

NAV680-SA

高精度双天线船用罗经
使用手册



原极（上海）科技有限公司

在使用本产品之前，请务必先仔细阅读本使用手册。

目录

1. 产品简介	1
1.1 产品清单	1
1.2 技术指标	2
2. 硬件组成	2
2.1 机械尺寸	2
2.2 接口说明	3
2.3 数据线接口定义	4
2.3 电气物理特性	5
2.4 天线	5
3. 使用范例	6
3.1 设备安装	6
4. 坐标系定义	8
5. 输出协议	9
5.1 二进制协议-组合导航数据流	9
5.2 nmea 协议	11
6. 参数配置	12
6.1 通过串口配置	12
6.1.1 配置 GNSS 双天线安装角	12
6.1.2 配置输出 NMEA 格式数据流	12
6.1.3 配置当前数据流停止输出	13
6.1.4 配置波特率	13
6.1.5 打印所有配置信息	13
6.1.6 查询版本号	13
6.1.7 配置安装旋转角	14
6.1.8 保存参数	14
7. 固件升级	15
7.1 通过上位机	15
7.1.1 R422 串口直连	15
7.2 串口推送	16
8. 附件	23
9. 更新记录	24

NAV680-SA_Datasheet_产品手册

1. 产品简介

1.1 产品清单

在您打开包装箱时，请确认包装箱中产品：


名称	数量	示意图
①NAV680-SA 双天线罗经	1	
②③卫星天线（不区分主副）	2	
④天线馈线 (型号:20 米长, TNC 公转公)	2	
⑤天线吸盘	2	
⑥天线柱 (20cm)	2	
⑦⑧集线束 (含 1 个 RS422, 1 个电源接口 (9-24V))	1	
⑨电源适配器 (12V, 1A)	1	

图 1 产品实物图

若有缺失，请及时与销售人员联系

1.2 技术指标

表 1

组合导航 指标	姿态精度	Roll/Pitch: $<0.2^{\circ}$ rms
	艏向精度	$<0.3^{\circ}$ rms
	航向精度	$<0.3^{\circ}$ rms
	更新率	100Hz
	速度精度 (RMS)	0.02m/s
IMU 指标	陀螺量程	$\pm 500^{\circ}/s$
	陀螺零偏不稳定性	4deg/h @1 σ
	加速度计量程	$\pm 6g$
	加速度计零偏稳定性	0.04mg @1 σ
RTK 指标	定位精度 (RMS)	单点: 1.5m RTK: 1cm+1ppm
	双天线定向精度 (RMS)	$0.1^{\circ}/1m$ 基线
	速度精度 (RMS)	0.03m/s
	PPS 精度 (RMS)	20ns
	更新率	20Hz
	RTK 初始化时间	$<5s$
接口		1 \times RS422 2 \times GNSS 天线接口 1 \times 电源接口

