FORSENSE
原极科技

Dual-Antenna Marine Satellite Compass

NAV680-SA

Product manual



Forsense (Shanghai) Technology Co., Ltd

Please read this user manual carefully before using this product.

Table of Contents

1. Product Introduction **2**
1.1 Packing List 2
1.2 Technical Specifications 2
1.3 Mechanical Dimensions 4
1.4 Interface Description 6
1.5 Data Cable Interface Definition 6
1.6 Electrical and Physical Characteristics 8
1.7 Antenna 8
2. Usage Example **9**
2.1 Device Installation 9
3. Coordinate System Definition **12**
4. Output Protocol **13**
4.1 Binary Protocol - Integrated Navigation Data Stream 13
4.2 NMEA Protocol 15
5. Parameter Configuration **24**
5.1 Via Serial Port Configuration 24
5.1.1 Configure GNSS Dual antenna Mounting Angle 24
5.1.2 Configure NMEA Format Data Stream Output 24
5.1.3 Configure Current Data Stream to Stop Output 25
5.1.4 Configure Baud Rate 25
5.1.5 Print All Configuration Information 26
5.1.6 Query Firmware Version 26
5.1.7 Configure Installation Rotation Angle 26
5.1.8 Save Parameter 26
6. Firmware Update **27**
6.1 Via PC software 27
6.1.1 R422 Direct Connection to Serial Port 27
6.2 serial push 27
7. Accessories **36**
8. Update Record **38**

1. Product Introduction

1.1 Packing List

When you open the packaging box, please verify that the product in the box:

Name	Quantity	Diagram
① NAV680-SA Dual antenna Compass	1	
②③ Satellite Antenna (No distinction between primary and secondary)	2	
④ Antenna feeder (Model: 20 m length, TNC Male to Male)	2	
⑤ Antenna Magnetic Base	2	
⑥ Antenna Mast (20 cm)	2	
⑦⑧ Hub Cable (includes 1 RS422, 1 Power Interface (9-24V)	1	
⑨ Power Adapter (12V, 1A)	1	

Figure 1 Product Physical Image

If any items are missing, please contact the sales representative promptly.

