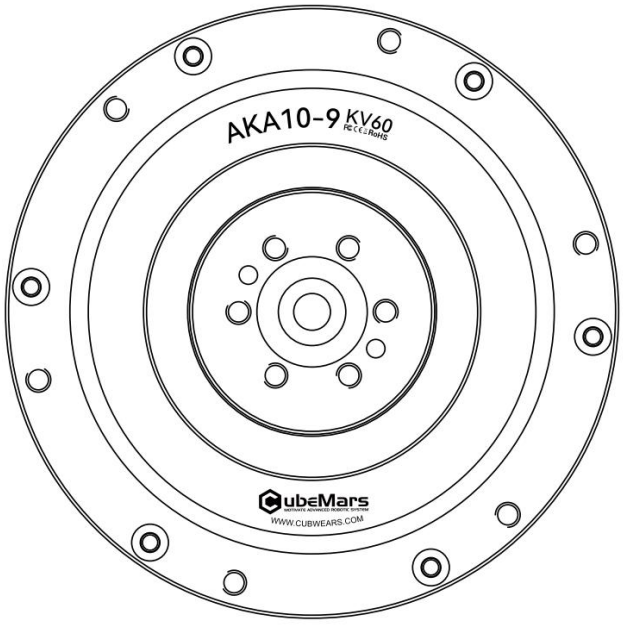
AKA 系列电机模组产品使用说明

V3.0.0



E-mail:info@cubemars.com
https://www.cubemars.com/

目录

E-mail:info@cubemars.com
https://www.cubemars.com/

E-mail:info@cubemars.com
https://www.cubemars.com/

注意事项

1.确保电路无短路，接口按照要求正确连接。

2. 驱动板输出时，会出现发热情况，请小心使用，以免烫伤。

3. 使用前请检查各零部件是否完好。如有部件缺失、老化，请停止使用并及时联系技术支持。

4. 请严格按照本文规定的工作电压、电流、温度等参数使用，否则会对产品造成永久性的损坏！

产品特色

AKA 系列电机驱动板采用同级别中高性能的驱动芯片，使用 Field Oriented Control（简称 FOC）算法，搭配先进自抗扰控制技术对速度和角度进行控制。可配合 CubeMarsTool 调参软件进行参数设置并升级固件。在硬件方面，内环采用了 16 位高精度编码器，最高支持 21 位（需定制固件），CAN 通讯采用了更安全的隔离 CAN 接口，并更换高可靠性插头，大大提升了产品的使用及通讯可靠性；在软件方面，对上位机 CubeMarsTool 进行了全新升级，伺服模式与力控模式使用无需进行切换，控制界面更为简洁，使用操作方面也做了大量精简，全面提升客户的使用体验。

免责声明

感谢您购买 AKA 系列模组电机。在使用之前，请仔细阅读本声明，一旦使用，即被视为对本声明全部内容的认可和接受。请严格遵守产品说明书和相关的法律法规、政策、准则安装和使用该产品。在使用产品过程中，用户承诺对自己的行为及因此而产生的所有后果负责。因用户不当使用、安装、改装造成的任何损失，CubeMars 将不承担法律责任。

版本变更记录

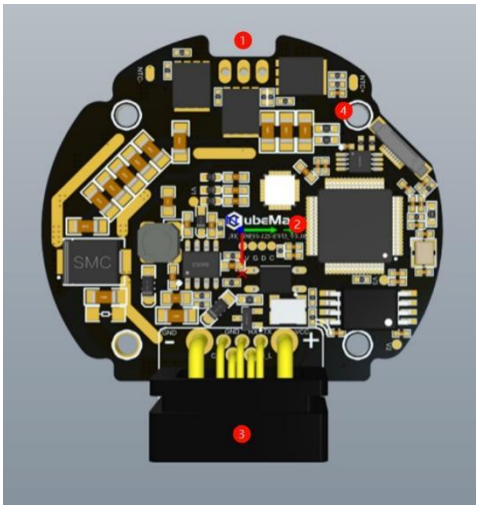
日期	版本	变更内容
2023.10.18	Ver.2.0.0	1.首次编写 2.驱动器版本为 V3.0 3.CubeMarsTool 版本为 V2.0
2024.07.29	Ver.2.0.1	1.1 将最大工作电压改为允许工作电压范围 4.1.6 设置原点示例代码修改 4.2 运控协议示例代码修改 4.3.2.1 修正外环位置公式

2024.08.06	Ver.2.0.2	修改封面 logo 1 驱动器产品信息修改 2 添加章节内容 3.4 修正固件更新示例图片 4.2 参数范围修改
2024.12.04	Ver.2.0.3	1.2.2 修正驱动器接口定义 2 修正 RLINK 定义 2.1 修改 RLINK 产品规格，增加 VCC 电压选择开关 3 修改 CubeMars 归属权从江西新拓实业有限公司到南昌酷德科技有限公司
2024.12.11	Ver3.0.0	正式变更版本为 3.0.0

E-mail:info@cubemars.com
https://www.cubemars.com/

1. 驱动器产品信息

1.1 驱动器外观简介&产品规格（小款）



①三相动力线连接端

②硬件版本号

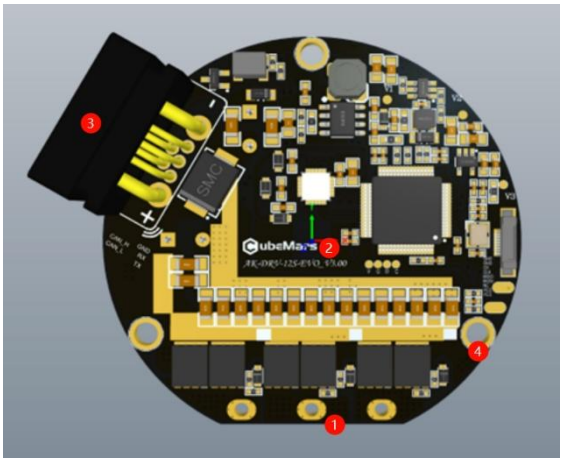
③连接端口

④安装孔

产品规格（小款）	
额定工作电压	48V
允许工作电压	18-52V
额定工作电流	10A
最大允许电流	30A
待机功耗	≤50mA
CAN 总线比特率	1Mbps
尺寸	54mm×49mm
工作环境温度	-20℃至 65℃
控制板最大允许温度	100℃
内环编码器位数	21bit（单圈绝对值）
外环编码器位数（限双编码器型号）	15bit（单圈绝对值）

E-mail:info@cubemars.com
https://www.cubemars.com/

1.2 驱动器外观简介&产品规格（大款）



①三相动力线连接端

②硬件版本号

③连接端口

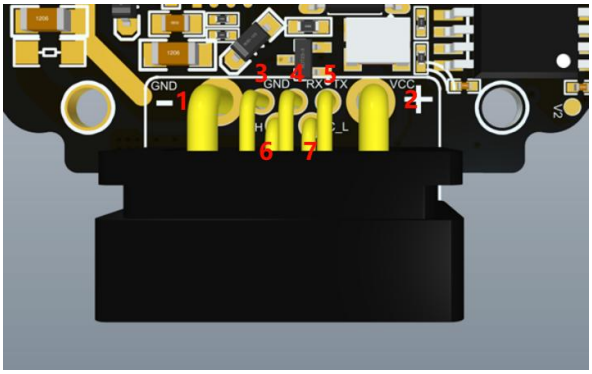
④安装孔

产品规格（大款）	
额定工作电压	48V
允许工作电压	18-52V
额定工作电流	20A
最大允许电流	60A
待机功耗	≤50mA
CAN 总线比特率	1Mbps
尺寸	57mm×60mm
工作环境温度	-20℃至 65℃
控制板最大允许温度	100℃
内环编码器位数	21bit（单圈绝对值）
外环编码器位数（限双编码器型号）	15bit（单圈绝对值）

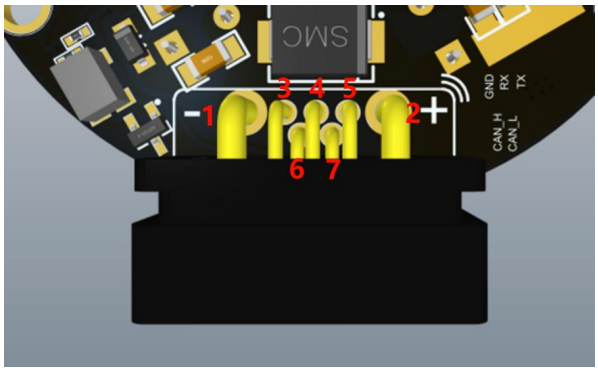
E-mail:info@cubemars.com
https://www.cubemars.com/

1.2 驱动器接口及定义

1.2.1 驱动器接口图



小款



大款

1.2.2 驱动器接口引脚定义

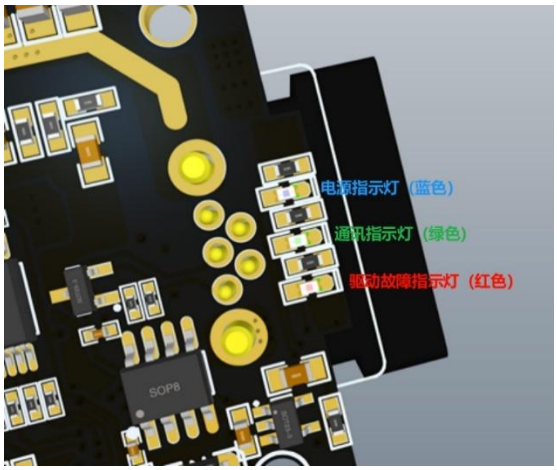
序号	接口功能	引脚	说明	颜色
1	**电源输入及通讯接口**	1	电源负极（-）	黑色
		2	电源正极（+）	红色
		3	串口信号地（GND）	黑色
		4	串口信号输入（RX）	黄色
		5	串口信号输出（TX）	绿色
		6	CAN 通讯高侧（CAN_H）	白色
		7	CAN 通讯低侧（CAN_L）	蓝色

