

# 小米 CyberGear 关节通讯协议整理

## 目录

小米 CyberGear 关节通讯协议整理.....	1
前言.....	2
总述.....	3
关节 ID 修改.....	4
关节零点修改.....	4
关节运控模式.....	4
关节位控模式（需指定速度）.....	4
关节速度模式.....	5
关节使能.....	5
关节停止.....	5
硬件准备：.....	6
24V 电源、导线、接线器若干，.....	6
Cybergear 关节.....	6
关节引脚定义：.....	6
硬件接线如图.....	8
上位机调试及参数设定.....	9
介绍.....	9
打开串口及参数设置.....	10
进入数据模式发送命令.....	11
关节通信协议解析及示例.....	11
指令包.....	12
应答包.....	13
指令类型.....	14
示例 1：ID 修改.....	14
示例 2：零点修改：.....	15
示例 3 关节使能：.....	16
示例 4 关节停止：.....	16
示例 5 速度模式：.....	16
示例 6 位置模式：.....	18
示例 7：运控模式：.....	19

部分资料搜集自互联网

本文仅供参考

禁止用于商业用途，违者后果自负

2023.9.4



# 正式开售

CyberGear 微电机



¥499

8月14日晚10点 小米商城开售

CyberGear 关节

## 前言

小米刚刚发布了最新款的机器人关节，额定负载 4Nm，峰值扭矩 12Nm，最大转速 30rad/s，直径 71mm，重量 317g，减

速比 7.75: 1, 价格 499, 性价比可以说是非常之高, 只有市面上同性能产品一半甚至三分之一的价格, 笔者买回来研究了一下, 性能确实不错, 但是官方的手册通信协议部分写的并不是很详细, 很多地方一笔带过, 而且对应的示例也比较少, 用官方指定 USB-CAN 硬件+上位机比较好控制, 但是用自己的设备 (比如 USB-CAN/STM32 等等) 就比较难控制, 笔者研究了一下协议内容, 在此分享给大家, 让大家不需要上位机或者官方提供的库也可以控制电机。

PS: 本说明只是对于官方说明书的补充, 在看完本说明后可以更好的理解官方手册并且可以用自己的 USB-CAN 模块或其他硬件去控制关节, 本说明没有提到的参数功能可以在官方书中查看。

## 总述

关节的 MCU 采用来自易兆创新的 GD32F303RET6ARM 基于 ARM Cortex-M4 内核, 价格大概 30 元一枚, 驱动芯片使用的是英飞凌的半桥驱动 6EDL714, 价格大概 60 元一枚, 位置传感采用的是来自 ams 的 AS5047P, 价格 25 元一枚, 通信采用 CAN 协议

关节具有以下功能:

## 关节 ID 修改

此命令可以修改关节 ID

- 在一个 CAN 总线中，可以连接多个关节，每个关节都有一个独特的 ID 号码。控制器发出的指令包含 ID 信息，只有与指令中 ID 号匹配的关节才能完整地接收并回应该指令。

## 关节零点修改

此命令可以修改关节的零点位置（下电丢失）

## 关节运控模式

此命令可以设置关节的扭矩、速度、位置、Kp、Kd 五个数值  
控制关节运动

- Kp、Kd 为比例增益和微分增益；

## 关节位控模式（需指定速度）

关节的角度控制范围：正负任意圈数

控制精度：<0.01 度（理论控制精度，非实际精度）

关节的速度控制范围为：-30-30rad/s。

备注：

要将角速度从 "rad/s" 转换为 "度/s", 可以使用以下公式:

$$\text{度/秒} = \frac{\text{弧度}}{\pi} \times \frac{180}{\text{秒}}$$

其中, 弧度 表示以 "rad/s" 为单位的角速度, 度/秒 表示以 "度/s" 为单位的角速度。

所以, 要将角速度从 "rad/s" 转换为 "度/s", 只需将角速度值乘以  $\frac{180}{\pi}$ 。这是因为  $1 \text{ rad} = \frac{180}{\pi} \text{ degrees}$ 。