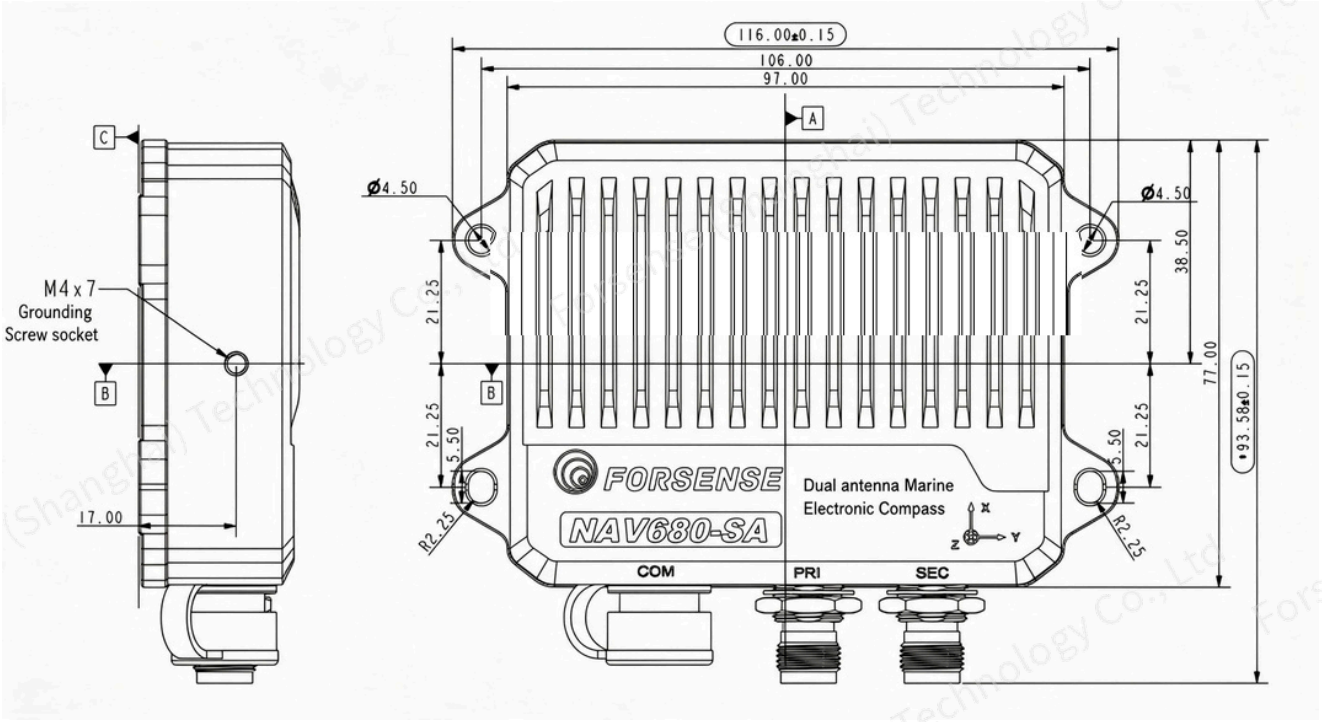
1.2 Technical Specifications

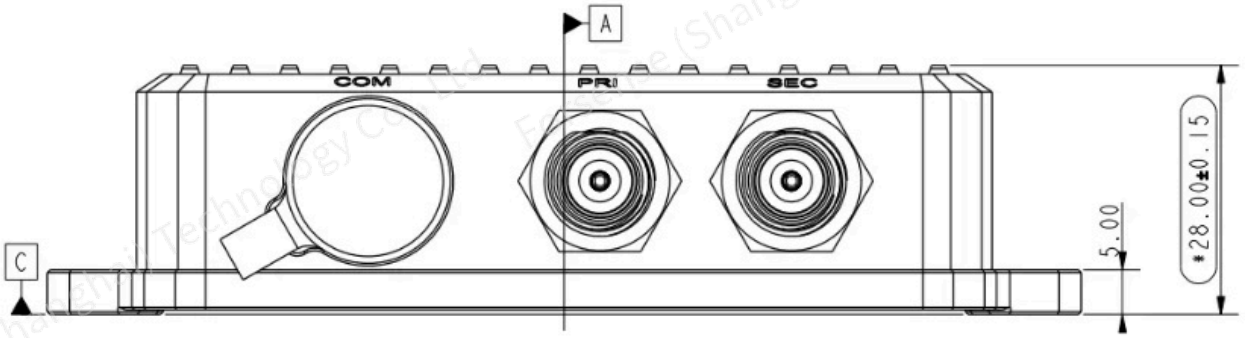
Table 1

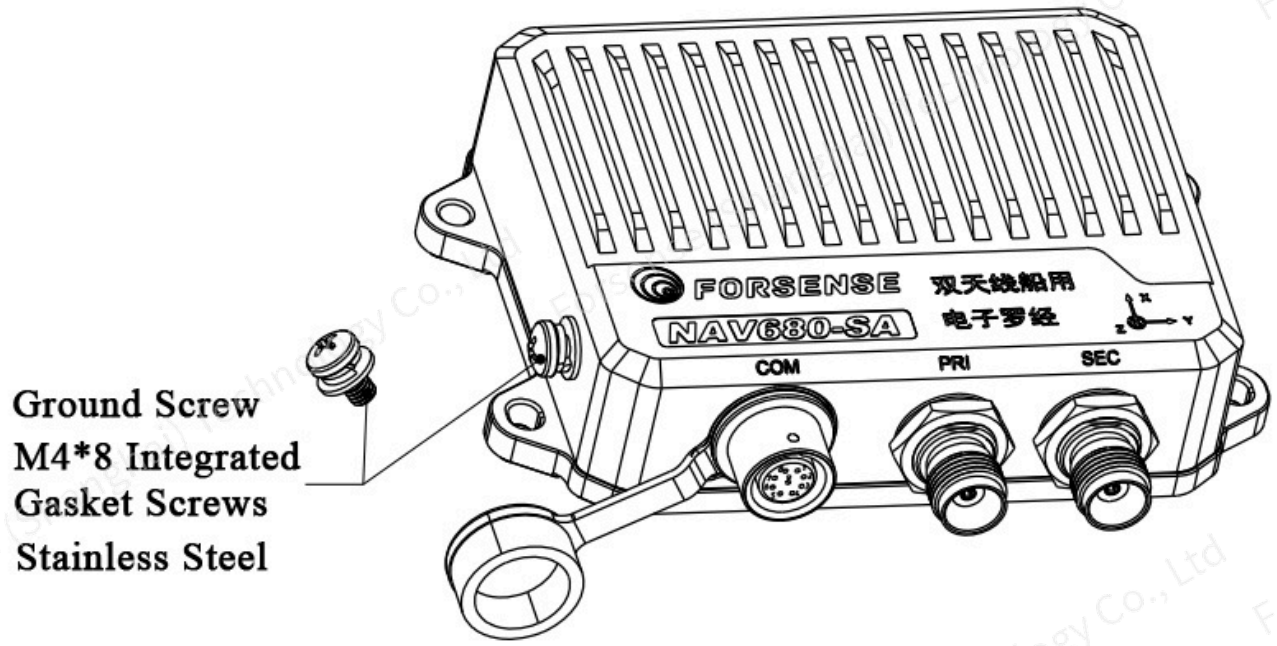
Integrated Navigation Specifications	**Attitude Accuracy**	**Roll/Pitch:<0.2°rms**
	bowhead accuracy	<0.3°rms
	heading accuracy	<0.3°rms
	Update Rate	100Hz
	velocity accuracy(RMS)	0.02m/s
IMU Specifications	Gyroscope Range	±500°/s
	Gyroscope Bias instability	4deg/h @1σ
	Accelerometer Range	±6g
	Accelerometer Bias stability	0.04mg @1σ
RTK Metrics	positioning accuracy(RMS)	Horizontal:5m Elevation:10m
	Dual antenna Heading Accuracy (RMS)	0.1°/1m baseline
	velocity accuracy(RMS)	0.03m/s
	PPS Accuracy (RMS)	20ns
	Update Rate	10Hz
	RTK initialize Time	<5s
Interface		1×RS422 2× GNSS Antenna Interface 1× Power Interface

Hardware Composition

1.3 Mechanical Dimensions







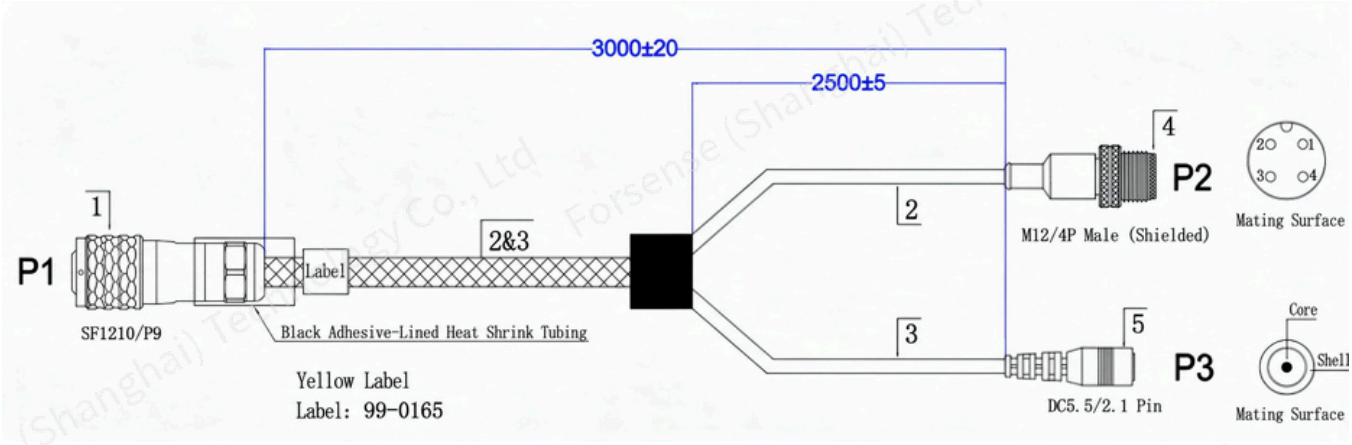
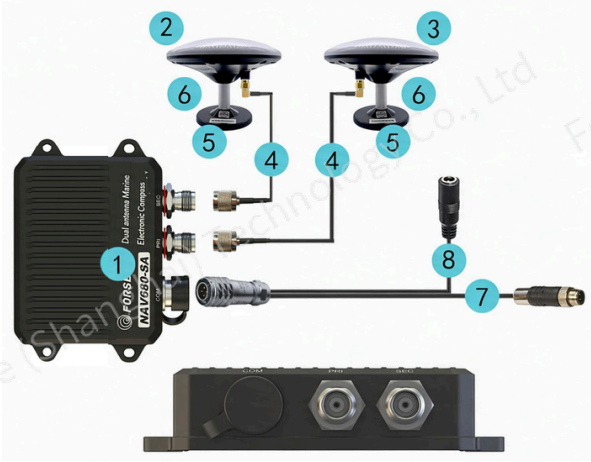


Figure 1 Mechanical Dimensions (Unit: mm)

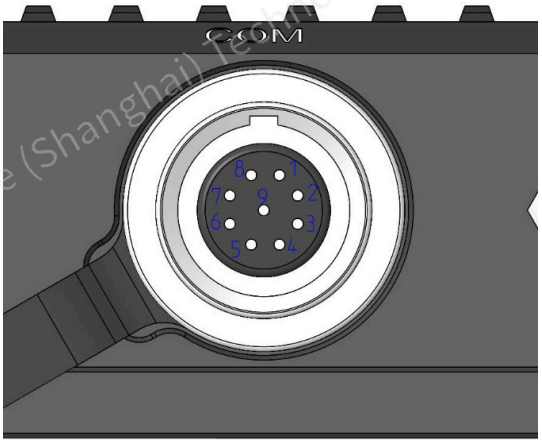
1.4 Interface Description

NAV680-SA includes a COM bus interface, with power and signal lines, PRI Primary antenna and SEC Secondary antenna.



