

## catalogue

1. Product introduction.....	1
1.1 Product List .....	1
1.2 Product Overview .....	2
1.3 Technical Indicators .....	2
2. Hardware composition.....	3
2.1 Mechanical dimensions .....	3
2.2 Interface Description .....	4
2.3 Data cable interface definition .....	5
2.4 Electrical physical properties .....	6
3. Hardware wiring method .....	7
3.1 Hardware wiring Methods .....	7
3.2 Antenna .....	7
4. Use examples .....	8
4.1 Equipment Installation .....	8
4.2 Parameter Settings .....	10
4.2.1 Configuring the main mast arm .....	10
4.2.2 Configure the output NMEA format data stream .....	11
4.2.3 Configure the current data stream to stop output .....	11
4.2.4 Configuring the Baud rate .....	11
4.2.5 Print all configuration information .....	11
4.2.6 Configure the mounting rotation Angle .....	12
4.2.7 Configure the position and velocity projection point of the integrated navigation output .....	12
4.2.8 Querying the version number .....	12
4.3 Saving Parameters .....	12
5. Coordinate system definition .....	13
6. Integrated navigation output protocol .....	14
6.1 nmea protocol .....	14
7. Upgrade the firmware .....	15
7.1 Using the Upper computer software .....	15
7.1.1 Direct Connection of the COM .....	15
7.2 COM Push .....	16
8. Accessories .....	23
9. Update the record .....	24

# NAV680-SA\_Datasheet\_Product Manual (waterproof version)

## 1. Product introduction

### 1.1 Product List

When you open the packing case, please confirm the products in the packing case:


Name	Quantity	Schematic diagram
① NAV680-SA Integrated Navigation system	1	
② Satellite antenna	2	
③ Antenna cable Main antenna model :TNC male head to TNC male head Secondary antenna model :TNC male head to TNC male head	2	
④ Antenna suction cup	2	
⑤ antenna column	2	
⑥ Harness (Includes 3 RS232, 2 CAN, 1 PPS, 1 power port (9-24V)).	1	

Figure 1 Product physical picture

If anything is missing, please contact the sales staff in time

## 1.2 Product Overview

Customized development is made for the navigation scene, which can output the heading and heading based on true North at the same time, and output the high-precision attitude Angle NAV680-SA integrated and the full-frequency high-precision positioning and orientation module UM982 of Core Star. It integrates the high-precision IMU developed by the original pole, and built-in intelligent position fusion algorithm of the original pole multiple models.

## 1.3 Technical Indicators

Table 1

Attitude accuracy	Roll/Pitch :<0.2° rms Heading accuracy :<0.3° rms	
Update rate	100Hz	
GNSS Indicators	positioning accuracy (RMS)	Single point: Horizontal 1.5m/ elevation 2.5m
	Dual antenna orientation Accuracy (RMS)	0.1°/1m baseline
Interface	1×RS422 2 x GNSS antenna interface 1 x power port	

## 2. Hardware composition

### 2.1 Mechanical dimensions

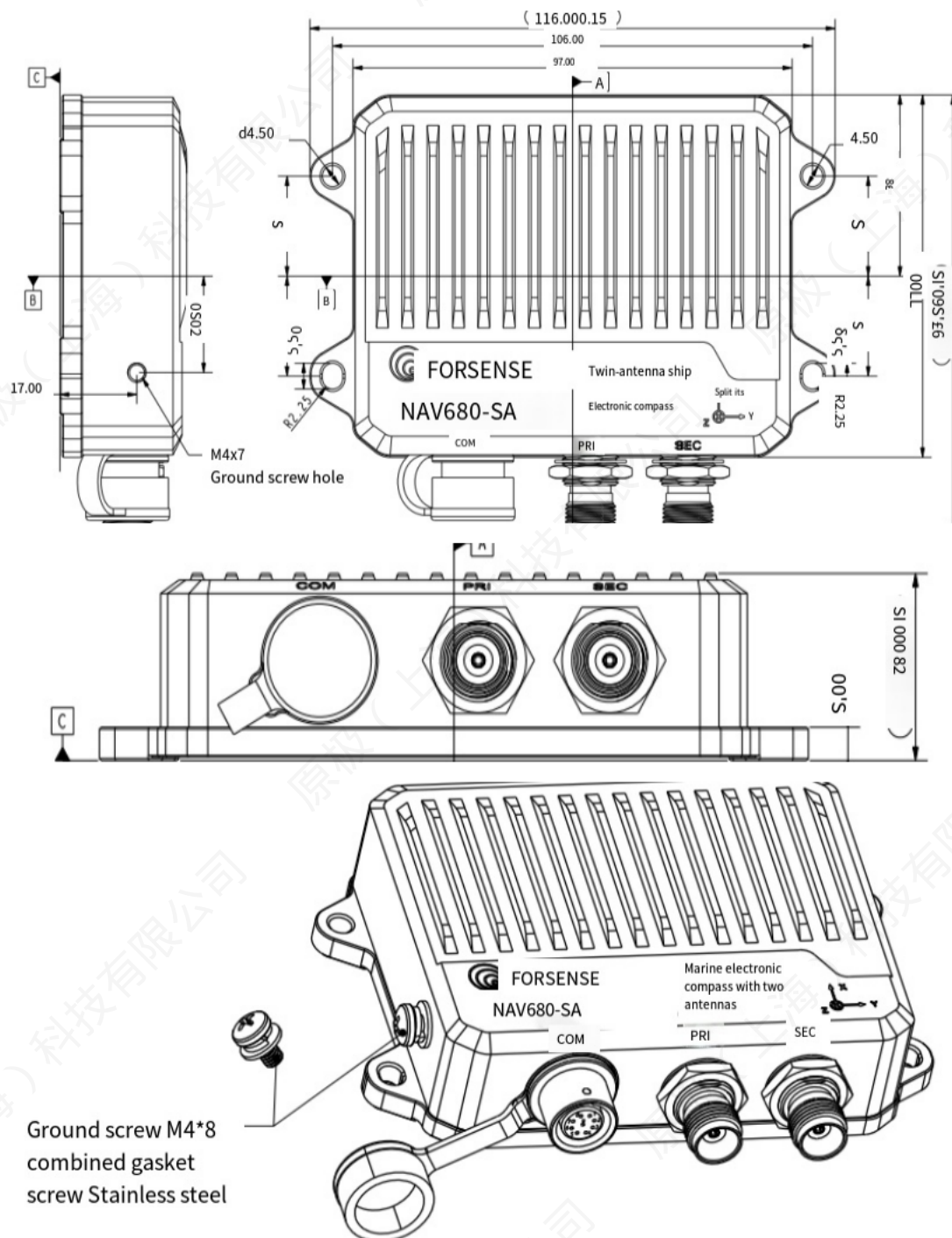
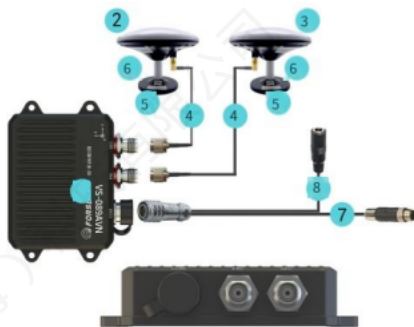


