indicates that it is invalid Bit2:1 means the integrated navigation is initialized and 0 means it is not initialized (Single antenna initialization condition: RTK positioning state 4+ speed reaches 0.5m/S or more + straight line driving with acceleration and deceleration, Dual antenna initialization condition: RTK positioning state 4+RTK orientation state 50) Bit3:1 indicates that the front wheel Angle is valid and 0 is not Bit4:1 indicates that the combined navigation has converged and 0 has not Bit5:1 means front wheel gyro data is valid, 0 is not (no external gyro can be ignored) Bit6:1 means the steering wheel motor data is valid and 0 is not Bit7 & Bit8: 01 for vehicle forward (Bit7=1, Bit8=0) 10 means the vehicle is moving backwards (Bit7=0, Bit8=1) 00 means invalid (Bit7=0, Bit8=0)	Uint16	90
Reserved 1	Uint32	92

Reserved 2	Uint32	96
CRC check	Uint32	100

6.3 NMEA Protocol

- Support the output of combined data in nmea format.
- The NMEA protocol and the binary data stream cannot be output at the same time, and before switching the data stream, stop the output of current data stream by following the commands in 7.2.2.
- The following statements are supported. See section 7.3 for configuration methods.

GPGGA

GPRMC

GPHDT (Heading information)

GPVTG (Ground velocity)

GPZDA (UTC time and date)

GPATT (Customized message)

The GPATT formats are shown in the following table

Name	*Unit*	*Format*	*Example*	*Description*
Sentence Identifier		String	$GNATT	
Time		hhmmss.sss	170834.000	17:08:34 UTC
Status		Character	1	0: invalid 1: valid
Roll Angle	degree	3 decimal places	-4.891	range ±90, right side down defined as positive
Indicator for roll		character	R	Roll indicator
Pitch Angle	degree	3 decimal places	3.122	range ±90, head up defined as positive
Indicator for Pitch		character	P	Pitch indicator
Heading Angle	degree	3 decimal places	124.005	range 0~360, to true North, counter clockwise defined as positive
Roll Angle uncertainty	degree	3 decimal places	0.432	range 0~360
Pitch Angle uncertainty	degree	3 decimal places	0.811	range 0~360
Heading Angle uncertainty	degree	3 decimal places	1.202	range 0~360
Checksum		Hex	*68	Used by program to check for transmission errors

6.4 CAN Protocol

Table 9 CAN Extended Frame Format 0x19FF CC9A (unit: deg*100, int16)

Extended Frame ID	1	2	3	4	5	6	7	8
0x19FF CE9A	Roll angle		Pitch angle		Heading angle		Track angle	

Table 10 CAN Extended Frame Format 0x19FF CD9A (unit: deg/s*52.0127, int16)

Note: "Reserved" means If connected to G200, the output will be the angle of the front wheels.

Extended Frame ID	1	2	3	4	5	6	7	8
0x19FF CD9A	gyro_x		gyro_y		gyro_z		Reserved	

Table 11 CAN Extended Frame Format 0x19FF CE9A (unit: g*3276.8, int16)

Status bits:

Bit0: RTK board data valid flag bit. 1 means valid and 0 means invalid.

Bit1: PPS valid flag bit. 1 means valid and 0 means invalid.

Bit2: Front wheel angle valid flag bit. 1 means valid and 0 means invalid

Bit3: Front wheel gyro valid flag bit. 1 means valid and 0 means invalid.

Bit4: RTK positioning fixed solution flag bit. 1 means fixed solution and 0 means unfixed solution.

Bit5: RTK directional fixed solution flag bit. 1 means fixed solution and 0 means unfixed solution.

Bit6: Heading initialization flag bit. 1 means initialized and 0 means uninitialized (only for the gyro of single antenna vehicle).

Extended Frame ID	1	2	3	4	5	6	7	8
0x19FF CE9A	accel_x		accel_y		accel_z		Status bits	

Table 12 CAN Extended Frame Format 0x19FFCE9B

Note: Speed direction (1: forward, -1: backward)

Extended Frame ID	1	2	3	4	5	6	7	8
0X19FFCE9B	Seconds in				Estimated antenna		Speed	Reserved

	GPS week	mounting deviation angle	direction	

Table 13 CAN Extended Frame Format 0X19CCFF9A (unit: deg / 1*107 , int32)

The equipment adopts the WGS84 coordinate system. By default, the output of the latitude and longitude is the position of the phase center of the main antenna. In the case of a configured projection point, the output is the position of the projection point.

The latitude is defined as 0° at the equator. “Latitude” greater than 0° indicates the Northern Hemisphere, and conversely, it indicates the Southern Hemisphere.

The longitude is defined as 0° at the Prime Meridian. “Longitude” greater than 0° indicates the Eastern Hemisphere, and conversely, it indicates the Western Hemisphere.

Extended Frame ID	1	2	3	4	5	6	7	8
0X19CCFF9A	Longitude				Latitude			

Table 14 CAN Extended Frame Format

(The elevation refers to the ellipsoidal height / 1*107 , int32)

Extended Frame ID	1	2	3	4	5	6	7	8
0X19CCFF9B	Elevation				Seconds in GPS week			

6.5 RTK Positioning Status Table

Binary	ASCII	Description
0	NONE	No solution
1	FIXEDPOS	The position is specified by the FIX POSITION command
2	FIXEDHEIGHT	Unsupported
8	DOPPLER_VELOCITY	Velocity is derived instantly from Doppler information
16	SINGLE	Single point positioning
17	PSRDIFF	Pseudo-range differential positioning
18	WAAS	SBAS positioning
32	L1_FLOAT	L1 floating solution
33	IONOFREE_FLOAT	Floating solution for mitigating ionospheric effects
34	NARROW_FLOAT	Narrow lane floating solution
48	L1_INT	L1 fixed solution
49	WIDE_INT	Wide lane fixed solution
50	NARROW_INT	Narrow lane fixed solution

7. Parameter Configuration

7.1 Use Serial Ports to Configure Parameters

7.1.1 Configure the main antenna lever arm

For example, configure the lever arm vector as X=0.5m, Y=-0.6m, Z=-1.0m.