2. 硬件组成

2.1 机械尺寸

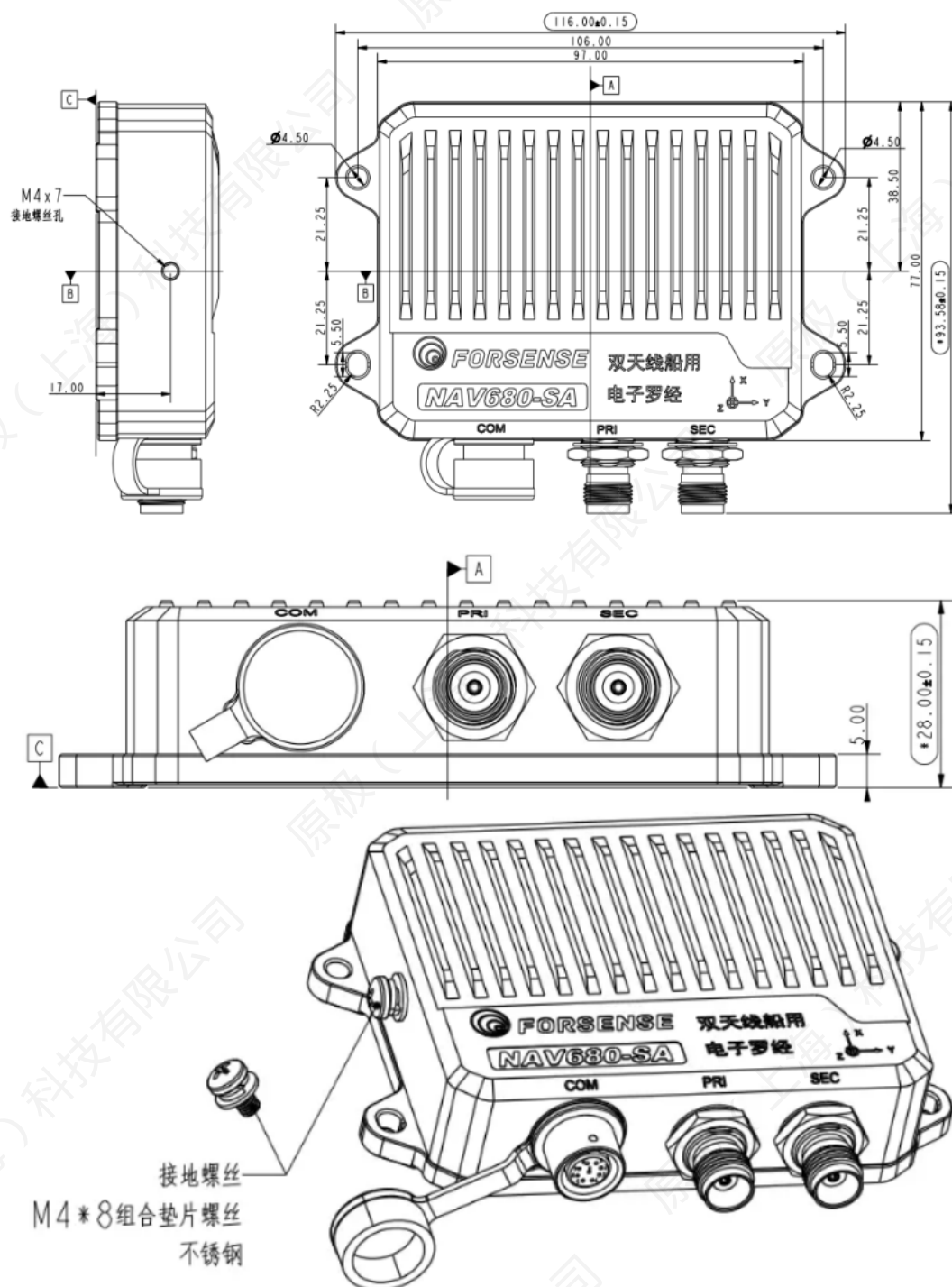
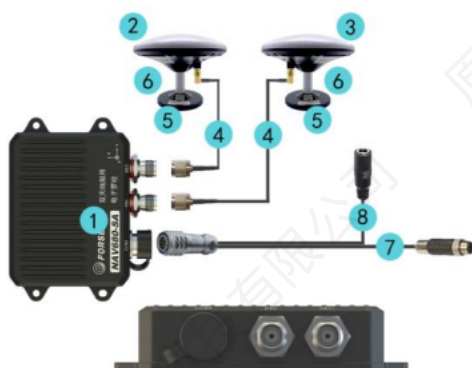


图 1 机械尺寸 (单位: mm)

2.2 接口说明

NAV680-SA 包含 COM 总线接口，引出电源线和信号线，PRI 主天线和 SEC 副天线。



名称	数量
1 NAV680-SA船用罗经	1个
2 主天线（定位天线）	1个
3 副天线（定位天线）	1个
4 天线馈线(20米)	2个
5 天线吸盘	2个
6 天线柱	2个
7 集线束-RS422接口	1个
8 集线束-电源接口	1个

2.3 数据线接口定义

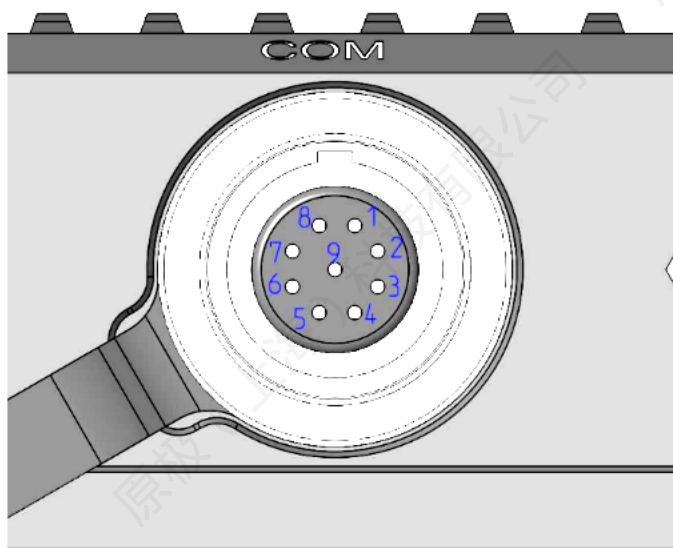


图 2 数据线接口定义图

对应引脚定义如下

PIN 序	定义	端口
1	RS422_TX+	M12 公头, Pin1
2	VIN	DC 母座
3	RS422_TX-	M12 公头, Pin2
4	GND	DC 母座
5	RS422_RX-	M12 公头, Pin3
6	GND	空
7	RS422_RX+	M12 公头, Pin4
8	PGND	M12 公头, 线缆屏蔽层
9	PPS	空