	Name	Quantity
1	NAV680-SA Marine Electronic Compass	1 pc
2	Primary antenna (positioning antenna)	1 pc
3	Secondary antenna (positioning antenna)	1 pc
4	Antenna feeder (20 m)	2 pcs
5	Antenna Suction Cup	2 pcs
6	Antenna Pole	2 pcs
7	Harness - RS422 Interface	1 pc
8	Harness - Power Interface	1 pc

1.5 Data Cable Interface Definition



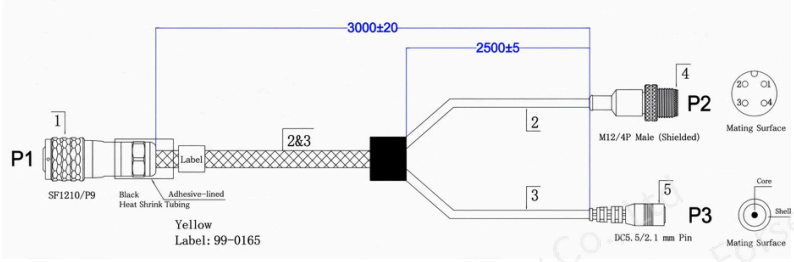


Figure 2 Data Cable Interface Definition Diagram

Pin No.	Definition	Port
1	RS422_TX+	M12 Male Connector, Pin1
2	VIN	DC Female Jack
3	RS422_TX-	M12 Male Connector, Pin2
4	GND	DC Female Jack
5	RS422_RX-	M12 Male Connector, Pin3
6	GND	Empty
7	RS422_RX+	M12 Male Connector, Pin4
8	PGND	M12 Male Connector, Cable Shield The definition of Pin is as follows
9	PPS	Empty

1.6 Electrical and Physical Characteristics

Table 2

Parameter	Symbol	Min.	Typ.	Maximum Value	Unit
Power Input	VIN	9	12	32	V
Ground (GND)	GND				
Power Consumption	P		1.8		W
Operating Temperature (Tot)	T	-40	\	85	℃
Storage Temperature (Tstg)	T	-40	\	95	℃
Waterproof Rating	IP67				
Humidity	95%non-condensing				
Dimensions	116*85.9*28mm				
Weight	263g				

1.7 Antenna

NAV680-SA provides PRI and SEC primary Secondary antenna interfaces. With no antenna connected, a multimeter test shows an open-circuit voltage of DC4.8–5.4V. When the module RF Port is connected to an antenna at room temperature with an operating current of 30–100mA, it can externally provide antenna power supply of DC4.6V±0.2V. When using NAV680-SA with active antenna, ensure a 50-ohm impedance matching between the device and the antenna.

2. Usage Example

2.1 Device Installation

1,Installation Requirements

1) For Dual antenna installation, you need to install two antennas (primary and Secondary antenna) on the vessel and connect them to the compass, following the requirements below:

The satellite antennas must have a clear view of the sky with no obstruction from the carrier. The baseline formed by Dual antenna should be kept as parallel to the ship's hull as possible.

Primary antenna connects to the PRI interface for positioning and is installed at the stern.

Secondary antenna connects to the SEC interface for heading determination and is installed at the bow.

The distance between the two antennas, i.e., the length of baseline, must be greater than 0.5 m.

The heights of Dual antenna should be kept consistent. See Figure 20 for the installation schematic.

2) Keep away from interference sources such as electrical components or systems that generate electromagnetic interference signals.

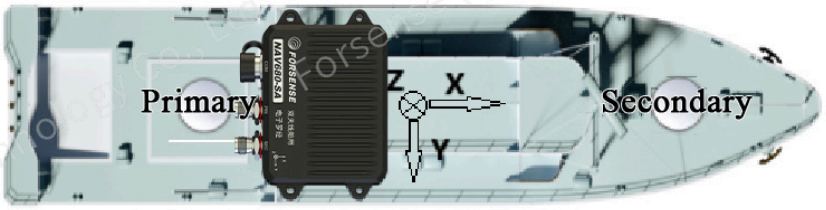
(3) Screw the GNSS antenna onto the strong magnetic suction cup and fix it as centrally as possible on the test platform'saxisposition, placing it at the highest point of the hull to ensure good GNSS signal reception;

(4) To achieve optimal performance, minimize the distance between the GNSS Primary antenna and the device Master, especially the horizontal distance.

(5) Securely mount on a rigid surface; avoid locations with high vibration.

(6) The X-axis orientation should maintain the relationship to the bow direction as shown in the figure below.

The Primary and Secondary Antenna is placed as follows,
the two Antennas should be in a straight line.
The Primary Antenna on the stern of the boat and the Secondary Antenna on the bow of the boat.
Antenna Angle Installing Deviation (0°)



