小豚智能
XIAOTUN INTELLIGECE

小豚智控-MC02

产品说明书

东莞小豚智能技术有限公司
DONGGUAN XIAOTUN INTELLIGENCE TECHNOLOGIESCO., LTD.

小豚智控-MC02
产品说明书

编　　制：梁俊杰　　校　核：

标准化审查：　　审　定：

出　版　号：V1.0

文 件 代 号：

出 版 日 期：

电话：0769-89887358
邮箱：marketing@xiaotunai.com

重 要 提 示

感谢您使用东莞小豚智能技术有限公司的产品。为了安全、正确、高效地使用本装置，请您务必注意以下重要提示：

1) 本说明书仅适用于小豚智控-MCO2 。

2) 请仔细阅读本说明书，并按照说明书的规定调整、测试和操作。如有随机资料，请以随机资料为准。

3) 为防止装置损坏，严禁带电插拔装置各插件、触摸印制电路板上的芯片和器件。

4) 请使用合格的测试仪器和设备对装置进行试验和检测。

5) 装置如出现异常或需维修，请及时与本公司服务热线联系。

目　　录

第一篇 技术说明

1 概述

小豚智控-MCO2是一款高性能、易扩展无人船运动控制器套件，具有集成度高、性能可靠、扩展性好等优点，助力于客户快速搭建无人船系统，便于功能集成及二次开发。

小豚智控-MCO2具有无人船的运动控制、多无人船协同/编队控制等功能，可用于各种尺度船舶及无人船自动控制功能。小豚智控-MCO2控制器采用高性能STM32控制器，内置双天线定位模块和433MHz无线通信模块，搭载操作系统和多种先进控制算法，使控制器具有高实时性、扩展性和稳定性，易于系统集成。小豚智控-MCO2控制器结构与电气部分采用防水设计，防水级别可达IP66。同时提供了丰富的IO接口和多种总线接口，对第三方设备具有良好的兼容性。

1.1 适用范围

小豚智控-MCO2是一款无人船运动控制器，主要用于无人艇的运动控制、自动控制，适用于各种船艇。

1.2 主要特点

具有以下特点：

a)集成度高；

b)性能可靠；

c)扩展性好；

d)便于集成与二次开发；

1.3 主要功能

小豚智控-MCO2 具有以下功能：

a)无人船的基本运动控制；

b)无人船多船协同/编队控制；

c)无人船自动控制；

1.4 主要技术参数

1) 输入电压：DC24-54.6V；
2) 定位精度：≤0.3° /1m
3) 水平定位精度：≤2cm+1ppm
4) 速度分辨精度：≤0.2m/s
5) 最大更新频率：≥10Hz
6) 通信距离：≥2km

7） 功耗：≤25W
8） 工作温度：-20° C~+50° C
9） 防护等级：不低于 IP65
10）质量：≤1kg
11）外形尺寸：≤180mm×120mm×60mm
12）通信方式：CAN 总线、RS232、433M 和 5.8G 无线；

2 组成部分

小豚智控-MC02 主要由外壳结构及电路板组成。

2.1 外壳结构

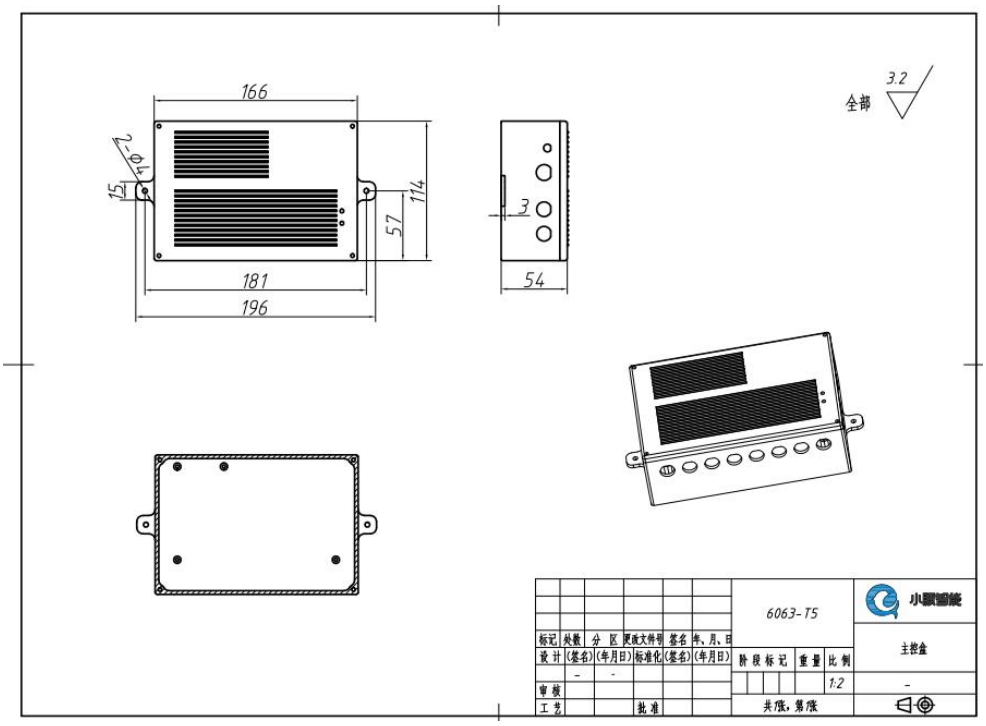
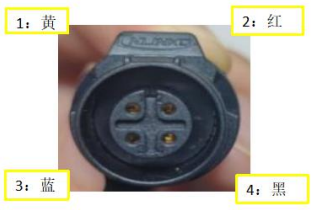