Figure 2 Mechanical dimensions (unit: mm)

## 2.2 Interface Description

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



	Name	Quantity	Remarks
one	NAV680-SA Marine Compass	1	
2	Main antenna (positioning antenna)	1	Salt spray proof
3	Secondary antenna (positioning antenna)	1	Salt spray proof
4	Antenna feeder (20 m)	2 for	Nautical class
5	Antenna sucker	2	
6	Antenna posts	for 2	
7	Harness -RS422 interface	1	
8	Harness - power interface	1	

## 2.3 Data cable interface definition

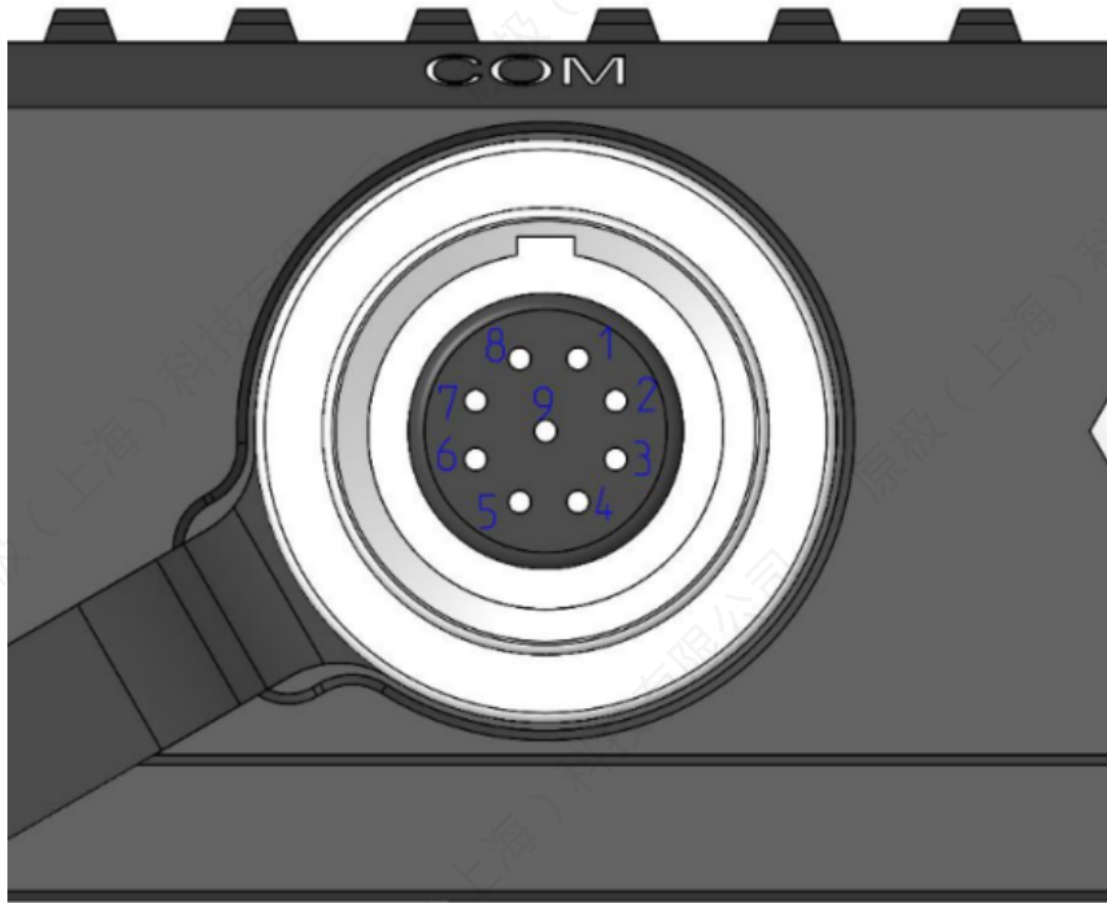


Figure 2 Data cable interface definition diagram

The corresponding pins are defined as follows

Table 2

PIN Sequence	Definition	Ports
1	RS422_TX+	M12 male head,Pin1
2	VIN	DC base
3	RS422_TX-	M12 male head,Pin2
4	GND	DC busbar
5	RS422_RX-	M12 male head,Pin3
6	GND	empty
7	RS422_RX+	M12 male head,Pin4
8	PGND	M12 male head, cable shielding layer
9	PPS	empty