E-mail:info@cubemars.com
https://www.cubemars.com/

1.2.3 驱动器接口推荐品牌及型号

序号	板载接口型号	品牌	线端接口型号	品牌
1	A1257WR-S-3P	CJT（长江连接器）	A1257H-3P	CJT（长江连接器）

1.3 驱动器指示灯定义



指示灯定义		
1. 电源指示灯（亮时蓝灯）	灯亮	驱动板通电
	灯灭	驱动板未上电
2. 运行指示灯（亮时绿色）	灯亮	电机在运行状态
	灯灭	电机未在运行状态
3. 驱动故障指示灯（亮时红色）	灯亮	驱动板故障
	灯灭	驱动板正常

⚠：驱动板上电后，正常状态下蓝灯常亮，绿灯和红灯点亮 2 秒后熄灭。

1.4 主要配件及规格

序号	项目	规格	数量	备注
1	电源及通信插头	16AWG 硅胶线 颜色红黑 和 30AWG 铁氟龙镀银线 颜色 黑白黄蓝绿 不含插头长 200±5mm，一端插头 CTZ-B，一端剥皮浸锡 5mm	各 1PCS	±2MM
2	热敏电阻	MF51B103F3950-10K-3950	2PCS	
3	电解电容	120Uf-63V-10x12MM	2PCS	AKA10-9 V3.0 标配
4	功率 MOS	HYG035N08NS2C2-80V-2.8mΩ	12PCS	

E-mail:info@cubemars.com
https://www.cubemars.com/

2.连接

2.1 R-link 外观简介&产品规格及接口定义

产品规格	
额定工作电压	5V
VCC 电压选择开关	3.3V（默认）
待机功耗	≤30mA
外形尺寸	73.8x23.6x14.5MM
工作环境温度	-20℃至 65℃
控制板最大允许温度	85℃

序号	接口功能	引脚	说明
1	串行接口	RXD	串口信号输入
		TXD	串口信号输出
		GND	串口信号地
		VCC	VCC 电源输出
		RTS	发送数据请求
		CTS	清除发送
		GND	电源地

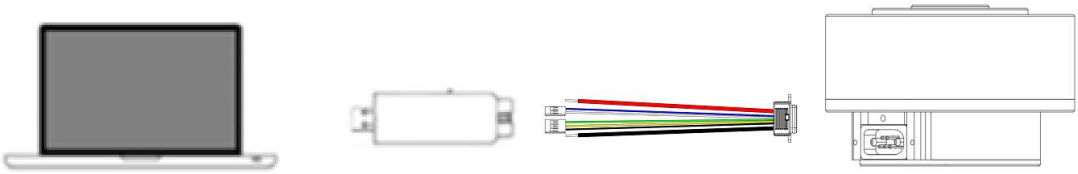
E-mail:info@cubemars.com
https://www.cubemars.com/

序号	接口功能	引脚	说明
		5V	5V 电源输出

2.2 R-link 指示灯定义

序号	指示灯	颜色定义	说明
1	PWR	红色	电源指示灯，用于指示 R-link 电源情况，正常情况下电源插入时都会亮起红色，如插入电源时红色不亮，请立刻移除电源，切不可再上电。
2	TXD	红色	串口通讯输出指示灯，常灭，当 R-link 串口有数据输出时，闪烁。
3	RXD	红色	串口通讯输入指示灯，常灭，当 R-link 串口有数据输入时，闪烁。

2.3 驱动器与 R-link 连接及注意事项

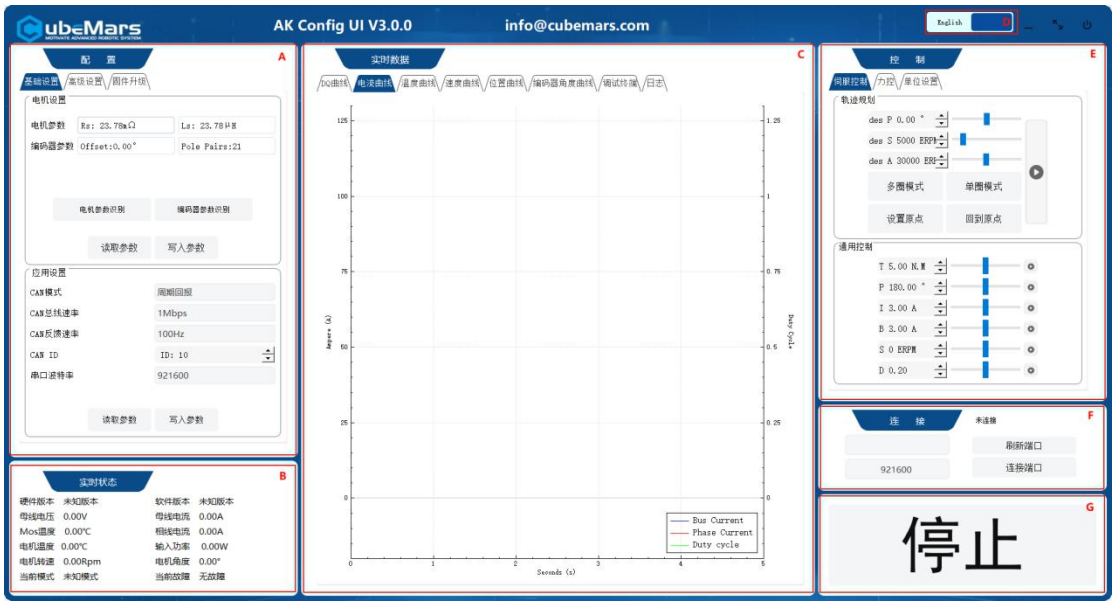


R-Link 上的 USB ----> PC 端
3Pin 口 ----> RLink8Pin 口端
7Pin 端子 ----> 电机上的 7Pin 口

E-mail:info@cubemars.com
https://www.cubemars.com/

3.上位机使用说明

3.1 上位机界面及说明



A. 配置
B. 实时状态
C. 实时数据
D. 中英文切换
E. 控制
F. 连接
G. 停止

E-mail:info@cubemars.com
https://www.cubemars.com/

3. 1. 1 配置

3.1.1.1 基础设置

基础设置界面可供用户电机参数与通讯参数配置；电机设置栏支持电机参数识别、编码器参数校准、参数读取与写入，具体电机校准操做见 3.2；应用设置栏中支持对 CAN 口模式、CAN 总线速率、CAN 反馈速率、CAN ID、串口波特率设置，用户可根据实际工况匹配。

CAN 口模式分为周期回报和问答模式两种：周期回报会根据 CAN 反馈速率大小定时反馈消息报文；问答模式只有在电机接受到正确的控制命令时才会反馈消息报文。

电机参数识别：进行电机参数识别，可以在位置曲线中查看电机是否在运动，识别完成后会自动停止并更新电机参数至上位机。

⚠：电机参数识别全过程应保持空载状态，否则可能会导致识别参数不准或者电机损坏。识别过程开始会出现较尖锐短促的声音，随后电机会开始旋转，伴随声音较大。

编码器参数识别：进行编码器参数识别，可以在位置曲线中查看电机是否在运动，识别完成后会自动停止并更新编码器参数至上位机。

⚠：电机编码器参数识别全过程应保持空载状态，否则可能会导致识别参数不准或者电机损坏；识别过程将持续发出较尖锐的声音且电机发热量较大，多次连续进行此过程会使电机温度骤升。

E-mail:info@cubemars.com
https://www.cubemars.com/

读取参数 ：将驱动板电机参数及编码器参数读取至上位机。

写入参数 ：将上位机电机参数及编码器参数写入驱动板。

3.1.1.2 高级设置

⚠：注意，初次使用本产品客户请不要随意修改高级设置中的相关参数，否则电机可能会出现运行异常。

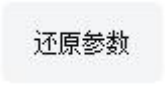
高级设置界面可供用户更多的自定义参数，用户可根据需求进行设置；为方便使用便捷，除参数读取、写入功能外，另提供参数还原、载入、导出功能。

读取参数 读取参数：点击后将驱动板参数读取至上位机。

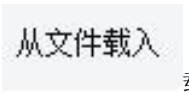
写入参数 写入参数：点击后将当前上位机参数写入至驱动板。

⚠：改写电机参数前务必先读取参数，否则电机其他参数将会出错，如果发生此类情况，请到官网下载对应电机的默认 APP 参数，再通过”导入设置”将该电机的默认参数写入电机！

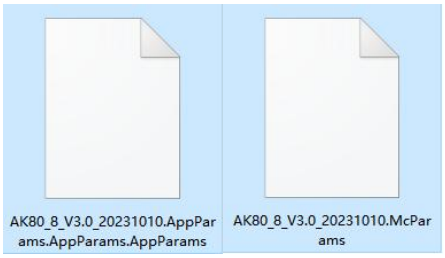
E-mail:info@cubemars.com
https://www.cubemars.com/