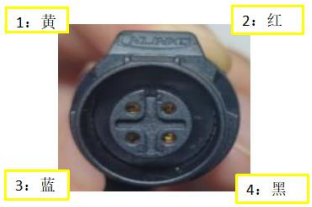
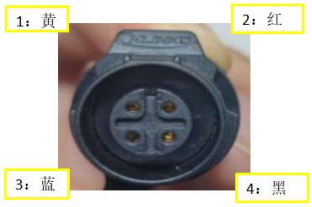
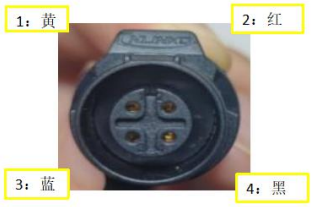
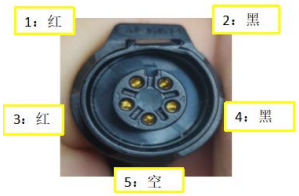
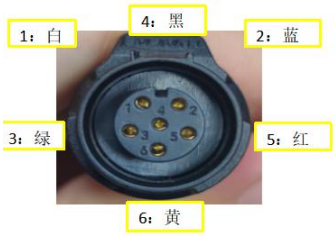
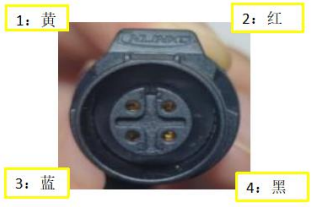
图 1 外壳结构三视图

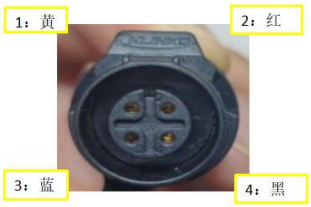
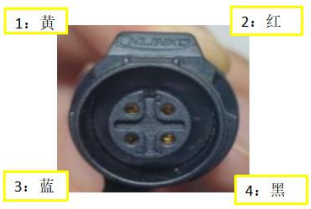
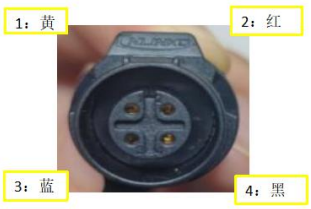
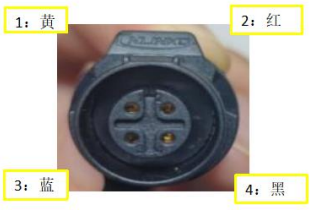
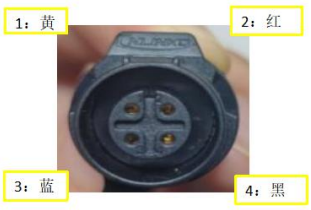
2.2 外部接口名称及功能介绍

具体如下图所示：

表 5 接口功能介绍

序号	名称	功能	座子线序以及对应信号定义	
1	433-天线 1	E32 模块馈线座子	无	
2	433-天线 1	E22 模块馈线座子	无	
3	定位天线 -1	GPS 模块馈线座子	无	
4	定位天线 -2	GPS 模块馈线座子	无	
5	备用-1	MCU1 调试口	1. GND 2. VCC(3.3V) 3. CLK 4. DIO	

6	备用-2	MCU2 调试口	1. GND 2. VCC(3.3V) 3. CLK 4. DIO	
7	CAN-1	CAN 通信	1. CAN L 2. VCC(24V) 3. CAN H 4. GND	
8	CAN-2	CAN 通信	1. CAN L 2. VCC(24V) 3. CAN H 4. GND	
9	电源	直流电源输入 (24-60V)	1. VCC 2. GND 3. VCC 4. GND 5. NULL(无效)	
10	网口	网络模块通信口,提供 POE 电源(24V)	1. TD+ 2. TD- 3. RD+ 4. GND 5. VCC(24V) 6. RD-	
11	232-1	RS232 通信口	1. RX 2. VCC(24V) 3. TX 4. GND	

12	232-2	RS232 通信口 (可外接惯导模块)	1. RX 2. VCC(5V) 3. TX 4. GND	
13	232-3	RS232 通信口 (GPS 数据输出口)	1. RX 2. VCC(5V) 3. TX 4. GND	
14	232-4	RS232 通信口 (可外接 4G 模块)	1. RX 2. VCC(5V) 3. TX 4. GND	
15	232-5	RS232 通信口 (IPC 接口)	1. RX 2. VCC(24V) 3. TX 4. GND	
16	232-6	RS232 通信口 (可外接其他设备)	1. RX 2. VCC(24V) 3. TX 4. GND	