Command: AT+CLUB_VECTOR=0.5,-0.6,-1.0\r\n

Response: GPS_POS_X=0.5,GPS_POS_Y=-0.6,GPS_POS_Z=-1.0/r/n

Note: The lever arm vector is the three dimensional vector (X, Y, Z) representing the position of the RTK main antenna phase center relative to the IMU phase center. (unit: m)

In the Front-Right-Down coordinate system,

the value is positive if the RTK main antenna is in front of the IMU, and it is negative otherwise;

the value is positive if the RTK main antenna is on the right side of the IMU, and it is negative otherwise;

the value is negative if the RTK main antenna is above the IMU, and it is positive otherwise.

The coordinate system diagram is shown in the following figure: (The sticker needs to face upwards, and the IMU needs to be reconfigured in terms of mounting orientation by referring to the section 8.2.2 if it is not installed in accordance with the following.)

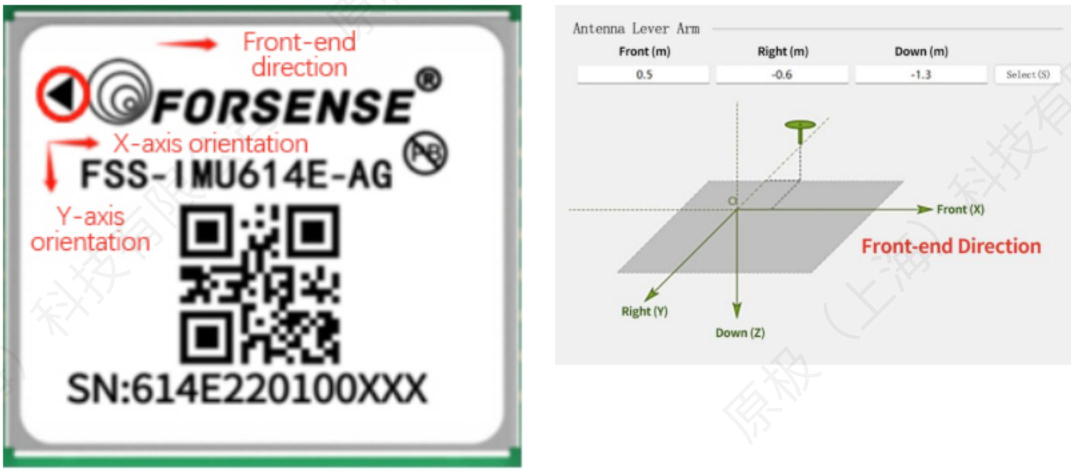


Fig. 5 Antenna Lever Arm Diagram

7.1.2 Configure Lever Arms

For example, configure the lever arm vector as X=0.5m, Y=-0.6m, Z=1.0m.
Command: AT+OD_VECTOR=0.5,-0.6,1.0\r\n
Response: OD_POS_X=0.5, OD_POS_Y=-0.6, OD_POS_Z=1.0/r/n

Note: The lever arm vector is the three dimensional vector (X, Y, Z) representing the position of the RTK main antenna phase center relative to the IMU phase center. (unit: m)

In the Front-Right-Down coordinate system,
the value is positive if the RTK main antenna is in front of the IMU, and it is negative otherwise;
the value is positive if the RTK main antenna is on the right side of the IMU, and it is negative otherwise;
the value is **Positive** if the RTK main antenna is above the IMU, and it is positive otherwise.

The coordinate system diagram is shown in the following figure: (The sticker needs to face upwards, and the IMU needs to be reconfigured in terms of mounting orientation by referring to the section 7.2.2 if it is not installed in accordance with the following.)

Figure 6 Coordinate System Diagram

7.1.3 Configure Vehicle Wheelbase

If the vehicle wheelbase is configured to be 2m, the configuration command is:

AT+WHEEL_BASE=2\r\n

Note: The command needs to be saved after configuration.

7.1.4 Configure RTK Dual Antenna Mounting Angle

If the RTK dual antenna mounting angle is configured to be 0 °, the configuration command is:

Command: AT+RTK_ANGLE=0\r\n

Response: ANGLE=0\r\n

Explanation: The mounting angle is the angle between the ray extending from the main antenna to the secondary antenna and the direction of the vehicle's front. Clockwise motion is considered positive, while counterclockwise motion is negative. The angle input range is -180° to 180°.

Note: After saving the configuration commands, it is crucial to power off and restart. The distance between the two antennas should be more than 50cm.

7.1.5 Enable Dual Antenna Fusion

Note: By default, dual antenna fusion is in the off state and is only used for rapid initialization when stationary. If dual antenna heading is to be used throughout the entire process, enable dual antenna fusion by following the instructions below. 1 indicates enabling dual antenna fusion, while other parameters indicate disabling it. Note that the installation error of the dual antennas needs to be calibrated synchronously; otherwise, there will be a fixed error between the vehicle body heading.
Command: AT+DRTK_USE=1\r\n\$

Response: OK

Note: After saving the configuration command, a power-off and restart is required for it to take effect. The separation distance between the dual antennas needs to be greater than 50 cm.

7.1.6 Calibrate Dual Antenna

The dual antenna calibration process is as follows:

2. Connect serial ports and enter AT+SETNO\r\n to stop all data output;

2. Send the command AT+RTK_BIAS_EST=1\r\n to start the calibration process;

3. Vehicles travel straight at the speed of no less than 3km/h, and the calibration is successfully done when the serial port sends the statement RTK_BIASE_CORRECT_DONE;

4. Send the command AT+CONFIG\r\n to view the current configuration, and send the

statement RTK_BIAS_FLAG_AND_VALUE=99,XX (°) to confirm the calibration angle. The installation angle is normally less than 1°; otherwise, reinstallation is a must;

5. Enter AT+SAVE\r\n to save the result and re-power on.

7.1.7 Configure the INS to Output Projection Points of Position and Velocity

If the INS is configured to output projection points, the configuration command is:

Command: AT+PROJ_VECTOR=1.0,2.0,3.0\r\n

Response: PROJ_VECTOR_X=1.0, PROJ_VECTOR_Y=2.0, PROJ_VECTOR_Z=3.0/r/n

Explanation: The INS default output is projection points of the antenna phase center. To output results from other positions, you need to configure the corresponding lever arm vectors. The configuration method is the same as 7.1 lever arm configuration.