2.4 电气物理特性

表 2

参数	符号	最小值	典型值	最大值	单位
电源输入	VIN	9	12	32	V
电源地	GND				
功耗	P		1.8		W
使用温度	T	-40	\	85	°C
存储温度	T	-40	\	95	°C
防水等级	IP67				
湿度	95%非凝露				
尺寸	116*85.9*28mm				
重量	263g				

2.5 天线

NAV680-SA 引出 PRI 和 SEC 主副天线接口，不接天线，使用万用表测试，即空载时提供电压为 DC4.8~5.4V；模块射频口接天线时，常温下，工作电流为 30~100mA 时测试，能对外提供 DC4.6V±0.2V 的天线馈电。NAV680-SA 采用有源天线时注意与天线间的 50 欧姆阻抗匹配。

3. 使用范例

3.1 设备安装

1、安装要求

1) 双天线安装，需要您安装两个天线（主，副天线）在船上并连接罗经，并按以下要求安装

卫星天线对空且载体对天线无遮挡，双天线形成的射线应与船身尽量保持平行。

主天线连接 PRI 接口，用于定位，装与船尾，

副天线连接 SEC 接口，用于定向，装与船尾，

两天线间距离，即基线长度需大于 0.5m，

双天线高度尽量保持一致，安装示意图如图 20 所示。；

2) 远离干扰源如电器器件或系统等能产生电磁干扰信号的位置；

3) GNSS 天线拧到强磁吸盘上并尽量固定摆放在测试载体的中心位置，尽可能的将其安置于船体的最高处以保证能够接收到良好的 GNSS 信号；

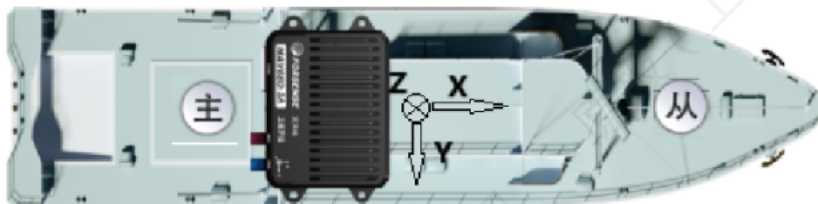
4) 为获取最佳性能，应尽量减小 GNSS 主天线与设备主机之间的距离，尤其是水平距离。