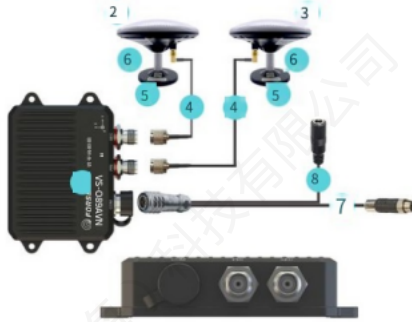
## 2.4 Electrical physical properties

Table 3

Parameters	Symbols	Minimum value	Typical value	Maximum value	Units
Power input	VIN	9	12	32	V
Ground (GND)	GND				
Power consumption	P		1.8		W
Operating Temperature (Tot)	T	-40	\	85	°C
Storage Temperature (Tstg)	T	-40	\	95	°C
Humidity	95% non-condensation				
Size	116 * 85.9 * 28 mm				
Weight	263 g				

### 3. Hardware wiring method

#### 3.1 Hardware wiring Methods



	Name	Quantity	Remarks
one	NAV680-SA Marine Compass	1	
2	Main antenna (positioning antenna)	1	Salt spray proof
3	Secondary antenna (positioning antenna)	1 for	Salt spray proof
4	Antenna feeder (20 m)	2	Nautical class
5	Antenna suckers	for 2	
6	Antenna posts	2	
7	Harness -RS422 interface	1	
8	Harness - power interface	1	

#### 3.2 Antenna

NAV680-SA leads out the PRI and SEC primary and secondary antenna interfaces, without antenna, using a multimeter test, that is, the voltage provided when no load is DC4.8 ~ 5.4V; When the module RF port is connected to the antenna, it can provide DC4.6V±0.2V antenna feed when the working current is 30 ~ 100mA at normal temperature. When NAV680-SA uses an active antenna, pay attention to the 50 ohm impedance matching between the antenna and the antenna.



## 4. Use examples

### 4.1 Equipment Installation

#### 1. Installation requirements

1) Dual antenna installation, you need to install two antennas (main and secondary antennas) on the ship and connect the compass, and install according to the following requirements:

The satellite antenna is in the air and the carrier does not block the antenna. The rays formed by the Dual antenna should be parallel to the ship's body as far as possible.

The main antenna is connected to the PRI interface for positioning and mounting with the stern.

The secondary antenna is connected to the SEC interface for orientation, mounting to the stern.

The distance between the two antennas, that is, the base line length should be greater than 0.5m.

The height of Dual antenna should be consistent as far as possible. The installation diagram is shown in the following figure.

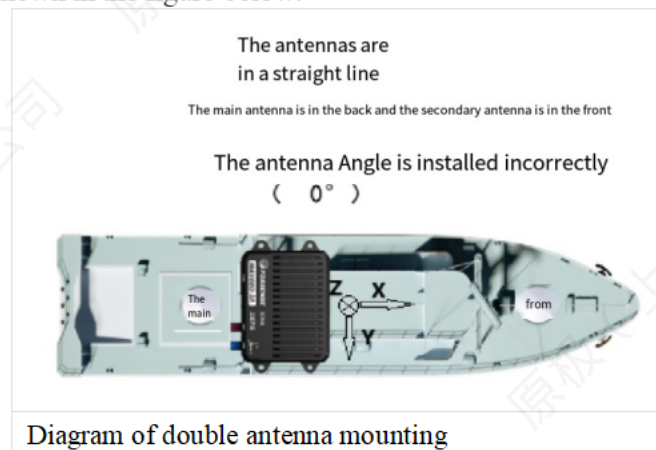
2) Away from interference sources such as electrical devices or systems that can produce electromagnetic interference signals;

3) The GNSS antenna is screwed on the strong magnetic chuck and fixed in the center of the test carrier as far as possible, and it is placed at the highest point of the hull to ensure that it can receive good GNSS signals;

4) In order to obtain the best performance, the distance between the GNSS main antenna and the equipment host should be minimized, especially the horizontal distance.

5) The antenna should be firmly fixed on a rigid plane and avoid being installed in a position with large vibration.

6) The orientation of the X-axis installation should maintain the relationship with the bow direction as shown in the figure below.