第二篇 使用方法

3 安装以及使用

3.1 开箱检查

3.1.1 打开包装后，检查设备外观是否完好无损；

3.1.2 检查设备的合格证明书、配套文件、附件、备品备件等是否与订货要求一致，是否与装箱单规定的型号、名称、数量等一致和齐备；

3.1.3 如有问题，请与制造厂及时联系。

3.2 安装前注意事项

3.2.1 将设备安装使用前必须确保，设备不上电。设备不支持热插拔。

3.2.2 433 天线接入时，注意馈线与座子是否对应上，馈线是否是内孔的。

3.3 安装与使用步骤

第 1 步：将 433MHz 馈线接入 433-天线 2 中。

第 2 步：将 GPS 两个天线分别接入定位天线-1 和定位天线-2 中。

第 3 步：将 CAN 通信线接到扩展板 DC-DC 口。

第 4 步：将电源线接入电源接口。

第 5 步：检查电源线另一端使用的电源是否接入正确，并上电。若设备正常，则可以看到指示灯闪烁。

第 6 步：使用遥控器选择相应的船号，若遥控器能够正常控制船只，则设备通信正常。

第 7 步：使用 4 芯航插转 RS232 线，接入 RS232-3 接口，可以通过串口调试助手看到 GPS 数据输出。

第 8 步：使用 6 芯航插转网线，接入网口接口，并将网线接入电脑，同时配置电脑为固定 IP,IP 网段为 192.168.1.xx，可以使用网络调试助手与设备通信。

第三篇 通讯介绍

4 通讯

4.1 通讯概要

设备可以使用 CAN、RS232 以及网络控制。

4.2 CAN 通讯接口

CAN-1 接口和 CAN-2 接口都可以接收和发送指令。

4.2.1 通信参数

方式：CAN 总线；
波特率：1Mbps。

4.2.2 帧格式

协议数据单元由 CAN 扩展数据帧 29 位 ID 和 0～8 个字节数据组成。基于 NMEA2000 协议提供的数据通信方法，扩展标识符由优先权、PGN 消息参数群、源地址组成，如图 8 所示。

图 8 扩展标识符由优先权、PGN 消息参数群、源地址组成

4.2.3 设备地址

表 9 设备定义地址

序号	节点名称	地址（Source Address）
1	主控板	0X10
2	主推电机左	0X20
3	主推电机右	0X21

4	转向舵机左	0X30
5	转向舵机右	0X31
6	倒车斗舵机左	0x40
7	倒车斗舵机右	0x41

4.2.4 消息标识符

注：此消息标识符对应 CAN 通信中 29 位扩展标识符；

表 10 消息标识符地址

序号	标识符地址（Hex）	优先级	名称
1	0x05ff2010	1	左主推电机转速指令
2	0x05ff2110	1	右主推电机转速指令
3	0x05ff3010	1	左转向舵机角度指令
4	0x05ff3110	1	右转向舵机角度指令
5	0x05ff4010	1	左倒车斗舵机角度指令
6	0x05ff4110	1	右倒车斗舵机角度指令
7	0x09ff1020	2	左主推电机实时状态
8	0x09ff1021	2	右主推电机实时状态
9	0x09ff1030	2	左转向舵机实时状态
10	0x09ff1031	2	右转向舵机实时状态
11	0x09ff1040	2	左倒车斗舵机实时状态
12	0x09ff1041	2	右倒车斗舵机实时状态

4.2.5 消息数据

备注：在消息数据定义中，左右侧数据定义一致；

4.2.5.1 主推电机转速

表 11 主推电机对应地址

OUT	IN	ID		周期（ms）
主控板	主推电机	P	PGN	20*10ms
		1	0x11020/0x11021	
数据				
位置	数据名		备注	
BYTE1	模式		0x00 为遥控模式（该模式下电机转速指令经过柔和处理后再给电机） 0x01 为自动控制模式（该模式下电机转速由控制器直接给定，不经过柔和处理）	
BYTE2	电机转速（r/min）		UINT8：0~7 位	
BYTE3			UINT8：8~15 位	
BYTE4	保留			
BYTE5	保留			
BYTE6	保留			
BYTE7	保留			
BYTE8	保留			

4.2.5.2 转向舵机角度

表 12 转向舵机对应地址

OUT	IN	ID		周期（ms）
主控板	转向舵机	P	PGN	200
		1	0x11030/0x11031	
数据				
位置	数据名		备注	
BYTE1	角度（700~1100，中位值 900，720 代表左舵极限值，1080 代表右舵极限值）		UINT8：0~7 位	
BYTE2			UINT8：8~15 位	
BYTE3	保留			
BYTE4	保留			
BYTE5	保留			
BYTE6	保留			
BYTE7	保留			
BYTE8	保留			

4.2.5.3 倒车斗舵机角度

表 13 倒车斗舵机对应地址

OUT	IN	ID	周期（ms）

主控板	倒车斗舵机	P	PGN	200
		1	0x11040/0x11041	
数据				
位置	数据名		备注	
BYTE1	角度（0~100，0 代表倒车斗升起最高点，100 代表倒车斗放下最低点）		UINT8：0~7 位	
BYTE2			UINT8：8~15 位	
BYTE3	保留			
BYTE4	保留			
BYTE5	保留			
BYTE6	保留			
BYTE7	保留			
BYTE8	保留			

4.2.5.4 主推电机实时状态

表 14 主推电机实时状态监控地址

OUT	IN	ID		周期（ms）
主推电机	主控板	P	PGN	200
		2	0x1ff10	
数据				
位置	数据名		备注	
BYTE1	实际转速（r/min）		UINT8：0~7 位	
BYTE2			UINT8：8~15 位	
BYTE3	保留			
BYTE4	保留			
BYTE5	保留			
BYTE6	保留			
BYTE7	保留			
BYTE8	保留			