7.1.8 Configure the Output of Binary - AG Data Stream

If the AG data stream output is configured, neither the 7.2 combined navigation data stream nor the 7.3 NMEA protocol will be output.

If it is necessary to switch to the output of the 7.2 combined navigation data stream or the 7.3 NMEA protocol, the current data stream output needs to be stopped first by following the instruction 8.1.11.

Instruction: AT+SETAG$\r\n$

Response: OK$\r\n$

If the output is configured to stop

Instruction: AT+SETNO$\r\n$

Response: OK$\r\n $

7.1.9 Configuring the Output binary Integrated Navigation Data Flow

If the output of the combined navigation data stream is configured, then neither the 7.1 AG data stream nor the 7.3 NMEA protocol will be output.

If it is necessary to switch to the output of the 7.2 combined navigation data stream or the 7.3 NMEA protocol, the current data stream output needs to be stopped first by following the instruction 7.1.11.

The configuration instructions are as follows:

Instruction: AT+SETNAV\r\n

Answer: OK\r\n

Stop output if configured

Instruction: AT+SETNO\r\n

Answer: OK\r\n

7.1.10 Configure the output NMEA format data stream

If NEMA statement output is configured, the 7.2 Integrated Navigation data flow is not output

To switch to 7.2 Integrated Navigation Data Flow Output, it is required to stop the current data stream output by following the instruction 8.1.11 first.

The configuration instructions are as follows:

GPGGA

Example: Output GPGGA statement AT 5Hz: AT+GPGGA=5\r\n

Answer: OK\r\n

GPRMC SS-NAV680-AG User manual

Example: Output GPRMC statement AT 1Hz: AT+GPRMC=1\r\n

Answer: OK\r\n

GPHDT (Heading information)

Example: Output GPHDT statement AT 1Hz frequency: AT+GPHDT= 1\r\n

Answer: OK\r\n

GPVTG (Ground Speed information)

Example: Output GPVTG AT 1Hz frequency statement: AT+GPVTG= 1\r\n

Answer: OK\r\n

GPZDA (UTC time and date)

Example: Output GPZDA statement AT 1Hz: AT+GPZDA= 1\r\n

Answer: OK\r\n

GPATT (Custom message)

Example: Output GPATT statement AT 1Hz frequency: AT+GPATT= 1\r\n

Answer: OK\r\n

Stop output if configured

Instruction: AT+SETNO\r\n

Answer: OK\r\n

7.1.11 Configure the current data stream to stop output

Instruction: AT+SETNO\r\n

Answer: OK\r\n

7.1.12 Configure Data Output Frequency

If the configuration data output frequency is 10Hz, the configuration command is:

Command: AT+OUTRATE=10\r\n

Response: OK\r\n

7.1.13 Configure Baud Rate

Only support the configured baud rate 115200 and 230400, and the default is 115200.

If the IMU serial baud rate is configured to 230400, the configuration command is:

Command: AT+BAUD=230400\r\n

Response: BAUD=230400\r\n

Note: It is essential to power off and restart after configuring and saving commands.

7.1.14 Configure the CAN Baud rate

Only the configuration baud rates of 250K, 500K, and 1M are supported, with the default baud rate being 500K.

Example: If the CAN baud rate is configured as 500K, the configuration instruction is:

Instruction: AT+CAN_BAUD=500$\r\n$

Response: OK

Note: The configuration instruction takes effect after saving and requires a power-off and restart.

7.1.15 Configuring CAN Output Frequency

If the CAN data output frequency is configured as 10Hz, the configuration instruction is:
Instruction: AT+CAN_ODR=10
Response: OK$\r\n$
Remarks: The maximum supported frequency is 100Hz. The modification takes effect immediately, and it remains effective after saving and powering off and restarting.

7.1.16 Query All Configuration Information

If you query all configured information, the configuration command is:

AT+CONFIG\r\n

7.1.17 Query Version Number

AT+VERSION\r\n

7.1.18 Configuring Carrier XYZ Velocity

After configuration, the northward, eastward, and downward velocities in the combined data stream will be switched to the XYZ-axis velocities of the carrier upon the completion of initialization.

"1" enables the output of the carrier velocity, while other parameters disable it.

Instruction: AT+NAV_OUTPUT_XYZ=1$\r\n$

Response: OK

7.1.19 Deduction of Gravity Acceleration from Acceleration Data

1. Command for removing the gravity acceleration: AT+DEDUCTIONG=1$\r\n$

The return of OK after sending indicates successful sending and immediate effect.

It is necessary to send AT+SAVE$\r\n$ to save the configuration; otherwise, it will be invalid after restart.

2. Command for restoring the gravity acceleration: AT+DEDUCTIONG=0$\r\n$

The return of OK after sending indicates successful sending and immediate effect.

It is necessary to send AT+SAVE$\r\n$ to save the configuration; otherwise, it will be invalid after restart.

7.1.20 Configure Mounting Rotation Angle

Currently only the following rotation angles are supported

X-axis rotation 180°

Z-axis rotation 90° 180° 270°

The configuration commands are as follows

If the mounting rotation angle is 180° on the X-axis, the configuration command is:

Command: AT+INSTALL_ANGLE=180,0,0\r\n

Response: INST_ANGLE_X=180.000,INST_ANGLE_Y=0.000,INST_ANGLE_Z=0.000

If the mounting rotation angle is 180° on the Z-axis, the configuration command is:

Command: AT+INSTALL_ANGLE=0,0,180\r\n

Response: INST_ANGLE_X=0.000,INST_ANGLE_Y=0.000,INST_ANGLE_Z=180.000

7.1.21 Turn on the Slope Mode

For scenarios where there is always a slope of more than 5°, you can turn on the slope mode for compensation.

The configuration command is:

AT+SLOPE=0.05\r\n Agricultural scenarios with slopes can be configured with this statement, which defaults to 0.

Then enter AT+SAVE\r\n to save the configuration and the slope mode will work after reboot.

7.1.22 Save Parameters

The above parameters involve lever arm, data stream, output frequency, etc. After the configuration, you need to enter the SAVE command and restart the software.