Dual antenna Installation Diagram

The correct installation diagram is shown below

X Axis Facing Forward

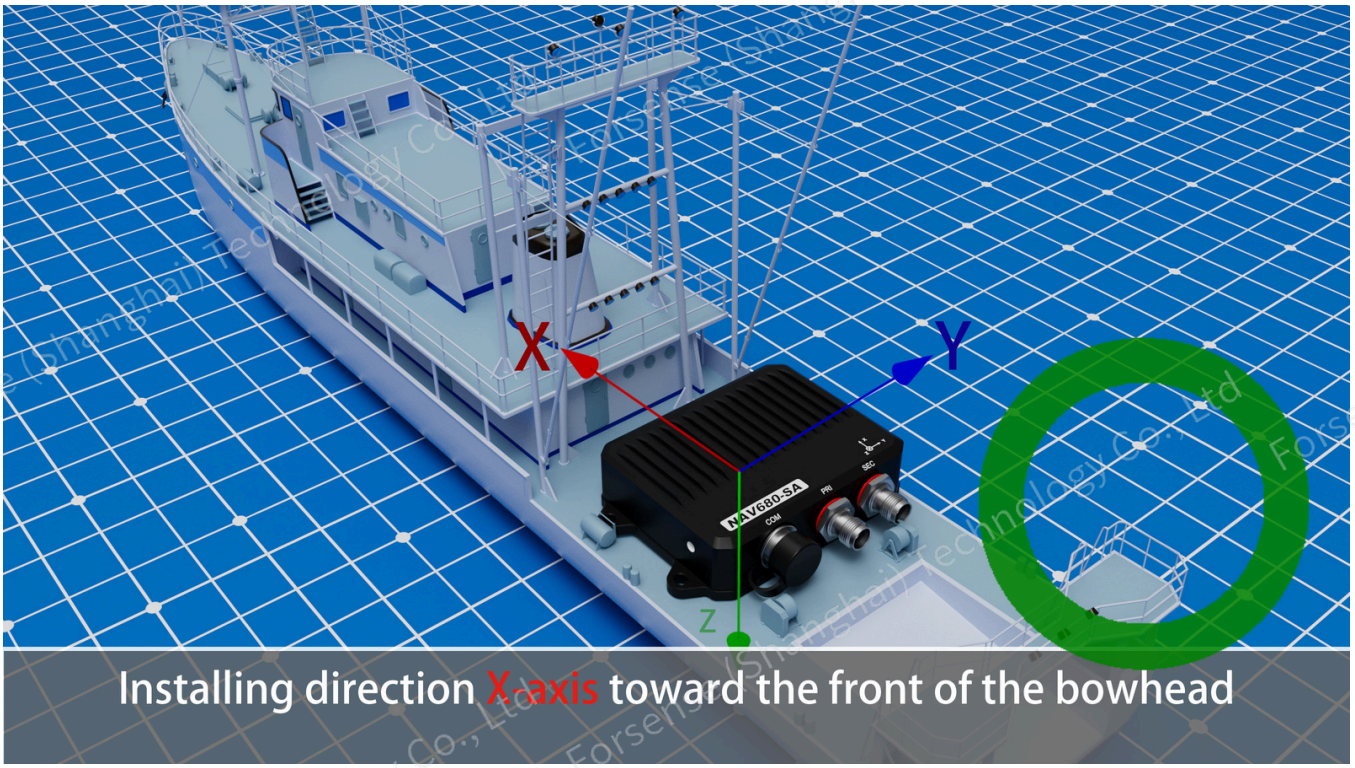
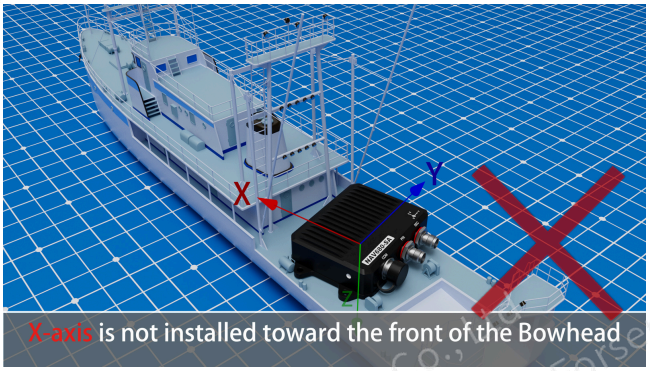


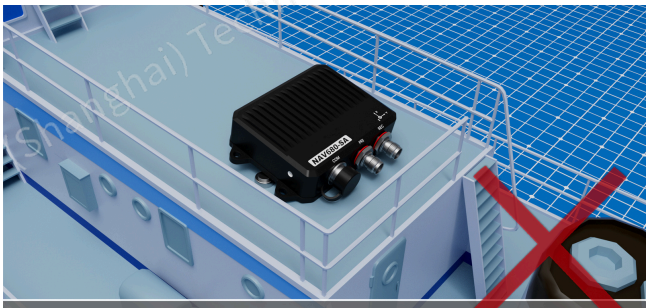
Figure 3 Correct Installation Diagram

The following installation methods are allIncorrect Installation.

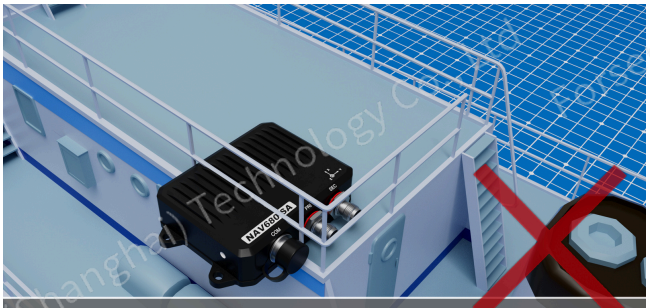


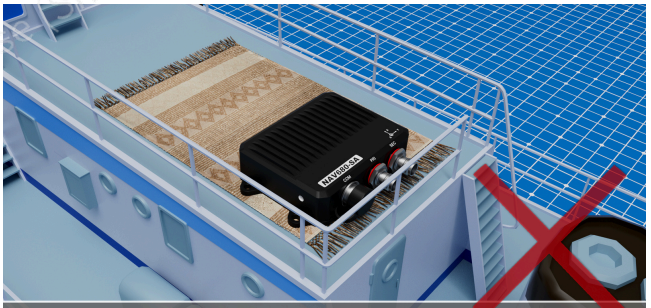
X-axis is not installed toward the front of the Bowhead


The Compass is installed on a bracket


The Compass is not installed horizontally


The Campass is fixed by hand


The Campass is installed outside the vehicle


The compass is installed on a non-rigid surface

3. Coordinate System Definition

Figure 7 Coordinate System Schematic



This product uses the Front-Right-Down (FRD) coordinate system. The Euler angle ranges are as follows:

Rotation about the Z-axis: Yaw angle Yaw range: 0° to 360°;

Rotation about the X-axis: Roll angle Roll range: -180° to 180°;

Rotation about the Y-axis: Pitch angle Pitch range: -90° to 90°.

4. Output Protocol

4.1 Binary Protocol - Integrated Navigation Data Stream

Note:

- The CRC check starts from Frame header and does not include the CRC check bit itself. It covers all bytes in the frame. For the calculation method and example code, refer to the appendix.
- The frame length is the total number of data bytes excluding Frame header, Frame ID, Frame Length, and Check Bit.
- little-endian mode, send the low byte first.