例如, 如果角速度为 5 rad/s, 则转换为度/秒的值为:

$$\text{度/秒} = 5 \times \frac{180}{\pi} \approx 286.48 \text{ 度/秒}$$

所以, 5 rad/s 约等于 286.48 度/秒。

## 关节速度模式

让关节以恒定速度转动 ( $-30-30\text{rad/s}$ )

## 关节使能

当一个关节被使能时, 它被允许执行运动或控制任务。

这意味着电机可以驱动关节进行运动, 凡是运动前必须先使能。

## 关节停止

使关节停止运动

## 硬件准备：

24V 电源、导线、接线器若干，

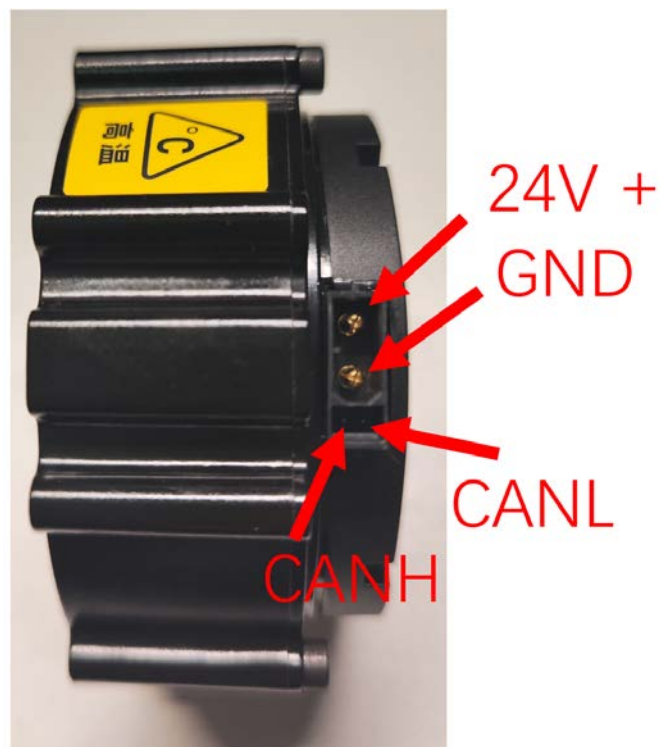
XT30PB(2+2)-F： 购买链接：

<https://buyertrade.taobao.com/trade/detail/tradeSnap.htm?spm=a1z09.2.0.0.6f612e8dh6PYTb&tradeID=1961727027068168298&snapShot=true>

注意要买-F（母头），不要买错了

Cybergear 关节一个（推荐官网/APP 原价 499 购买）

关节引脚定义：（附图



注意好接口方向，按图接线，CANH 对应 USB 转 CAN 模块的 CANH，CANL 对应 USB 转 CAN 模块的 CANL

- 因为需要和关节进行 CAN 通信，所以需要准备相应硬件（单片机，USB 转 CAN 模块），本文的例子都是由电脑端控制，通过 USB 转 CAN 模块对关节进行控制，CAN 模块需要购买，视频同款模块淘宝链接：