Command: AT+SAVE\r\n

Response: OK\r\n

7.2 Use CAN Interface to Configure Parameters

7.2.1 Query Version Number

Example:

ID = 0x19FFF326

Data = 0x9A 0x07 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF

Response (version number V1.21.2.2)

ID = 0x19FFF29A

Data = 0x26 0xC7 0x01 0x15 0x02 0x02 0xFF 0xFF

7.2.2 Configure Wheelbase and Lever Arm

Example

ID = 0X0DFFC126

Data = wheelbase 2 bytes + lever arm 2 bytes*3 (unit: cm)

Response

ID = 0X19FFC09A

Data = wheelbase 2 bytes + lever arm 2 bytes*3 (unit: cm)

7.2.3 Configure Dual Antenna Mounting Angle

ID = 0X0DFFC326

Data = angle (int16, unit: deg*100) + 0xFF*6

7.2.4 Query Wheelbase, Lever Arm and Dual Antenna Mounting Angle

ID = 0X0DFFC226

Data =0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF

Response

ID = 0X19FFC29A

Data = wheelbase 2 bytes + lever arm 2 bytes*3 (unit: cm)

Mounting angle query command:

ID = 0X0DFFC526

Data =0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF

Response

ID = 0X19FFC29A

Data = angle*100 2 bytes

7.2.5 Configure CAN Interface to Output Update Rate and Baud Rate

Baud rates: 1M, 500K, 250K (0x01, 0x02, 0x03 respectively)

Update rates: 20Hz, 50Hz, 100Hz (0x14, 0x32, 0x64 respectively).

After the configuration, restart the software.

ID = 0x19FFF326

Data = 0x9A 0x4A Baud Rate Update Rate 0xFE 0xFB 0xF9 0xFF

Response

ID = 0x19FFF59A

Data = 0xFF 0x4A Baud Rate Update Rate 0xFF 0xFF 0xFF 0xFF

7.2.6 Query Update Rate and Baud Rate

ID = 0x19FFF426

Data =0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF

Response

ID = 0x19FFF49A

Data = 0xFF 0x4A Baud Rate Update Rate 0XFF 0XFF 0XFF 0xFF

7.2.7 Save Parameter Instructions

ID = 0X0DFFCF26

Data =0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF

Response

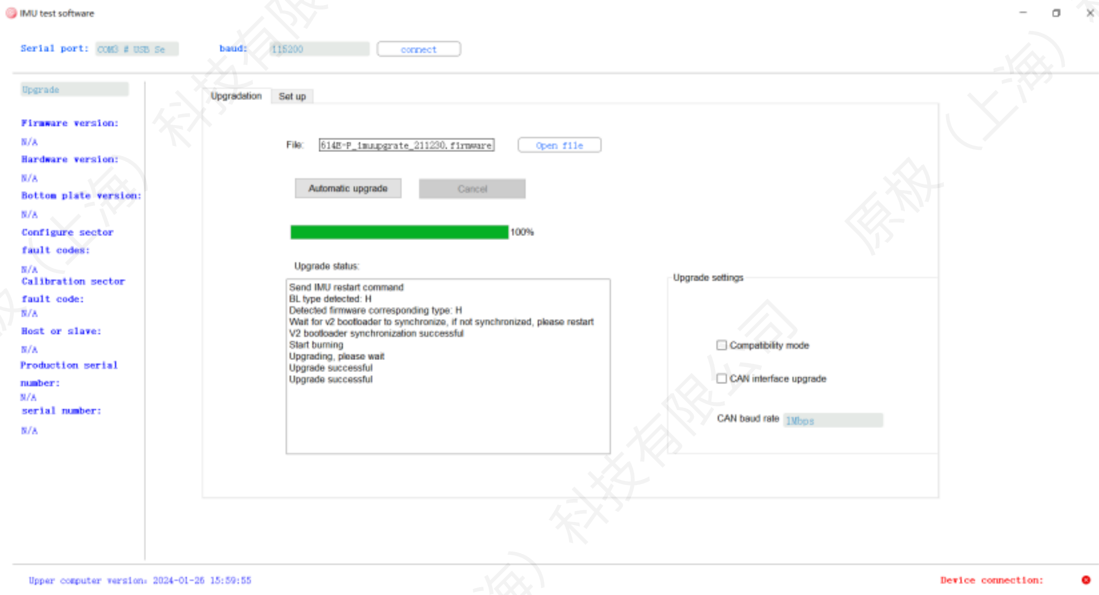
ID = 0X19FFCF9A

8. Firmware Upgrading

8.1 Firmware Upgrading via the PC Software

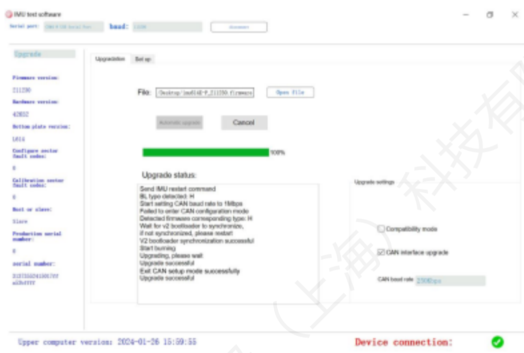
8.1.1 RS232

Use FSS IMUs to test the PC software - Select firmware upgrade - Open the firmware - Select the CAN interface upgrade - Set up the baud rate of the upgraded firmware - Click Auto Upgrade.



8.1.2 Firmware Upgrading via the CAN Interface

Use FSS IMUs to test the PC software - Select firmware upgrade - Open the firmware - Select the CAN interface upgrade - Set up the baud rate of the upgraded firmware - Click Auto Upgrade.



8.2 Serial OTA

The upgrade process is divided into the following steps:

Step 1: Send upgrade command

Send upgrade command to the IMU module. After that, the device will record the upgrade flag bit in a certain area of FLASH and then perform a soft reboot. The module will enter the BOOTLOADER.

The PC software sends the upgrade command as follows:

```
cmd_bl[34] ={0x55,0xaa,0xbb,0x88,0x18,0x00,0x00,0x00,0xc8,0x42,0x00,0x00,
   0x48,0x43,0x2c,0x01,0x00,0x00,0x90,0x01,0x00,0x00,0xf4,0x01,0x00,0x00,
0x58,0x02,0x00,0x00,0x40,0x97,0x46,0x6a};
```

Then the imu module will reboot and enter the bootloader

Step 2: Send the string **HC32MCU_FORSENSE**

After entering the bootloader, the IMU module will actively send the string HC32_UPLOADER. During this period, the PC software needs to send the string HC32MCU_FORSENSE. If the module receives this string, it will not jump to APP, but stay in a pending upgrade state in the bootloader and stop actively sending any messages.