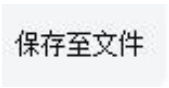


还原参数：点击后上位机参数将还原成对应机型的默认数值。



载入参数：将参数文件载入到上位机；点击后，选择对应电机型号的参数文件后即可，两份参数文件格式为.AppParams 与.McParams。





导出参数：点击导出参数，选择保存文件路径后即可得到自定义的参数文件。

⚠：请务必严格按照规定电压、电流、功率、温度使用。因违规操作本产品导致对人体造成伤害，或对驱动板及电机造成不可逆的损伤，我司将不承担任何法律责任。

E-mail:info@cubemars.com
https://www.cubemars.com/

3.1.1.3 固件升级

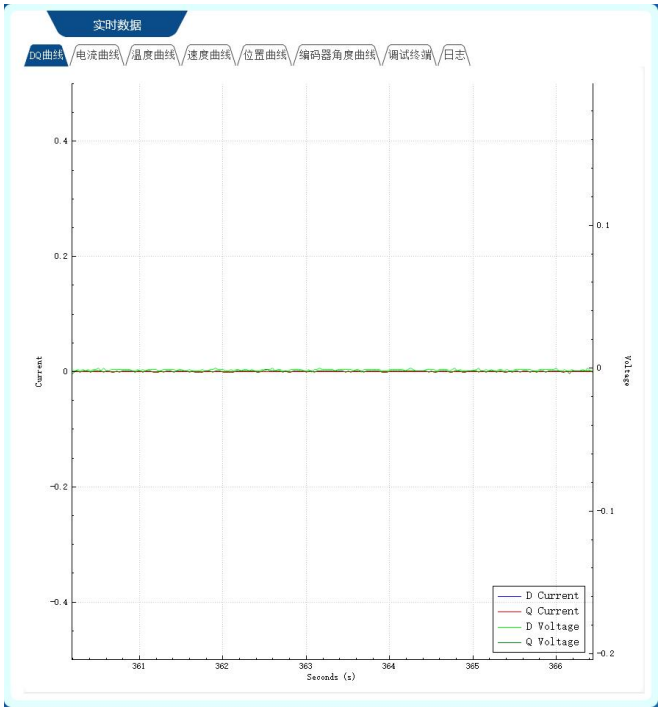
固件升级界面包含固件信息、升级选项、升级进度三个栏目，客户可根据自主需求进行固件升级服务，具体升级固件步骤见 3.4。

3.1.2 实时状态

实时状态界面会显示当前的软硬件版本、电机的固件版本、电机参数状态、运行模式及故障状态。

E-mail:info@cubemars.com
https://www.cubemars.com/

3.1.3 实时数据



该页面支持观看实时数据反馈与调试中端界面，并绘制图像。数据包括：

DQ：DQ 电压表示电机电压换算在 D 轴和 Q 轴上的电压，DQ 电流表示电机电流换算在 D 轴和 Q 轴上的电流。

电流:Current in 表示电机的输入电流，Current motor 表示电机的相线电流，Duty cycle 表示电机的占空比。

温度:Temperature MOSFET 表示电机 MOS 管温度，Temperature Motor 表示电机温度。

速度:ERPM 表示电机转子的电气转速。

位置:Position 表示电机转子的位置。

编码器角度:Encoder angle 表示编码器角度。

3.1.4 中英文切换

上位机目前支持中文、英语两种语言切换，点击界面右上角红色框内的“中”“EN”即可进行语言切换。

E-mail:info@cubemars.com
https://www.cubemars.com/

3. 1. 5 控制

3.1.5.1 伺服控制

多圈模式：在位置速度环模式中，电机可设置位置在-36000°-36000°；

单圈模式：在位置速度环模式中，电机可设置位置在 0-359°；

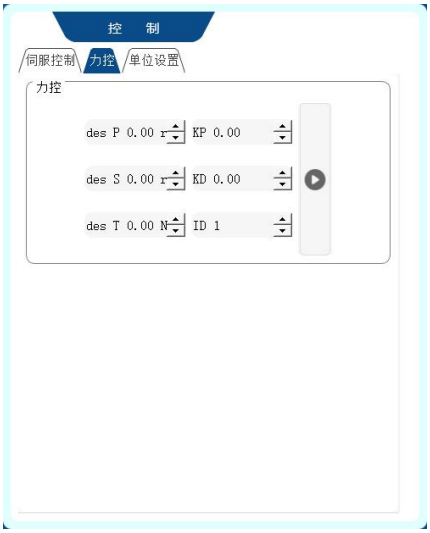
设置原点：电机将把当前转子位置设置为零点位置；

回到原点：电机旋转至零点位置；（电机会以最快速度运行，注意安全）


：对应模式启动按键。

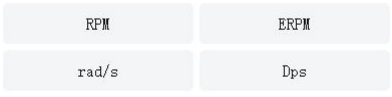
E-mail:info@cubemars.com
https://www.cubemars.com/

3.1.5.2 力控



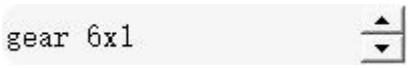
力控模式（运动控制模式、MIT MODE）提供位置、速度、扭矩环三种不同的控制模式，由发送的数据决定，具体操作方式可见 3.3.2。

3.1.5.3 单位设置



:每个按键对应所需要转换成为的单位，支持转速（RPM、ERPM、rad/s、Dps）转换显示；

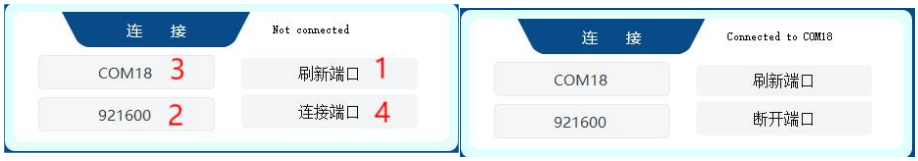
:填写电机极数，如左图所示，21 为极对数；



:填写电机减速比。

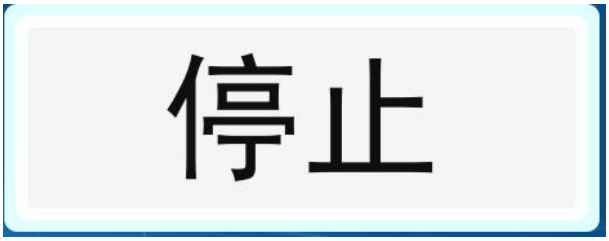
E-mail:info@cubemars.com
https://www.cubemars.com/

3.1.6 连接



连接模块可对串口通讯进行配置，如左图所示，首先点击“刷新端口”，随后选择通讯波特率，选择正确端口，最后电机“连接端口”。显示字样由“Not connected”变为“Connected to COMX”表明串口连接无误。连接成功状态如右图，点击“断开端口”即断开通讯。

3.1.7 停止



实验停止按键，点击后电机将停止运动。

3.2 驱动板校准

当你重新安装了驱动板在电机上，或者更换过电机三相线的线序，亦或者更新了固件之后，需重新进行校准（第一次出厂时已校准，无需再次校准）。校准之后，便可以正常使用电机。

3.2.1 校准步骤

STEP0:

确认电机输入电源稳定，连接器连接正常，在与上位机成功连接后，进入系统设置页面;

STEP1:

单击“读取参数”，待连接界面显示 

；

STEP2:

点击“电机参数识别”，伴随短促蜂鸣声后，出现 

；随后电机开始

E-mail:info@cubemars.com
https://www.cubemars.com/

旋转，大约等待 10S 后电机停止旋转，待出现 

后，表明电机参数识别过程完成。

STEP3:

点击“编码器参数识别”,电机慢速旋转，大约等待 45S,连接界面显示 

即完成编码器参数识别；

STEP4:

点击“写入参数”，连接界面显示 

即完成校准。

⚠：电机参数与编码器参数识别全过程应保持空载状态，否则可能会导致识别参数不准或者电机损坏；编码器参数识别过程电机发热量较大，多次连续进行此过程会使电机温度骤升。

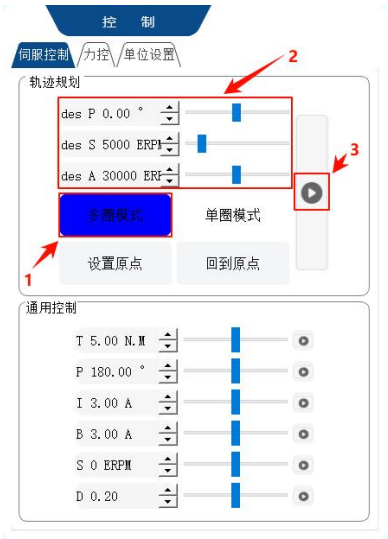
3.3 控制演示

在确认电机电源、负载正确安装、接线无误并已经完成了驱动板校准后，即可开始使用电机，下面分别对伺服控制和力控模式使用进行说明。

3. 3. 1 伺服模式

3.3.1.1 多圈位置速度环模式

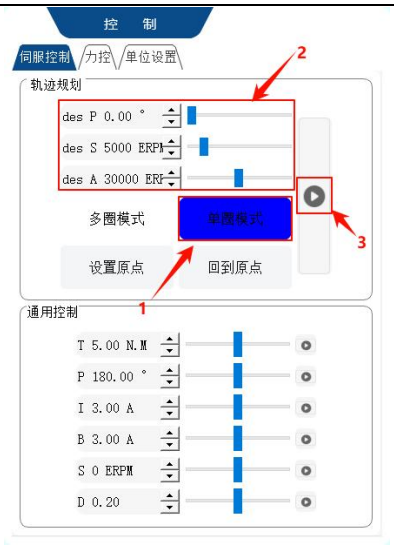
确认电机输入电源稳定，连接器连接正常，在与上位机成功连接后，在“伺服控制”界面单击“多圈模式”，输入期望位置（此时位置为±100 圈，即-36000°-36000°）、期望速度和加速度后，电机将会按照期望速度运动直到期望位置。



3.3.1.2 单圈位置速度环模式

确认电机输入电源稳定，连接器连接正常，在与上位机成功连接后，在“伺服控制”界面单击“单圈模式”，输入期望位置（此时位置只有一圈，即 0° -359°）、期望速度和加速度后，电机将会按照期望速度运动直到期望位置。