Content	Type	Relative Position
Frame header 1:0xAA	Uint8	0
Frame header 2:0x55	Uint8	1
Frame ID: 0x0166	Uint16	2
Frame Length: 0x005E	Uint16	4
seconds in GPS week(ms)	Uint32	6
GPS week number count	Uint16	10
Latitude (Deg × 10000000)	Int32	12
Longitude (Deg × 10000000)	Int32	16
Altitude (mm)	Int32	20
North Velocity (m/s)	Float	24
East Velocity (m/s	Float	28
Down Velocity (m/s)	Float	32
Roll angle (deg)	Floa 40t	36

Pitch angle (deg)	Float	40
heading angle (deg)	Float	44
Dual antenna Heading (deg)	Float	48
track angle (deg)	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
Gyroscope Y-axis (deg/s)	Float	72
Gyroscope Z-axis (deg/s)	Float	76
IMU Temperature (°C)	Float	80
RTK Positioning Status (same as the positioning status in GGA) 0: No Fix 1: single point positioning 2: Pseudo-range differential positioning4:fixed solution 5:floating solution	Uint8	84
Number of Satellites	Uint8	85
differential delay	Uint8	86
Dual antenna Orientation status 50 indicates alignedOthers indicate not aligned	Uint8	87
Position Dilution of Precision (DOP) (cm) Integrated Navigation initialize Valid After	Uint16	88
Status bits: bit0: 1 indicates RTK data is valid, 0 indicates invalid Bit1: 1 indicates PPS signal is valid, 0 indicates invalidBit2: 1 indicates Integrated Navigation has been initialize, 0 indicates not initialize	Uint16	90
Reserved 1	Uint32	92
Reserved 2	Uint32	96
CRC Check	Uint32	100

4.2 NMEA Protocol

- Supports output of fused data in NMEA format,
- Cannot be output simultaneously with the binary data stream. If the NMEA data stream is enabled, the binary data stream cannot be output. Before switching data streams, stop the current data stream output using the command in Section 7.2.2.
- The following sentences are currently supported. For configuration methods, refer to Chapter 7.3.

GPGGA

Message Description

Field	Structure	Description	Style	Example
1	$GPGGA	Message Header		$GPGGA
2	utc	UTC Time (Hour/Minute/Second/Fractional Second)	hhmmss.ss	202134
3	lat	Latitude (DDmm.mm)	llll.ll	5106.9847
4	lat dir	Latitude Direction (N = North, S = South)	a	N
5	lon	Longitude (DDDmm.mm)	yyyyy.yy	11402.2986
6	lon dir	Longitude Direction (E = East, W = West)	a	W
7	quality	Solution Status	x	1
8	# sats	Number of satellites used in the solution, which may differ from the number of visible satellites	xx	2
9	hdop	Horizontal Dilution of Precision (DOP)	x.x	1
10	alt	Antenna height above or below sea level	x.x	1062.22
11	a-units	Antenna height unit (M = metres)	M	M
12	undulation	Elevation Outlier	x.x	-16.271
13	u-units	Elevation Outlier Unit (M = metres)	M	M
14	age	Age of Differential Data, s	xx	1
15	stn ID	Differential Base Station ID	xxxx	8
16	*xx	Checksum	*hh	*48
17	[CR][LF]	Sentence Terminator		[CR][LF]

Official Message Description

Field	Structure	Description	Symbol	Example
1	$GPGGA	Log header		$GPGGA
2	utc	UTC time status of position (hours/minutes/seconds/ decimal seconds)	hhmmss.ss	202134.00
3	lat	Latitude (DDmm.mm)	llll.ll	5106.9847
4	lat dir	Latitude direction (N = North, S = South)	a	N
5	lon	Longitude (DDDmm.mm)	yyyyy.yy	11402.2986
6	lon dir	Longitude direction (E = East, W = West)	a	W
7	quality	refer to Table: GPS Quality Indicators	x	1
8	# sats	Number of satellites in use. May be different to the number in view	xx	10
9	hdop	Horizontal dilution of precision	x.x	1.0
10	alt	Antenna altitude above/below mean sea level	x.x	1062.22
11	a-units	Units of antenna altitude (M = metres)	M	M
12	undulation	Undulation - the relationship between the geoid and the WGS84 ellipsoid	x.x	-16.271
13	u-units	Units of undulation (M = metres)	M	M
14	age	Age of correction data (in seconds) The maximum age reported here is limited to 99 seconds.	xx	(empty when no differential data is present)
15	stn ID	Differential base station ID	xxxx	(empty when no differential data is present)
16	*xx	Check sum	*hh	*48
17	[CR][LF]	Sentence terminator		[CR][LF]

GPS Quality Indicators

Indicator	Description
0	Fix not available or invalid
1	Single point
	Converging PPP (TerraStar-L)
2	Pseudorange differential
	Converged PPP (TerraStar-L)
	Converging PPP (TerraStar-C PRO, TerraStar-X)
4	RTK fixed ambiguity solution
5	RTK floating ambiguity solution
	Converged PPP (TerraStar-C PRO, TerraStar-X)
6	Dead reckoning mode
7	Manual input mode (fixed position)
8	Simulator mode
9	WAAS (SBAS)[1]

GPRMC

Message Description in Chinese

Field	Structure	Description	Style	Example
1	$GPRMC	Message Header		$GPRMC
2	utc	Fixed UTC Time	hhmmss.ss	144326
3	pos status	Positioning Status (A for Valid Positioning, V for Invalid Positioning)	A	A
4	lat	Latitude (DDmm.mm)	llll.ll	5107.0017737
5	lat dir	Latitude Direction (N = North, S = South)	a	N
6	lon	Longitude (DDDmm.mm)	yyyyy.yy	11402.3291611
7	lon dir	Longitude Direction (E = East, W = West)	a	W
8	speed Kn	Ground Speed	x.x	0.08
9	track true	Ground Course (True North Reference)	x.x	323.3
10	date	UTC Date: (Year/Month/Day)	xxxxxx	210307
11	mag var	Magnetic Variation	x.x	0
12	var dir	Magnetic Variation Direction	a	E
13	mode ind	Positioning Mode Indicator	a	A
14	*xx	Checksum	*hh	*20
15	[CR][LF]	Sentence Terminator		[CR][LF]

Official Message Description

Field	Structure	Description	Symbol	Example
1	$GPRMC	Log header		$GPRMC
2	utc	UTC of position	hhmmss.ss	144326.00
3	pos status	Position status (A = data valid, V = data invalid)	A	A
4	lat	Latitude (DDmm.mm)	llll.ll	5107.0017737
5	lat dir	Latitude direction: (N = North, S = South)	a	N
6	lon	Longitude (DDDmm.mm)	yyyyy.yy	11402.3291611
7	lon dir	Longitude direction: (E = East, W = West)	a	W
8	speed Kn	Speed over ground, knots	x.x	0.080
9	track true	Track made good, degrees True	x.x	323.3
10	date	Date: dd/mm/yy	xxxxxx	210307
11	mag var	Magnetic variation, degrees Note that this field is the actual magnetic variation and will always be positive. The direction of the magnetic variation is always positive.	x.x	0.0
12	var dir	Magnetic variation direction E/W Easterly variation (E) subtracts from True course. Westerly variation (W) adds to True course.	a	E
13	mode ind	Positioning system mode indicator, see Table: NMEA Positioning System Mode Indicator	a	A
14	*xx	Check sum	*hh	*20
15	[CR][LF]	Sentence terminator		[CR][LF]