//After sending the upgrade command, send the following string to ensure that the imu module will not jump to the APP area, but stay in a pending upgrade state

```
QString str = "HC32MCU_FORSENSE" ;
for(int i=0;i<10;i++)
{
  _port_device->write(str.toLatin1()); sleep_ms(50);
}
```

Step 3: Send synchronization command

When the IMU module stays in a pending upgrade state in the bootloader, it will not send messages actively, but respond to the commands of the PC software passively. At this time, the PC software enters Send_CMD_LONG(0X21,0,0,0,0,0,0,0,0) and waits for the response from the IMU module. After receiving this command, IMU replies with Send CMD ACK (0x64,0x10,0) immediately. Once the PC software determines that it has received this response, it indicates that the synchronization process is completed.

Synchronization process between the PC software and imu module:

1.The PC software sends the synchronization command with the command code of 0x21, and then waits for the imu module to respond.

Send_CMD_LONG(0x21,0,0,0,0,0,0);

2.After the imu module receives the command code of 0x21, it will send the response data with the command as follows:

Send_CMD_ACK(0x64,0x10,0);

3.The PC software determines whether it has received the response data from the imu. If received, the synchronization process is completed.

Step 4: Send erase command

The PC software sends the erase command Send_CMD_LONG(0x23,0,0,0,0,0,0), and then the IMU module will erase all the content in the APP area and send the final execution result to the PC software. The PC software will decide whether to send erase command again according to the result. Once the erase operation is successful, the APP of IMU module cannot be restored. In addition, the PC software must wait for the response of successful erase before proceeding to the next step, otherwise it may cause the subsequent upgrade failure.

1.The PC software sends the erase command with the command code of 0x23, and then waits patiently for imu to reply with the execution result.

Send_CMD_LONG(0x23,0,0,0,0,0,0);

2.After imu receives the command code of 0x23, it will erase the APP data and reply the final execution result to the PC software.

send_CMD_ACK(0x64,0x10,0); //Indicating that the erase process is successful

send_CMD_ACK(0x64,0x11,0); //Indicating that the erase process fails and needs to operate again

Note: The PC software must receive a successful erase response before proceeding to the next step.

Step 5: Send upgrade packets

After a successful erase operation, it will enter the most important part of sending firmware data by using the function Send_Upload_Data. The PC software will divide the firmware data into packets, and each packet has a fixed size of 64 bytes. If the last packet contains fewer than 64 bytes, it will be sent with the actual number of bytes. Each data frame contains the valid data length and the offset address of this packet within the entire firmware. Every time the PC software sends a single data frame, it must wait for the response from the IMU module and ensure that IMU has successfully obtained this frame before sending the next. After the IMU module successfully receives the data packet from the PC software, it will send the response data and write it to the specified Flash address according to the offset address.

If the writing process fails, it will send failure command; otherwise it will send nothing.

Example: send upgrade file uint8_t Upgrade_Data[1000] of 1000 bytes to imu module

//Send the 1st packet:

1.1 Send 0~63 bytes to imu module by using the function

Send_Upload_Data（0x27,0,0,0x40,Upgrade_Data);

//The first parameter 0x27 of the above function is a fixed value, the second parameter 0 is a fixed value, the third parameter 0 is the offset address, the fourth parameter 0x40 is the valid byte length, and the fifth parameter is the first address of the sent data.

1.2 After imu successfully receives the data, it will send the response data and write it to the specified flash address according to the offset address

send_CMD_ACK(0x753D,0x00,0);

//The first parameter of the above function 0x753D is a fixed value, the second parameter 0 is a fixed value, and the third parameter is the offset address.

The imu module will send the failure command if it fails to write data to flash send_CMD_ACK(0x64,0x11,0);

//Send the 2nd packet

2.1 After the PC software receives the response data from imu, it will send the 2nd data packet Send_Upload_Data （0x27,0,0x40,0x40,Upgrade_Data+0x40);

2.2 After the imu successfully receives the 2nd data packet, it will send the response: send_CMD_ACK(0x753D,0x00,0x40);

2.3 The imu module will send the failure command if it fails to write data to flash send_CMD_ACK(0x64,0x11,0);

//Send the 3rd packet:

3.1 After the PC software receives the response data from imu, iit will send the 3rd data packet （0x27,0,0x80,0x40,Upgrade_Data+0x80);

3.2 After the imu successfully receives the 3rd data packet, it will send the response: send_CMD_ACK(0x753D,0x00,0x80);

3.3 The imu module will send the failure command if it fails to write data to flash send_CMD_ACK(0x64,0x11,0);

…………………………

//Send the 15th packet:

15.1 After the PC software receives the response data from imu, it will send the 15th data packet Send_Upload_Data （0x27,0,0x380,0x40,Upgrade_Data+0x380);

15.2 After the imu successfully receives the 15th data packet, it will send the response: send_CMD_ACK(0x753D,0x00,0x380);

15.3 The imu module will send the failure command if it fails to write data to flash send_CMD_ACK(0x64,0x11,0);

//Send the 16th packet:

15.1 After the PC software receives the response data from imu, it will send the 16th data packet Send_Upload_Data（0x27,0,0x3C0,0x28,Upgrade_Data+0x3C0);

15.2 After the imu successfully receives the 16th data packet, it will send the response: send_CMD_ACK(0x753D,0x00,0x3C0);

15.3 The imu module will send the failure command if it fails to write data to flash send_CMD_ACK(0x64,0x11,0);

END

Step 6: Acquire CRC check digits

Generally speaking, the command rule for upgrading firmware is imu614e-b#CRC1373387121.firmware, and the CRC string is immediately followed by the calculated CRC checksum value. After the upgrade file is sent, the PC software needs to send a check command to determine if the upgrade file received by the IMU module is correct. After the PC software enters Send_CMD_LONG(0x29,0,0,0,0,0,0), it will acquire the CRC check digits calculated by the IMU module.