5) 应牢靠固定在刚性平面上，避免安装在震动大的位置。

6) X 轴安装朝向应与船头方向保持下图所述关系。

**天线一前一后
在同一直线
主天线在后，副天线在前**

**天线角度安装偏差
(0°)**



双天线安装示意图

正确安装示意图如下

X 轴朝向车头

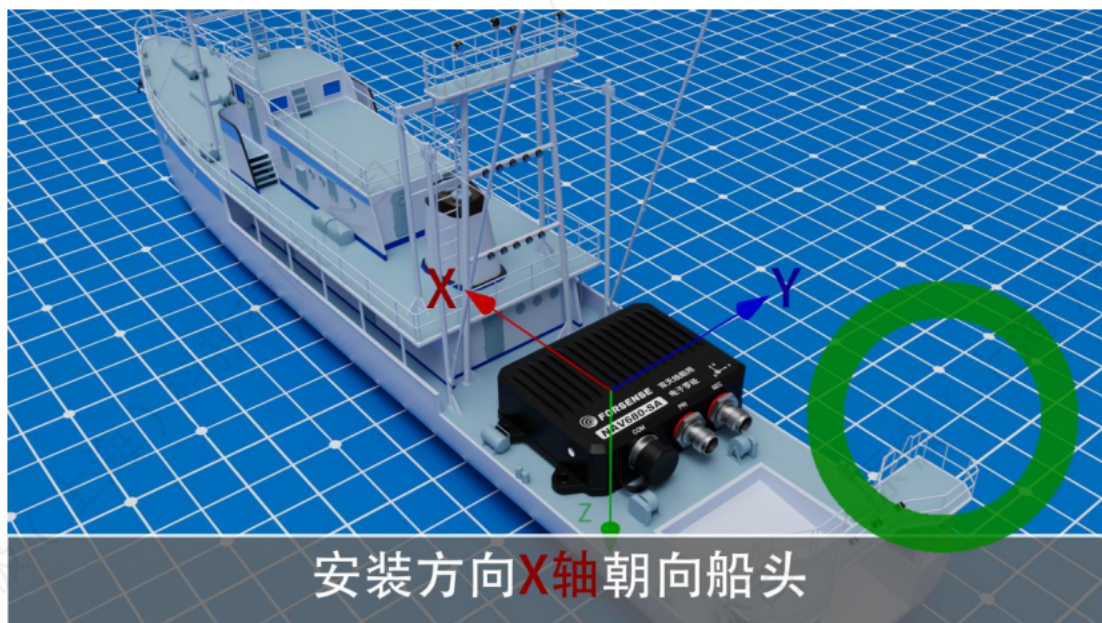
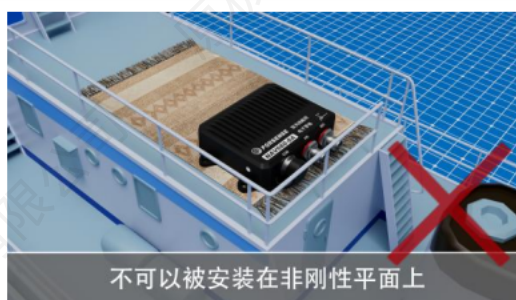
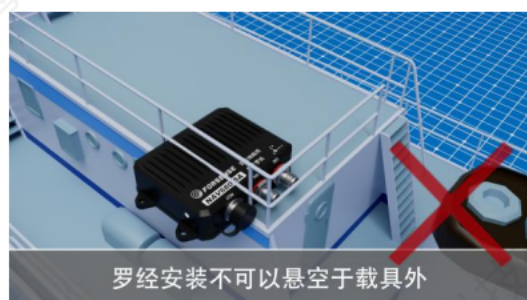
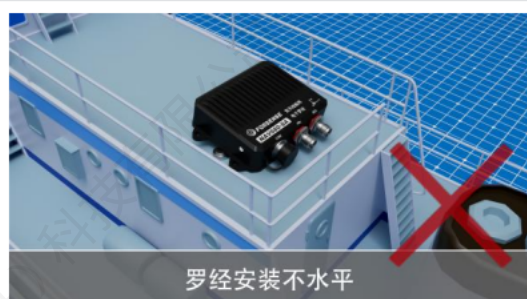
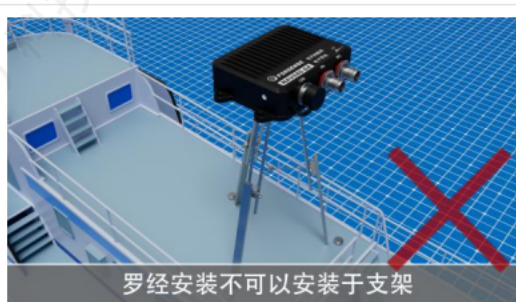
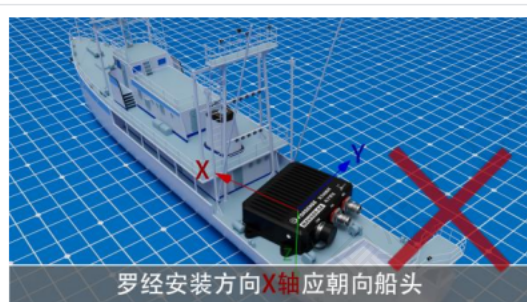


图 3 正确安装示意图

以下安装方式均是**错误安装**。



4. 坐标系定义

图 7 坐标系示意图



本产品坐标系使用 前-右-下 (FRD) 坐标系，欧拉角范围如下：

绕 Z 轴方向旋转：航向角 Yaw 范围： $0^{\circ} \sim 360^{\circ}$ ；

绕 X 轴方向旋转：横滚角 Roll 范围： $-180^{\circ} \sim 180^{\circ}$ ；

绕 Y 轴方向旋转：俯仰角 Pitch 范围： $-90^{\circ} \sim 90^{\circ}$ 。

5. 输出协议

5.1 二进制协议-组合导航数据流

注意:

- CRC 校验为从帧头开始, 不包含 CRC 校验位本身, 该帧所有字节的 CRC 校验, 校验计算方式和例程见附录。
- 帧长为除去帧头, 帧 ID, 帧长和校验位之外的所有数据字节总数。
- 小端模式, 先发送低字节。