NMEA Positioning System Mode Indicator

Mode	Indicator
A	Autonomous
D	Differential
E	Estimated (dead reckoning) mode
M	Manual input
N	Data not valid

GPHDT (Heading Information)

Message Description in Chinese

Field	Structure	Description	Style	Example
1	$GPHDT	Message Header		$GPHDT
2	heading	Yaw angle	x.x	75.5554
3	TRUE	True North Angle	T	T
4	*xx	Checksum	*hh	*36
5	[CR][LF]	Statement Terminator		[CR][LF]

Official Message Description

Field	Structure	Description	Symbol	Example
1	$GPHDT	Log header		$GPHDT
2	heading	Heading in degrees	x.x	75.5554
3	True	Degrees True	T	T
4	*xx	Check sum	*hh	*36
5	[CR][LF]	Sentence terminator		[CR][LF]

GPVTG (Ground Speed Information)

Message Description in Chinese

Field	Structure	Description	Style	Example
1	$GPVTG	Message Header		$GPVTG
2	track true	Heading Angle, referenced to True North	x.x	24.168
3	T	True North Indicator	T	T
4	track mag	Heading Angle, referenced to Magnetic North	x.x	24.168
5	M	Magnetic North Indicator	M	M
6	speed Kn	Horizontal Velocity	x.x	0.4220347
7	N	Velocity Indicator, Knots	N	N
8	speed Km	Horizontal Velocity, km/s	x.x	0.781608
9	K	Velocity Indicator (K = km/hr)	K	K
10	mode ind	Positioning Mode Indicator	a	A
11	*xx	Checksum	*hh	*7A
12	[CR][LF]	Statement Terminator		[CR][LF]

Official Message Description

Field	Structure	Description	Symbol	Example
1	$GPVTG	Log header		$GPVTG
2	track true	Track made good, degrees True	x.x	24.168
3	T	True track indicator	T	T
4	track mag	Track made good, degrees Magnetic; Track mag = Track true + (MAGVAR correction) See the **MAGVAR** command	x.x	24.168
5	M	Magnetic track indicator	M	M
6	speed Kn	Speed over ground, knots	x.x	0.4220347
7	N	Nautical speed indicator (N = Knots)	N	N
8	speed Km	Speed, kilometres/hour	x.x	0.781608
9	K	Speed indicator (K = km/hr)	K	K
10	mode ind	Positioning system mode indicator, see Table: NMEA Positioning System Mode Indicator	a	A
11	*xx	Check sum	*hh	*7A
12	[CR][LF]	Sentence terminator		[CR][LF]

NMEA Positioning System Mode Indicator

Mode	Indicator
A	Autonomous
D	Differential
E	Estimated (dead reckoning) mode
M	Manual input
N	Data not valid

GPZDA (UTC Time and Date)

Message Description in Chinese

Field	Structure	Description	Style	Example
1	$GPZDA	Message Header		$GPZDA
2	utc	UTC Time	hhmmss.ss	220238
3	day	Day, 01 -31	xx	15
4	month	Month, 01-12	xx	7
5	year	Year	xxxx	1992

6	null	Local Area Code, Not Available	xx	(empty when no data is present)
7	null	Local Zone Offset Description, Not Available	xx	(empty when no data is present)
8	*xx	Checksum	*hh	*6F
9	[CR][LF]	Statement Terminator		[CR][LF]

Official Message Description

Field	Structure	Description	Symbol	Example
1	$GPZDA	Log header		$GPZDA
2	utc	UTC time status	hhmmss.ss	220238.00
3	day	Day, 01 to 31	xx	15
4	month	Month, 01 to 12	xx	07
5	year	Year	xxxx	1992
6	null	Local zone description—not available Local time zones are not supported by OEM7 family receivers. Fields 6 and 7 are always null.	xx	(empty when no data is present)
7	null	Local zone minutes description—not available	xx	(empty when no data is present)
8	*xx	Check sum	*hh	*6F
9	[CR][LF]	Sentence terminator		[CR][LF]

The GPATT format is shown in the table below

Name	Unit	Format	Example	Description
Message Header		String	$GNATT	
UTC Time		hhmmss.sss	170834.000	17:08:34 UTC
Status		Character	1	0: Unavailable 1: Available
Roll angle	degree	3 decimal place	-4.891	Range is ±180°, defined with right side downward is positive
Roll angle Indicator		Character	R	Roll angle Indicator
Pitch angle	degree	3 decimal place	3.122	Range is ±90°, defined as upward is positive
Pitch angle Indicator		Character	P	Pitch angle Indicator
Yaw angle	degree	3 decimal place	124.005	Range from 0° to 360°, true North, clockwise is positive
Roll angle uncertainty	degree	3 decimal place	0.432	Range from 0 to 360
Pitch angle uncertainty	degree	3 decimal place	0.811	Range from 0 to 360
Yaw angle uncertainty	degree	3 decimal place	1.202	Range from 0 to 360
Checksum		HEX	*68	Used to check for transmission errors

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

5. Parameter Configuration

5.1 Via Serial Port Configuration

5.1.1 Configure GNSS Dual antenna Mounting Angle

If the GNSS Dual antenna mounting angle is configured as 0 degrees, the configuration command is:

Command: AT+RTK_ANGLE=0\r\n

Response: ANGLE=0\r\n

Note: The mounting angle is the angle between the ray from Primary antenna pointing to Secondary antenna and the vehicle heading direction. Clockwise is positive, counterclockwise is negative. Angle input range: -180°~180°

Note: Configuration commands take effect after a power cycle; the Dual antenna distance must be greater than 50 cm,Speed projection points

If the output of projection points results as set by Integrated Navigation is configured, 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

Description: The default output for Integrated Navigation is the antenna phase center coordinates. To output coordinates for other positions, configure the lever arm vector of the IMU phase center relative to this projection points position. The configuration method is the same as the lever arm configuration in Section 8.1.1.

