



DR EMPOWER

# DrEmpower 电机库函数使用说明

**DrEmpower**

DR TECH

大然机器人  
Dr Robot

## 目录

第一章 函数库安装 .....	1
1.1 添加 Python 函数库 .....	1
1.2 添加 STM32 函数 .....	1
1.3 添加 Arduino 函数库 .....	10
1.4 添加 ROS 函数库 .....	13
第二章 运动控制功能 .....	13
2.1 单个电机位置控制 .....	13
2.2 多个电机位置控制 .....	14
2.3 单个电机相对角度控制 .....	15
2.4 多个电机相对角度同步控制 .....	16
2.5 单个电机速度控制 .....	18
2.6 多个电机速度控制 .....	19
2.7 单个电机电矩控制 .....	20
2.8 多个电机电矩控制 .....	21
2.9 单个电机阻抗控制 .....	22
2.10 急停 .....	23
第三章 参数设置功能 .....	24
3.1 设置 ID 号 .....	24
3.2 设置 uart 串口波特率(M) .....	24
3.3 设置 can 波特率 .....	25
3.4 设置模式 .....	26
3.5 设置零点 .....	27
3.6 设置 GPIO 控制接口模式(M) .....	27
3.7 设置电机软件限位极限位置 .....	28
3.8 设置梯形轨迹模式下速度曲线类型 .....	29
3.9 设置指定参数 .....	30
第四章 参数回读功能 .....	32
4.1 获取 ID 号 .....	32
4.2 读取位置和速度 .....	32
4.3 读取电压和电流 .....	33
4.4 读取 GPIO 控制接口模式(M) .....	34
4.5 读取指定参数 .....	34
第五章 辅助功能 .....	36
5.1 清除错误标志 .....	36
5.2 显示错误标志 .....	36
5.3 保存电机配置 .....	37
5.4 电机重启 .....	38
5.5 单个电机等待函数(等待单个电机运动到目标角度) .....	38
5.5 多个电机等待函数(等待多个电机运动都到目标角度) .....	38
附录：常用参数及其地址表 .....	40

## 第一章 函数库安装

为方便用户使用，公司特开发基于 python 语言和 C/C++语言（STM32 平台及 Arduino 平台）的电机控制库。当使用库函数控制电机时，需要事先安装对应平台的电机函数库，具体安装方法如下。

### 1.1 添加 Python 函数库

Python 函数库分为 ascii、can 和 uart 三种。使用 ascii 库函数时，需要使用一条 mini USB 线缆通过 USB 接口与电机相连（仅 M 系列支持）；使用 can 库函数时，需要使用专用的 USB 转 CAN 模块通过 CAN 接口与电机相连。使用 uart 库函数时，需要使用 USB 转 TTL 模块通过 UART 接口与电机相连（仅 G 系列支持）。

在接好电机并正确供电后，将 python 库函数对应 python 文件夹添加至 pycharm，(PyCharm 配置详见《windows 下通过 pycharm 调用 Python 函数库指南》) 然后打开 DrEmpower 开头文件查看函数，在 python 控制台 import 该文件名 as 别名，再使用别名.函数名( )，就能运行库函数中代码。

如图，以 ACSII 库函数运行例程为例：



图 1-1 ACSII 库函数截图（部分）

（图 1-1 中，在函数 serial.Serial（）中，'COM26'位置应输入实际通信端口编号，115200 位置输入实际波特率。如需使用其它设备，则需取消注释对应设备的代码）

```

import DrEmpower_ascii as drep
drep.set_angle(1, 20, 30, 10, 1)
  