内容	类型	相对位置
帧头 1: 0xAA	UInt8	0
帧头 2: 0x55	UInt8	1
帧 ID: 0x0166	UInt16	2
帧长: 0x005E	UInt16	4
GPS 周内秒 (ms)	UInt32	6
GPS 周计数	UInt16	10
纬度 (度 $\times 10000000$)	Int32	12
经度 (度 $\times 10000000$)	Int32	16
高度 (毫米)	Int32	20
北向速度 (m/s)	Float	24
东向速度 (m/s)	Float	28
地向速度 (m/s)	Float	32
横滚角 (度)	Float	36
俯仰角 (度)	Float	40
艏向角 (度)	Float	44
双天线航向 (度)	Float	48
航迹角 (度)	Float	52
加速度计 X 轴 (g)	Float	56
加速度计 Y 轴 (g)	Float	60
加速度计 Z 轴 (g)	Float	64
陀螺仪 X 轴 (deg/s)	Float	68
陀螺仪 Y 轴 (deg/s)	Float	72

陀螺仪 Z 轴 (deg/s)	Float	76
IMU 温度 (°C)	Float	80
RTK 定位状态(同 GGA 中定位状态) 0:未定位 1:单点定位 2:伪距差分定位 4:固定解 5:浮点解	Uint8	84
卫星数量	Uint8	85
差分延时	Uint8	86
双天线定向状态 50 表示已定向 其他表示未定向	Uint8	87
位置精度因子 (cm) 组合导航初始化后有效	Uint16	88
状态位: bit0:1 表示 RTK 数据有效, 0 表示无效 Bit1:1 表示 PPS 信号有效, 0 表示无效 Bit2:1 表示组合导航已初始化, 0 表示未初始化	Uint16	90
预留 1	Uint32	92
预留 2	Uint32	96
CRC 校验	Uint32	100

5.2 nmea 协议

- 支持按 nmea 格式输出组合后数据，
- 与二进制数据流无法同时输出，输出 nmea 数据流则不能输出二进制数据流，切换数据流前需先按 7.2.2 指令停止当前数据流输出。
- 目前支持以下语句。配置方式见 7.3 章节

GPGBA

GPRMC

GPHDT （航向信息）

GPVTG （地面速度信息）

GPZDA （UTC 时间及日期）

GPATT （原极自定义报文）

GPATT 格式如下表

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. 参数配置

6.1 通过串口配置

6.1.1 配置 GNSS 双天线安装角

若配置 gnss 双天线安装角为 0 度，则配置指令为：

指令：AT+RTK_ANGLE=0\r\n

应答：ANGLE=0\r\n

说明：安装角为主天线指向副天线的射线与车头方向的夹角，顺时针为正，逆时针为负，角度输入范围-180° ~180°

注意：配置指令保存后需断电重启生效；双天线间隔距离需大于 50cm、速度投影点

若配置输出组合导航设定的投影点结果，则配置指令为：

指令：AT+PROJ_VECTOR=1.0,2.0,3.0\r\n

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

说明：组合导航输出默认为天线相位中心坐标，若需输出其他位置坐标，则需配置 IMU 相位中心相对此投影点位置的杆臂向量，配置方法同 8.1.1 杆臂配置

6.1.2 配置输出 NMEA 格式数据流

若配置 NEMA 语句输出，则 7.2 组合导航数据流不输出

如需切换为 7.2 组合导航数据流输出，需先按 8.1.11 指令停止当前数据流输出

配置指令如下

GPGLA

例：以 5hz 频率输出 GPGLA 语句：AT+GPGLA=5\r\n

应答：OK\r\n

GPRMC

例：以 1hz 频率输出 GPRMC 语句：AT+GPRMC=1\r\n

应答：OK\r\n

GPHDT（航向信息）

例：以 1hz 频率输出 GPHDT 语句：AT+GPHDT=1\r\n

应答：OK\r\n

GPVTG（地面速度信息）

例：以 1hz 频率输出 GPVTG 语句： AT+GPVTG=1\r\n

应答： OK\r\n

GPZDA (UTC 时间及日期)

例：以 1hz 频率输出 GPZDA 语句： AT+GPZDA=1\r\n

应答： OK\r\n

GPATT (自定义报文)