5.1.2 Configure NMEA Format Data Stream Output

If NMEA sentence output is configured,4.1Integrated Navigation data stream will not be output

To switch to4.1Integrated Navigation data stream output, first send the5.1.3command to stop the current data stream output

The configuration command is as follows

GPGGA

Example: Output GPGGA sentences at a frequency of 5 Hz: AT+GPGGA=5\r\n

Response: OK\r\n

GPRMC

Example: Output GPRMC sentences at a frequency of 1 Hz: AT+GPRMC=1\r\n

Response: OK\r\n

GPHDT (Heading Information)

Example: Output GPHDT sentences at a frequency of 1 Hz: AT+GPHDT=1\r\n

Response: OK\r\n

GPVTG (Ground Speed Information)

Example: Output GPVTG sentences at a frequency of 1 Hz: AT+GPVTG=1\r\n

Response: OK\r\n

GPZDA (UTC Time and Date)

Example: Output GPZDA sentences at a frequency of 1 Hz: AT+GPZDA=1\r\n

Response: OK\r\n

GPATT (Custom Message)

Example: Output GPATT sentences at a frequency of 1 Hz: AT+GPATT=1\r\n

Response: OK\r\n

If configured to stop output

Command: AT+SETNO\r\n

Response: OK\r\n

5.1.3 Configure Current Data Stream to Stop Output

Command: AT+SETNO\r\n

Response: OK\r\n

5.1.4 Configure Baud Rate

The default baud rate is 115200

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

Command: AT+BAUD=230400\r\n

Response: BAUD=230400\r\n

Note: The configuration takes effect after saving and power cycling

5.1.5 Print All Configuration Information

To query all configured information, use the following command:

AT+CONFIG\r\n

5.1.6 Query Firmware Version

AT+VERSION\r\n

5.1.7 Configure Installation Rotation Angle

When the compass is not installed in the default orientation, configure the rotation angle.

Currently, only the following rotation angles are supported:

Rotate 180° around the X-axis

Rotate 90°, 180°, or 270° around the Z-axis

The configuration command is as follows:

If the installation rotation angle is 180° around 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 degrees around the Z-axis, the configuration command is:

Command: AT+INSTALL_ANGLE=0,0,180

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

5.1.8 Save Parameter

The above settings, including lever arm, data stream, and output frequency Parameter, require this save command after configuration to take effect upon restart.

Command: AT+SAVE

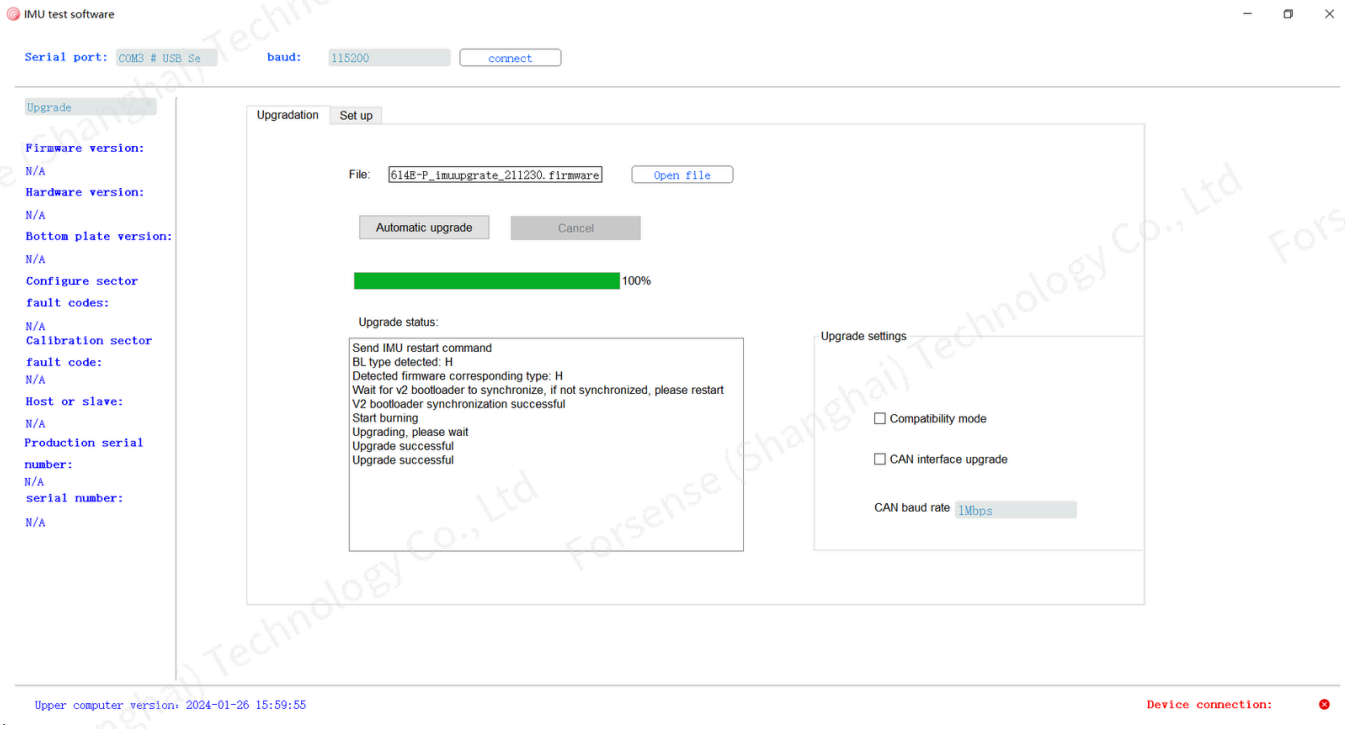
Response: OK

6. Firmware Update

6.1 Via PC software

6.1.1 R422 Direct Connection to Serial Port

Use Forsense IMU to test PC software: Select Firmware Update, Open Firmware, and Click Auto Upgrade.



6.2 serial push

The upgrade process consists of the following steps:

Step 1: Send Upgrade Command

Send the upgrade command to prepare the module for updating. After the command is sent, the device records an upgrade flag in a specific flash area and performs a soft reboot. At this point, the IMU module enters

In BOOTLOADER.

```
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};
   At this point, the IMU module will restart and enter the bootloader.
```

Step 2: Send **HC32MCU_FORSENSE** String

After the IMU module enters bootloader mode, it actively sends the HC32_UPLOADER string. During this period, PC software must send the HC32MCU_FORSENSE string. Once the IMU module receives this string, it will not jump to the APP and will remain in bootloader mode in a pending upgrade state, while also

stopping the active transmission of any messages.

```
After sending the upgrade command, immediately send the following string to the IMU module. The IMU module does not jump to the APP region and remains in the 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 is in bootloader mode with a pending upgrade status, it does not actively send information but only passively responds to commands from PC software. At this point, after PC software sends the command Send_CMD_LONG(0X21,0,0,0,0,0,0), it waits for the response from the IMU module. Upon receiving this command, the IMU immediately replies with Send CMD ACK (0x64,0x10,0)

data. Once PC software detects that this data has been received, synchronization is considered complete.