```

### 1.2 添加 STM32 函数

#### (1) STM32Cude 的基本配置

STM32 函数库基于 HAL 库，使用串口操纵电机，适用于使用了 HAL 库的 STM32 工程。

如需新建 HAL 库工程，使用 STM32 电机库函数，需打开 STM32CubeMX，新建项目，选择完对应的芯片型号之后，进入配置界面。

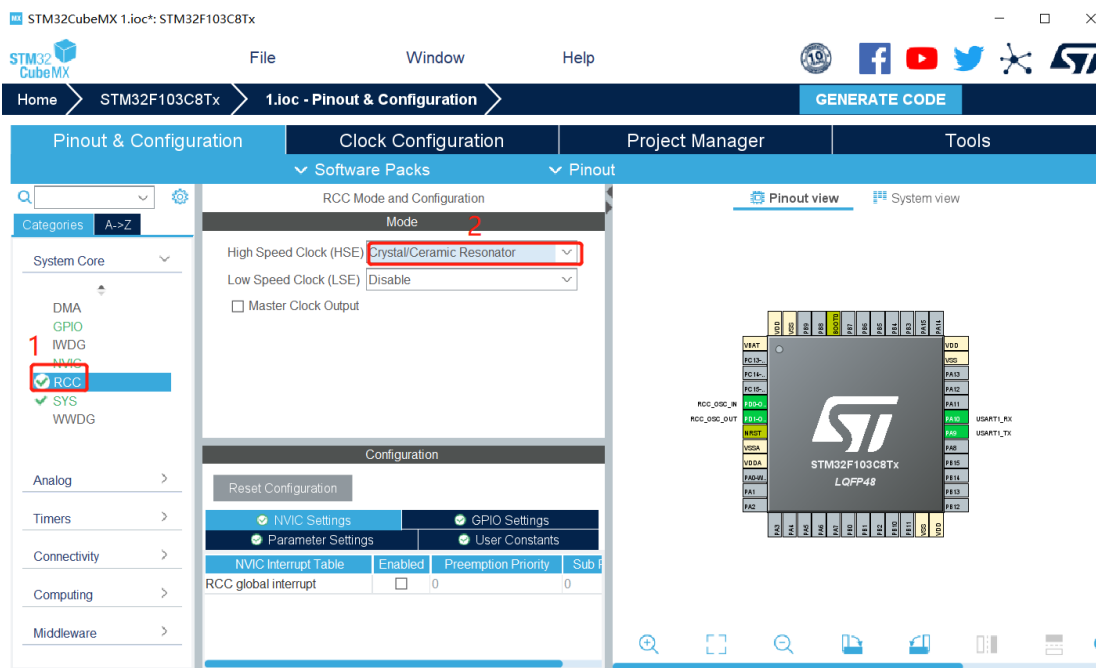


图 1-2 STM32CubeMX 界面 RCC 配置

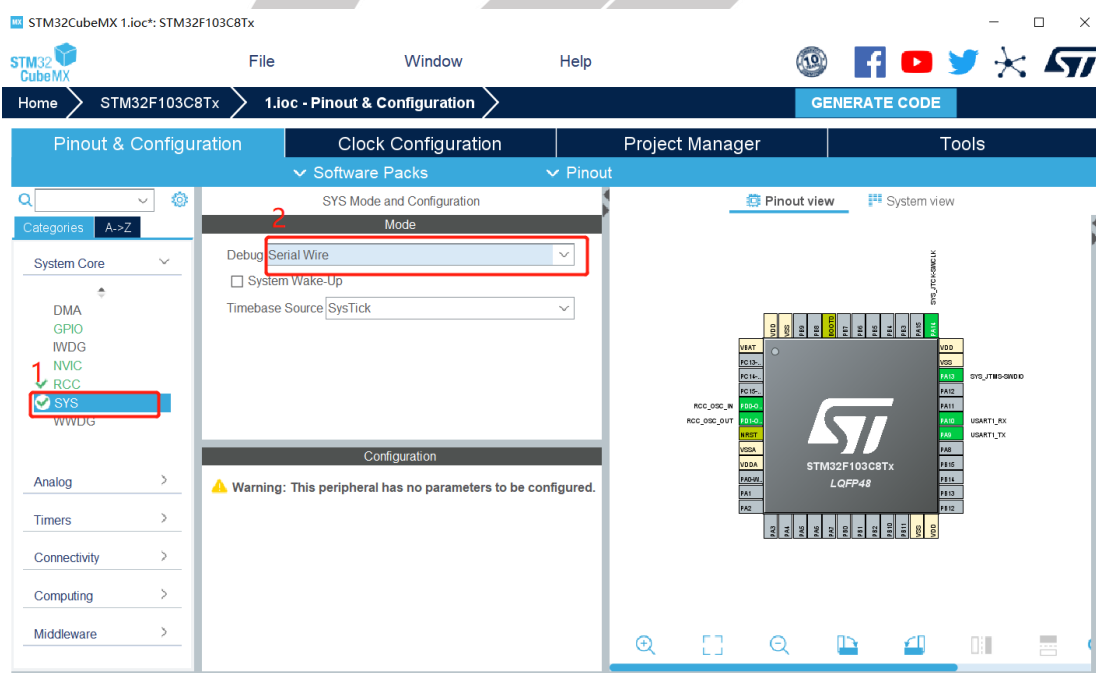


图 1-3 STM32CubeMX 界面 SYS 配置

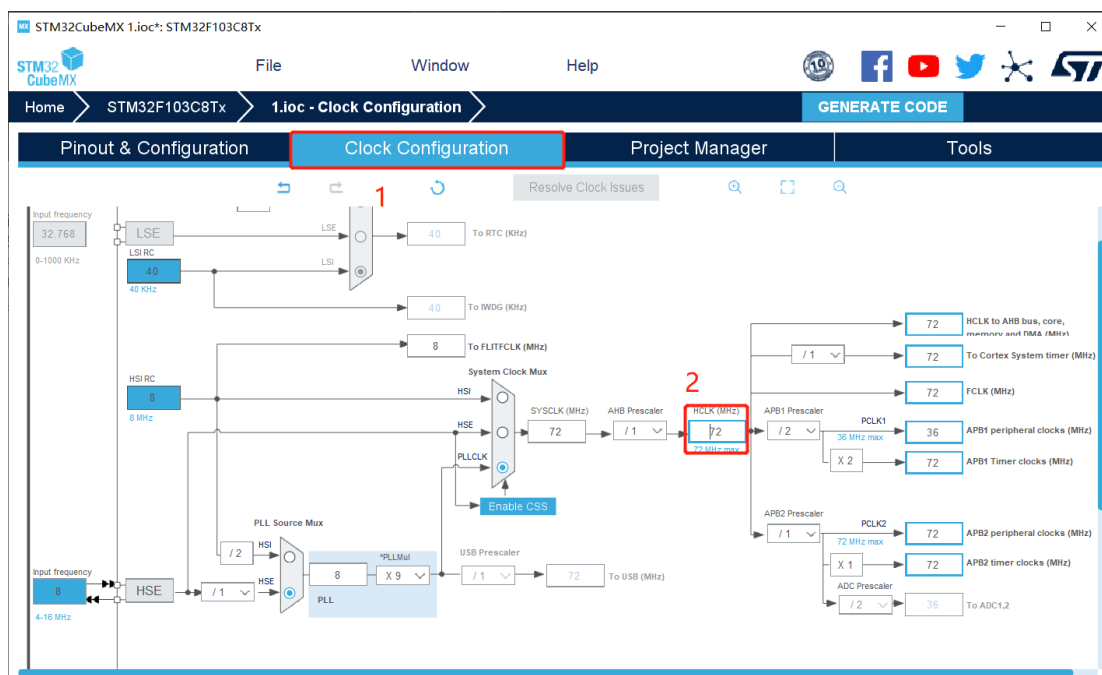


图 1-4 STM32CubeMX 界面 时钟配置

首先如图 1-2 所示，先在 Pinout&Configuration 下找到 RCC 点击进去，然后选择 Crystal/Ceramic Resonator；在如图 1-3 所示，点击 SYS，在里面选择 Serial Wire，做完这两步操作后，点击 Clock Configuration，调配时钟，如图 1-4 所示，将标记点 2 改为 72M，然后回车键确认。

## (2) 使用 UART 接口（ascii 协议）进行控制（M）

STM32 函数库基于 HAL 库，使用串口操纵电机，适用于使用了 HAL 库的 STM32 工程。

如需新建 HAL 库工程，使用 STM32 电机库函数，需打开 STM32CubeMX，新建项目，选择完对应的芯片型号之后，进入配置界面。

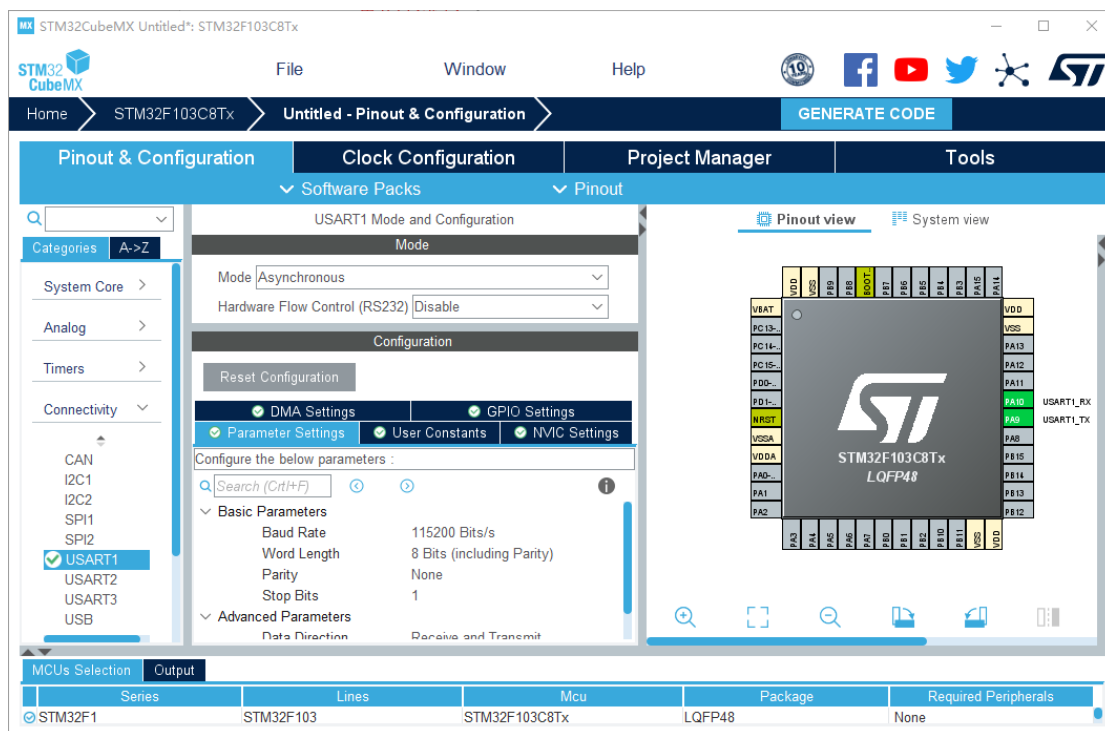


图 1-5 STM32CubeMX 界面 USART 配置

在左侧列表选中任意一个串口（或在右侧引脚图中直接选择串口引脚），将其模式配置为 **Asynchronous**，再将波特率设置为与电机串口波特率对应的数值。

设置完串口之后，再进行时钟、选择生成目标 IDE(MDK-ARM、STM32CubeIDE 等)工程及其它配置，再在左侧 Code Generator 中勾选 **Generate peripheral initialization as a pair of '.c/.h' files per peripheral**，最后点击右上角的 **GENERATE CODE** 生成代码，就完成了项目的创建。

如已有工程是使用 HAL 库的，也可以点开 STM32CubeMX 的工程文件，新配置一个操作电机的串口，如不使用 STM32CubeMX，可以手动配置串口。

配置完成串口之后，项目代码中会使用串口句柄 `UART_HandleTypeDef` 进行对串口的一系列操作。将 `DrEmpower_uart.c` 和 `DrEmpower_uart.h` 文件添加至项目中，修改 `DrEmpower_uart.c` 中的宏定义为自己配置的 `UART_HandleTypeDef`（如 `huart1`）。

最后由于 STM32 函数库使用了 `sprintf` 函数输出浮点数，所以也要在 IDE 中进行一些配置才可以让 STM32 的 `sprintf` 函数正确输出浮点数：

**Keil 项目：**在工程属性的 **Target → Code Generation** 中勾选 **Use MicroLIB** 选项。

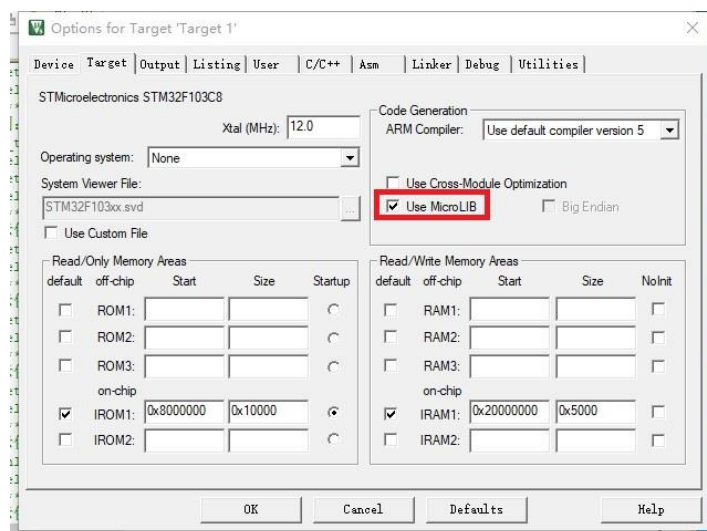


图 1-6 Keil Use MicroLIB

STM32CubeIDE 项目：点击菜单栏中的 Project → Properties，打开 Properties 窗口，在 MCU Settings 中选中 Use float with printf from newlib-nano。