If the PC software judges that there is an error in the CRC check digits, it will restart the upgrade process by resuming the 4th step of sending erase command.

The PC software sends the command to acquire the crc check digits, and waits for the imu response

Send_CMD_LONG(0x29,0,0,0,0,0,0);

The imu module responds by sending the crc checksum value data:

send_CMD_ACK(0x753C,0x10,crc32_data);

The crc32_data value is calculated by the imu module itself.

Step 7: Send reboot command

After the PC software judges that the CRC check digits are correct, it will send reboot command, indicating successful upgrading.

After determining that the crc checksum value is correct, send the reboot command:

Send_CMD_LONG(0x30,0,0,0,0,0,0);

Once the firmware upgrading is completed, you can determine whether the upgrading process is successful by reading the version number after power cycling.

Function definitions:

1. The definition of Send_CMD_LONG is as follows:

```
struct  MULTI_LONG_CMD_STRUCT
{
uint8_t header1;
uint8_t header2;
uint16_t id;
uint16_t length;
float param1;
float param2;
uint32_t param3;
uint32_t param4;
int32_t param5;
int32_t param6;
uint32_t check_crc;
}__attribute__((packed));
```

```
void  :Send_CMD_LONG(uint16_t cmd_id,float cm1,float cm2,uint32_t cm3,uint32_t cm4,int32_t cm5,int32_t cm6)
{
    uint8_t check_sum=0;
    struct MULTI_LONG_CMD_STRUCT data_cmd_long __attribute__((packed));
    data_cmd_long.header1=0x55;
    data_cmd_long.header2=0xAA;
    data_cmd_long.id=cmd_id;
    data_cmd_long.length=sizeof(data_cmd_long)-10;
    data_cmd_long.param1=cm1;
    data_cmd_long.param2=cm2;
    data_cmd_long.param3=cm3;
    data_cmd_long.param4=cm4;
    data_cmd_long.param5=cm5;
    data_cmd_long.param6=cm6;
    int len=sizeof(data_cmd_long)-4;
    uint32_t check_crc=1;
    data_cmd_long.check_crc=crc_crc32(check_crc,(uint8_t *)(&data_cmd_long), len);
    send((uint8_t *)(&data_cmd_long),sizeof(data_cmd_long));
}
```

2. The definition of Send_CMD_ACK is as follows:

```
struct  CMD_ACK_STRUCT
{
uint8_t header1;
uint8_t header2;
uint16_t id;
```

```
uint16_t length;
uint32_t command;/*<  Command ID (of acknowledged command).*/
uint32_t result;/*<  Result of command.*/
uint32_t check_crc;
}__attribute__((packed));

void Send_CMD_ACK(uint16_t cmd_id,uint16_t ack_id,uint32_t result)
{
    uint32_t check_crc=0;
    struct CMD_ACK_STRUCT data_cmd_ack __attribute__((packed));
    data_cmd_ack.header1=0xAA;
    data_cmd_ack.header2=0x55;
    data_cmd_ack.id=cmd_id;
    data_cmd_ack.length=sizeof(data_cmd_ack)-10;
    data_cmd_ack.command=ack_id;
    data_cmd_ack.result=result;
    int len=sizeof(data_cmd_ack)-4;
    check_crc=1;
    data_cmd_ack.check_crc=crc_crc32(check_crc,(uint8_t *)(&data_cmd_ack), len);
    Cout((uint8_t *)(&data_cmd_ack),sizeof(data_cmd_ack));
}
```

3. The definition of Send_Upload_Data is as follows:

```
struct  UPLOAD_DATA
{
uint8_t header1;
uint8_t header2;
uint16_t id;
uint16_t length;
uint8_t param[64];
uint32_t offset;
uint16_t size;
uint8_t cmd;
uint32_t check_crc;
}__attribute__((packed));

struct UPLOAD_DATA upload_data;
void Send_Upload_Data(uint8_t cmd_id,uint8_t cmd,uint32_t offset,uint16_t size,uint8_t*  param)
{
    upload_data.header1=0x55;
```

```
    upload_data.header2=0xAA;
    upload_data.id=cmd_id;
    upload_data.length=sizeof(UPLOAD_DATA)-10;
    upload_data.cmd=cmd;
    for(int i=0;i<size;i++)
    upload_data.param[i] = *(param+i);
    upload_data.offset=offset;
    upload_data.size=size;
    int len=sizeof(UPLOAD_DATA)-4;
    uint32_t check_crc=1;
    upload_data.check_crc=crc_crc32(check_crc,(uint8_t *)(&upload_data), len);
    send((uint8_t *)(&upload_data),sizeof(UPLOAD_DATA));
}
```

4. The definition of CRC32 checksum is as follows:

```
static const uint32_t 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, 0x4adfa541, 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,
    0xb7bd5c3b, 0xc0ba6cad, 0xedb88320, 0x9abfb3b6, 0x03b6e20c, 0x74b1d29a,
    0xead54739, 0x9dd277af, 0x04db2615, 0x73dc1683, 0xe3630b12, 0x94643b84,
    0x0d6d6a3e, 0x7a6a5aa8, 0xe40ecf0b, 0x9309ff9d, 0x0a00ae27, 0x7d079eb1,
    0xf00f9344, 0x8708a3d2, 0x1e01f268, 0x6906c2fe, 0xf762575d, 0x806567cb,
    0x196c3671, 0x6e6b06e7, 0xfed41b76, 0x89d32be0, 0x10da7a5a, 0x67dd4acc,
```