4.2.5.5 转向舵机实时状态

表 15 转向舵机实时监控地址

OUT	IN	ID		周期（ms）
转向舵机	主控板	P	PGN	200
		2	0x1ff10	
数据				
位置	数据名		备注	
BYTE1	保留			
BYTE2	实际角度		UINT8：0~7 位	
BYTE3			UINT8：8~15 位	
BYTE4	保留			
BYTE5	保留			
BYTE6	保留			
BYTE7	保留			
BYTE8	保留			

4.2.5.6 倒车斗舵机实时状态

表 16 倒车斗舵机实时监控地址

OUT	IN	ID		周期（ms）
倒车斗舵机	主控板	P	PGN	倒车斗舵机
		2	0x1ff10	
数据				
位置	数据名		备注	
BYTE1	保留			
BYTE2	实际角度（单位）		UINT8：0~7 位	
BYTE3	保留		UINT8：8~15 位	
BYTE4	保留			
BYTE5	保留			
BYTE6	保留			
BYTE7	保留			
BYTE8	保留			

4.2.5.7 测试用例

测试环境

集成开发环境：Keil µVision V5.23.0.0

IC 型号：STM32F407VGTx

Software Pack 版本：Keil.STM32F4xx_DFP.2.15.0

编译器：J-LINK/J-TRACE Cortex

STM32F10x_FWLib 版本:V1.4.0

测试流程

左舵机转动→右舵机转动→左电机加减速→右电机加减速→左倒车斗转动→右倒车斗转动

样例程序

声明：本样例程序仅为在此特定环境下程序，仅供参考，具体请以实际环境为准，如有问题请咨询本公司！

```
#include"stm32f4xx_conf.h"
#include"stdio.h"
//延时函数 1ms 左右
void delayms(intt) {
    int i,j;
    for (i=0;i<t;i++)
    for (j=0;j<35000;j++);
}
//CAN 外设初始化，波特率：1M
void CAN1ModeInit(unsignedchartsjw, unsignedchartbs2, unsignedchartbs1, unsignedintbrp, unsignedcharmode) {
    GPIO_InitTypeDef GPIO_InitStructure;
    CAN_InitTypeDef CAN_InitStructure;
    CAN_FilterInitTypeDef CAN_FilterInitStructure;
    NVIC_InitTypeDef NVIC_InitStructure;
    //使能相关时钟
    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE);
    //使能 PORTA 时钟
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_CAN1, ENABLE);
    //使能 CAN1 时钟
    //初始化 GPIO
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_12;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
    GPIO_Init(GPIOA, &GPIO_InitStructure);
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_11;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;
    GPIO_Init(GPIOA, &GPIO_InitStructure);
    //CAN 单元设置
    CAN_InitStructure.CAN_TTCM = DISABLE;
    //非时间触发通信模式
    CAN_InitStructure.CAN_ABOM = DISABLE;
```

```
//软件自动离线管理
CAN_InitStructure.CAN_AWUM = DISABLE;
//睡眠模式通过软件唤醒(清除 CAN->MCR 的 SLEEP 位)
CAN_InitStructure.CAN_NART = ENABLE;
//禁止报文自动传送
CAN_InitStructure.CAN_RFLM = DISABLE;
//报文不锁定,新的覆盖旧的
CAN_InitStructure.CAN_TXFP = DISABLE;
//优先级由报文标识符决定
CAN_InitStructure.CAN_Mode = mode;
//模式设置
CAN_InitStructure.CAN_SJW = tsjw;
//重新同步跳跃宽度(Tsjw)为 tsjw+1 个时间单位  CAN_SJW_1tq~CAN_SJW_4tq
CAN_InitStructure.CAN_BS1 = tbs1;
//Tbs1 范围 CAN_BS1_1tq ~CAN_BS1_16tq
CAN_InitStructure.CAN_BS2 = tbs2;
//Tbs2 范围 CAN_BS2_1tq ~CAN_BS2_8tq
CAN_InitStructure.CAN_Prescaler = brp;
//分频系数(Fdiv)为 brp+1
CAN_Init(CAN1, &CAN_InitStructure);
//初始化 CAN1
//配置过滤器
CAN_FilterInitStructure.CAN_FilterNumber = 0;
//过滤器 0
CAN_FilterInitStructure.CAN_FilterMode = CAN_FilterMode_IdMask;
CAN_FilterInitStructure.CAN_FilterScale = CAN_FilterScale_32bit;
//32 位
CAN_FilterInitStructure.CAN_FilterIdHigh = 0x0000;
//32 位 ID
CAN_FilterInitStructure.CAN_FilterIdLow = 0x0000;
CAN_FilterInitStructure.CAN_FilterMaskIdHigh = 0x0000;
//32 位 MASK
CAN_FilterInitStructure.CAN_FilterMaskIdLow = 0x0000;
CAN_FilterInitStructure.CAN_FilterFIFOAssignment = CAN_Filter_FIFO0;
//过滤器 0 关联到 FIFO0
CAN_FilterInitStructure.CAN_FilterActivation = ENABLE;
//激活过滤器 0
CAN_FilterInit(&CAN_FilterInitStructure);
//滤波器初始化
CAN_ITConfig(CAN1,CAN_IT_FMP0,ENABLE);
//FIFO0 消息挂号中断允许.
NVIC_InitStructure.NVIC_IRQChannel = CAN1_RX0_IRQn;
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 1;
//  主优先级为 1
```

```
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
    // 次优先级为 0
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
}
//CAN 接收中断服务函数
void USB_LP_CAN1_RX0_IRQHandler (void) {
    //09ff1020      左电机状态 ID
    //09ff1021      右电机状态 ID
    //09ff1030      左舵机状态 ID
    //09ff1031      右舵机状态 ID
    CanRxMsg RxMessage;
    //09ff1030
    CAN_Receive(CAN1, 0, &RxMessage);
}
//主函数
int main(void) {
    unsignedint speed, steerangle, bucketangle, i;
    CanTxMsg TxMessage;
    //CAN 消息定义
    //CAN 外设初始化
    CAN1ModeInit(CAN_SJW_1tq,CAN_BS2_3tq,CAN_BS1_3tq,6,CAN_Mode_Normal);
    //motor_l   0x5ff2010      左电机 ID
    //motor_l   0x5ff2110      右电机 ID
    //steer_l   0x5ff3010      左舵机 ID
    //steer_l   0x5ff3110      右舵机 ID
    //bucket_l 0x5ff4010       左倒车斗 ID