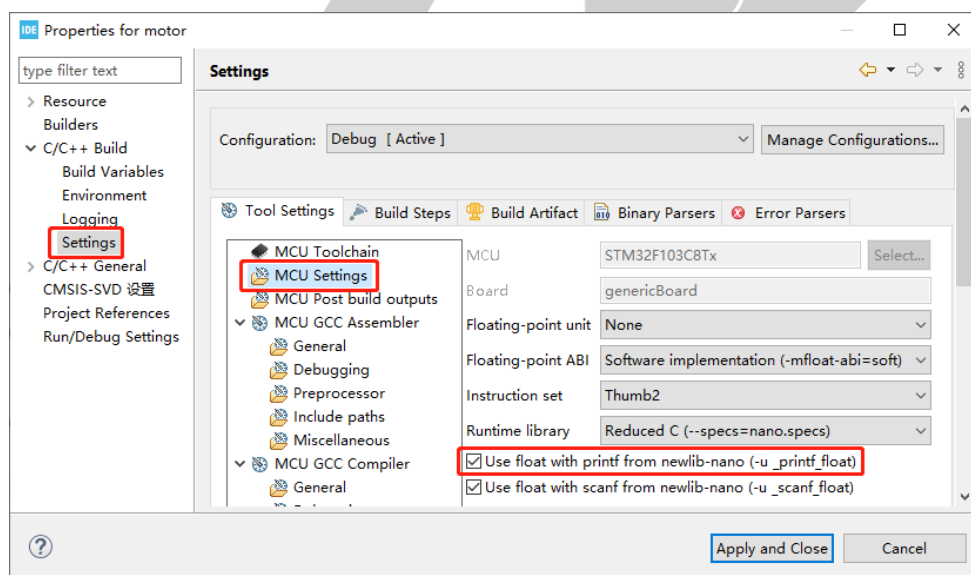


图 1-7 STM32CubeIDE 的 properties 窗口

至此，STM32 函数库已配置完成，可以任意使用库中的函数（前提是要包含 DrEmpower\_uart.h 文件）。

### (3) 使用 UART 接口（uart 协议）进行控制（G）

STM32 函数库基于 HAL 库，使用串口操纵电机，适用于使用了 HAL 库的 STM32 工程。

如需新建 HAL 库工程，使用 STM32 电机库函数，需打开 STM32CubeMX，新建项目，选择完对应的芯片型号之后，进入配置界面。

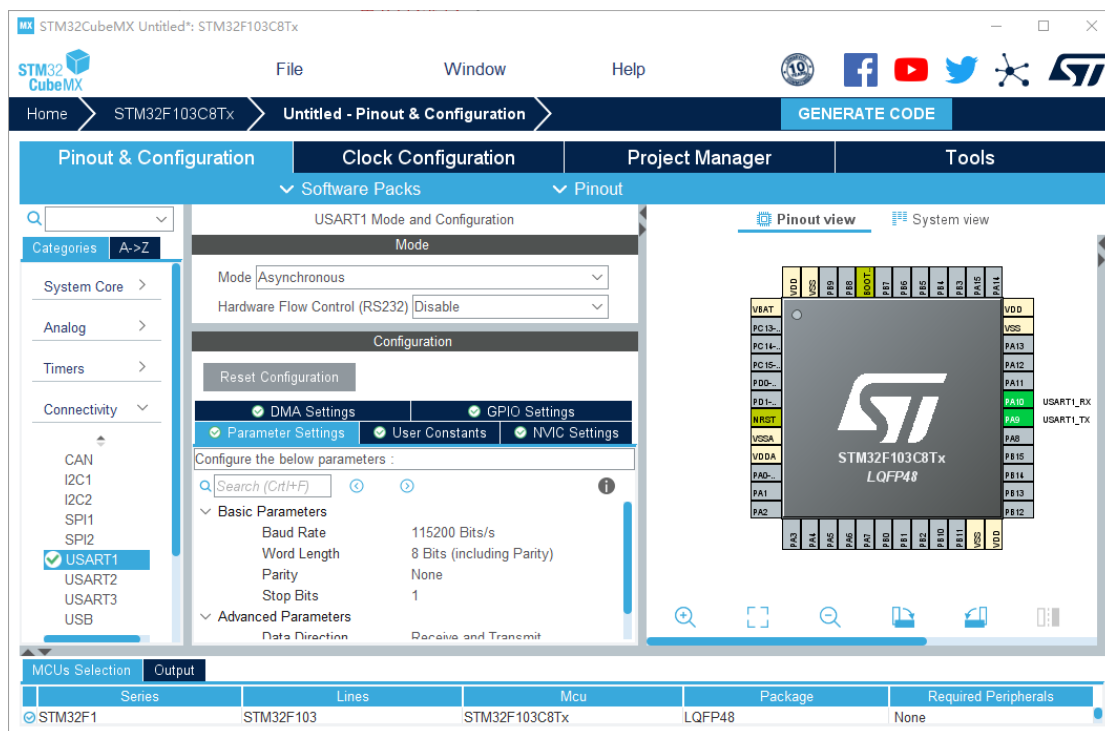


图 1-8 STM32CubeMX 界面 USART 配置

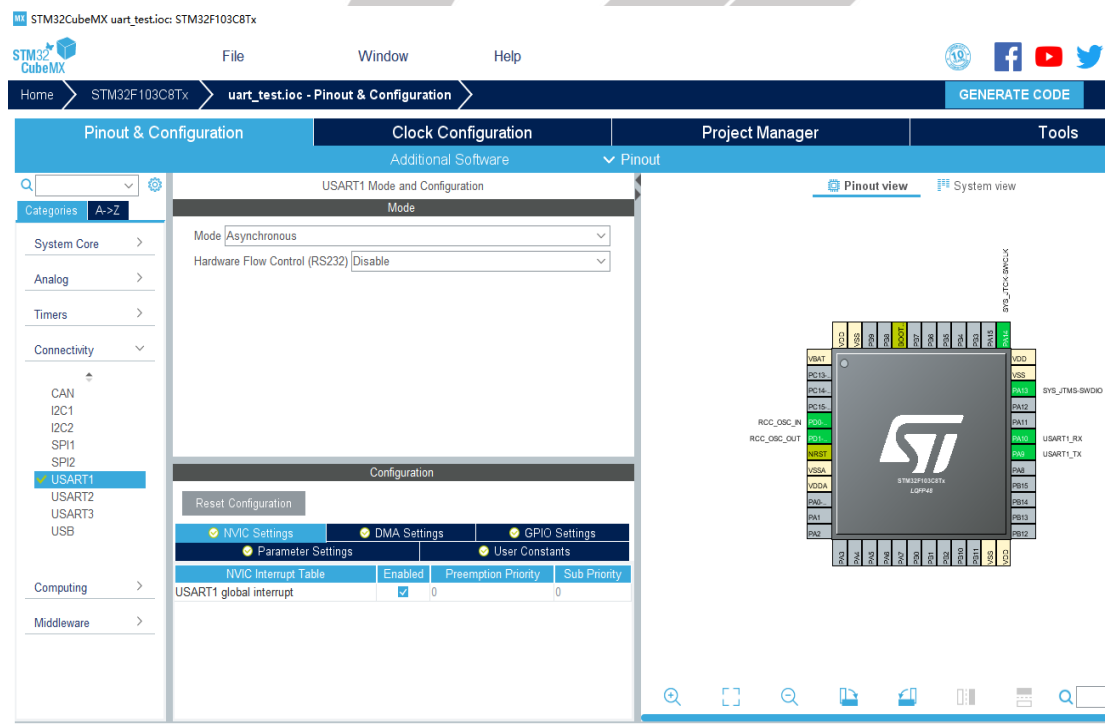


图 1-9 USART 配置打开串口接收中断

在左侧列表选中任意一个串口（或在右侧引脚图中直接选择串口引脚），将其模式配置为 **Asynchronous**，再将波特率设置为与电机串口波特率对应的数值。然后还需要在中断配置中将接收中断打开，如图 1-9 所示。