9. ROS Driver

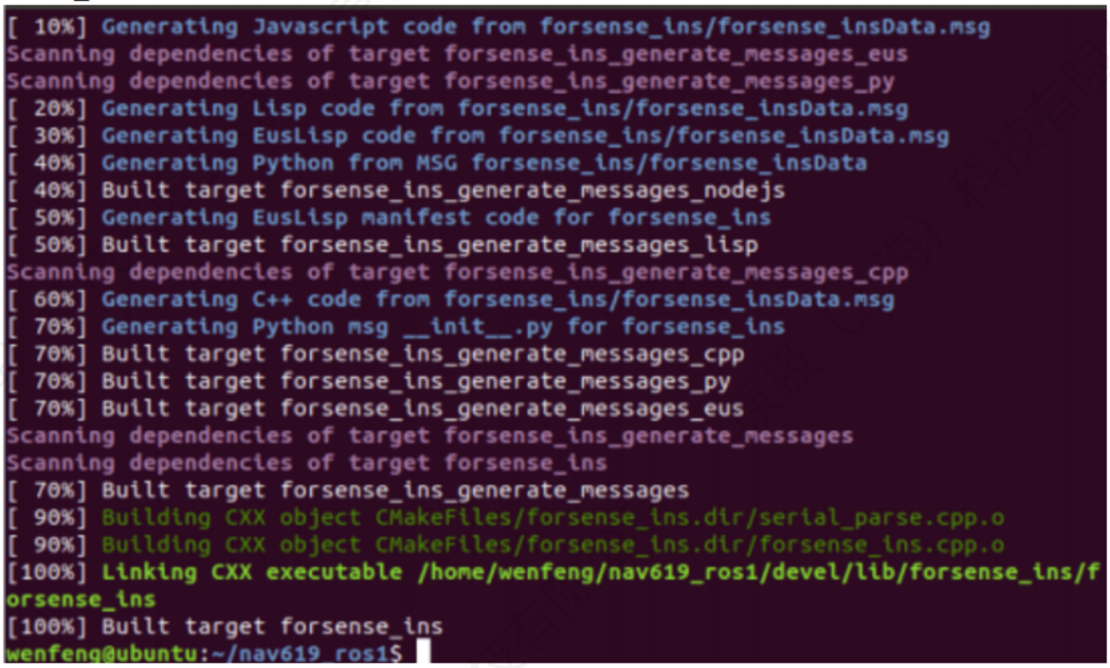
ROS driver download address:
https://data.forsense-imu.com/page/download.html

9.1 Install ROS Serial

Install the ROS serial package. This example program relies on the serial package provided by ROS to achieve serial communication.
First, execute the following command to download and install the serial package:
sudo apt-get install ros -melodic-serial
Second, enter the roscd serial command to find the serial download location. If the installation is successful, the following message will appear:
/opt/ros/melodic/share/serial

9.2 Compile Code

cd FS982_ros/
catkin_make



```
[ 10%] Generating Javascript code from forsense_ins/forsense_insData.msg
Scanning dependencies of target forsense_ins_generate_messages_eus
Scanning dependencies of target forsense_ins_generate_messages_py
[ 20%] Generating Lisp code from forsense_ins/forsense_insData.msg
[ 30%] Generating EusLisp code from forsense_ins/forsense_insData.msg
[ 40%] Generating Python from MSG forsense_ins/forsense_insData
[ 40%] Built target forsense_ins_generate_messages_nodejs
[ 50%] Generating EusLisp manifest code for forsense_ins
[ 50%] Built target forsense_ins_generate_messages_lisp
Scanning dependencies of target forsense_ins_generate_messages_cpp
[ 60%] Generating C++ code from forsense_ins/forsense_insData.msg
[ 70%] Generating Python msg __init__.py for forsense_ins
[ 70%] Built target forsense_ins_generate_messages_cpp
[ 70%] Built target forsense_ins_generate_messages_py
[ 70%] Built target forsense_ins_generate_messages_eus
Scanning dependencies of target forsense_ins_generate_messages
Scanning dependencies of target forsense_ins
[ 70%] Built target forsense_ins_generate_messages
[ 90%] Building CXX object CMakeFiles/forsense_ins.dir/serial_parse.cpp.o
[ 90%] Building CXX object CMakeFiles/forsense_ins.dir/forsense_ins.cpp.o
[100%] Linking CXX executable /home/wenfeng/nav619_ros1/devel/lib/forsense_ins/forsense_ins
[100%] Built target forsense_ins
wenfeng@ubuntu:~/nav619_ros1$
```

Compilation completed.

9.3 Connect IMU to Linux System via USB

Check if it is connected:

lsusb

```
wenfeng@ubuntu:~$ lsusb
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 005: ID 0403:6001 Future Technology Devices International, Ltd FT
232 USB-Serial (UART) IC
Bus 002 Device 004: ID 0e0f:0008 VMware, Inc.
Bus 002 Device 003: ID 0e0f:0002 VMware, Inc. Virtual USB Hub
Bus 002 Device 002: ID 0e0f:0003 VMware, Inc. Virtual Mouse
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
wenfeng@ubuntu:~$
```

Check the USB port number:

ls /dev/ttyU*

```
wenfeng@ubuntu:~$ ls /dev/ttyU*
/dev/ttyUSB0
wenfeng@ubuntu:~$
```

Configure and enable USB to serial port permissions:

sudo chmod 777 /dev/ttyUSB0

9.4 Check Data

Execute roscore to start ROS
Go back to the serial_imu_ws folder and run
source devel/setup.bash
Start rosrun
rosrun forsense_ins forsense_ins

```
wenfeng@ubuntu:~/nav619_ros1$ rosrun forsense_ins forsense_ins
[ INFO] [1695457979.128623440]: /dev/ttyUSB0 is opened.
```

Open a new window
source devel/setup.bash
rostopic list

```
wenfeng@ubuntu:~/nav619_ros1$ rostopic list
/nav619Data
/rosout
/rosout_agg
```

Enter a command to check IMU data

rostopic echo /FS982Data

```
  frame_id: "WGS84"
itow: 549636980
week_num: 2280
lat: 312627286
lon: 1216155393
hgt: 38859
vn: 0.00240602344275
ve: 0.000262897461653
vd: 0.00270945159718
roll: -0.169113516808
pitch: -0.286453634501
yaw: 0.0
rtk_yaw: 359.766906738
wheel_angle: 0.0
imu: [-0.005366197787225246, 0.0035326573997735977, -1.004271149635315, -0.04756
217822432518, -0.11066819727420807, -0.06515973061323166, 35.8017578125]
fix_type: 16
sv_num: 28
diff_age: 0
heading_type: 0
pos_acc: 0
status: 3
---
```

10. Used with G200

1. G200 installation requirements:

A. G200 must be mounted on the front axle of the front wheel (a horizontally rigid position that can rotate with the front wheel);

B. G200 port is installed toward the rear of the car.

2. The CAN interface of G200 is connected to CAN A of the hub, and the power supply is 5V.

Fig. 10 Wiring Diagram

3. Measure the wheelbase of vehicles. Take the 2m wheelbase as an example. The serial port of the host computer connects to the serial port assistant, sends AT+WHEEL_BASE=2, receives the response WHEEL_BASE=2, and then enters AT+SAVE. If it receives the response OK, the device needs to be re-powered on.