E-mail:info@cubemars.com
https://www.cubemars.com/



3.3.1.3 抱闸模式

确认电机输入电源稳定，连接器连接正常，在与上位机成功连接后，在“伺服控制”界面输入期望扭矩 T，电机将以期望扭矩抱闸。

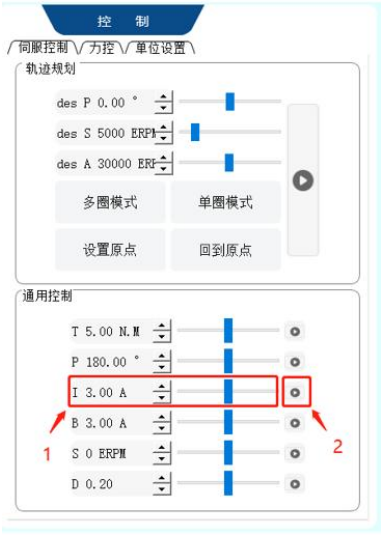
3.3.1.4 位置环模式

确认电机输入电源稳定，连接器连接正常，在与上位机成功连接后，在“伺服控制”界面输入期望位置 P，电机将以最大转速和加速度到达期望位置。

E-mail:info@cubemars.com
https://www.cubemars.com/

3.3.1.5 电流环模式

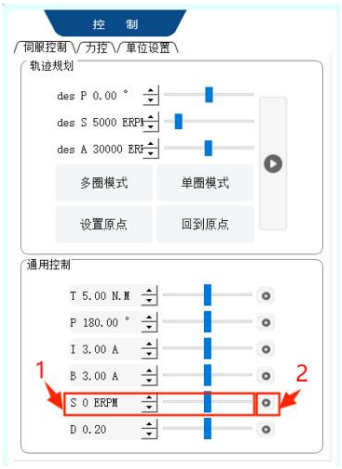
确认电机输入电源稳定，连接器连接正常，在与上位机成功连接后，在“伺服控制”界面输入期望电流 I，电机将以期望电流运动。



3.3.1.6 速度环模式

确认电机输入电源稳定，连接器连接正常，并且电机处于伺服模式时，在与上位机成功连接后，在“伺服控制”界面输入期望转速 S（±50000ERPM），电机将以期望转速运动（加速度默认最大）。

E-mail:info@cubemars.com
https://www.cubemars.com/



3.3.1.7 刹车环模式

确认电机输入电源稳定，连接器连接正常，在与上位机成功连接后，在“伺服控制”界面输入期望刹车电流 B，电机将以期望电流进行刹车。

3.3.1.8 占空比模式

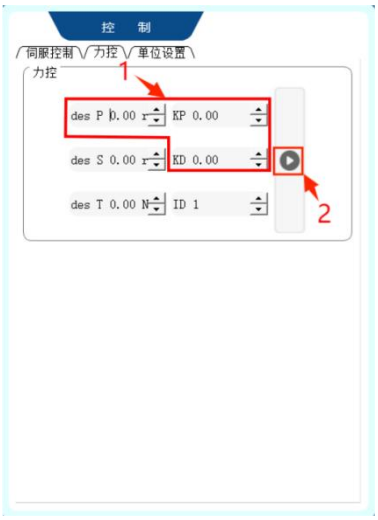
确认电机输入电源稳定，连接器连接正常，并且电机处于伺服模式时，在与上位机成功连接后，在“伺服控制”界面输入期望占空比（默认 0.005-0.95），电机将以期望占空比运动。

E-mail:info@cubemars.com
https://www.cubemars.com/

3. 3. 2 力控模式

3.3.2.1 力控位置模式

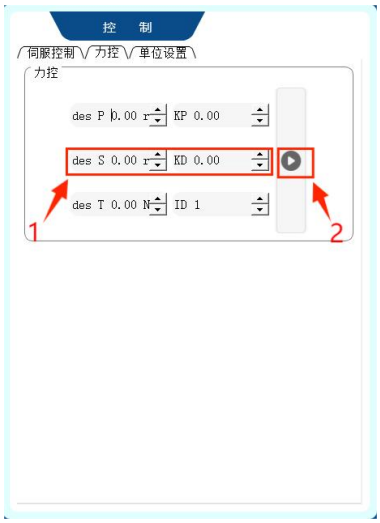
确认电机输入电源稳定，连接器连接正常，在与上位机成功连接后，在“运动控制”界面输入对应的“CAN ID”，输入期望位置与 KP、KD 后，再单击“启动”，电机将会进行位置运动。（只给定电机指定的位置及 Kp、Kd 值运行，电机将向指定位置转动）



3.3.2.2 力控速度模式

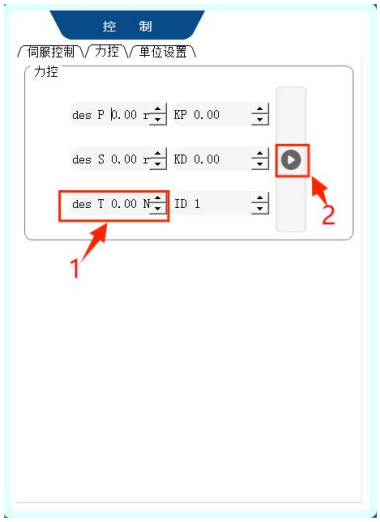
确认电机输入电源稳定，连接器连接正常，输入期望速度 S 和 KD 后，再单击“启动”，电机将会进行速度运动。（只给定电机运行速度及 Kd 值运行，电机将以指定速度转动）

E-mail:info@cubemars.com
https://www.cubemars.com/



3.3.2.3 力控扭矩模式

确认电机输入电源稳定，连接器连接正常，输入期望扭矩 T 后，再单击“启动”，电机将会进行扭矩运动。（只给定扭矩值运行，电机将以指定扭矩值转动）



E-mail:info@cubemars.com
https://www.cubemars.com/

3.4 固件更新

1.在下拉列表选择对应固件；
2.单击“跳转引导”；
3.单击“开始升级”，等待升级进度条完成至 100%；
4.单击”跳转应用”，等待 5S，电机进入运行模式，固件更新完毕。

E-mail:info@cubemars.com
https://www.cubemars.com/

4. 驱动板通讯协议及说明

4.1 伺服模式控制模式及说明

伺服模式有 6 种控制模式，每个控制模式对应特定的 ID 号。

占空比模式：给定电机指定占空比电压，类似方波驱动形式；
电流环模式：给定电机指定的 Iq 电流，由于电机输出扭矩=iq*KT，所以可以当作扭矩环使用（KT 数据可参考官网）；
电流刹车模式：给定电机指定的刹车电流，让电机固定在当前位置（使用时注意电机温度）；
速度模式：给定电机指定的运行速度（加速度默认为最大值）；
位置模式：给定电机指定的位置，电机将运行到该指定的位置（速度和加速度默认为最大值）；
位置速度环模式：给定电机指定的位置、速度、加速度。

伺服模式驱动板发送数据定义
标识符： 控制模式 ID+驱动器 ID　　帧类型：拓展帧
帧格式： DATA　　DLC: 8 字节
伺服电机协议为 CAN 协议，采用扩展帧格式如下示

Can ID bits	[28]-[8]	[7]-[0]
Field name (功能定义)	Control mode ID （控制模式 ID）	Drive ID （驱动器 ID）

Control mode 有{0,1,2,3,4,5,6} 7 个特征值分别对应 7 种控制模式
占空比模式：0
电流环模式：1
电流刹车模式：2
转速模式：3
位置模式：4
设置原点模式：5
位置速度环模式：6

以下提供各种模式控制电机实例
下面为各种实例调用库 函数与宏定义

```
typedef enum {
    CAN_PACKET_SET_DUTY = 0,            //占空比模式
    CAN_PACKET_SET_CURRENT,             //电流环模式
    CAN_PACKET_SET_CURRENT_BRAKAE,      // 电流刹车模式
    CAN_PACKET_SET_RPM,                 // 转速模式
    CAN_PACKET_SET_POS,                 // 位置模式
    CAN_PACKET_SET_ORIGIN_HERE,         //设置原点模式
    CAN_PACKET_SET_POS_SPD,             //位置速度环模式
} CAN_PACKET_ID;
```


E-mail:info@cubemars.com
https://www.cubemars.com/

```
void comm_can_transmit_eid(uint32_t id, const uint8_t *data, uint8_t len) {
	uint8_t i=0;
	if (len > 8) {
		len = 8;
	}
	CanTxMsg TxMessage;
	TxMessage.StdId = 0;
	TxMessage.IDE = CAN_ID_EXT;
	TxMessage.ExtId = id;
	TxMessage.RTR = CAN_RTR_DATA;
	TxMessage.DLC = len;
	for(i=0;i<len;i++)
		TxMessage.Data[i]=data[i];
	CAN_Transmit(CHASSIS_CAN, &TxMessage);    //CAN 口发送 TxMessage 数据
}

void buffer_append_int32(uint8_t* buffer, int32_t number, int32_t *index) {
	buffer[(*index)++] = number >> 24;
	buffer[(*index)++] = number >> 16;
	buffer[(*index)++] = number >> 8;
	buffer[(*index)++] = number;
}

void buffer_append_int16(uint8_t* buffer, int16_t number, int16_t *index) {
	buffer[(*index)++] = number >> 8;
	buffer[(*index)++] = number;
}
```

4.1.1 占空比模式

占空比模式发送数据定义

数据位	Data[0]	Data[1]	Data[2]	Data[3]
范围	0~0xff	0~0xff	0~0xff	0~0xff
对应变量	占空比 25-32 位	占空比 17-24 位	占空比 9-16 位	占空比 1-8 位

```
void comm_can_set_duty(uint8_t controller_id, float duty) {
	int32_t send_index = 0;
	uint8_t buffer[4];
	buffer_append_int32(buffer, (int32_t)(duty * 100000.0), &send_index);
	comm_can_transmit_eid(controller_id |
```