设置完串口之后，再进行时钟、选择生成目标 IDE(MDK-ARM、STM32CubeIDE 等)工程等其它配置，再在左侧 Code Generator 中勾选 **Generate peripheral**



initialization as a pair of '.c/.h' files per peripheral, 最后点击右上角的 GENERATE CODE 生成代码, 就完成了项目的创建。

如已有工程是使用 HAL 库的, 也可以点开 STM32CubeMX 的工程文件, 新配置一个操作电机的串口, 如不使用 STM32CubeMX, 可以手动配置串口。

配置完成串口之后, 项目代码中会使用串口句柄 UART\_HandleTypeDef 进行对串口的一系列操作。将 DrEmpower\_uart.c 和 DrEmpower\_uart.h 文件添加至项目中, 修改 DrEmpower\_uart.c 中的宏定义为自己配置的 UART\_HandleTypeDef (如 huart1)。

最后由于 STM32 函数库使用了 sprintf 函数输出浮点数, 所以也要在 IDE 中进行一些配置才可以让 STM32 的 sprintf 函数正确输出浮点数:

**Keil 项目:** 在工程属性的 Target → Code Generation 中勾选 Use MicroLIB 选项。

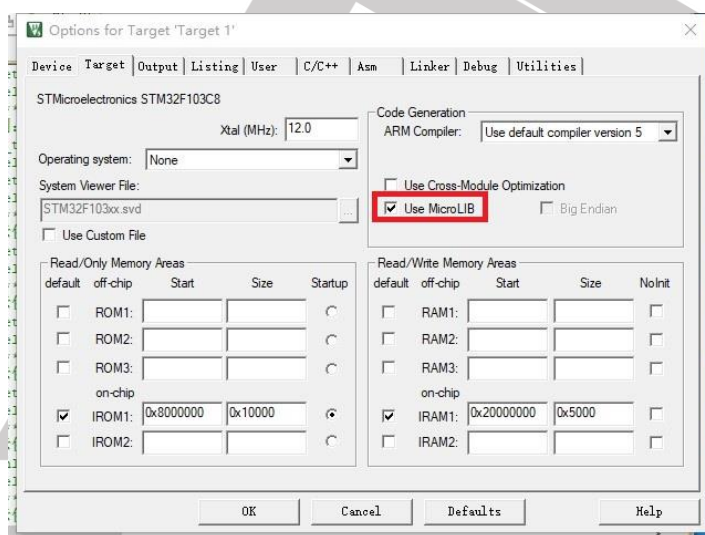


图 1-10 Keil Use MicroLIB

**STM32CubeIDE 项目:** 点击菜单栏中的 Project → Properties, 打开 Properties 窗口, 在 MCU Settings 中选中 Use float with printf from newlib-nano。

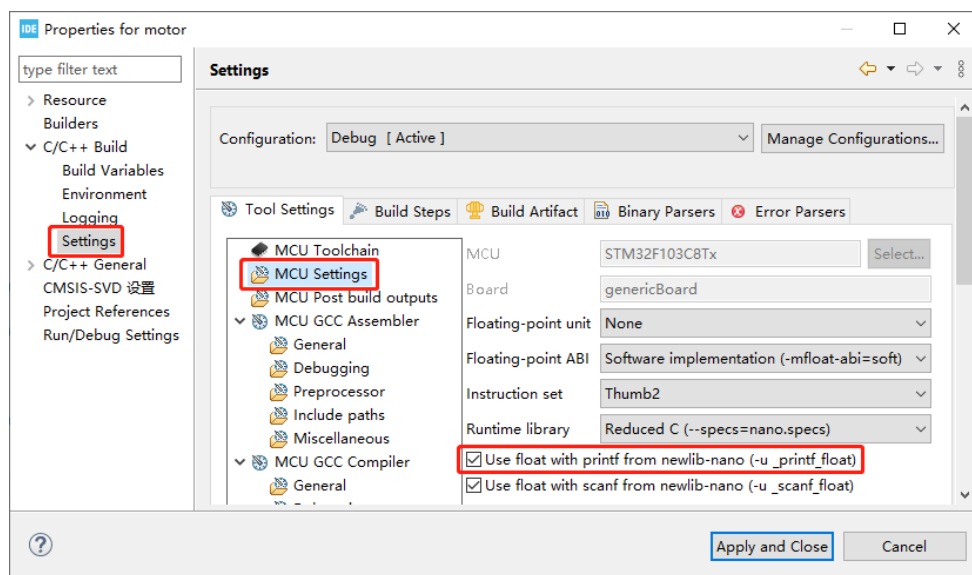


图 1-11 STM32CubeIDE 的 properties 窗口

至此，STM32 函数库已配置完成，可以任意使用库中的函数（前提是要包含 DrEmpower\_uart.h 文件）。

#### (4) 使用 CAN 接口进行控制

STM32 函数库基于 HAL 库，使用 CAN 接口控制电机，适用于使用了 HAL 库的 STM32 工程。需要注意的是，使用 CAN 接口控制电机，需要使用一个 CAN 收发模块，例如 TJA1050 模块，如图 1- 所示。其中 TJA1050 模块与 STM32 单片机接线方式如下：

VCC：接单片机上的 5.0V 引脚(注意 TJA1050 模块不能用 3.3V 供电)

GND：接单片机上的 GND 引脚

TXD：接单片机上的 CAN\_TX 引脚

RXD：接单片机上的 CAN\_RX 引脚(注意这里不是反接)

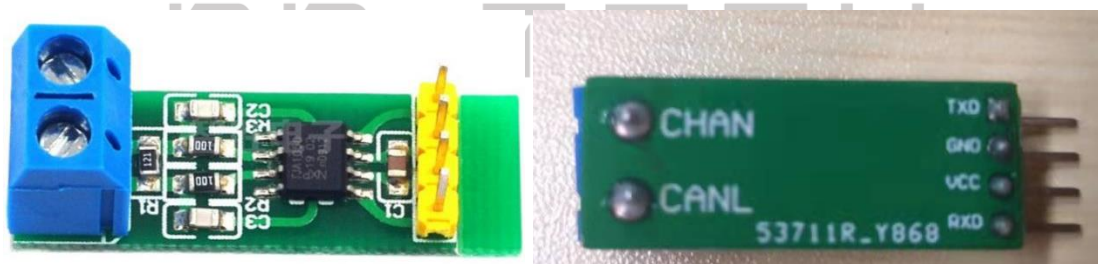


图 1- 12 TJA1050 模块实物图

如需新建 HAL 库工程，使用 STM32 电机库函数，需打开 STM32CubeMX，新建项目，选择完对应的芯片型号之后，进入配置界面，配置好系统时钟（RCC, SYS, Clock Configuration）。