The correct installation diagram is below

The X axis faces the bow

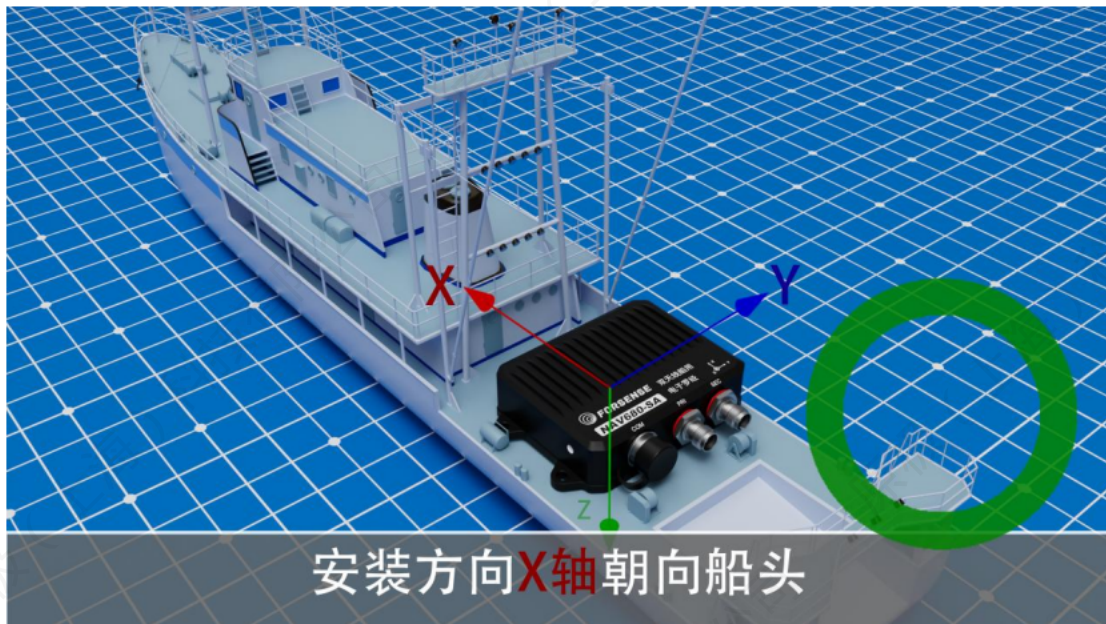
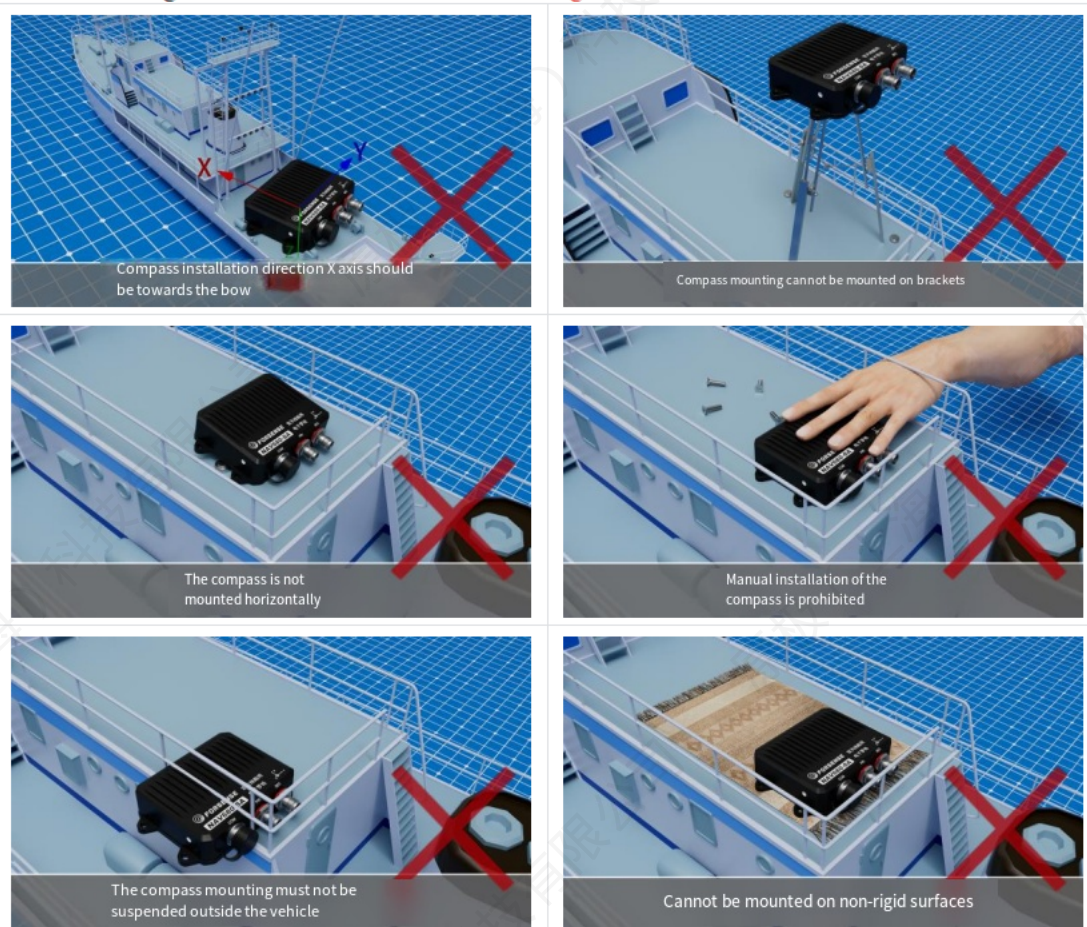


Figure 3 Diagram of proper installation

The following installation methods are all **wrong installation**.





## 4.2 Parameter Settings

### 4.2.1 Configuring the main mast arm

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

Instruction: AT+CLUB\_VECTOR=0.5,-0.6,-1.0\r\n

Reply: GPS\_POS\_X = 0.5, GPS\_POS\_Y = 0.6, GPS\_POS\_Z = 1.0 / r/n

Description: The rod arm vector is the three-dimensional vector (X,Y,Z) of the phase center of the RTK main antenna relative to the phase center of the IMU, in meters. Where,

Measure the distance from the center of the antenna to the center of the device along the X/Y/Z axis of the machine coordinate system, and get three values of X1 Offset/Y1 Offset/Z1 Offset, paying attention to the positive and negative.