E-mail:info@cubemars.com
https://www.cubemars.com/

```
                ((uint32_t)CAN_PACKET_SET_DUTY << 8), buffer, send_index);
}
```

4.1.2 电流环模式

电流环模式发送数据定义

数据位	Data[0]	Data[1]	Data[2]	Data[3]
范围	0~0xff	0~0xff	0~0xff	0~0xff
对应变量	电流 25-32 位	电流 17-24 位	电流 9-16 位	电流 1-8 位

其中，电流数值为 int32 类型，数值-60000-60000 代表-60-60A。

电流环模式发送例程

```
void comm_can_set_current(uint8_t controller_id, float current) {
    int32_t send_index = 0;
    uint8_t buffer[4];
    buffer_append_int32(buffer, (int32_t)(current * 1000.0), &send_index);
    comm_can_transmit_eid(controller_id |
                ((uint32_t)CAN_PACKET_SET_CURRENT << 8), buffer, send_index);
}
```

4.1.3 电流刹车模式

电流刹车模式发送数据定义

数据位	Data[0]	Data[1]	Data[2]	Data[3]
范围	0~0xff	0~0xff	0~0xff	0~0xff
对应变量	刹车电流 25-32 位	刹车电流 17-24 位	刹车电流 9-16 位	刹车电流 1-8 位

其中，刹车电流数值为 int32 类型，数值 0-60000 代表 0-60A。

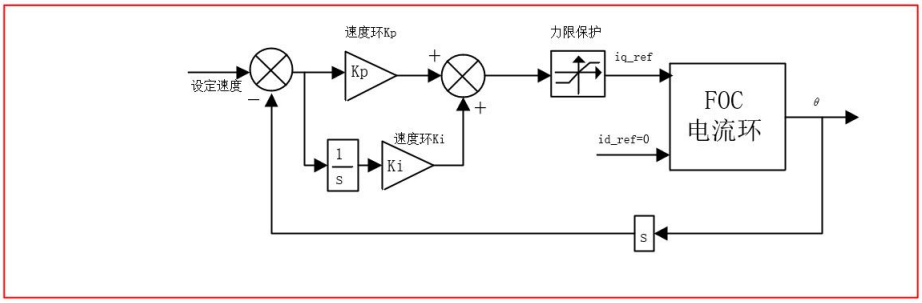
电流刹车模式发送例程

```
void comm_can_set_cb(uint8_t controller_id, float current) {
    int32_t send_index = 0;
    uint8_t buffer[4];
    buffer_append_int32(buffer, (int32_t)(current * 1000.0), &send_index);
    comm_can_transmit_eid(controller_id |
            ((uint32_t)CAN_PACKET_SET_CURRENT_BRAKAE << 8), buffer, send_index);
}
```


E-mail:info@cubemars.com
https://www.cubemars.com/

4.1.4 速度环模式

速度环简略控制框图



速度环模式发送数据定义

数据位	Data[0]	Data[1]	Data[2]	Data[3]
范围	0~0xff	0~0xff	0~0xff	0~0xff
对应变量	速度 25-32 位	速度 17-24 位	速度 9-16 位	速度 1-8 位

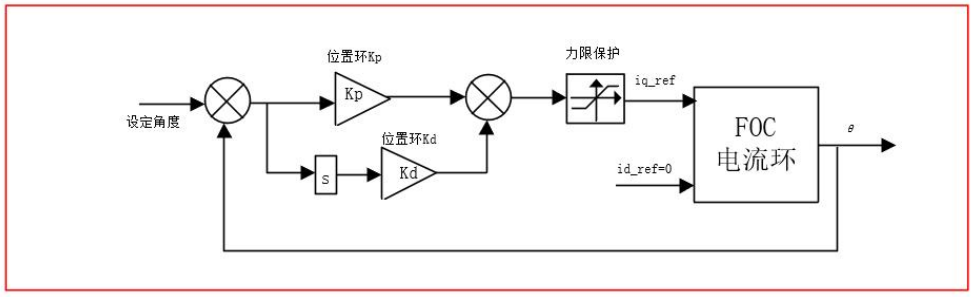
其中，速度数值为 int32 型，范围-100000-100000 代表-100000-100000 电气转速。

速度环发送例程

```
void comm_can_set_rpm(uint8_t controller_id, float rpm) {
	int32_t send_index = 0;
	uint8_t buffer[4];
	buffer_append_int32(buffer, (int32_t)rpm, &send_index);
	comm_can_transmit_eid(controller_id |
			((uint32_t)CAN_PACKET_SET_RPM << 8), buffer, send_index);
}
```

4.1.5 位置环模式

位置环简略控制框图



E-mail:info@cubemars.com
https://www.cubemars.com/

位置环模式发送数据定义

数据位	Data[0]	Data[1]	Data[2]	Data[3]
范围	0~0xff	0~0xff	0~0xff	0~0xff
对应变量	位置 25-32 位	位置 17-24 位	位置 9-16 位	位置 1-8 位

其中，位置为 int32 型，范围-360000000-360000000 代表位置-36000°~36000°；

位置环发送例程

```
void comm_can_set_pos(uint8_t controller_id, float pos) {
    int32_t send_index = 0;
    uint8_t buffer[4];
    buffer_append_int32(buffer, (int32_t)(pos * 10000.0), &send_index);
    comm_can_transmit_eid(controller_id |
            ((uint32_t)CAN_PACKET_SET_POS << 8), buffer, send_index);
}
```

4.1.6 设置原点模式

数据位	Data[0]
范围	0~0x01
对应变量	设置指令

其中，设置指令为 uint8_t 型，0 代表设置临时原点(断电消除)，1 代表设置永久零点(参数自动保存);

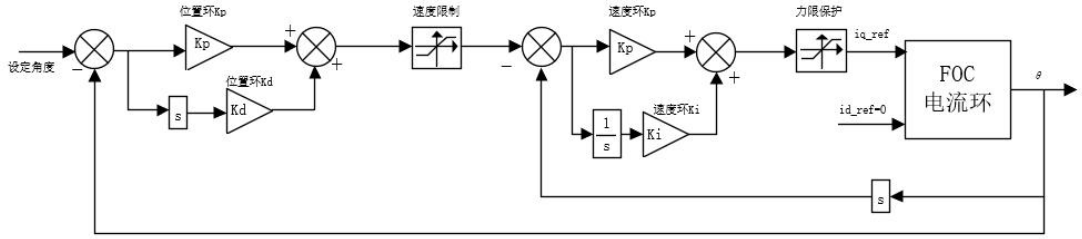
设置原点发送例程

```
void comm_can_set_origin(uint8_t controller_id, uint8_t set_origin_mode) {
    int32_t send_index = 1;
    uint8_t buffer;
    buffer=set_origin_mode;
    comm_can_transmit_eid(controller_id |
            ((uint32_t) CAN_PACKET_SET_ORIGIN_HERE << 8), buffer, send_index);
}
```

4.1.7 位置速度环模式

位置速度环简略框图

E-mail:info@cubemars.com
https://www.cubemars.com/



位置速度环模式发送数据定义

数据位	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
范围	0~0xff	0~0xff	0~0xff	0~0xff	0~0xff	0~0xff	0~0xff	0~0xff
对应变量	位置 25-32 位	位置 17-24 位	位置 9-16 位	位置 1-8 位	速度高八位	速度低八位	加速度高八位	加速度低八位

其中，位置为 int32 型，范围-360000000~360000000 对应-位置-36000°~36000°；
其中，速度为 int16 型，范围-32768~32767 对应-327680~-327680 电气转速；
其中，加速度为 int16 型，范围 0~32767，对应 0~327670，1 单位等于 10 电气转速/s²。

```
void comm_can_set_pos_spd(uint8_t controller_id, float pos,int16_t spd, int16_t RPA ) {
    int32_t send_index = 0;
    Int16_t send_index1 = 4;
    uint8_t buffer[8];
    buffer_append_int32(buffer, (int32_t)(pos * 10000.0), &send_index);
    buffer_append_int16(buffer,spd/10.0, & send_index1);
    buffer_append_int16(buffer,RPA/10.0, & send_index1);
    comm_can_transmit_eid(controller_id |
            ((uint32_t)CAN_PACKET_SET_POS_SPD << 8), buffer, send_index1);
}
```

4.2 力控模式通讯协议

力控模式有 3 种控制模式，三种模式共用一个控制 ID 号，根据发送数据来决定控制模式，各种模式上位机操作详细见 3.3.2。

位置环模式： 只给定电机指定的位置及 Kp、Kd 值运行，电机将向指定位置转动；
速度环模式： 只给定电机运行速度及 Kd 值运行，电机将以指定速度转动；
电流环模式： 只给定扭矩值运行，电机将以指定扭矩值转动。

力控模式使用 can 协议，采用扩展帧格式如下所示

Can ID bits	[28]-[8]	[7]-[0]
Field name (功能定义)	Control mode ID （控制模式 ID）	Drive ID （驱动器 ID）