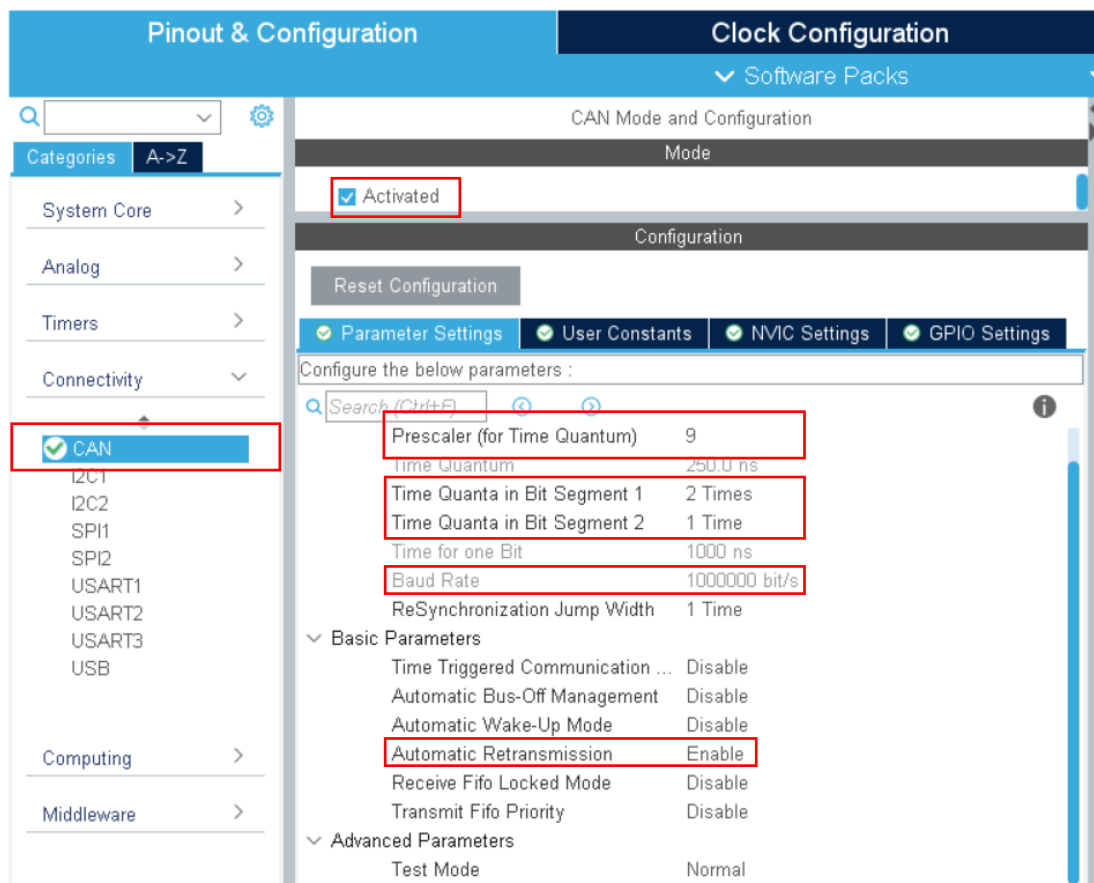


图 1- 13 STM32CubeMX CAN 配置界面截图

在左侧列表选中 CAN（或在右侧引脚图中直接选择串口引脚），首先将上面的 Activated 复选框选中，然后将 CAN 波特率（Baud Rate）配置为电机 can 通信波特率（默认为 1M）。CAN 波特率是由四个参数共同决定的，计算公式如下：

$$\text{BaudRate} = \text{APB1CLK} / \text{分频系数} / (1 + \text{BS1} + \text{BS2})$$

其中 APB1CLK 为 TM32 中 APB1 时钟频率，这里为 36M

分频系数为图 1- 中的 Prescaler 参数，图中分频系数=9

BS1 为图 1- 中的 Time Quanta in Bit Segment 1 参数，图中对应值为 2

BS2 为图 1- 中的 Time Quanta in Bit Segment 2 参数，图中对应值为 1

所以最终波特率：BaudRate=36M/9/(1+2+1)=1M.

此外需要将 Automatic Retransmission 设置为 Enable。

配置好 CAN 波特率后,选择 NVIC Settings，将 CAN 接收中断打开，如图 1- 0 所示。

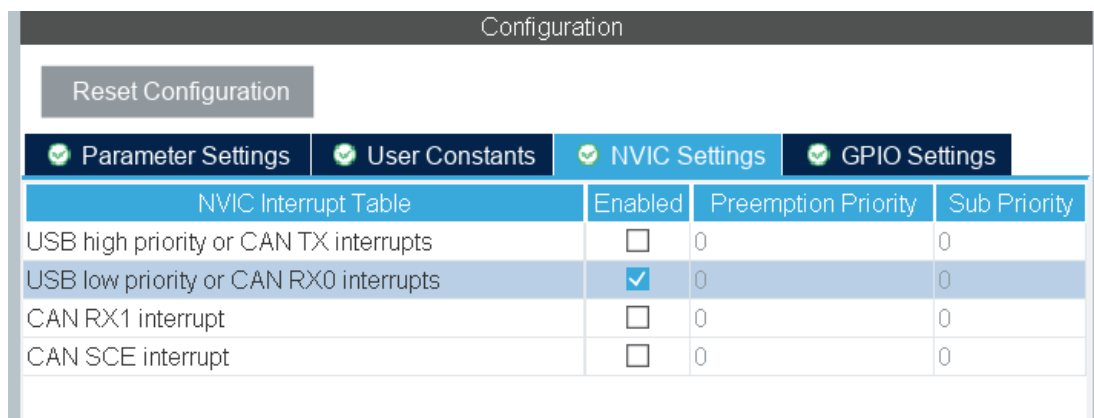


图 1-14 打开 CAN 接收中断

配置好 CAN 之后，选择生成目标 IDE(MDK-ARM、STM32CubeIDE 等)工程及其它配置，再在左侧 Code Generator 中勾选 Generate peripheral initialization as a pair of '.c/.h' files per peripheral，最后点击右上角的 GENERATE CODE 生成代码，就完成了项目的创建。

使用 CAN 接口创建完上述项目后，还需要手动修改一部分代码。修改说明参考《创建项目后手动添加代码说明》，该文件在 STM32 CAN 版本库函数所在文件夹下，按照说明添加相应代码后，将 CAN 版本库函数复制到项目中即可。

另外部分 STM32 开发板的 USB 接口采用了 PA11 和 PA12 引脚，这时如果使用电脑 USB 口给 STM32 单片机供电，可能造成 CAN 通信异常；解决办法是采用 5V 适配器或插座 USB 口进行供电，或者通过重定向将 CAN 接口改为 PB8 和 PB9；

### 1.3 添加 Arduino 函数库

#### (1) Arduino 使用 UART 接口进行控制

Arduino 的电机库函数使用 UART 串口接口控制电机，所以将占用 Arduino 的串口。将 DrEmpower.cpp 和 DrEmpower.h 文件所在文件夹 DrEmpower 添加至 arduino 第三方库文件所在目录 libraries 中，在要用到库函数的 arduino 程序文件 (.ino) 中包含 DrEmpower.h 头文件，并在使用库函数前调用 Serial.begin() 方法初始化串口，即可使用库函数中的功能。

Program Files (x86) > Arduino > libraries

名称	修改日期	类型
Adafruit_Circuit_Playground	2018/10/9 21:06	文件夹
Bridge	2018/10/9 21:06	文件夹
DrEmpower	2021/10/8 12:55	文件夹
Esplora	2018/10/9 21:06	文件夹
Ethernet	2018/10/9 21:06	文件夹
Firmata	2018/10/9 21:06	文件夹

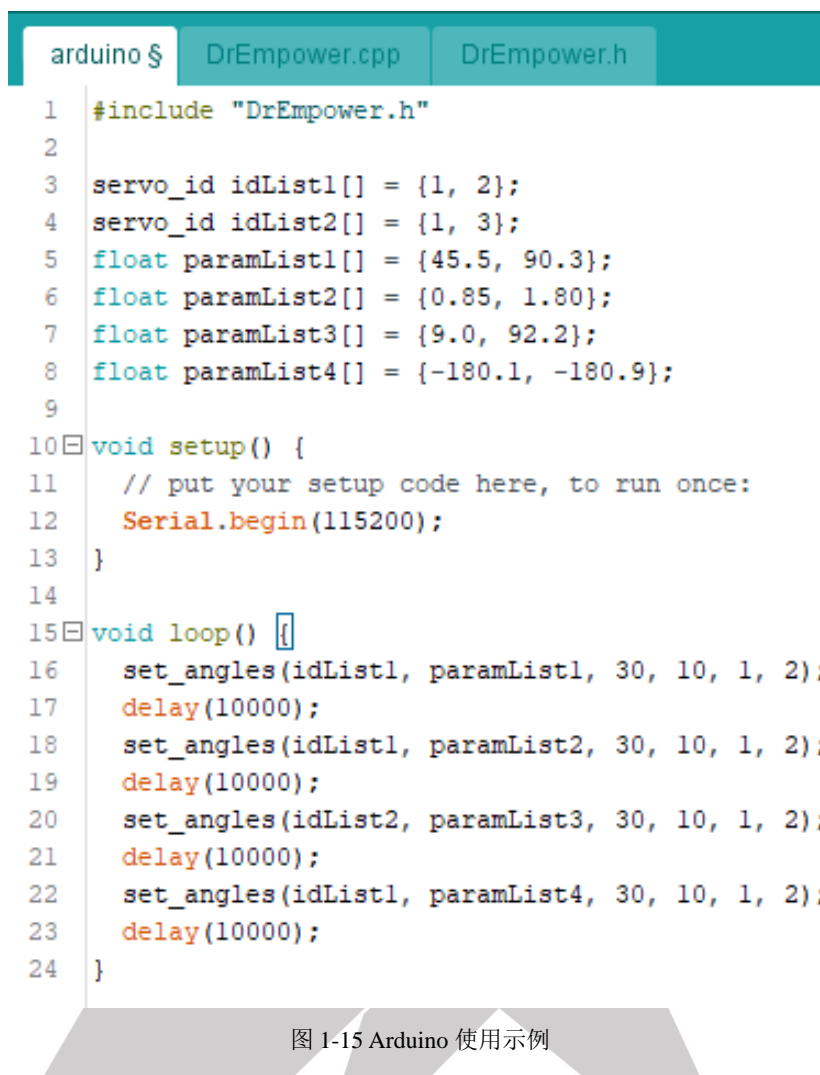


图 1-15 Arduino 使用示例

## (2) Arduino 使用 CAN 接口进行控制

硬件部分：

Arduino 通过 SPI 接口连接 MCP2515\_CAN 模块进行控制电机，如图 1-12 所示，MCP2515\_CAN 模块与 Arduino SPI 接口相连接。