    //bucket_l 0x5ff4110       右倒车斗 ID
    //motor speed   data[2]=speed>>8; data[1]=speed&0xff    电机转速范围 700~8000（rpm）
    //steer angle   data[1]=steerangle>>8; data[0]=steerangle&0xff   舵机舵角范围 700~1100（0.1°），
中间值为 900
    //bucket angle   data[0]=bucketangle    倒车斗范围 0~100（%）
    delayms(3000);
    TxMessage.StdId = 0x12;
    // 标准标识符为 0
    TxMessage.IDE = CAN_Id_Extended;
    // 使用扩展标识符
    TxMessage.RTR = 0;
    // 消息类型为数据帧，一帧 8 位
    TxMessage.DLC = 8;
    // 发送两帧信息
    //左舵机往左 1s
    steerangle = 700;
    TxMessage.ExtId = 0x5ff3010 & 0x01FFFFFF;
```

```
TxMessage.Data[1]=steerangle>>8;
TxMessage.Data[0]=steerangle&0xff;
CAN_Transmit(CAN1, &TxMessage);
delayms(1000);
//左舵机往右 1s
steerangle = 1100;
TxMessage.ExtId = 0x5ff3010 & 0x01FFFFFF;
TxMessage.Data[1]=steerangle>>8;
TxMessage.Data[0]=steerangle&0xff;
CAN_Transmit(CAN1, &TxMessage);
delayms(1000);
//左舵机往左 1s
steerangle = 700;
TxMessage.ExtId = 0x5ff3010 & 0x01FFFFFF;
TxMessage.Data[1]=steerangle>>8;
TxMessage.Data[0]=steerangle&0xff;
CAN_Transmit(CAN1, &TxMessage);
delayms(1000);
//左舵机往右 1s
steerangle = 1100;
TxMessage.ExtId = 0x5ff3010 & 0x01FFFFFF;
TxMessage.Data[1]=steerangle>>8;
TxMessage.Data[0]=steerangle&0xff;
CAN_Transmit(CAN1, &TxMessage);
delayms(1000);
//右舵机往左 1s
steerangle = 700;
TxMessage.ExtId = 0x5ff3110 & 0x01FFFFFF;
TxMessage.Data[1]=steerangle>>8;
TxMessage.Data[0]=steerangle&0xff;
CAN_Transmit(CAN1, &TxMessage);
delayms(1000);
//右舵机往右 1s
steerangle = 1100;
TxMessage.ExtId = 0x5ff3110 & 0x01FFFFFF;
TxMessage.Data[1]=steerangle>>8;
TxMessage.Data[0]=steerangle&0xff;
CAN_Transmit(CAN1, &TxMessage);
delayms(1000);
//左电机加速 5s
speed = 1000;
for (i = 0; i<5; i++) {
    speed += 300*i;
    TxMessage.ExtId = 0x5ff2010 & 0x01FFFFFF;
```

```
        TxMessage.Data[2]=speed>>8;
        TxMessage.Data[1]=speed&0xff;
        TxMessage.Data[0]=0;
        CAN_Transmit(CAN1, &TxMessage);
        delayms(1000);
    }
    //左电机减速 5s
    for (i = 0; i<5; i++) {
        speed -= 300*i;
        TxMessage.ExtId = 0x5ff2010 & 0x01FFFFFF;
        TxMessage.Data[2]=speed>>8;
        TxMessage.Data[1]=speed&0xff;
        TxMessage.Data[0]=0;
        CAN_Transmit(CAN1, &TxMessage);
        delayms(1000);
    }
    //左电机停止
    speed = 0;
    TxMessage.ExtId = 0x5ff2010 & 0x01FFFFFF;
    TxMessage.Data[2]=speed>>8;
    TxMessage.Data[1]=speed&0xff;
    TxMessage.Data[0]=0;
    CAN_Transmit(CAN1, &TxMessage);
    delayms(1000);
    //右电机加速 5s
    speed = 1000;
    for (i = 0; i<5; i++) {
        speed += 300*i;
        TxMessage.ExtId = 0x5ff2110 & 0x01FFFFFF;
        TxMessage.Data[2]=speed>>8;
        TxMessage.Data[1]=speed&0xff;
        TxMessage.Data[0]=0;
        CAN_Transmit(CAN1, &TxMessage);
        delayms(1000);
    }
    //右电机减速 5s
    for (i = 0; i<5; i++) {
        speed -= 300*i;
        TxMessage.ExtId = 0x5ff2110 & 0x01FFFFFF;
        TxMessage.Data[2]=speed>>8;
        TxMessage.Data[1]=speed&0xff;
        TxMessage.Data[0]=0;
        CAN_Transmit(CAN1, &TxMessage);
        delayms(1000);
```

```
}
//右电机停止
speed = 0;
TxMessage.ExtId = 0x5ff2110 & 0x01FFFFFF;
TxMessage.Data[2]=speed>>8;
TxMessage.Data[1]=speed&0xff;
TxMessage.Data[0]=0;
CAN_Transmit(CAN1, &TxMessage);
delayms(1000);
//左倒车斗拉升 20%
bucketangle = 20;
TxMessage.ExtId = 0x5ff4010 & 0x01FFFFFF;
TxMessage.Data[2]=0;
TxMessage.Data[1]=0;
TxMessage.Data[0]=bucketangle;
CAN_Transmit(CAN1, &TxMessage);
delayms(1000);
//左倒车斗拉升 80%
bucketangle = 80;
TxMessage.ExtId = 0x5ff4010 & 0x01FFFFFF;
TxMessage.Data[2]=0;
TxMessage.Data[1]=0;
TxMessage.Data[0]=bucketangle;
CAN_Transmit(CAN1, &TxMessage);
delayms(1000);
//右倒车斗拉升 20%
bucketangle = 20;
TxMessage.ExtId = 0x5ff4110 & 0x01FFFFFF;
TxMessage.Data[2]=0;
TxMessage.Data[1]=0;
TxMessage.Data[0]=bucketangle;
CAN_Transmit(CAN1, &TxMessage);
delayms(1000);
//右倒车斗拉升 80%
bucketangle = 80;
TxMessage.ExtId = 0x5ff4110 & 0x01FFFFFF;
TxMessage.Data[2]=0;
TxMessage.Data[1]=0;
TxMessage.Data[0]=bucketangle;
CAN_Transmit(CAN1, &TxMessage);
delayms(1000);
//测试结束
while(1) {
    ;
```

```
    }
}
```