In the front right lower hull coordinate system

If the RTK main antenna is in front of the IMU, the number is positive, otherwise it is negative.

If the RTK main antenna is to the right of the IMU, it is positive, otherwise it is negative;

If the RTK main antenna is above the IMU, it is a **negative number**, otherwise it is a positive number (generally the antenna is above the device).

The coordinate system diagram is as follows: (Label should face up; if the IMU is not installed according to the following figure, the orientation should be configured in section 8.2.2)

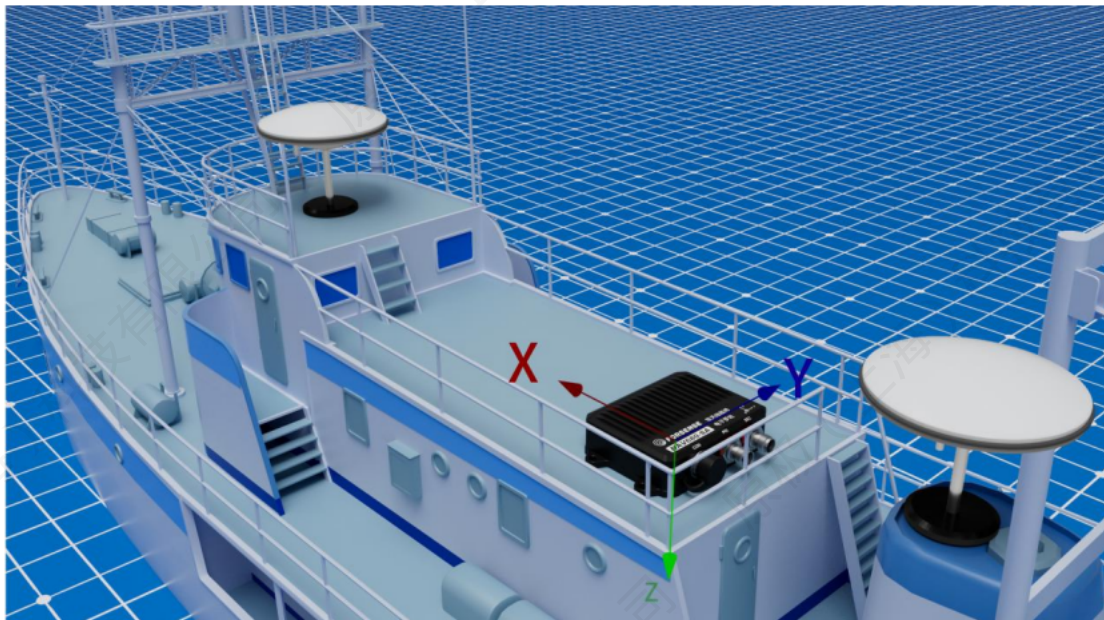


FIG. 5-day pole arm diagram

### 4.2.2 Configure the output NMEA format data stream

If NEMA statement output is configured, 7.2 Integrated Navigation data flow is not output  
To switch to 7.2 Integrated Navigation data stream output, stop the current data stream output  
by following instructions 8.1.11

The configuration instructions are as follows

GPGGA

Example: Output GPGGA statement with a frequency of 5Hz: AT+GPGGA=5\r\n

Answer: OK\r\n

GPRMC

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 (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

### 4.2.3 Configure the current data stream to stop output

Instruction: AT+SETNO\r\n

Answer: OK\r\n

### 4.2.4 Configuring the Baud rate

Only the baud rate can be set to 115200 or 230400. The default baud rate is 115200

If the baud rate of the IMU serial port is set to 230400, the configuration instructions are as follows:

Instruction: AT+BAUD=230400\r\n

Answer: BAUD=230400\r\n

Note: After the configuration instruction is saved, power off and restart will take effect

### 4.2.5 Print all configuration information

To query all configured information, the configuration instructions are:

AT+CONFIG\r\n

### 4.2.6 Configure the mounting rotation Angle

Currently, only the following rotation angles are supported

Rotate the X-axis 180 degrees

z axis rotated 90 180 270

Configuration instructions are as follows

If the installation rotation Angle is 180 degrees X axis rotation, the configuration instructions are:

Instruction: AT+INSTALL\_ANGLE=180,0,0\r\n

Answer: INST\_ANGLE\_X=180.000,INST\_ANGLE\_Y=0.000,INST\_ANGLE\_Z=0.000

If the installation rotation Angle is 180 degrees of Z-axis rotation, the configuration instructions are:

Instruction: AT+INSTALL\_ANGLE=0,0,180\r\n

Answer: INST\_ANGLE\_X=0.000,INST\_ANGLE\_Y=0.000,INST\_ANGLE\_Z=180.000

### 4.2.7 Configure the position and velocity projection point of the integrated navigation output

If you need to configure the result of the projection point set by the output integrated navigation, the configuration instruction is:

Instruction: AT+PROJ\_VECTOR=1.0,2.0,3.0\r\n

Answer: PROJ\_VECTOR\_X=1.0, PROJ\_VECTOR\_Y=2.0, PROJ\_VECTOR\_Z=3.0/r/n

Note: The integrated navigation output default is the antenna phase center coordinate, if you need to output other position coordinates, you need to configure the IMU phase center relative to this projection point of the pole arm vector, the configuration method is the same as 8.1.1 pole arm configuration