图 1-16 MCP2515\_CAN 模块

MCP2515 CAN 模块与 Arduino 硬件连接如图 1-13 所示：(这里以 Arduino

Nano 为例，其它 Arduino 主板也是类似连接方式)

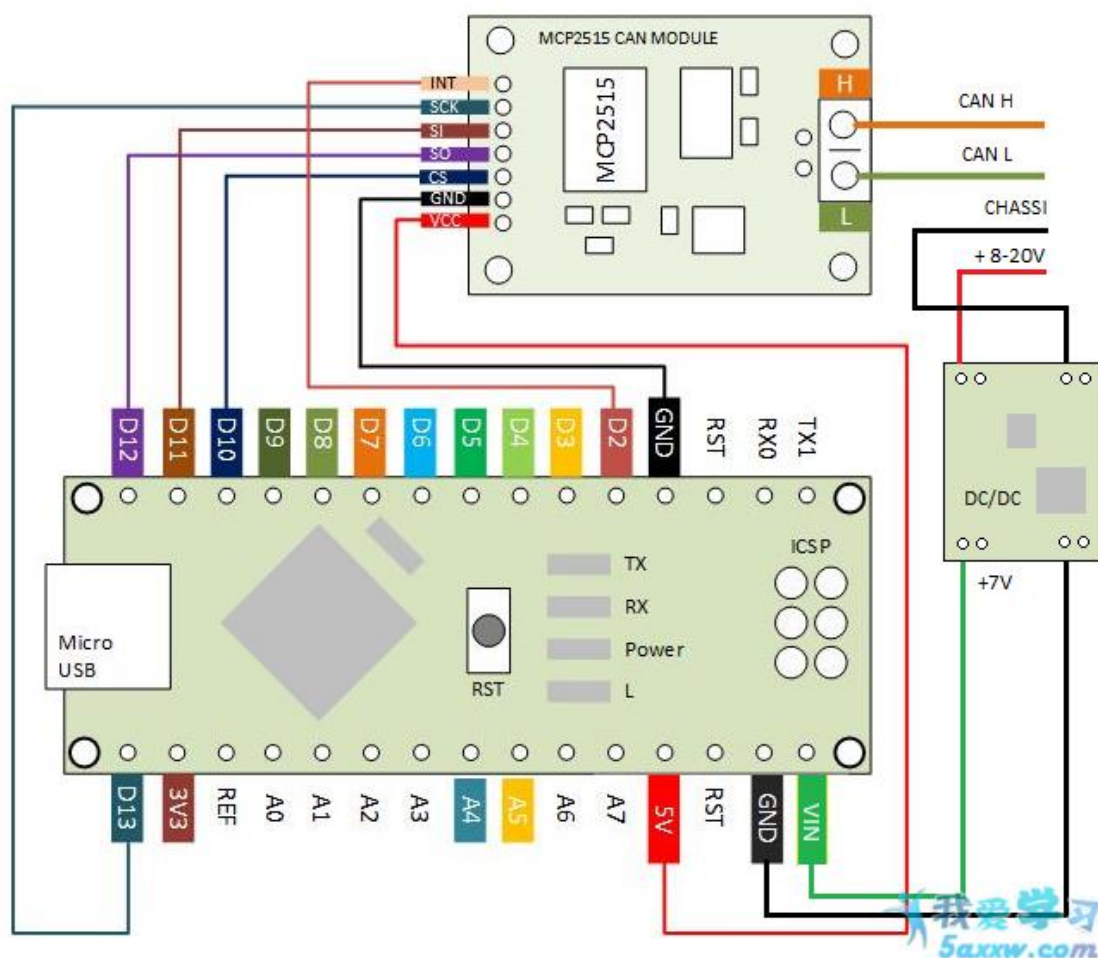


图 1-17 MCP2515\_CAN 模块与 Arduino 的接线图

软件部分:

安装 Arduino mcp2515 驱动库

1. 从 <https://github.com/autowp/arduino-mcp2515/archive/master.zip> 下载 ZIP 文件
2. 从 ArduinoIDE:Sketch->Include Library。。。->添加.ZIP 库。。。
3. 重新启动 Arduino IDE 以查看新的“mcp2515”库是否安装完成。

Arduino 工程建立

1. 使用 CAN 接口，将 DrEmpower.cpp 和 DrEmpower.h 文件所在文件夹 DrEmpower 添加至 arduino 第三方库文件所在目录 libraries 中，如图 1-14 所示
2. 打开 Arduino 新建工程,在工程开始处包含头文件: `#include "DrEmpower_can.h"`
3. 在 void setup() 中加入 MCP2515 模块的初始化函数: `MCP2515_CAN_Init();`
4. 打开 examples 文件，把.ino 文件内容复制进去或者自己根据所需函数自己操作。



Program Files (x86) &gt; Arduino &gt; libraries

名称	修改日期	类型
Adafruit_Circuit_Playground	2018/10/9 21:06	文件夹
Bridge	2018/10/9 21:06	文件夹
DrEmpower	2021/10/8 12:55	文件夹
Esplora	2018/10/9 21:06	文件夹
Ethernet	2018/10/9 21:06	文件夹
Firmata	2018/10/9 21:06	文件夹

图 1-18 Arduino 文件存放

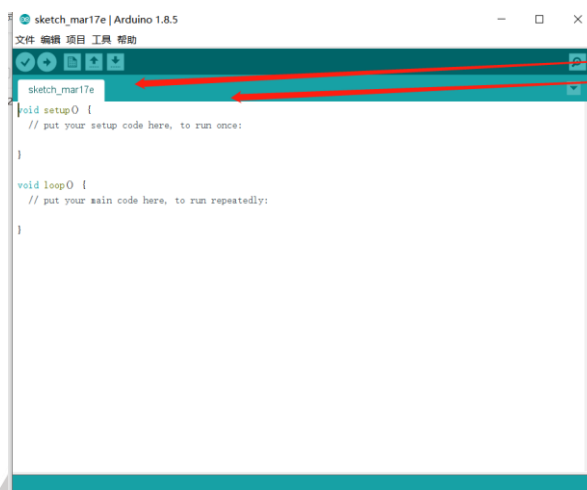


图 1-19 Arduino 创建新工程

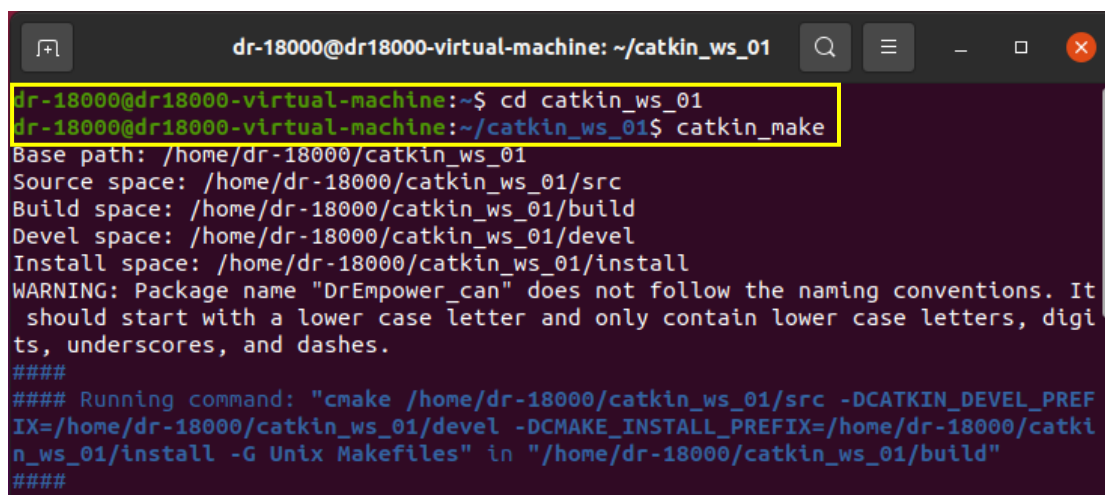
## 1.4 添加 ROS 函数库

DrEmpower 系列驱动器的 ROS 函数库基于 CAN 通信，采用消息发布和订阅模式。驱动器各控制函数充当消息订阅者，用户充当消息发布者。当订阅者收到消息后会执行相应的控制功能。