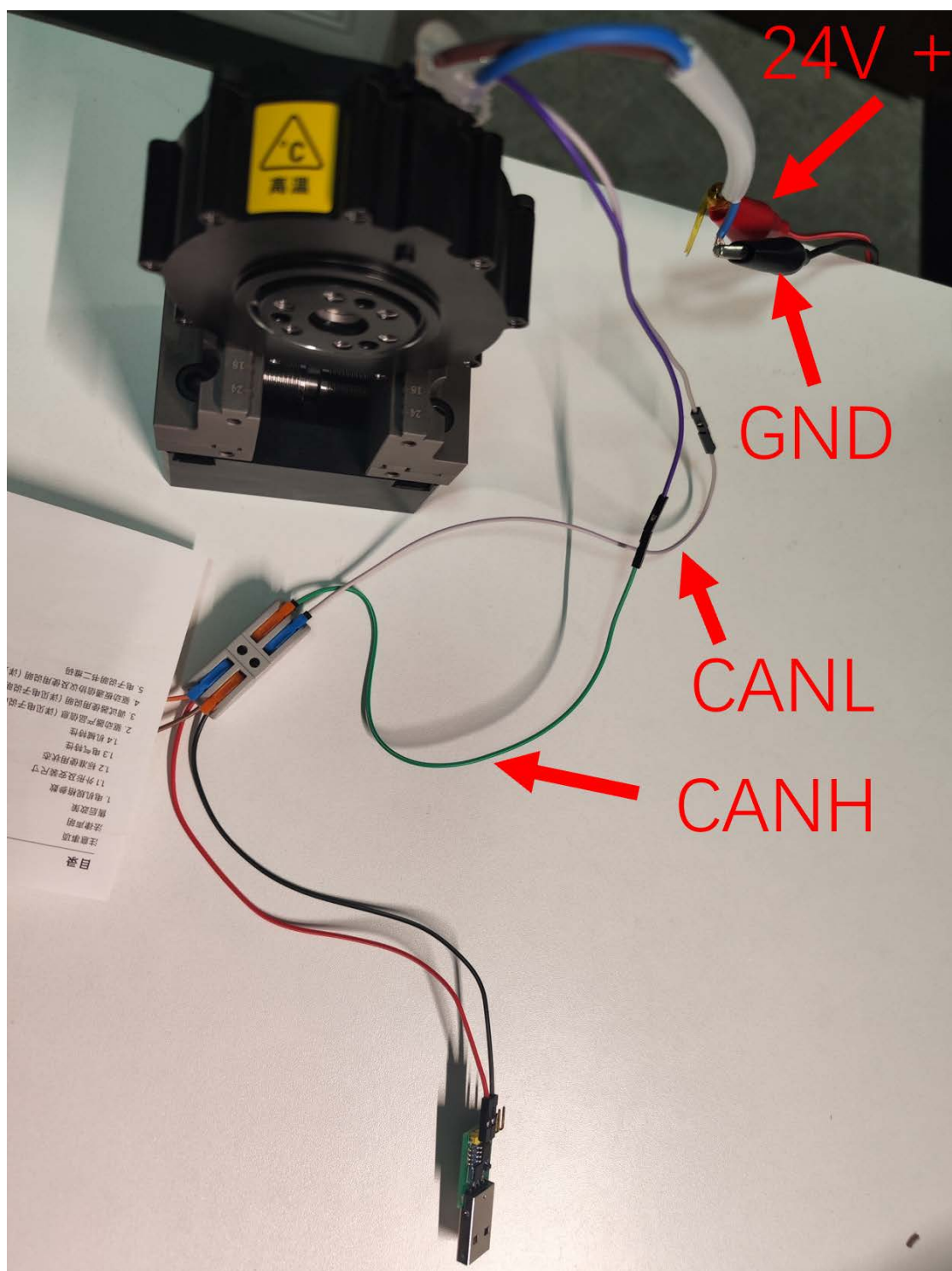
【淘宝】

[USB 转 CAN modbus CANOpen 工业级转换器 CAN 分析仪 串口转 CAN TTL-淘宝网](#)  
CZ0001 「USB 转 CAN modbus CANOpen 工业级转换器 CAN 分析仪 串口转 CAN TTL」

点击链接直接打开 或者 淘宝搜索直接打开

，只要可以进行 CAN 通信，买哪款都可以，但是如果没有使用官方推荐的模块是无法使用官方的上位机软件的，只能用串口软件进行通信（本文使用的是非官方推荐的模块，进行串口通信演示）