例：以 1hz 频率输出 GPATT 语句： AT+GPATT=1\r\n

应答： OK\r\n

若配置停止输出

指令： AT+SETNO\r\n

应答： OK\r\n

6.1.3 配置当前数据流停止输出

指令： AT+SETNO\r\n

应答： OK\r\n

6.1.4 配置波特率

默认波特率为 115200

例：若配置 IMU 串口波特率为 230400，则配置指令为：

指令： AT+BAUD=230400\r\n

应答： BAUD=230400\r\n

注意：配置指令且保存后需断电重启生效

6.1.5 打印所有配置信息

若查询所有配置过的信息，则配置指令为：

AT+CONFIG\r\n

6.1.6 查询版本号

AT+VERSION\r\n

6.1.7 配置安装旋转角

当罗经未按默认朝向安装时，可配置旋转角

目前仅支持如下旋转角度

x 轴旋转 180 度

z 轴旋转 90 180 270

配置指令如下

若 安装旋转 角度为 X 轴旋转 180 度，则配置指令为：

指令：AT+INSTALL_ANGLE=180,0,0\r\n

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

若 安装旋转 角度为 Z 轴旋转 180 度，则配置指令为：

指令：AT+INSTALL_ANGLE=0,0,180\r\n

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

6.1.8 保存参数

以上包括杆臂，数据流，输出频率等参数，配置完以后都需要输入保存指令，重启后才能生效

指令：AT+SAVE\r\n

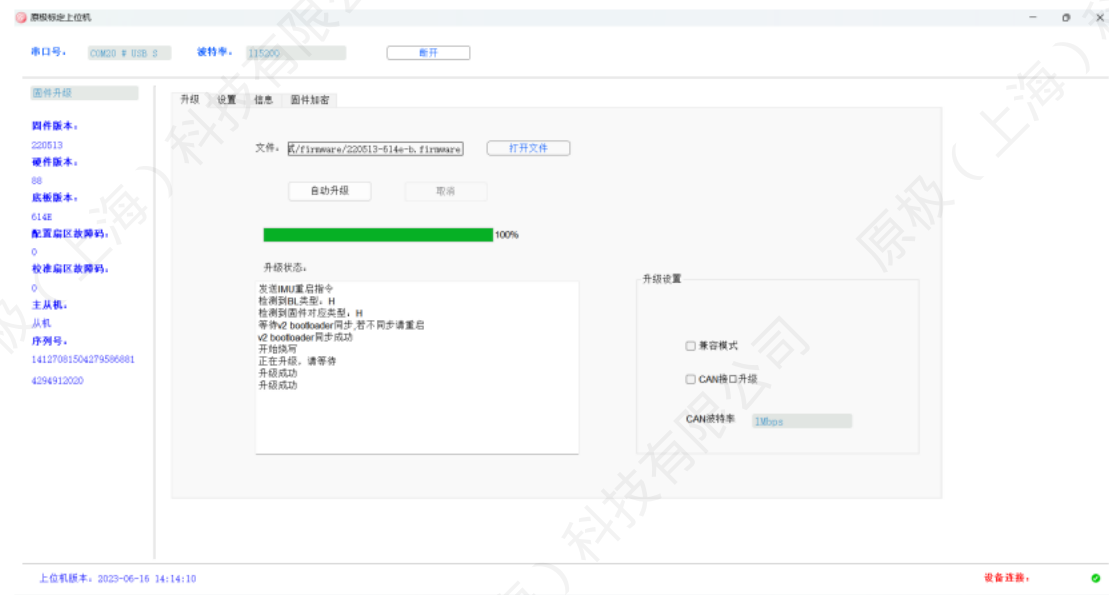
应答：OK\r\n

7. 固件升级

7.1 通过上位机

7.1.1 R422 串口直连

使用原极 IMU 测试上位机——选择固件升级——打开固件——点击自动升级。



7.2 串口推送

升级总共分为以下几个步骤：

第一步：发送升级指令

发送升级指令，告诉模块做好升级准备，命令发送完成后，设备会在 flash 某个区域记录下升级标志位，然后进行软重启，此时 imu 模块将进入

BOOTLOADER 中。

上位机发送升级指令如下：

```
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};
```

此时 imu 模块将会重启后进入 bootloader 中

第二步：发送 HC32MCU_FORSENSE 字符串

imu 模块进入 bootloader 中后，imu 模块会主动发送 HC32_UPLOADER 字符串，在此期间，上位机需要发送 HC32MCU_FORSENSE 字符串，imu 模块一旦收到此字符串，将不会跳转进 APP，将留在 bootloader 中处于程序待升级状态，同时也