### 4.2.8 Querying the version number

AT+VERSION\r\n

## 4.3 Saving Parameters

After all configuration instructions are configured, send the SAVE parameter instruction "AT+SAVE\r\n"

## 5. Coordinate system definition

Figure 7 Diagram of coordinate system



This product coordinate system uses the pre-right-down (FRD) coordinate system, Euler Angle range is as follows:

Rotation around the Z axis: course Angle Yaw range:  $0^{\circ} \sim 360^{\circ}$ ;

Rotation around X axis: Roll Angle roll range:  $-180^{\circ} \sim 180^{\circ}$ ;

Rotation around the Y-axis direction: Pitch Angle range:  $-90^{\circ} \sim 90^{\circ}$ .



## 6. Integrated navigation output protocol

### 6.1 nmea protocol

- Support the output of combined data in nmea format,
- And binary data stream can not be output at the same time, output nmea data stream can not output binary data stream, before switching data stream need to stop the current data stream output according to 7.2.2 instruction.
- Currently, the following statements are supported. Section 7.3 between Configuration Methods

GPGBA

GPRMC

GPHDT (Heading information)

GPVTG (Speed information)

GPZDA (UTC time and date)

GPATT (Primitive Custom message)

The following table describes the format of GPATT

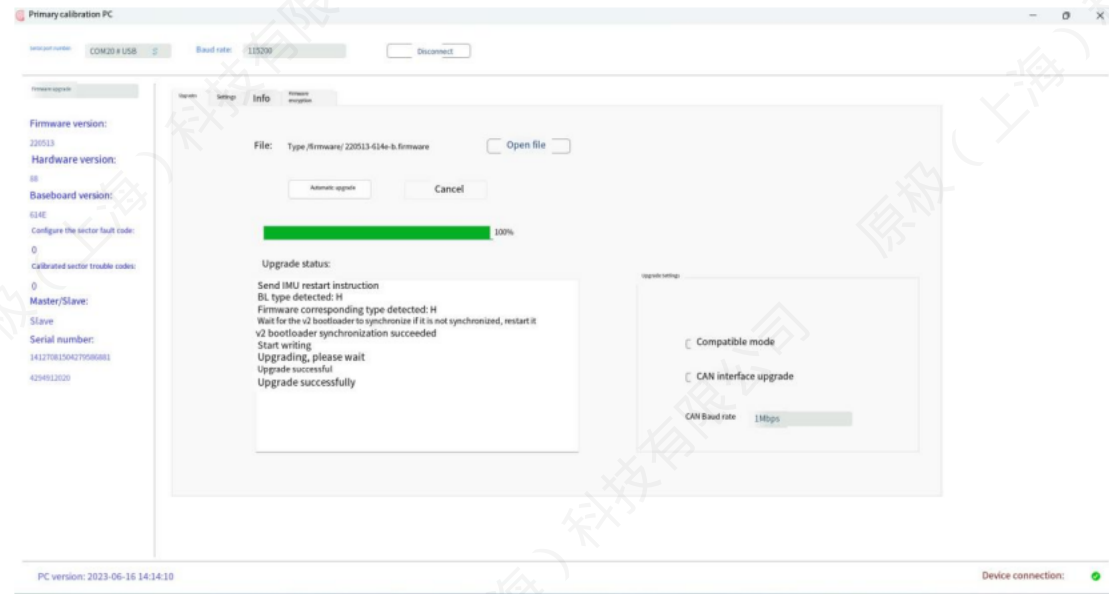
Name	Unit	Format	Example	Description
Sentence Identifier		String	\$GNATT	
Time		hhmmss.sss	170834.000	17:08:34 UTC
Status		Character	one	0: invalid 1: valid
Roll Angle	degree	3 decimal places	4.891	range $\pm 90$ , right side down defined as positive
Indicator for roll		character	R	Roll indicator
Pitch Angle	degree	3 decimal places	3.122	range $\pm 90$ , 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

## 7. Upgrade the firmware

### 7.1 Using the Upper computer software

#### 7.1.1 Direct Connection of the COM

Test the Upper computer software using the Forsense IMU -- select Firmware upgrade -- Open firmware -- click Automatic Upgrade.



## 7.2 COM Push

The upgrade is divided into the following steps:

### Step 1: Send upgrade instructions

Send the upgrade command to tell the module to prepare for the upgrade. After the command is sent, the device will record the upgrade flag bit in a certain area of the flash, and then perform a soft restart. At this time, the IMU module will enter

In BOOTLOADER.

The upper computer sends upgrade instructions as follows:

```
cmd_b[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};
```

The imu will restart and enter the bootloader

### Step 2: Send the **HC32MCU\_FORSENSE** string

After the IMU module enters the bootloader, the IMU module actively sends the HC32\_UPLOAD string. During this period, the Upper computer software needs to send the HC32MCU\_FORSENSE string. Once the IMU module receives the HC32MCU\_for sense string, it will not jump to the APP. Will remain in the bootload r program to be upgraded, and will stop sending any messages

11 After sending the upgrade command, send the following character string to ensure that the imu does not jump to the APP area and is in the waiting state

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

### Step 3: Send synchronization commands

When the IMU is in the upgrade state in the bootloader, it does not actively send information but passively responds to the commands of the Upper computer software. The Upper computer software sends the Send\_CMD\_LONG(0x21,0,0,0,0,0,0) command and waits for the response from the IMU module. After receiving this command, the IMU immediately replies Send CMD ACK (0x64,0x10,0).