4. Connect the device to FS982-AG decoding PC software V3.3.4. Enter the home page of the PC software, select the serial port number and baud rate (default 115200), and then click 'Connect'. The device information on the left side indicates a successful connection. Click 'Mode Activation' in the lower-left corner, and reach correct access to G200. The valid bit for the front wheel gyroscope will be displayed as green; otherwise, it will be red. Once the vehicle initialization is successful, the valid bit for the front wheel angle will be displayed as green; otherwise, it will be red. The front wheel angle data will be output in real time when the vehicle is travelling.

Fig. 11 Correct Access to G200

11. Accessories

Standard Accessories:

 FSS-NAV680-AG	 RS232 Serial Cable
 Primary Antenna (Positioning Antenna)	 Secondary Antenna (Directional Antenna)
 Primary Antenna Connector	 Secondary Antenna Connector

Optional Accessories:

RS232 Serial Cable

XT60

FSS-G200

11. List of Frequent Questions

Problem	Answers
The upper computer cannot connect	Please check whether the serial port is occupied and whether the product is powered on normally. If the host computer is disconnected during the connection process, the serial port may be loose. Plug and remove the serial cable and turn on the host computer again.
The positioning status is always 1	Confirm that the differential data is connected normally, check whether the baud rate, differential account, mount point and other information is correct
The directional state of the double antenna has not reached 50	Check for the following factors: 1. If the directional status is always 0, check whether the secondary antenna is connected and whether the antenna feeder is damaged. 2. If the directional status is not 0 but does not reach 50, please confirm whether the test environment is open and whether the distance between the two antennas is more than 50CM.
When the active antenna is connected, the signal is normal. No signal when connected to a passive antenna	The feed line of the passive antenna cannot exceed 1.5M.
When the antenna is connected to an active power splitter, the signal is normal. When the device supplies power to the antenna, there is no signal	Check whether there is a short circuit in the antenna feeder or whether the antenna is inserted or removed during power-on. The antenna power supply circuit is abnormal due to static electricity. A fuse has been designed in the current circuit design and can be restored after power-on.
Failure to initialize	The following conditions must be met for initialization, Single antenna: positioning state fixed solution + driving speed above 0.5m/s + linear driving with acceleration and deceleration Dual antenna: positioning state fixed solution +

	directional state 50
Packet loss over serial port	Check the following factors: 1. The serial cable supports at least 115,200 baud 2. The serial port delay must be set to 2ms
How to determine whether the G200 physical connection is correct	By parsing the number of status bits, see if bit5 is valid
With G200 connected, the front gyro significant bit is not valid	Use the CAN analyzer to check whether the CAN baud rate of G200 is consistent with the CAN Baud rate of AG. If not, please modify it to be consistent. For specific modification instructions, refer to the parameter configuration page of the G200 and AG manual
When G200 is connected, the effective bit of the front wheel gyro is valid, and the front wheel Angle is abnormal	Please check the following factors: 1. The premise that the front wheel Angle output is correct is that the AG device can output the correct front wheel Angle after completing the initialization 2. Wheelbase is configured. If the wheelbase is not configured, the front wheel Angle will be invalid
CAN configuration rod arm, double antenna Angle and other information configuration results are inconsistent with the configuration	Check whether the CAN is configured in the following ways: Small-endian mode, low bytes first, unit is CM, negative numbers need to be complemented
Can the nmea protocol be output synchronously with the binary protocol?	No, only one of these protocols can be output simultaneously
When the G200 is connected, the front wheel Angle is asymmetrical	Check for the following factors: 1. Whether the G200 is installed horizontally 2. If it is only powered on in the static state, it needs to be powered on at 0 degrees, or it can be corrected by manually driving straight for a section
Why is there still a satellite number when the antenna is unplugged?	UM982 signal tracking ability is strong, if there is a signal source around, unplugging the antenna will continue to track for a period of time

	through the coupling mode, the actual satellite lock-out scenario antenna is still connected, external signals can not enter, does not affect the use
The use of a metal pot cover to block the antenna simulates the loss of lock, and the state is still fixed	Error test method: The metal pot lid can only block the signal directly above, and the side signal is reflected through the roof to the bottom of the pot lid and then reflected to the antenna, which will cause false fixation
The ROS drive can't read the data	The default power-on output combination data stream is not configured for the device. Configuring the default power-on output combination data stream can solve the problem
Large heading error	Several reasons may exist: 1. The rod arm, the wheelbase is misconfigured, the unit is magnified 10x, 2. Confirm whether it is rigid and fixed 3. Whether it is away from strong vibration sources
The attitude error with the reference datum exceeds the standard	Confirm whether the installation error is deducted during statistics (generally, the average error between the benchmark and the installation error)
How to complement the CAN command when configuring the small end	The complement is performed on negative numbers, -20, X-65536=-20 X=65516 The hexadecimal corresponding to 65516 is FFEC, the hexadecimal corresponding to 65535 is FFFF, because it is calculated from 0, the maximum is 65536, the current result minus the maximum is the complement value

12. Revision History

Version	Dates	Status/Comments
Version 1.0	2024.01.09	First Issue
Version 2.0	2024.01.12	Added RTK performance metrics
Version 3.0	2024.01.15	Add Update record
Version 4.0	2024.01.18	Adjust primary and secondary antenna attachment picture
Version 5.0	2024.01.24	Remove CAN B
Version 6.0	2024.01.26	1. Add CAN protocol related content, 2. Add RTK positioning status markers
Version 7.0	2024.02.20	Fixed rod arm configuration description
Version 8.0	2024.02.22	Fixed rotation Angle configuration answer description
Version 9.0	2024.03.26	Added Slope Mode/Match G200 Use description
Version 10.0	2024.06.12	Added some AT commands new. Added Latitude, longitude and elevation messages based on the CAN protocol. Added List of Frequent Questions.