会停止主动发送任何消息

//发送完升级指令后，紧接着发送如下字符串，确保 imu 模块不会跳转到 APP 区域，同时处于待升级状态

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

第三步：发送同步指令

imu 模块在 bootloader 中处于待升级状态时，不会主动发送信息，只会被动的响应上位机的指令。此时上位机发送命令 Send_CMD_LONG(0x21, 0, 0, 0, 0, 0, 0) 后，等待 imu 模块的响应。imu 收到此命令后，立即回复 Send_CMD_ACK(0x64, 0x10, 0)

数据。上位机一旦判断接收到此数据，就表示完成了同步。

上位机与 imu 模块同步过程：

1. 上位机发送同步命令，命令码为 0x21，发送后等待 imu 模块响应

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

2. imu 模块收到 0x21 的命令码后，发送响应数据，数据命令如下：

```
Send_CMD_ACK(0x64, 0x10, 0);
```

3. 上位机判断是否收到了 imu 的响应数据，如收到则表示完成了同步

注：Send_CMD_LONG 函数与 Send_CMD_ACK 函数见文章末尾

第四步：发送擦除命令

上位机发送擦除命令 `Send_CMD_LONG(0x23, 0, 0, 0, 0, 0, 0)`，指令发送后，imu 模块 会将 APP 区域的内容全部擦除干净，并将最终的执行结果告诉上位机。上位机根据执行结果是否要再次发送擦除命令。一旦擦除成功，imu 模块的 APP 将无法还原。另外上位机必须等到 imu 擦除成功的响应后，才能进行下一步，否则有可能造成后续的升级失败。

1. 上位机发送擦除命令，命令码为 `0x23`，发送完此命令后耐心等待 `imu` 回复执行结果

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

2. `imu` 收到 `0x23` 的命令码后，将会把 APP 区域数据擦除干净，并将最终的执行结果回复给上位机

```
send_CMD_ACK(0x64, 0x10, 0); // 表示擦除成功
```

```
send_CMD_ACK(0x64, 0x11, 0); // 表示擦除失败，需要重新擦除
```

注意：上位机必须收到擦除成功的响应，才能进行下一步

第五步：发送升级数据包

擦除成功后，将进入最重要的发送固件数据环节，发送数据使用函 `Send_Upload_Data` 进行发送。上位机将升级固件进行分包，每包固定大小 64 字节，最后一包不足 64 字节的按照实际字节数进行发送。每一帧数据包有效数据长度与此包在整个固件中的偏移地址。上位机每发送一帧数据，必须等待 IMU 模块的响应，判断 IMU 成功获取到此帧数据后，再发送下一帧。IMU 模块成功收到上位机的数据包后，会发送响应数据，并根据偏移地址写入指定的 flash 地址。

如果写入 flash 失败，发送失败命令，写入成功则不发。

```
例：将大小 1000 字节的升级文件 uint8_t Upgrade_Data[1000]发送给 imu 模块

//发送第一包：

1.1 通过函数将 0~63 字节发送到 imu 模块
Send_Upload_Data (0x27,0,0,0x40,Upgrade_Data);
//上面函数第一参数 0x27 为固定值，第二参数 0 为固定值，第三参数 0 为偏移地址，
    第四参数 0x40 为有效字节长度，第五参数为发送数据的首地址

1.2 imu 成功收到数据后，会发送响应数据，并根据偏移地址写入指定的 flash 地址
send_CMD_ACK(0x753D,0x00,0);
//上面函数第一参数 0x753D 为固定值，第二参数 0 为固定值，第三参数为偏移地址。
imu 模块将数据写入 flash 过程中如果写入失败将发送失败命令 send_CMD_ACK(0x64,0x11,0);

//发送第二包：

2.1 上位机收到 imu 响应数据后，将发送第二包数据 Send_Upload_Data (0x27,0,0x40,0x40,Upgrade_Data+0x40);
2.2 imu 成功收取到第二包数据后，发送响应： send_CMD_ACK(0x753D,0x00,0x40);
2.3 imu 模块将数据写入 flash 过程中如果写入失败将发送失败命令 send_CMD_ACK(0x64,0x11,0);

//发送第三包：

3.1 上位机收到 imu 响应数据后，将发送第三包数据 Send_Upload_Data (0x27,0,0x80,0x40,Upgrade_Data+0x80);
3.2 imu 成功收取到第三包数据后，发送响应： send_CMD_ACK(0x753D,0x00,0x80);
3.3 imu 模块将数据写入 flash 过程中如果写入失败将发送失败命令 send_CMD_ACK(0x64,0x11,0);

-----
//发送第十五包：

15.1 上位机收到 imu 响应数据后，将发送第十五包数据 Send_Upload_Data (0x27,0,0x380,0x40,Upgrade_Data+0x380);
15.2 imu 成功收取到第十五包数据后，发送响应： send_CMD_ACK(0x753D,0x00,0x380);
15.3 imu 模块将数据写入 flash 过程中如果写入失败将发送失败命令 send_CMD_ACK(0x64,0x11,0);

//发送第十六包：

15.1 上位机收到 imu 响应数据后，将发送第十六包数据 Send_Upload_Data (0x27,0,0x3C0,0x28,Upgrade_Data+0x3C0);
15.2 imu 成功收取到第十六包数据后，发送响应： send_CMD_ACK(0x753D,0x00,0x3C0);
15.3 imu 模块将数据写入 flash 过程中如果写入失败将发送失败命令 send_CMD_ACK(0x64,0x11,0);

发送结束
```