```
PC software and IMU Module Synchronization Process:
1. PC software sends the synchronization command with command code 0x21, then waits to a response from the IMU module.
 Send_CMD_LONG(0x21,0,0,0,0,0,0);
2. Upon receiving command code 0x21, the IMU module sends response data. The data command is as follows:
Send_CMD_ACK(0x64,0x10,0);
3. PC software checks whether it has received response data from the IMU. If received, synchronization is considered complete.
Note: See the end of the article for the D_ACK function
```

Step 4: Send Erase Command

PC software sends the erase command Send_CMD_LONG(0x23,0,0,0,0,0,0). After the command is sent, the IMU module erases all content in the APP area and reports the final execution result to PC software. PC software determines whether to resend the erase command based on the execution result. Once the erase operation is successful, the APP in the IMU module cannot be restored. Additionally, PC software must wait for the successful erase response from the IMU before proceeding to the next step; otherwise,

subsequent upgrade failures may occur.

```
1. PC software sends the erase command with command code 0x23. After sending this command, wait patiently for the IMU to reply with the execution result.
   Send_CMD_LONG(0x23,0,0,0,0,0,0,0);
2. Upon receiving the command code 0x23, the IMU will erase all data in the APP area and reply to the PC software with the final execution result.
    send_CMD_ACK(0x64,0x10,0);  // Indicates Erase Success
```

```
   send_CMD_ACK(0x64,0x11,0); // indicates erase failure; re-erasing is required
   Note: PC software must receive a successful erase response before proceeding to the next step
```

Step 5: Send Upgrade Data Packet

After successful erasure, the critical phase of sending firmware data begins. The function Send_Upload_Data is used to send the data. PC software splits the upgrade firmware into packets, each with a fixed size of 64 bytes; the last packet sends the actual number of bytes if it is less than 64. Each frame contains the valid data length and its offset address within the entire firmware. PC software After sending each frame, wait for the response from the IMU module. Only after confirming that the IMU has successfully received the current frame should the next frame be sent. Once the IMU module successfully receives the data packet PC software, it sends response data and writes to the specified flash address based on offset address.

If writing to flash fails, send a failure command; if writing succeeds, do not send anything.

Example: Send the 1000-byte upgrade file uint8_t Upgrade_Data[1000] to the imu module

// Send the first packet:

1.1 Send bates 0~63 to the imu module via function

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

// In the above function, the first Parameter 0x27 is a fixed value, the second Parameter 0 is a fixed value, and the third Parameter 0 is offset address, The fourth Parameter 0x40 is the valid byte length, and the fifth Parameter is the starting address of the data to be sent

1.2 After the imu successfully receives the data, it sends response data and writes to the specified flash address based on offset address

```
send_CMD_ACK(0x753D,0x00,0);
```

// In the above function, the first Parameter 0x5D is a fixed value, the second Parameter 0 is a fixed value, and the third Parameter 0 is offset address.

If the write fails during the process of writing data to flash by the imu module, it will send the failure command `send_CMD_ACK(0x64,0x11,0);`

// Send the second packet:

2.1 After the host computer receives the response data from imu, it will send the second packet of data

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

2.2 After the imu successfully receives the second packet of data, it sends a response: `send_CMD_ACK(0x753D,0x00,0x40);`

2.3 If the write fails during the process of writing data to flash by the imu module, it will send the failure command `send_CMD_ACK(0x64,0x11,0);`

// Send the third packet:

3.1 After the host computer receives the response data from imu, it will send the third packet of data

```
Send_Upload_Data (0x27,0,0,0x80,0x40,Upgrade_Data+0x80);
```

3.2 After the imu successfully receives the third packet of data, it sends a response: `send_CMD_ACK(0x753D,0x00,0x80);`

3.3 If the write fails during the process of writing data to flash by the imu module, it will send the failure command `send_CMD_ACK(0x64,0x11,0);`

// Send the fifteenth packet:

15.1 Hostr computer abe rec eiving the response data from the IMU, sends `Send_Upload_Data (0x27,0,0x380,0x40,Upgrade_Data+0x380);`

15.2 After the Imu successfully receives the 15th packet of data, it sends a response: `send_CMD_ACK(0x753D,0x00,0x380);`

15.3 If a write failure occurs when the imu module write data to flash, ite will sends a failure command `send_CMD_ACK(0x64,0x11,0);`

" Send the sixth packet:"

15.1 Hostr computer ate rec eiving the response data from the IMU, sends `Send_Upload_Data (0x27,0,0x3C0,0x28,Upgrade_Data+0x3C0);`

15.2 After the imu successfully receives the 16th packet of data, ite sends a response: `send_CMD_ACK(0x753D,0x00,0x3C0);`

15.3 If a write failure occurs when the imu module write data to flash, the will send a failure command `send_CMD_ACK(0x64,0x11,0);`

Transmission ended

Step 6: Obtain **CRC** Checksum

Generally, the firmware upgrade command format is imu614e-b#CRC1373387121.firmware. The string following CRC is the pre-calculated CRC checksum value. After sending the upgrade file, the host software must send a verification command to check if the IMU module received the upgrade file correctly. After PC software sends the Send_CMD_LONG(0x29,0,0,0,0,0,0) command, it retrieves the CRC check digit calculated by the IMU module.

If PC software determines that CRC checksum value is incorrect, restart the upgrade process from Step 4 (Erase Command).

```
PC software Send Fetch CRC Check Code Command, Wait for IMU Response
 Send_CMD_LONG(0x29,0,0,0,0,0,0);
IMU Module Response Send CRC Check Value Data:
send_CMD_ACK(0x753C,0x10,crc32_data);
Where the crc32_data value is the crc32 data calculated by the IMU module itself
```

Step 7: Send Reboot Command

After PC software verifies that the CRC checksum is correct, send the reboot command to complete the upgrade successfully.

```
After verifying that the CRC checksum is correct, send the restart command:
 Send_CMD_LONG(0x30,0,0,0,0,0,0);
```

Once the Firmware Update is complete, power cycle the device and verify the success of the upgrade by reading the version number.

Function Definition:

- The definition of the Send_CMD_LONG function 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));
}
```

- The definition of the Send_CMD_ACK function 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));
}
```

- The definition of the Send_Upload_Data function 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));
}
```

- The CRC32 checksum function 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, 0xfbd44c69,
    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,
```

Forsense (Shanghai) Technology Co., Ltd

7. Accessories

Standard accessories:



NAV680-SA Master



Cable Harness



Primary antenna(positioning antenna)



Secondary antenna (Directional Antenna)



Primary antenna Cable



Secondary antenna Cable

Forsense (Shanghai) Technology Co., Ltd

8. Update Record

Latest Manual Version: NAV680-SA_Datasheet_Product Manual(en)

Version	Date	Status/Notes
Version 1.0	2024.01.09	Initial Release
Version 1.1	2026.03.18	Updated positioning accuracy in RTK Metrics

Forsense (Shanghai) Technology Co., Ltd