力控模式的控制模式 ID 特征值为 8。

E-mail:info@cubemars.com
https://www.cubemars.com/

力控模式驱动板发送数据定义

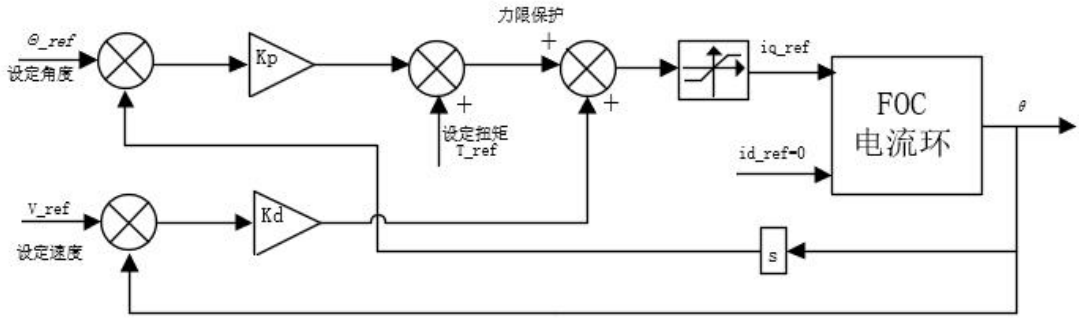
标识符： 控制模式 ID+驱动器 ID　　帧类型：拓展帧

帧格式： DATA　　DLC: 8 字节

数据域	DATA[0]	DATA[1]		DATA[2]	DATA[3]
数据位	7-0	7-4	3-0	7-0	7-0
数据内容	KP 值高 8 位	KP 值低 4 位	KD 值高 4 位	KD 值低 8 位	位置值高 8 位

数据域	DATA[4]	DATA[5]	DATA[6]		DATA[7]
数据位	7-0	7-0	7-4	3-0	0-7
数据内容	位置值低 8 位	速度值高 8 位	速度值低 4 位	电流值高 4 位	电流值低 8 位

力控模式简略框图



参数范围

模组电机	AKA10-9	AKA60-6	AKA70-9
KV	60	80	60
电机位置（rad）	-12.56f-12.56f		
电机速度（rad/s）	-28.0f-28.0f	-60.0f-60.0f	-30.0f-30.0f
电机扭矩（N.M）	-54.0f-54.0f	-12.0f-12.0f	-32.0f-32.0f
Kp 范围	0-500		
Kd 范围	0-5		

力控模式发送代码例程

发送例程代码 （参数以 AKA10-9 为例）

```
u8 SERVO_Can_Send_Msg(uint32_t ExtId,u8* msg,u8 len)
{
        u8 mbox;
```


E-mail:info@cubemars.com
https://www.cubemars.com/

```
    u16 i=0;
//      TxMessage.StdId=stdId;          // 标准标识符
    TxMessage.ExtId=ExtId;          // 设置扩展标示符
    TxMessage.IDE=CAN_Id_Extended;  // 标准帧
    TxMessage.RTR=CAN_RTR_Data;     // 数据帧
    TxMessage.DLC=len;              // 要发送的数据长度
    for(i=0;i<len;i++)
    TxMessage.Data[i]=msg[i];
    mbox= CAN_Transmit(CANx, &TxMessage);
    i=0;
    while((CAN_TransmitStatus(CAN1, mbox)==CAN_TxStatus_Failed)&&(i<0XFFF))i++;  //
等待发送结束
    if(i>=0XFFF)return 1;
    return 0;
}

float fmaxf(float x, float y){
    /// Returns maximum of x, y ///
    return (((x)>(y))?(x):(y));
    }

float fminf(float x, float y){
    /// Returns minimum of x, y ///
    return (((x)<(y))?(x):(y));
    }

float fmaxf3(float x, float y, float z){
    /// Returns maximum of x, y, z ///
    return (x > y ? (x > z ? x : z) : (y > z ? y : z));
    }

float fminf3(float x, float y, float z){
    /// Returns minimum of x, y, z ///
    return (x < y ? (x < z ? x : z) : (y < z ? y : z));
    }


float P_MIN =-12.56f;
float P_MAX =12.56f;
float V_MIN =-33.0f;
float V_MAX =33.0f;
float T_MIN =-65.0f;
float T_MAX =65.0f;
```


E-mail:info@cubemars.com
https://www.cubemars.com/

```
float Kp_MIN =0;
float Kp_MAX =500.0f;
float Kd_MIN =0;
float Kd_MAX =5.0f;
float Test_Pos=0.0f;


int p_int ;
int v_int ;
int kp_int ;
int kd_int ;
int t_int ;


void pack_cmd(uint8_t controller_id, float p_des, float v_des, float kp, float kd, float t_ff){
uint8_t buffer[8];

p_des = fminf(fmaxf(P_MIN, 0), P_MAX);
v_des = fminf(fmaxf(V_MIN, 0), V_MAX);
kp = fminf(fmaxf(Kp_MIN, kp), Kp_MAX);
kd = fminf(fmaxf(Kd_MIN, kd), Kd_MAX);
t_ff = fminf(fmaxf(T_MIN, t_ff), T_MAX);
/// convert floats to unsigned ints ///

  p_int = float_to_uint(p_des, P_MIN, P_MAX, 16);
  v_int = float_to_uint(v_des, V_MIN, V_MAX, 12);
  kp_int = float_to_uint(kp, Kp_MIN, Kp_MAX, 12);
  kd_int = float_to_uint(kd, Kd_MIN, Kd_MAX, 12);
  t_int = float_to_uint(t_ff, T_MIN, T_MAX, 12);

/// pack ints into the can buffer ///
buffer[0] = kp_int>>4;          //KP 高 8 位
buffer[1] = ((kp_int&0xF)<<4)|( kd_int>>8); //KP 低 4 位，Kd 高 4 位
buffer[2] = kd_int&0xFF;        //Kd 低 8 位
buffer[3] = p_int>>8;             //位置高 8 位
buffer[4] = p_int&0xFF; //位置低 8 位
buffer[5] = v_int>>4;     //速度高 8 位
buffer[6] = ((v_int&0xF)<<4)|(t_int>>8);      //速度低 4 位，扭矩高 4 位
buffer[7] = t_int&0xff;   //扭矩低 8 位
SERVO_Can_Send_Msg(controller_id |((uint32_t) CAN_PACKET_SET_mit << 8), buffer, 8);
        }
```

发包时所有的数都要经以下函数转化成整型数之后再发给电机。

```
        int float_to_uint(float x, float x_min, float x_max, unsigned int bits){
```


E-mail:info@cubemars.com
https://www.cubemars.com/

```
    /// Converts a float to an unsigned int, given range and number of bits ///
    float span = x_max - x_min;
    if(x < x_min) x = x_min;
    else if(x > x_max) x = x_max;
    return (int) ((x- x_min)*((float)((1<<bits)/span)));
}
```


E-mail:info@cubemars.com
https://www.cubemars.com/

4.3 电机报文格式

4.3.1 CAN 上传报文协议

电机 CAN 报文使用定时上传模式，上传频率可设置为 1~500HZ，上传字节为 8 字节

数据位	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
范围	0~0xff	0~0xff	0~0xff	0~0xff	0~0xff	0~0xff	0~0xff	0~0xff
对应变量	位置高八位	位置低八位	速度高八位	速度低八位	电流高八位	电流低八位	电机温度	报错代码

其中，位置为 int16 型，范围-32000~32000 代表位置-3200°~3200°；
其中，速度为 int16 型，范围-32000~32000 代表-320000~320000rpm 电气转速；
其中，电流为 int16 类型，数值-6000~6000 代表-60~60A
其中，温度为 int8 型，范围-20~127 代表驱动板温度-20℃~127℃；
其中，报错代码为 uint8 型，0 为无故障，1 为电机过温度故障，2 为过电流故障，3 为过压故障，4 为欠压故障，5 为编码器故障，6 为 mos 管过温度故障，7 为电机堵转。

以下为报文接受实例

```
void motor_receive(float* motor_pos,float*
motor_spd,float* cur,int_8* temp,int_8* error,rx_message)
  {
      int16_t pos_int = (rx_message)->Data[0] << 8 | (rx_message)->Data[1]);
      int16_t spd_int = (rx_message)->Data[2] << 8 | (rx_message)->Data[3]);
      int16_t cur_int = (rx_message)->Data[4] << 8 | (rx_message)->Data[5]);
    &motor_pos= (float)( pos_int * 0.1f); //电机位置
    &motor_spd= (float)( spd_int * 10.0f);//电机速度
    &motor_cur= (float) ( cur_int * 0.01f);//电机电流
    &motor_temp= (rx_message)->Data[6] ;//电机温度
    &motor_error= (rx_message)->Data[7] ;//电机故障码
  }
```

4.3.2 串口报文协议

串口收发报文协议如下

帧头（0xAA）	数据长度（不含帧头帧尾和校验位）	数据帧	数据位	校验高 8 位	校验低 8 位	帧尾（0xBB）

校验位计算代码参考 4.3.2.2 节。

E-mail:info@cubemars.com
https://www.cubemars.com/

数据帧定义

```
typedef enum {
    COMM_FW_VERSION = 65,
    COMM_JUMP_TO_BOOTLOADER=66,
    COMM_ERASE_NEW_APP=67,
    COMM_WRITE_NEW_APP_DATA=68,
    COMM_GET_VALUES=69,        //获取电机运行参数
    COMM_SET_DUTY=70,          //电机以占空比模式运行
    COMM_SET_CURRENT=71,    //电机以电流环模式运行
    COMM_SET_CURRENT_BRAKAE=72, //电机电流刹车模式模式运行
    COMM_SET_RPM=73,               //电机以速度环模式运行
    COMM_SET_POS=74,               //电机以位置环模式运行
    COMM_SET_HANDBRAKAE=75,     //电机以手刹电流环模式运行
    COMM_SET_DETECT=76,           //电机实时反馈当前位置指令
    COMM_ROTOR_POSITION=87,//电机反馈当前位置
    COMM_GET_VALUES_SETUP=16,//电机单个或多个参数获取指令
    COMM_SET_POS_SPD=60,        //    电机以位置速度环模式运行
    COMM_SET_POS_MULTI=61,      //    设置电机运动为单圈模式
    COMM_SET_POS_SINGLE=62,   //      设置电机运动为多圈模式  范围±100 圈
    COMM_SET_POS_UNLIMITED=63, //保留
    COMM_SET_POS_ORIGIN=64, //设置电机原点
} COMM_PACKET_ID;
```