Data. Once the Upper computer software determines that it has received this data, it indicates that the synchronization is complete.

Synchronization process between the host computer and imu module:

1. The host sends the synchronization command with the command code 0x21 and waits for the response from the imu  
X21 Send\_CMD\_LONG (0, 0,0,0,0,0,0);
2. The imu module sends the response data after receiving the command code 0x21. The data command is as follows:  
X64 Send\_CMD\_ACK (0, 0 x10, 0).
3. The host computer determines whether it has received the response data from the imu. If yes, it indicates that the synchronization is complete

Note: Send\_CMD\_LONG and Send\_CMD\_ACK functions are shown at the end of the article

### Step 4: Send the erase command

The Upper computer software sends the erasing command

Send\_CMD\_LONG(0x23,0,0,0,0,0,0). After the command is sent, the IMU module erases all the contents of the APP area and tells the Upper computer software the final execution result.

The Upper computer software will send the erase command again according to the execution result. Once the erasing is successful, there is no way to restore the IMU module APP. In addition, the Upper computer software must wait until the IMU erases the response successfully before proceeding to the next step. Otherwise, subsequent upgrades may fail.

1. The host sends the erase command, whose command code is 0x23. After the command is sent, wait for the imu to reply the execution result

```
X23 Send_CMD_LONG (0, 0, 0, 0, 0, 0);
```

2. After receiving the command code 0x23, the imu erases the data in the APP area and sends the final execution result back to the host

```
X64 send_CMD_ACK (0, 0 x10, 0). 11 indicates that the erase was successful
```

```
X64 send_CMD_ACK (0, 0 x11, 0). 11 indicates that the erase failed and needs to be erased again
```

Note: The upper computer must receive a response that the erase was successful before it can proceed to the next step

#### Step 5: Send the upgrade packet

After the erasing is successful, the most important step of sending firmware data is to use the Send\_Upload\_Data function to send data. The upper computer will upgrade the firmware to subcontract, each packet fixed size 64 bytes, the last packet less than 64 bytes according to the actual number of bytes sent. Each frame of data contains the valid data length and the packet's offset address in the entire firmware. Each time the host computer sends a frame of data, it must wait for the response of the IMU module and judge that the IMU has successfully obtained this frame of data before sending the next frame. After the IMU module successfully receives the data packet from the host computer, it sends the response data and writes it to the specified flash address according to the offset address.

If the flash fails to be written, the command fails to be sent, and the command is not sent if the flash fails to be written.

For example, the upgrade file uint8\_t Upgrade\_Data[1000] is sent to the imu module

// Send the first packet:

1.1 Send 0 to 63 bytes to imu module by 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, and 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 receiving data successfully, the imu sends the response data and writes the response data to the specified flash address based on the offset address

```
X753d send_CMD_ACK (0, 0x00 to 0).
```

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

The imu module sends the send\_CMD\_ACK (0x64, 0x11, 0) command if the data fails to be written to the flash.

// Send the second packet:

2.1 After receiving the imu response data, the host sends the second packet Send\_Upload\_Data (0x27, 0, 0x40, 0x40, Upgrade\_Data+0x40).

2.2 After receiving the second packet of data, the imu sends send\_CMD\_ACK(0x753D, 0x00, 0x40);

2.3 If the imu fails to write data into the flash, send\_CMD\_ACK(0x64, 0x11, 0) is sent.

// Send the third packet

3.1 After receiving the imu response data, the host sends the third packet Send\_Upload\_Data (0x27, 0, 0x80, 0x40, Upgrade\_Data+0x80).

3.2 After receiving the third packet of data, the imu sends send\_CMD\_ACK(0x753D, 0x00, 0x80);

3.3 If the imu fails to write data into the flash, send\_CMD\_ACK(0x64, 0x11, 0) is sent.

// Send the fifteenth packet:

Upper receive 15.1 to imu response data, to send packet data Send\_Upload\_Data 15 (x27 0, 0, 0x380, 0x40, Upgrade\_Data + 0x380);

15.2 After receiving the 15th packet, the imu sends send\_CMD\_ACK (0x753D, 0x00, 0x380);

15.3 If data fails to be written to the flash by the imu, send\_CMD\_ACK(0x64, 0x11, 0) is sent.

// Send the 16th packet:

15.1 After receiving the imu response data, the host sends the 16th packet Send\_Upload\_Data (0x27, 0, 0x3C0, 0x28, Upgrade\_Data+0x3C0).

15.2 After receiving the 16th packet of data, the imu sends send\_CMD\_ACK(0x753D, 0x00, 0x3C0);

15.3 If the imu fails to write data to the flash, send\_CMD\_ACK(0x64, 0x11, 0) is sent. End of sending

#### Step 6: Get the CRC check code

To upgrade firmware, run the imu614e-B #CRC1373387121.firmware command. The CRC string is followed by a calculated CRC value. After the upgrade file is sent, the host sends a verification command to check whether the IMU receives an incorrect upgrade file. After sending Send\_CMD\_LONG(0x29, 0, 0, 0, 0) command, the host computer obtains the CRC check code calculated by IMU module.

If the upper computer judges that the CRC check value is incorrect, it should start the upgrade from the fourth step of the erase command.