第六步：获取 CRC 校验码

一般情况下升级固件的命令规则为 imu614e-b#CRC1373387121.firmware，CRC 字符串后面紧跟着的是已经计算好的 CRC 校验值。升级文件发送完成后，上位机需要发送校验指令，以判断 IMU 模块收到的升级文件是否有误。

上位机发送 Send_CMD_LONG (0x29, 0, 0, 0, 0, 0, 0) 命令后，获取 IMU 模块自身计算的 CRC 校验码，

如果上位机判断 CRC 校验值有误，应该从第四步擦除命令开始进行重新升级。

```
上位机发送获取 crc 校验码指令，等待 imu 响应
Send_CMD_LONG(0x29, 0, 0, 0, 0, 0, 0);

imu 模块响应发送 crc 校验值数据：
send_CMD_ACK(0x753C, 0x10, crc32_data);

其中 crc32_data 值为 imu 模块本身计算的 crc32 数据
```

第七步：发送重启命令

升在上位机判断 crc 校验值正确后，发送重启命令，升级成功

```
判断 crc 校验值正确后，发送重启命令：
Send_CMD_LONG(0x30, 0, 0, 0, 0, 0, 0);
```

固件升级完毕，断电重启后可以通过读取版本号判断是否升级成功。

函数定义：

1. Send_CMD_LONG 函数定义如下：

```
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. Send_CMD_ACK 函数定义如下：

```
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));
}
```

Send_Upload_Data 函数定义如下：

```
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. CRC32 校验函数如下:

```

static const uint32_t crc32_tab[] = {
    0x00000000, 0x77073096, 0xee0e612c, 0x990951ba, 0x076dc419, 0x706af48f,
    0xe963a535, 0x9e6495a3, 0xedb8832, 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, 0xbfdb6116, 0x21b4f4b5, 0x56b3c423,
    0xcfba9599, 0xb8bda50f, 0x2802b89e, 0x5f058808, 0xc60cd9b2, 0xb10be924,
    0x2f6f7c87, 0x58684c11, 0xc1611dab, 0xb6662d3d, 0x76dc4190, 0x01db7106,
    0x98d220bc, 0xefd5102a, 0x71b18589, 0x06b6b51f, 0x9fbfe4a5, 0xe8b8d433,
    0x7807c9a2, 0x0f00f934, 0x9609a88e, 0xe10e9818, 0xf76a0dbb, 0x086d3d2d,
    0x91646c97, 0xe6635c01, 0xb6b51f4, 0xc1c6c162, 0x856530d8, 0xf262004e,
    0x6c0695ed, 0x1b01a57b, 0x8208f4c1, 0xf50fc457, 0x65b0d9c6, 0x12b7e950,
    0x8bbeb8ea, 0xfcb9887c, 0x62dd1ddf, 0x15da2d49, 0x8cd37cf3, 0xfbd44c65,
    0x4db26158, 0x3ab551ce, 0xa3bc0074, 0xd4bb30e2, 0xa4adfa541, 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,
    0xeada54739, 0x9dd277af, 0x04db2615, 0x73dc1683, 0xe3630b12, 0x94643b84,
    0x0d6d6a3e, 0x7a6a5aa8, 0xe40ecf0b, 0x9309ff9d, 0x0a00ae27, 0x7d079eb1,
    0xf00f9344, 0x8708a3d2, 0x1e01f268, 0x6906c2fe, 0xf762575d, 0x806567cb,
    0x196c3671, 0x6e6b06e7, 0xfed41b76, 0x89d32be0, 0x10da7a5a, 0x67dd4acc,
    
```

8. 附件

标配件：



NAV680-SA 主机



集线束



主天线（定位天线）



副天线（定向天线）



主天线连接线



副天线连接线

9. 更新记录

版本	日期	状态/注释
版本 1.0	2024. 01. 09	首次发行