将功能包 DrEmpower\_can 粘贴到 ROS 工作空间的 SRC 文件夹下，如图 1-15 所示。在终端使用 cd 命令打开 ROS 工作空间；随后使用 catkin\_make 命令进行编译，如图 1-16 所示，图中 catkin\_ws\_01 是 ROS 工作空间名称。编译成功后使用 roslaunch DrEmpower\_can launch\_actuator.launch 命令启动控制节点，如图 1-17 所示。节点启动成功后再打开一个新终端，在新终端中发布对应的消息控制电机。具体操作方式参见后续各章节特定功能操作说明。

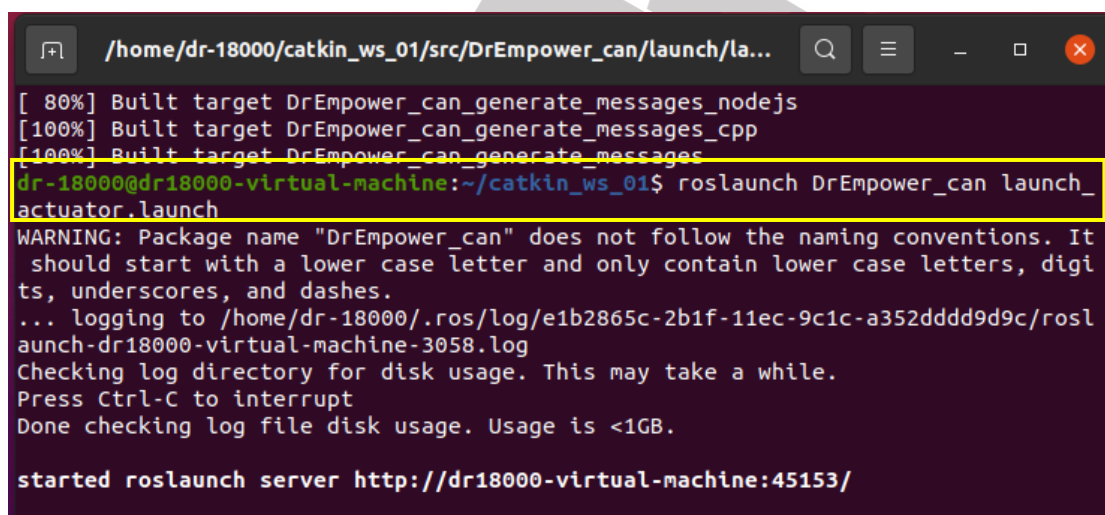


图 1-20 将功能包 DrEmpower\_can 粘贴到 ROS 工作空间的 SRC 文件夹下



```
dr-18000@dr18000-virtual-machine: ~/catkin_ws_01
dr-18000@dr18000-virtual-machine:~$ cd catkin_ws_01
dr-18000@dr18000-virtual-machine:~/catkin_ws_01$ catkin_make
Base path: /home/dr-18000/catkin_ws_01
Source space: /home/dr-18000/catkin_ws_01/src
Build space: /home/dr-18000/catkin_ws_01/build
Devel space: /home/dr-18000/catkin_ws_01/devel
Install space: /home/dr-18000/catkin_ws_01/install
WARNING: Package name "DrEmpower_can" does not follow the naming conventions. It
should start with a lower case letter and only contain lower case letters, digi
ts, underscores, and dashes.
####
#### Running command: "cmake /home/dr-18000/catkin_ws_01/src -DCATKIN_DEVEL_PREF
IX=/home/dr-18000/catkin_ws_01/devel -DCMAKE_INSTALL_PREFIX=/home/dr-18000/catki
n_ws_01/install -G Unix Makefiles" in "/home/dr-18000/catkin_ws_01/build"
####
```

图 1-21 在终端使用 cd 命令打开 ROS 工作空间



```
/home/dr-18000/catkin_ws_01/src/DrEmpower_can/launch/la...
[ 80%] Built target DrEmpower_can_generate_messages_nodejs
[100%] Built target DrEmpower_can_generate_messages_cpp
[100%] Built target DrEmpower_can_generate_messages
dr-18000@dr18000-virtual-machine:~/catkin_ws_01$ roslaunch DrEmpower_can launch_
actuator.launch
WARNING: Package name "DrEmpower_can" does not follow the naming conventions. It
should start with a lower case letter and only contain lower case letters, digi
ts, underscores, and dashes.
... logging to /home/dr-18000/.ros/log/e1b2865c-2b1f-11ec-9c1c-a352ddd9d9c/rosl
aunch-dr18000-virtual-machine-3058.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://dr18000-virtual-machine:45153/
```

图 1-22 在终端使用 roslaunch 命令启动控制节点

注：启动节点有时会出现端口无法打开的情况，此时请使用 `ls -l /sys/class/tty/ttyUSB*` 命令检查 USB 转 CAN 模块在系统中的端口号，并在 `DrEmpower.py` 文件开头端口号设置处加以修改。

（如果是新版本虚拟串口版本 USB 转 CAN 模块，则使用 `ls -l /dev/ttyACM*` 命令进行检查）



## 第二章 运动控制功能

## 2.1 单个电机位置控制

单个电机位置控制函数 `set_angle()`，控制指定电机编号的电机按照指定的速度转动到指定的角度（绝对角度，相对于电机零点位置）。其原型及参数解释如表 2-1 和表 2-2。

表2-1 单个电机位置控制函数原型说明

函数原型说明	
函数名	<code>set_angle</code>
函数原型	<code>set_angle(id_num=1,angle=0,speed=0, param=0, mode=0)</code>
功能描述	电机角度控制
输入参数	<code>id_num,angle、 speed param、 mode</code>
返回值	<code>Tryue、 False</code>

表2-2 单个电机位置控制函数参数解释说明

序号	参数	解释
1	<code>id_num</code>	需要设置的电机ID编号,如果不知道当前电机ID,可以用0广播,如果总线上有多个电机,则多个电机都会执行该操作
2	<code>angle</code>	电机角度 $(-360 \sim 360) * n$ ( $n < 2000$ ), 支持大角度转动
3	<code>speed</code>	最大速度限制或前馈速度 (r/min)
4	<code>param</code>	<code>mode=0</code> 时为角度输入滤波带宽; ( $0 < param < 300$ ); <code>mode=1</code> 时为启动和停止阶段加速度 $((r/min)/s)$ ; <code>mode =2</code> 时为前馈扭矩 (Nm)。
5	<code>mode</code>	<code>mode=0</code> : 轨迹跟踪模式,用于一般绕轴运动,特别适合多个轨迹点输入,角度输入带宽参数需设置为指令发送频率的一半。 <code>mode=1</code> : 梯形轨迹模式,这种模式下可以指定运动过程中的速度 ( <code>speed</code> ) 和启停加速度 ( <code>accel</code> )。 <code>mode=2</code> : 前馈控制模式,这种模式下 <code>speed</code> 和 <code>torque</code> 为前馈控制量,前馈控制在原有PID控制基础上加入速度和扭矩前馈,提高系统的响应特性和减少静态误差。
备注		在 <code>mode=1</code> , 梯形轨迹模式中, <code>speed</code> 和 <code>accel</code> 都需要大于0。如果 <code>speed=0</code> 会导致电机报 <code>motor_error</code> 并退出闭环控制模式,所以在这种模式下如果 <code>speed=0</code> ,会被用0.01代替。 另外如果这种模式下 <code>accel=0</code> ,电机以最快速度运动到 <code>angle、 speed</code> 参数不再起作用。 在 <code>mode=0</code> , 轨迹追踪模式中, 一条轨迹的采样点数应尽量密, 根据采样点和角度发送频率计算的电机转速不可超过其最大转速。

例如, 控制 1 号电机以梯形轨迹模式, 转动到  $180^\circ$ , 转速为 10r/min, 角加速度为 20r/min/s。其代码如下:

`XX.set_angle(1, 180, 10, 20, 1)` # `XX` 代表用户定义的对象名称, 1 代表电机 ID 号, 180 代表目标位置, 10 代表转速, 20 代表角加速度, 1 代表电机运动模式。