4.3 RS232 通讯接口

4.3.1 数据定义

序号	数据结构名称	定义

1	动作指令	typedef struct (21 个数据)
		{
		(1)UINT8 Model; //舵角转速(0)、航向航速(1)、航向转速(2)
		(2,3)UINT16 Angle_l; //舵角(0.1°)，航向（0.1°）左
		(4,5)UINT16 Angle_r; //舵角(0.1°)，航向（0.1°）右
		(6,7)UINT16 Speed_l; //转速(rpm)，航速（cm/s）左
		(8,9)UINT16 Speed_r; //转速(rpm)，航速（cm/s）右
		(10)UINT8 Direction_l; //转速方向左(前推(0)，后退(1))
		(11)UINT8 Direction_r; //转速方向右(前推(0)，后退(1))
		(12)UINT8 Bucket_l; //倒车斗左(拉起~下放(0~100))
		(13)UINT8 Bucket_r; //倒车斗右(拉起~下放(0~100))
		(14,15,16,17)UINT8 reserve[4];
		}ActionData;

2	执行层状态	typedef struct (31 个数据)
		{
		(1,2,3,4)UINT32 Lat; //纬度(e-7)
		(5,6,7,8)UINT32 Lon; //经度(e-7)
		(9,10)UINT16 Heading; //航向(0.01°)
		(11,12)UINT16 Velocity; //水平速度(cm/s)
		(13,14)UINT16 VelAngle; //速度角(0.01°)
		(15,16)UINT16 Angle_l; //舵角(0.1°)
		(17,18)UINT16 Angle_r; //舵角(0.1°)
		(19,20)UINT16 Speed_l; //电机转速(rpm)
		(21,22)UINT16 Speed_r; //电机转速(rpm)
		(23)UINT8 Direction_l; //转速方向左(前推(0)，后退(1))
		(24)UINT8 Direction_r; //转速方向右(前推(0)，后退(1))
		(25)UINT8 Bucket_l; //倒车斗状态(拉起~下放(0~100))
		(26)UINT8 Bucket_r; //倒车斗状态(拉起~下放(0~100))
		(27)UINT8 Mileage; //续航里程(km)
		(28,29)UINT16 DriverError; //故障信息
		(30,31)UINT8 reserve[2]; //预留信息
		}ActuatorState;

3	位置点	typedef struct (4 个数据)
		{
		(1,2)UINT16 PosX; //X 轴位置(-32767m~32768m)
		(3,4)UINT16 PosY; //Y 轴位置(-32767m~32768m)
		}DotPositon;

4	船只序列号	typedef struct (4 个数据)
		{
		(1)UINT8 Type; //产品类型
		(2)UINT8 Year; //出产年
		(3)UINT8 Month; //出产月
		(4)UINT8 Day; //出产日
		(5)UINT8 seq; //当日序号
		}ControllerSeq;