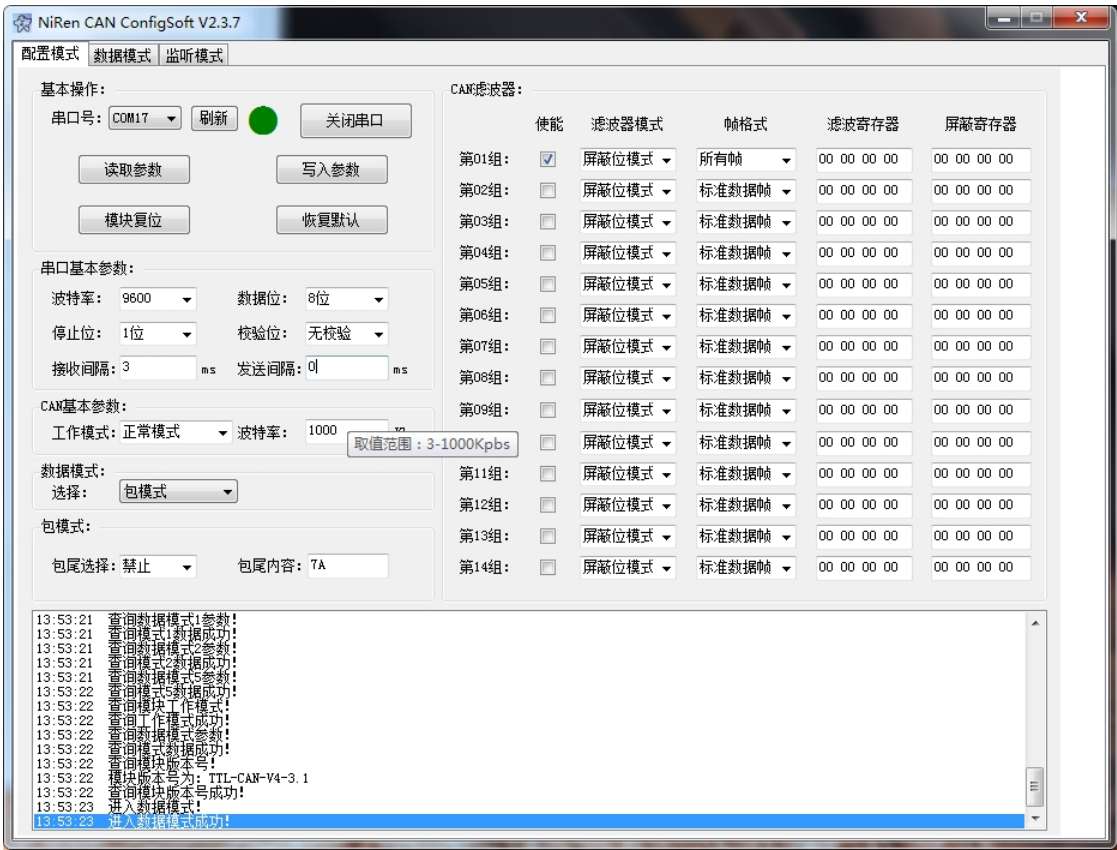
硬件接线如图





# 上位机调试及参数设定


本文所使用的 USB 转 CAN 模块的上位机具有配置模式、数据模式，接下来以这个模块为例进行 CAN 总线参数的设置



选择正确串口，打开串口，读取模块目前参数，按下图进行参数设置

1. 选择正确的串口

基本操作：

串口号：COM17 刷新  关闭串口

读取参数 写入参数

模块复位 恢复默认

串口基本参数：

波特率：9600 数据位：8位

停止位：1位 校验位：无校验

接收间隔：3 ms 发送间隔：0 ms

CAN基本参数：

工作模式：正常模式 波特率：1000 取值范围：3-1000000

数据模式：

选择：包模式

包模式：

包尾选择：禁止 包尾内容：7A

2. 打开串口

5. 写入参数

3. 读取参数

4. 参数设置

## 进入数据模式发送命令



（具体命令环节在下面，图中的命令忽略）

## 关节通信协议解析及示例

首先希望读者明白 USB-CAN 通信的基本结构：



本文主要的内容是介绍 USB-CAN 模块与 CAN 之间的 Cybergear 关节的 CAN 协议，至于模块与电脑之间的串口 (COM)

协议，对于不同模块，协议是不一样的，需要你自己去看模块的使用手册。

在讲解命令之前，对于零基础的人需要注意的是，我使用的是泥人 USB-CAN 模块，其含义如下：



对于不同的模块，需要不同的命令，具体的需要参照你的模块手册（AA 01 00 07 不同），扩展帧部分也不一定相同（00 00 05 AB 不一定相同），数据部分是相同的（11 22 33 44 55 66 77 00），请忽略 7A。（具体命令环节在下面，图中的命令忽略）

指令包

我们位置控制的指令为例：

AA 01 00 08 12 00 00 01 16 70 00 00 00 00 80 3F

字头	扩展帧信息（四位）	功能参数	参数
0xAA 0x01 0x00 0x08	0x12 0x00 0x00 0x01	0x16 0x70	0x00 0x00 0x80 0x3F