4.3.2.1 获取电机参数实例

串口指令 ：AA 01 45 18 61 BB   // 获取电机参数指令 电机接收后反馈一次电机状态

电机回传串口指令实例：AA 49 45 01 27 FC DF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 F2 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF FF FF FD 00 00 00 07 00 FF EF 54 F1 68 00 00 00 00 00 00 00 00 00 00 00 FF FF FF FC 86 D5 BB

// AA（帧头）+49（数据长度）+ 45（数据帧） + mos 温度（2byte）+电机温度（2byte）+输出电流（4byte）+输入电流（4byte）+Id 电流（4byte）+Iq 电流（4byte）+电机油门值（2byte）+电机转速（4byte）+输入电压（2byte）+保留值（24byte）+电机状态码（1byte）+电机外环位置值（4byte）+电机控制 ID 号（1byte）+温度保留值（6byte）+Vd 电压值（4byte）+Vq 电压值（4byte）+CRC+BB(帧尾)

电机发出的部分参数转化公式如下

MOS 温度=(float)buffer_get_int16(data, &ind) / 10.0)

电机温度=(float)buffer_get_int16(data, &ind) / 10.0)

输出电流=(float)buffer_get_int32(data, &ind) / 100.0)

输入电流=(float)buffer_get_int32(data, &ind) / 100.0)

E-mail:info@cubemars.com
https://www.cubemars.com/

Id 电流=(float)buffer_get_int32(data, &ind) / 100.0)
Iq 电流=(float)buffer_get_int32(data, &ind) / 100.0)
电机油门值=(float)buffer_get_int16(data, &ind) / 1000.0)
电机转速=(float)buffer_get_int32(data, &ind))
输入电压=(float)buffer_get_int16(data, &ind) / 10.0)
电机外环位置=(float)buffer_get_int32(data, &ind) / 1000.0)
电机 ID 号=data
电机 Vd 电压=(float)buffer_get_int32(data, &ind) / 1000.0)
电机 Vq 电压=(float)buffer_get_int32(data, &ind) / 1000.0)

电机反馈位置报文指令
串口指令：AA 02 4C 04 08 25 BB　// 电机接收后隔 10ms 发送一次当前位置

电机反馈位置值发送实例（需要提前向电机发反馈位置报文指令，电机接收后每隔 10ms 发送一次当前位置）
串口指令：AA 05 57 00 1A B6 64 6E CD BB
Pos=(float)buffer_get_int32(data, &ind) / 1000.0

电机单个或多个参数获取指令实例
串口指令 AA 05 13 00 00 00 01 FA A9 BB　// 获取电机温度指令
指令说明： 此指令可以获取单个 或者多个电机参数　获取参数由数据段 4 字节 位决定，对应位位 1 时，电机将回传对应位的电机参数，为 0 时除去该字段
位对应的电机参数如下表格

32-19 位	18 位	17 位	16 位	10-15 位	9 位	8 位	7 位
保留值	电机 ID（1byte）	电机位置（4byte）	电机错误标志（1byte）	保留值	输入电压（2byte）	电机转速（4byte）	占空比（2byte）
6 位	5 位	4 位	3 位	2 位	1 位		
Iq 电流（4byte）	Id 电流（4byte）	输入电流（4byte）	输出电流（4byte）	电机温度（2byte）	MOS 温度（2byte）		

电机收到该指令后会发出对应参数
实例：AA 07 13 00 00 00 01 01 21 DF BB BB　// 反馈电机温度
电机发出的部分参数转化公式如下
MOS 温度=(float)buffer_get_int16(data, &ind) / 10.0)
电机温度=(float)buffer_get_int16(data, &ind) / 10.0)
输出电流=(float)buffer_get_int32(data, &ind) / 100.0)
输入电流=(float)buffer_get_int32(data, &ind) / 100.0)
电机油门值=(float)buffer_get_int16(data, &ind) / 1000.0)
电机转速=(float)buffer_get_int32(data, &ind))
输入电压=(float)buffer_get_int16(data, &ind) / 10.0)

E-mail:info@cubemars.com
https://www.cubemars.com/

电机位置=(float)buffer_get_int32(data, &ind) / 1000000.0)
电机 ID 号=data
电机错误状态码

```
typedef enum {
	FAULT_CODE_NONE = 0,
	FAULT_CODE_OVER_VOLTAGE,// 过压
	FAULT_CODE_UNDER_VOLTAGE,// 欠压
	FAULT_CODE_DRV,// 驱动故障
	FAULT_CODE_ABS_OVER_CURRENT,// 电机过流
	FAULT_CODE_OVER_TEMP_FET,// MOS 过温
	FAULT_CODE_OVER_TEMP_MOTOR,//电机过温
	FAULT_CODE_GATE_DRIVER_OVER_VOLTAGE,// 驱动过压
	FAULT_CODE_GATE_DRIVER_UNDER_VOLTAGE,// 驱动欠压
	FAULT_CODE_MCU_UNDER_VOLTAGE,// MCU 欠压
	FAULT_CODE_BOOTING_FROM_WATCHDOG_RESET,// 欠压
	FAULT_CODE_ENCODER_SPI,// SPI 编码器故障
	FAULT_CODE_ENCODER_SINCOS_BELOW_MIN_AMPLITUDE,//编码器超限
	FAULT_CODE_ENCODER_SINCOS_ABOVE_MAX_AMPLITUDE,//编码器超限
	FAULT_CODE_FLASH_CORRUPTION,// FLASH 故障
	FAULT_CODE_HIGH_OFFSET_CURRENT_SENSOR_1,// 电流采样通道 1 故障
	FAULT_CODE_HIGH_OFFSET_CURRENT_SENSOR_2,// 电流采样通道 2 故障
	FAULT_CODE_HIGH_OFFSET_CURRENT_SENSOR_3,// 电流采样通道 1 故障
	FAULT_CODE_UNBALANCED_CURRENTS,// 电流不平衡

} mc_fault_code;
```

4.3.2.2 串口校验

```
unsigned short crc16(unsigned char *buf, unsigned int len) {
	unsigned int i;
	unsigned short cksum = 0;
	for (i = 0; i < len; i++) {
	cksum = crc16_tab[(((cksum >> 8) ^ *buf++) & 0xFF)] ^ (cksum << 8);
	}
	return cksum;
}
const unsigned short crc16_tab[] = { 0x0000, 0x1021, 0x2042, 0x3063, 0x4084,
0x50a5, 0x60c6, 0x70e7, 0x8108, 0x9129, 0xa14a, 0xb16b, 0xc18c, 0xd1ad,
0xe1ce, 0xf1ef, 0x1231, 0x0210, 0x3273, 0x2252, 0x52b5, 0x4294, 0x72f7,
0x62d6, 0x9339, 0x8318, 0xb37b, 0xa35a, 0xd3bd, 0xc39c, 0xf3ff, 0xe3de,
0x2462, 0x3443, 0x0420, 0x1401, 0x64e6, 0x74c7, 0x44a4, 0x5485, 0xa56a,
0xb54b, 0x8528, 0x9509, 0xe5ee, 0xf5cf, 0xc5ac, 0xd58d, 0x3653, 0x2672,
0x1611, 0x0630, 0x76d7, 0x66f6, 0x5695, 0x46b4, 0xb75b, 0xa77a, 0x9719,
0x8738, 0xf7df, 0xe7fe, 0xd79d, 0xc7bc, 0x48c4, 0x58e5, 0x6886, 0x78a7,
0x0840, 0x1861, 0x2802, 0x3823, 0xc9cc, 0xd9ed, 0xe98e, 0xf9af, 0x8948,
```