The upper computer sends the command to obtain the crc verification code and waits for the imu response

```
X29 Send_CMD_LONG (0, 0,0,0,0,0,0);
```

imu module sends crc check value data in response:

```
send_CMD_ACK(0x753C,0x10,crc32_data);
```

The crc32\_data value is the crc32 data calculated by the imu

#### Step 7: Send the restart command

After the host computer judges that the crc check value is correct, it sends the restart command, and the upgrade succeeds

After the crc check value is correct, send the restart command:

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

After the firmware upgrade is complete, power off and restart, you can judge whether the upgrade is successful by reading the version number.



Function definition:

1. The Send\_CMD\_LONG function is defined 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;
    uint32_t param5;
    uint32_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)l
{
    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 Send\_CMD\_ACK function is defined 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=crc32(check_crc, (uint8_t*)&data_cmd_ack, len);

    Cout((uint8_t*)&data_cmd_ack, sizeof(data_cmd_ack));
}
```

The Send\_Upload\_Data function is defined 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(UPLD_DATA)-10;

upload_data.cmd=cmd;

for(int i=0;i<size;j++)

upload_data.param[i]=*(param+i);

upload_data.offset=offset;

upload_data.size=size;

int len=sizeof(UPLD_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(UPLD_DATA));

}

```

#### 4. The CRC32 check function is as follows:

```

static const uint32_t crc32_tab[] = {

    0x00000000, 0x77073096, 0xee0e612c, 0x990951ba, 0x076dc419, 0x706af48f,
    0x9e63a535, 0x9e6495a3, 0x0edb8832, 0x79dcb8a4, 0x0e0d5e91e, 0x97d2d988,
    0x09b64c2b, 0x7eb17cbd, 0xe7b82d07, 0x90bf1d91, 0x1db71064, 0x6ab020f2,
    0xf3b97148, 0x84be41de, 0x1dad47d, 0x6ddde4eb, 0xf4d4b551, 0x83d385c7,

    0x136c9856, 0x646ba8c0, 0xdd6297a, 0x8a65c9ec, 0x14015c4f, 0x63066cd9, 0xf3a@f3d63, 0x8d080df5, 0x3b6e20c8,
    0x4c69105e, 0xd56041e4, 0xa2677172, 0x3c03e4d1, 0x4b04d447, 0xd20d85fd, 0xa50ab56b, 0x35b5a8fa, 0x42b2986c,

    0xdbbbc9d6, 0xacbcf940, 0x32d86ce3, 0x45df5c75, 0xdcd60dcf, 0xabd13d59,
    0x26d930ac, 0x51de003a, 0xc8d75180, 0xbfdd06116, 0x21b4f4b5, 0x56b3c423,
    0xcfb99599, 0xb8bda50f, 0x2802b89e, 0x5f058808, 0xc60cd9b2, 0xb10be924,
    0x2ff677c7, 0x58684c11, 0xc1611dab, 0xb6662d3d, 0x76dc4190, 0x01db7106,
    0x98d220bc, 0xefd5102a, 0x71b18589, 0x06b6b51f, 0x9fbfe4a5, 0xe8b8d433,
    0x7807c9a2, 0xf00f934, 0x9609a88e, 0xe10e9818, 0x7f6a0dbb, 0x086d3d2d,

    0x91646c97, 0xe6635c01, 0xb66b51f4, 0x1c6c6162, 0x856530d8, 0xd262004e, 0xc6c0695ed, 0x1b01a57b, 0x8208f4c1,
    0xd50fc457, 0x65b0d9c6, 0x12b7e950, 0x8bbbeb8e, 0xfcb9887c, 0x62dd1ddf, 0x15da2d49, 0x8cd37cf3, 0xfbd44c65,
    0x4db26158, 0x3ab551ce, 0xa3bc0074, 0xd4bb30e2, 0x4adfa541, 0x3dd895d7,

    0xa4d1c46d, 0xd3d6f4fb, 0x4369e96a, 0x346ed9fc, 0xad678846, 0xda60b8d0,
    0x44042d73, 0x33031de5, 0xaa0a4c5f, 0xdd@d7cc9, 0x5005713c, 0x270241aa,
    0xbe0b1010, 0xc90c2086, 0x5768b525, 0x206f85b3, 0xb966d409, 0xce61e49f,
    0x5edef90e, 0x29d9c998, 0xb@d09822, 0xc7d7a8b4, 0x59b33d17, 0x2eb40d81,

    0xb7bd5c3b, @xc@ba6cad, @xedb88320, 0x9abfb3b6, 0x03b6e20c, 0x74b1d29a, 0xeadd54739, 0x9dd277af,
    0x04db2615, 0x73dc1683, 0xe3630b12, 0x94643b84,

    0x0d6d6a3e, 0x7a6a5aa8, 0xe40ecf0b, 0x9309ff9d, 0x0a00ae27, 0x7d079eb1,

    0xf00f9344, 0x8708a3d2, 0x1e01f268, 0x6906c2fe, 0xf762575d, 0x806567cb, 0x196c3671, 0x6e6b06e7, 0xfed41b76,
    0x89d32be0, 0x10da7a5a, 0x67dd4acc,

```

## 8. Accessories

Standard accessories:



NAV680-SA mainframe



NAV680-SA harness



Main antenna (positioning antenna)



Secondary antenna (directional antenna)



Salt-proof fog wire feeder (20m)

## 9. Update the record

Address for the latest manual version: [NAV680-SA\\_Datasheet\\_Product Manual \(waterproof version\)](#)

Version	Dates	Status/Comments
Version 1.0	2024.01.09	First Issue