AA 01 00 字头中红色的部分是泥人模块的协议；

08 橙色部分是有效数据长度（一般默认 8 就好，在这里指的是 16 70 00 00 00 00 80 3F）

12 浅绿色部分是功能码；

01 蓝色部分是 ID；

16 70 深绿色部分是功能参数

注意，上文带有 0x 字样的意为十六进制数字，解释见下面小字部分；蓝色的代表舵机的 ID，紫色部分代表数据内容

PS:

- 在通讯协议中，“0x”是一个常用的前缀，表示接下来的数字是十六进制（base 16）的。例如，“0xFF”表示一个十六进制的数，其中“F”在十六进制中对应于十进制的 15，所以“FF”在十六进制中表示的数值就是  $(15 \times 16) + 15 = 255$ 。类似的，“0x10”在十六进制中表示的数值就是  $(1 \times 16) + 0 = 16$ 。使用十六进制的一个主要原因是它可以更方便地表示二进制数，因为每一个十六进制的位都可以精确地对应四个二进制的位。例如，二进制的“1111”就等于十六进制的“F”。在串口通信中，数据通常是以二进制形式传输的，所以经常用十六进制来表示和操作这些数据。例如，你可能会看到类似“0xFF”或“0x00”这样的表达，分别对应于二进制的“11111111”和“00000000”。
- 二进制（binary）是一种计数系统，只使用两个数字 0 和 1 来表示数值。在计算机系统中，二进制被广泛应用，因为计算机处理器只能理解二进制数据。十进制（decimal）是我们平时使用的计数系统，使用 10 个数字 0~9 来表示数值。十六进制（hexadecimal）也是一种计数系统，使用 16 个数字 09 和字母 AF（分别表示 10~15）来表示数值。在计算机系统中，十六进制被广泛应用，因为它可以方便地表示二进制数，而且十六进制数的位数比二进制数少得多，便于人们阅读和输入。在计算机编程中，常常使用二进制、十进制和十六进制来表示数据和内存地址等信息。

## 应答包

我们位置控制指令的应答包为例：

AA 01 00 08 02 00 01 00 7F FE 7F EA 7F FF 01 39

字头	扩展帧信息（四位）	参数
0xAA 0x01 0x00 0x08	0x02 0x00 0x01 0x00	0x7F 0xFE 0x7F 0xFF 0x01 0x39

不同的指令包对应的应答包的功能参数不同

# 指令类型

功能指令	功能	功能码	功能参数	参数长度 (字节)
ID 修改	修改关节 ID	0x07	无	0
零点修改	修改关节零点	0x06	无	1
运控模式	控制关节的五个参数 (前面有讲)	0x01	无	8
位控模式	控制关节到指定位置	0x12 (写入参数功能码)	详见官方说明书	8+2
速度模式	使关节可以匀速转动	0x12 (写入参数功能码)	0X7016	5
关节使能	使关节能够运动	0x03	无	0
关节停止	停止运动	0x04	无	0

看不懂不要紧，看下面示例：

## 示例 1：ID 修改

- 将关节的 ID 号从 1 改成 2，关节在上电初期会向总线发送命令，可以在这个命令中知道关节的 ID 号  
(下文都以 ID 为 1 举例，所以记得改回来)