E-mail:info@cubemars.com
https://www.cubemars.com/

```
0x9969, 0xa90a, 0xb92b, 0x5af5, 0x4ad4, 0x7ab7, 0x6a96, 0x1a71, 0x0a50,
0x3a33, 0x2a12, 0xdbfd, 0xcbdc, 0xfbbf, 0xeb9e, 0x9b79, 0x8b58, 0xbb3b,
0xab1a, 0x6ca6, 0x7c87, 0x4ce4, 0x5cc5, 0x2c22, 0x3c03, 0x0c60, 0x1c41,
0xedae, 0xfd8f, 0xcdec, 0xddcd, 0xad2a, 0xbd0b, 0x8d68, 0x9d49, 0x7e97,
0x6eb6, 0x5ed5, 0x4ef4, 0x3e13, 0x2e32, 0x1e51, 0x0e70, 0xff9f, 0xefbe,
0xdfdd, 0xcffc, 0xbf1b, 0xaf3a, 0x9f59, 0x8f78, 0x9188, 0x81a9, 0xb1ca,
0xa1eb, 0xd10c, 0xc12d, 0xf14e, 0xe16f, 0x1080, 0x00a1, 0x30c2, 0x20e3,
0x5004, 0x4025, 0x7046, 0x6067, 0x83b9, 0x9398, 0xa3fb, 0xb3da, 0xc33d,
0xd31c, 0xe37f, 0xf35e, 0x02b1, 0x1290, 0x22f3, 0x32d2, 0x4235, 0x5214,
0x6277, 0x7256, 0xb5ea, 0xa5cb, 0x95a8, 0x8589, 0xf56e, 0xe54f, 0xd52c,
0xc50d, 0x34e2, 0x24c3, 0x14a0, 0x0481, 0x7466, 0x6447, 0x5424, 0x4405,
0xa7db, 0xb7fa, 0x8799, 0x97b8, 0xe75f, 0xf77e, 0xc71d, 0xd73c, 0x26d3,
0x36f2, 0x0691, 0x16b0, 0x6657, 0x7676, 0x4615, 0x5634, 0xd94c, 0xc96d,
0xf90e, 0xe92f, 0x99c8, 0x89e9, 0xb98a, 0xa9ab, 0x5844, 0x4865, 0x7806,
0x6827, 0x18c0, 0x08e1, 0x3882, 0x28a3, 0xcb7d, 0xdb5c, 0xeb3f, 0xfb1e,
0x8bf9, 0x9bd8, 0xabbb, 0xbb9a, 0x4a75, 0x5a54, 0x6a37, 0x7a16, 0x0af1,
0x1ad0, 0x2ab3, 0x3a92, 0xfd2e, 0xed0f, 0xdd6c, 0xcd4d, 0xbdaa, 0xad8b,
0x9de8, 0x8dc9, 0x7c26, 0x6c07, 0x5c64, 0x4c45, 0x3ca2, 0x2c83, 0x1ce0,
0x0cc1, 0xef1f, 0xff3e, 0xcf5d, 0xdf7c, 0xaf9b, 0xbfba, 0x8fd9, 0x9ff8,
0x6e17, 0x7e36, 0x4e55, 0x5e74, 0x2e93, 0x3eb2, 0x0ed1, 0x1ef0 };


//int16 数据位整理
void buffer_append_int16(uint8_t* buffer, int16_t number, int32_t *index) {
    buffer[(*index)++] = number >> 8;
    buffer[(*index)++] = number;
}

//uint16 数据位整理
void buffer_append_uint16(uint8_t* buffer, uint16_t number, int32_t *index) {
    buffer[(*index)++] = number >> 8;
    buffer[(*index)++] = number;
}

//int32 数据位整理
void buffer_append_int32(uint8_t* buffer, int32_t number, int32_t *index) {
    buffer[(*index)++] = number >> 24;
    buffer[(*index)++] = number >> 16;
    buffer[(*index)++] = number >> 8;
    buffer[(*index)++] = number;
}

//uint32 数据位整理
```


E-mail:info@cubemars.com
https://www.cubemars.com/

```
void buffer_append_uint32(uint8_t* buffer, uint32_t number, int32_t *index) {
    buffer[(*index)++] = number >> 24;
    buffer[(*index)++] = number >> 16;
    buffer[(*index)++] = number >> 8;
    buffer[(*index)++] = number;
}

//int64 数据位整理
void buffer_append_int64(uint8_t* buffer, int64_t number, int32_t *index) {
    buffer[(*index)++] = number >> 56;
    buffer[(*index)++] = number >> 48;
    buffer[(*index)++] = number >> 40;
    buffer[(*index)++] = number >> 32;
    buffer[(*index)++] = number >> 24;
    buffer[(*index)++] = number >> 16;
    buffer[(*index)++] = number >> 8;
    buffer[(*index)++] = number;
}

//uint64 数据位整理
void buffer_append_uint64(uint8_t* buffer, uint64_t number, int32_t *index) {
    buffer[(*index)++] = number >> 56;
    buffer[(*index)++] = number >> 48;
    buffer[(*index)++] = number >> 40;
    buffer[(*index)++] = number >> 32;
    buffer[(*index)++] = number >> 24;
    buffer[(*index)++] = number >> 16;
    buffer[(*index)++] = number >> 8;
    buffer[(*index)++] = number;
}

//CRC 校验
unsigned short crc16(unsigned char *buf, unsigned int len) {
    unsigned int i;
    unsigned short cksum = 0;
    for (i = 0; i < len; i++) {
        cksum = crc16_tab[(((cksum >> 8) ^ *buf++) & 0xFF)] ^ (cksum << 8);
    }
    return cksum;
}

 //数据包的整理发送
void packet_send_packet(unsigned char *data, unsigned int len, int handler_num) {
```


E-mail:info@cubemars.com
https://www.cubemars.com/

```
    int b_ind = 0;
    unsigned short crc;
    if (len > PACKET_MAX_PL_LEN) {
        return;
    }
    if (len <= 256) {
        handler_states[handler_num].tx_buffer[b_ind++] = 2;
        handler_states[handler_num].tx_buffer[b_ind++] = len;
    } else {
        handler_states[handler_num].tx_buffer[b_ind++] = 3;
        handler_states[handler_num].tx_buffer[b_ind++] = len >> 8;
        handler_states[handler_num].tx_buffer[b_ind++] = len & 0xFF;
    }

    memcpy(handler_states[handler_num].tx_buffer + b_ind, data, len);
    b_ind += len;

    crc = crc16(data, len);
    handler_states[handler_num].tx_buffer[b_ind++] = (uint8_t)(crc >> 8);
    handler_states[handler_num].tx_buffer[b_ind++] = (uint8_t)(crc & 0xFF);
    handler_states[handler_num].tx_buffer[b_ind++] = 3;

    if (handler_states[handler_num].send_func) {
        handler_states[handler_num].send_func(handler_states[handler_num].tx_buffer,
b_ind);
    }
}
```


E-mail:info@cubemars.com
https://www.cubemars.com/

4.4 控制命令实例

4.4.1 CAN 口控制命令实例

帧格式：拓展帧-数据帧（以电机 ID 为 0x68 示例）

模式	ID	DATA	说明
占空比模式	00 00 00 68	00 00 4E 20	0.2 占空比
	00 00 00 68	FF FF B1 E0	-0.2 占空比
电流环	00 00 01 68	FF FF F0 60	-4A IQ 电流
	00 00 01 68	00 00 0F A0	4A IQ 电流
刹车电流模式	00 00 02 68	FF FF F0 60	-4A 刹车电流
	00 00 02 68	00 00 0F A0	4A 刹车电流
速度环	00 00 03 68	00 00 13 88	5000 ERPM 电气速度
	00 00 03 68	FF FF EC 78	-5000 ERPM 电气速度
位置环	00 00 04 68	00 5B 8D 80	电机转动到 600 度
	00 00 04 68	FF A4 72 80	电机转动到-600 度
位置速度环	00 00 06 68	00 98 96 80 03 E8 03 E8	电机转动到 1000 度 10000 ERPM 电气速度 10000 电气加速度
	00 00 06 68	FF 67 69 80 FC 18 FC 18	电机转动到-1000 度 -10000 ERPM 电气速度 -10000 电气加速度
MIT 速度环	00 00 08 68	00 06 66 7F FF 8F 57 FF	Kd 设置为 2 速度设置为 6rad/s
	00 00 08 68	00 06 66 7F FF 70 97 FF	Kd 设置为 2 速度设置为-6rad/s
MIT 位置环	00 00 08 68	01 06 66 BD 70 7F F7 FF	Kp 设置为 2 Kd 设置为 2 电机转动到 6rad
	00 00 08 68	01 06 66 42 8F 7F F7 FF	Kp 设置为 2 Kd 设置为 2 电机转动到-6rad
MIT 力矩环	00 00 08 68	00 00 00 7F FF 7F F8 3F	2A IQ 电流
	00 00 08 68	00 00 00 7F FF 7F F8 7E	4A IQ 电流

E-mail:info@cubemars.com
https://www.cubemars.com/

4.4.2 串口控制命令实例

模式	串口指令	说明
占空比模式	AA 05 46 00 00 4E 20 D6 4C BB	0.20 占空比
	AA 05 46 FF FF B1 E0 88 3F BB	-0.20 占空比
刹车电流模式	AA 05 48 00 00 13 88 55 E5 BB	5A 刹车电流
	AA 05 48 FF FF EC 78 3D C5 BB	- 5A 刹车电流
速度环	AA 05 49 00 00 03 E8 90 61 BB	1000 ERPM 电气转速
	AA 05 49 FF FF FC 18 F8 41 BB	- 1000 ERPM 电气转速
位置环	AA 05 4A 0A BA 95 00 E1 4D BB	电机转动到 180 度
	AA 05 4A 05 5D 4A 80 84 93 BB	电机转动到 90 度
位置速度环	AA 0D 3C 00 02 BF 20 00 00 13 88 00 00 75 30 18 1C BB	180 度 转速 5000ERPM 加速度 30000/S
电流环	AA 05 47 00 00 13 88 30 1C BB	5 A IQ 电流
	AA 05 47 FF FF EC 78 58 3C BB	- 5 A IQ 电流
MIT 速度环	AA 15 60 00 00 00 00 00 00 17 70 00 00 00 00 00 00 00 00 00 00 00 07 D0 93 DA BB	Kd 设置为 2 速度设置为 6rad/s
	AA 15 60 00 00 00 00 FF FF E8 90 00 00 00 00 00 00 00 00 00 00 00 07 D0 87 5C BB	Kd 设置为 2 速度设置为-6rad/s
MIT 位置环	AA 15 60 00 00 17 70 00 00 00 64 00 00 00 00 00 00 07 D0 00 00 07 D0 91 BC BB	Kp 设置为 2 Kd 设置为 2 电机转动到 6rad
	AA 15 60 FF FF E8 90 00 00 00 64 00 00 00 00 00 00 07 D0 00 00 07 D0 C9 10 BB	Kp 设置为 2 Kd 设置为 2 电机转动到-6rad
MIT 力矩环	AA 15 60 00 00 00 00 00 00 00 64 00 00 07 D0 00 00 00 00 00 00 00 00 CB B7 BB	2A IQ 电流
	AA 15 60 00 00 00 00 00 00 00 64 00 00 0F A0 00 00 00 00 00 00 00 00 2A 27 BB	4A IQ 电流

E-mail:info@cubemars.com
https://www.cubemars.com/