5	船只状态	typedef struct (14 个数据)
		{
		(1)UINT8 Class; //无人艇种类
		(2)UINT8 BoatNum; //无人艇编号
		(3)UINT8 Vol; //主控板电压(V)
		(4)UINT8 Temp; //主控板温度(℃)
		(5)UINT8 ComError; //故障信息
		(6,7,8,9,10)ControllerSeq Seq; //产品序列号：型号，年，月，日，序号
		(11)UINT8 reserve[0]; //工作状态(TaskFlag|IPCModel|Model)
		(12,13,14)UINT8 reserve[3]; //预留信息
		}BoardState

6	遥控指令	typedef struct (13 个数据)
		{
		//舵角转速(0)、航向航速(1)、航向转速(2)、待命模式（0x0f）
		(1)UINT8 Model;
		(2)UINT8 BoatNum; //船号
		(3,4)UINT16 Angle; //舵角(0.1°)，航向（0.1°）
		(5,6)UINT16 Speed; //转速(rpm)，航速（cm/s）
		(7)UINT8 Bucket ; //倒车斗指令(拉起~下放(0~100))
		(8)UINT8 IPCModel; //是否工控机控制(否(0)，是(1))
		(9)UINT8 StartFlag; //是否启动无人船(否(0)，是(1))
		(10,11,12,13)UINT8 reserve[4]; //预留信息
		}CentralAction;

7	轨迹指令	typedef struct (13+4*DotNum 个数据)
		{
		(1)UINT8 Model; //任务类型(锚点跟踪（0）/轨迹跟踪（1）/

		区域扫描(2))
		(2)UINT8 IPCModel; //是否工控机控制(否(0)，是(1))
		(3)UINT8 Velocity; //速度（0.1m/s）
		(4)UINT8 DotNum； //轨迹点个数
		(5,6,7,8)UINT32 BaseLat; //基点维度(e^{-7}度)
		(9,10,11,12)UINT32 BaseLon; //基点经度(e^{-7}度)
		(13,14,15,16)UINT8 reserve[4]; //预留信息
		(17~17+4*DotNum)DotPositon Positions[DotNum]; //点位置
		}TrajectoryAction;

8	控制指令	typedef struct (6 个数据)
		{
		(1)UINT8 TaskFlag; //是否执行任务(否(0)，是(1))
		(2)UINT8 Model; //运行模式(舵角转速(0)、航向航速(1)、航向转速(2)、待命模式(0x0f))
		(3)UINT8 IPCModel; //是否工控机控制(否(0)，是(1))
		(4,5)UINT16 SetHeading; //航向（0.1°）
		(6,7)UINT16 SetVelocity; //航速（cm/s）
		}BoatCmd；

9	状态数据	typedef struct （18 个数据）
		{
		(1,2,3,4)UINT32 Lat; //纬度(e^{-7}度)
		(5,6,7,8)UINT32 Lon; //经度(e^{-7}度)
		(9,10)UINT16 Heading; //航向(0.01°)
		(11,12)UINT16 Velocity; //水平速度(cm/s)
		(13,14)UINT16 VelAngle; //速度角(0.01°)
		(15,16,17,18)UINT8 reserve[4];
		} BoatStateToDevice;

4.3.2 通信协议

4.3.2.1 帧格式

Header	CMD		Len		Checksum	Data		
	cmdL	cmdH	LenL	LenH				
0xFA	1BYTE	1BYTE	1BYTE	1BYTE	1BYTE	1BYTE	…	NBYTE

Header：协议头 0Xfa；
CMD：协议命令码；
Len：数据长度 N，data1，…，dataN；
Checksum：校验和，Checksum = Header + cmdL + cmdH + LenL + LenH + Data1+ … + DataN

4.3.2.2 通信协议(设备与数传电台\4G 模块\网口模块通信)

序号	命令码	说明
1	0201	通讯层接收舵角转速/航向航速和动作数据： CentralAction centralAction;
2	0210	通讯层接收运行模式和轨迹数据： TrajectoryAction trajectoryAction;

3	0301	通讯层发送状态数据：
		typedef struct
		{
		(7~37)ActuatorState actuatorState;
		(38~51)BoardState boardState;
		(52~71)DeviceState waterState; //应用设备数据
		}BoatStateToStation;

4	8202	通讯层发送响应信息： BoardState boardState;

4.3.2.3 通信协议(设备与应用设备通信)

序号	命令码	说明
1	0201	通讯层接收水采样数据： DeviceState deviceState;
1	0301	通讯层发送状态数据： BoatStateToDevice boatStateToDevice;

4.3.2.3 通信协议(设备与上位机通信)

序号	命令码	说明
1	0201	通讯层接规划层动作数据： ActionData actionData;
7	8302	通讯层接规划层响应数据： （0 成功，1 失败）；

4	0301	通讯层发送运行模式和状态信息：
		typedef struct
		{
		(7~37)ActuatorState actuatorState;
		(38~51)BoardState boardState;
		(52~71)DeviceState waterState; //应用设备数据
		(72~78)BoatCmd boatCmd;
		}BoatStateToIPC;

2	0310	通讯层发送运行模式和轨迹数据： TrajectoryAction trajectoryAction;