发送：AA 01 00 08 07 02 00 01 00 00 00 00 00 00 00 00

接收：AA 01 00 08 00 00 02 FE 58 10 31 31 30 33 31 0A

解析：  
AA 01 00 为字头；  
08 有效字节长度  
07 对应功能码  
02 01 表示 ID；

●

## 示例 2：零点修改：

修改 ID 为 1 的关节以现有位置作为零点（下电后会丢失）：

发送：AA 01 00 08 06 00 00 01 01 00 00 00 00 00 00 00

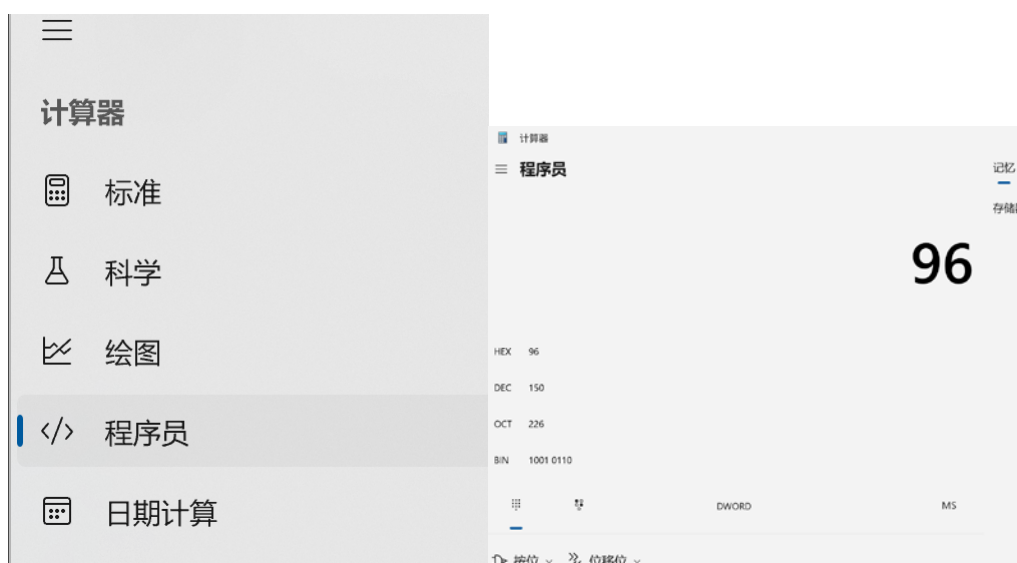
接收：AA 01 00 08 02 00 01 00 7F FE 80 C2 7F FF 01 1E

第一个 7F FE 代表关节位置，0x7FFE 转化为 10 进制可表示为 32766，当前角度 $[0^{\circ} \sim 65535^{\circ}]$ 对应 $(-4\pi \sim 4\pi)$ ，所以此时关节的位置正好为中点，即 0 度。

第二个 7F FE 代表关节力矩限制，当前力矩 $(-32768 \sim 32767)$ 对应 $(-12\text{Nm} \sim 12\text{Nm})$

01 1E 对应十进制数字 286，除以 10 即是当前电机温度（28.6 摄氏度）

- Windows 计算器可以进行进制换算，Hex 为 16 进制、DEC 为 10 进制



### 示例 3 关节使能：

#### ■ 使能：

发送：AA 01 00 08 03 00 00 01 00 00 00 00 00 00 00 00

接收：AA 01 00 08 02 80 01 00 7F FF 80 14 7F FF 01 4D

### 示例 4 关节停止：

#### ■ 使能：

发送：AA 01 00 08 04 00 00 01 00 00 00 00 00 00 00 00

接收：AA 01 00 08 02 80 01 00 7F FF 80 14 7F FF 01 4D

### 示例 5 速度模式：

要进行速度控制需要以下步骤：

进入速度模式——关节使能——按特定速度驱动——停止

#### ■ 进入速度模式：

发送：AA 01 00 08 12 00 00 01 05 70 00 00 02 00 00 00

接收：AA 01 00 08 02 80 01 00 7F FF 7F C0 80 2A 01 4D

12 浅绿色部分是功能码；

01 蓝色部分是 ID；

05 70 深绿色部分是功能参数

02 代表速度模式（01 代表位置模式 00 代表运控模式 04 代表电流模式，本书主要举例位置以及速度模式，其他模式详见官方说明书）



## ■ 关节使能

见例 3

## ■ 按特定速度驱动关节 (1.5rad/s)

发送: AA 01 00 08 12 00 00 01 0A 70 00 00 00 00 C0 3F

接收: AA 01 00 08 02 80 01 00 FF FF 86 5D 7F FF 01 46

0A 70 深绿色部分是功能参数

这里着重讲解下速度的设置, 这里的速度设置采用的是 IEEE 754 浮点数, 具体原理有兴趣可以上网了解, 这个网址可以进行在线换算:

<http://www.speedfly.cn/tools/hexconvert/>



IEEE 754浮点数十六进制相互转换

32位 四字节 单精度

10进制	1.5
此处填写你想要的数值 填完后点击下方16进制按钮即可转换	
16进制	3F C0 00 00
此处为对应上方数值转换后的结果	
注意每个字节中间会有空格 填入到代码中的时候要去除这些空格	

[返回七支剑的WP](#)

比如我们像设置速度为 1.5rad/s, 我们需要在上图 10 进制框内输入 1.5, 然后点击 16 进制按钮, 网页会将 1.5 按照 IEEE 754 格式转化为 16 进制数 0x3FC00000, 此通讯协议中角度转化计算需要移位(因为其低字节在前, 高字节在后), 即 0x3FC00000 变为 0x0000C03F (倒过来), 放在最后四个字节。 同理在官方说

说明书中我们可以看到按特定速度驱动关节的参数为 0x700A，但是在命令发送时我们发送 0x0A 0x70

如果不太明白，请看下面内容

Ps:

在串口通讯中，16 进制数 0x3FC00000 在命令中需要调换顺序，变成 0x0000C03F，是因为串口通讯传输的数据是按照字节(byte)为单位传输的。在大端字节序中，数据的高字节在前、低字节在后；而在小端字节序中，数据的低字节在前、高字节在后。通常，CPU 的内部数据存储方式都采用一种字节序，而通讯协议中定义的数据字节序可能与 CPU 内部的字节序不同。因此，为了避免字节序的不一致，通讯协议中规定了数据的字节序。在该通讯协议中，发送命令中的数据采用小端字节序，即低字节在前、高字节在后。因此，16 进制数在发送的命令中需要调换顺序，这样接收方才能正确地解析出数据。

## ■ 关节停止

见示例 4

执行命令后可以看到电机以 1.5 rad/s 的速度运动

## 示例 6 位置模式：

要进行位置控制需要以下步骤：

进入位置模式——关节使能——设定运动速度——设定位置并开始运动

## ■ 进入位置模式：

发送：AA 01 00 08 12 00 00 01 05 70 00 00 01 00 00 00

接收：AA 01 00 08 02 80 01 00 FF FF 7F EE 7F FF 01 4D

01 代表位置模式（02 代表速度模式 00 代表运控模式 04 代表电流模式，本书主要举例位置以及速度模式，其他模式详见官方说明书）

## ■ 关节使能

见例 3

#### ■ 设定关节运动速度(10rad/s)

发送: AA 01 00 08 12 00 00 01 17 70 00 00 00 00 20 41

接收: AA 01 00 08 02 00 01 00 FF FF 7F CE 7F FF 01 39

17 70 深绿色部分是功能参数

00 00 20 41 为 (10 rad/s) 通过 IEEE 754 标准转化后再移位得到的结果

#### ■ 设定关节运动位置(10rad/s)

发送: AA 01 00 08 12 00 00 01 16 70 00 00 C3 F5 48 40

接收: AA 01 00 08 02 80 01 00 FF FF 7F F9 7F FF 01 39

16 70 深绿色部分是功能参数

C3 F5 48 40 为 (3.14 rad) 通过 IEEE 754 标准转化后再移位得到的结果

执行命令后可以看到电机会运动 180 度

### 示例 7: 运控模式:

此模式下可以设置关节的扭矩、速度、位置、Kp、Kd 五个数值控制关节运动，**此命令危险！新手不建议使用此功能**，除非你明白所有参数的含义。

这个模式在官方的说明书讲解的很详细，在此引用下

4.1.2 运控模式电机控制指令（通信类型1）用来向电机发送控制指令

数据域	29位ID			8Byte数据区
大小	Bit28~bit24	bit23~8	bit7~0	Byte0~Byte7
描述	1	Byte2:力矩 (0~65535)  对应 (-12Nm~12Nm)	目标电机 CAN_ID	Byte0~1: 目标角度[0~65535]对应 (-4 $\pi$ ~4 $\pi$ )  Byte2~3: 目标角速度[0~65535]对应 (-30rad/s~30rad/s)  Byte4~5: Kp [0~65535]对应(0.0~500.0)  Byte6~7: Kd [0~65535]对应(0.0~5.0)

如果你知道更多功能欢迎私信我补充更新