第四篇 常见故障现象及处理方法

5 常见故障及对策

常见故障、原因分析和处理方法见下表。

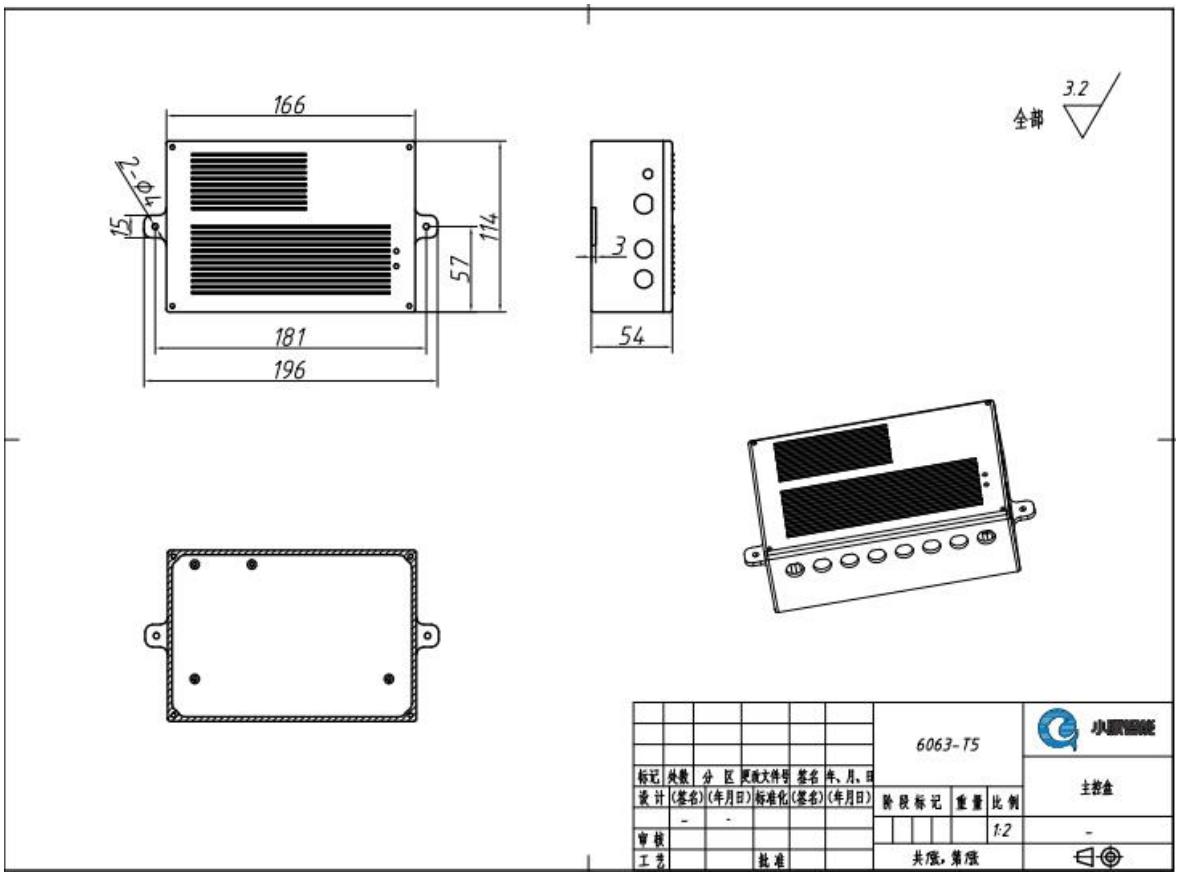
序号	常见故障现象	原 因 分 析	处 理 方 法
1	产品上电后无反应	1、可能设备供电异常或未供电 2、电缆线可能损坏 3、连接器插头松动，未连接好	1、检查供电电压是否正常，扩展板正常供电范围(24-60VDC)。如果供电电压不正常,请调整输入电压; 2、用电压表测量线缆插头上的电压是否正常,如果没有电压则线缆有问题，请更换电缆; 3、检查插头是否插好，请确保线缆连接正确，插头连接良好。
2	产品无法通信	1、产品供电系统异常，扩展板未上电 2、通信线插头未插好 3、遥控器没有选择正确船号 4、433M 天线接错位置 5、433M 天线馈线使用错误	1、检查产品供电是否正常，如产品不能正常供电请确保产品能够正常供电,在断电情况下使用万用表测量 CAN 总线 CAN_H 和 CAN_L 之前的电阻值,确保电阻值在 60 欧姆左右; 2、检查插头是否插好，请确保线缆连接正确，插头连接良好。 3、重新在遥控器上选择正确船号。 4、重新检查天线位置是否接错 5、检查馈线是否与座子对应

6 随装置供货的附件清单

货号	产品名称	品牌、型号规格	单位	数量
1	小豚智控-MCO2	小豚智控-MCO2	个	1
2	附件	电源线	条	1
3		CAN 通信线	条	2
4		4 芯航插转 RS232 线	条	2
5		6 芯航插转网线	条	1

7 附图和附表

附图 1 设备尺寸图



地址：广东省东莞市松山湖国际创业创新社区 G4 栋 1201 室

邮编：523808

电话：0769-89887358

网　址：http://www.xiaotunai.com