上述例子对应的 ROS 命令为 `rostopic pub -1 set_angle DrEmpower_can/arg_set_angle "{id: 0, angle: 0.0, speed: 0.0, param: 0.0, mode: 0}"`，填入对应参数后敲击回车便可，如图 2-1。（图 2-1 中代码比较长，但一般情况下只需键值 `set_angle` 后再空一格便可敲击 Tab 键由 ROS 系统将后续参数补足；命令中的 -1 为消息发布频率，可以不写）

```
dr-18000@dr18000-virtual-machine: ~
dr-18000@dr18000-virtual-machine:~$ rostopic pub -1 set_angle DrEmpower_can/arg_set_angle "{id: 1, angle: 180.0, speed: 10.0, param: 10.0, mode: 1}"
```

图 2-1 使用 ROS 命令对单个电机进行位置控制

## 2.2 多个电机位置控制

多个电机同步控制函数 `set_angles()`，同步控制多个电机按照指定的时间转动到指定的角度，保证多个电机同时到达目标角度。其原型及参数解释如表 2-3 和表 2-4。

表2-3 多个电机同步控制函数原型说明

函数原型说明	
函数名	<code>set_angles</code>
函数原型	<code>set_angles(id_list=[1, 2, 3], angle_list=[150.0, 150.0, 150.0], speed=10, param=10, mode=1, n)</code>
功能描述	多个电机角度控制
输入参数	<code>id_list</code> 、 <code>angle_list</code> 、 <code>speed</code> <code>param</code> 、 <code>mode</code> 、 <code>n</code>
返回值	无

表2-4 多个电机位置控制函数参数解释说明

序号	参数	解释
1	<code>id_list</code>	需要设置的电机ID编号组成的列表
2	<code>angle_list</code>	电机角度组成的列表
3	<code>speed</code>	角度转动范围最大的电机转动的速度（r/min）
4	<code>param</code>	角度输入滤波带宽（1/s）或者目标加速度（(r/min)/s）
5	<code>mode</code>	<p>角度控制模式选择，电机支持两种角度控制模式。</p> <p><code>mode = 0</code>: 多个电机轨迹跟踪模式，此时<code>speed</code>参数没有左右，<code>param</code>为角度输入滤波带宽（1/s），该参数需要需等于角度发送频率的一半。此时一条轨迹的采样点数应尽量密，根据采样点和角度发送频率计算的电机转速不可超过其最大转速。</p> <p><code>mode = 1</code>: 多个电机梯形轨迹模式，此时<code>speed</code>用运动时间<code>t</code>（s）表示，<code>param</code>为目标加速度（(r/min)/s）。</p> <p><code>mode = 2</code>: 前馈控制模式，这种模式下的<code>speed</code>和<code>torque</code>分别为前馈控制量。前馈控制在原有PID控制基础上加入速度和扭矩前馈，提高系统的响应特性和减少静态误差。</p>
6	<code>n</code>	<code>id_list</code> 和 <code>angle_list</code> 的长度（Python函数库中没有该参数）
备注		<code>id_list</code> 与 <code>angle_list</code> 列表内元素个数必须相同

例如，控制 1、2、3 号电机以梯形轨迹模式，分别转动到  $45^\circ$ 、 $90^\circ$ 、 $135^\circ$ ，转速为 10r/min，角加速度为 20r/min/s。其代码如下：

```
XX.set_angles([1,2,3], [45,90,135], 10, 20, 1)
```

# XX 代表用户定义的对象名称，[1,2,3]代表电机 ID 号列表，[45,90,135]代表电机目标角度列表，10 代表转速，20 代表角加速度，1 代表电机运动模式。

上述例子对应的 ROS 命令为 `rostopic pub -1 set_angles DrEmpower_can/arg_set_angles "id_list: [0]`

```
angle_list: [0]
```

```
speed: 0.0
```

```
param: 0.0
```

`mode: 0"`，填入对应参数后敲击回车便可，如图 2-2。（图 2-2 中代码比较长，但一般情况下只需键值 `set_angles` 后再空一格便可敲击 Tab 键由 ROS 系统将后续参数补足；命令中的 -1 为消息发布频率，可以不写）

```
dr-18000@dr18000-virtual-machine:~$ rostopic pub -1 set_angles DrEmpower_can/arg_set_angles "id_list: [1, 2, 3]
angle_list: [49, 90, 135]
speed: 10.0
param: 20.0
mode: 1"
```

图 2-2 使用 ROS 命令对多个电机进行位置控制

### 2.3 单个电机相对角度控制

单个电机相对角度控制函数 `step_angle()`，控制指定电机编号的电机按照指定的速度相对转动指定的角度（相对角度，相对于电机当前位置），其原型及参数解释如表 2-5 和表 2-6。

表2-5 单个电机相对角度控制函数原型说明

函数原型说明	
函数名	<code>step_angle</code>
函数原型	<code>step_angle(id_num=1, angle=0, speed=0, param=0, mode=0)</code>
功能描述	单个电机相对角度控制
输入参数	<code>id_num</code> 、 <code>angle</code> 、 <code>speed</code> 、 <code>param</code> 、 <code>mode</code>
返回值	True、False

表2-6 单个电机相对角度控制函数参数解释说明

序号	参数	解释
1	<code>id_num</code>	需要设置的电机ID编号,如果不知道当前电机ID,可以用0广播,如果总线上有多个电机,则多个电机都会执行该操作
2	<code>angle</code>	电机相对角度 $(-360 \sim 360) * n$ , 支持大角度转动
3	<code>speed</code>	最大速度限制或前馈速度 (r/min)

序号	参数	解释
4	param	mode=0 时为角度输入滤波带宽; (param<300); mode=1 时为启动和停止阶段加速度 ((r/min)/s); mode =2 时为前馈扭矩 (Nm)。
5	mode	mode=0: 轨迹跟踪模式, 用于一般绕轴运动, 特别适合多个轨迹点输入, 角度输入带宽参数需设置为指令发送频率的一半。 mode=1: 梯形轨迹模式, 这种模式下可以指定运动过程中的速度 (speed) 和启停加速度 (accel)。 mode=2: 前馈控制模式, 这种模式下speed和torque为前馈控制量, 前馈控制在原有PID控制基础上加入速度和扭矩前馈, 提高系统的响应特性和减少静态误差。
备注		在mode=1,梯形轨迹模式中, speed和accel都需要大于0.如果speed=0会导致电机报motor error 并退出闭环控制模式, 所以在这种模式下如果speed=0,会被用0.01代替。 另外如果这种模式下accel=0, 电机以最快速度运动到angle,speed参数不再起作用。

例如, 控制 1 号电机以梯形轨迹模式, 转动 180°, 转速为 10r/min, 角加速度为 20r/min/s。其代码如下:

```
XX. step_angle (1, 180, 10, 20, 1)
```

# XX 代表用户定义的对象名称, 1 代表电机 ID 号, 180 代表转动的相对角度, 10 代表转速, 20 代表角加速度, 1 代表电机运动模式。

上述例子对应的 ROS 命令为 `rostopic pub -1 step_angle DrEmpower_can/arg_step_angle "{id: 0, angle: 0.0, speed: 0.0, param: 0.0, mode: 0}"`, 填入对应参数后敲击回车便可, 如图 2-3。(图 2-3 中代码比较长, 但一般情况下只需键值 `step_angle` 后再空一格便可敲击 Tab 键由 ROS 系统将后续参数补足; 命令中的 -1 为消息发布频率, 可以不写)

```
dr-18000@dr18000-virtual-machine:~$ rostopic pub -1 step_angle DrEmpower_can/arg_step_angle "{id: 1, angle: 180.0, speed: 10.0, param: 20.0, mode: 1}"
```

图 2-3 使用 ROS 命令对单个电机进行相对角度控制

## 2.4 多个电机相对角度同步控制

多个电机相对角度同步控制函数 `step_angles ( )`, 同步控制多个电机按照指定的时间相对转动指定的角度, 保证多个电机同时到达目标角度。其原型及参数解释如表 2-7 和表 2-8。

表2-7 多个电机相对角度同步控制函数原型说明

函数原型说明	
函数名	<code>step_angles</code>
函数原型	<code>step_angles(id_list=[1, 2, 3], angle_list=[150.0, 150.0, 150.0], speed=10, param=10, mode=1, n)</code>

函数原型说明	
功能描述	多个电机相对角度同步控制
输入参数	id_list、angle_list、speed、param、mode
返回值	无

表2-8多个电机相对角度同步控制函数参数解释说明

序号	参数	解释
1	id_list	电机编号组成的列表
2	angle_list	电机角度组成的列表
3	speed	最大的电机转动的速度 (r/min) 或前馈速度
4	param	mode=0,角度输入滤波带宽 (<300) , mode=1,启动和停止阶段加速度 ((r/min)/s) ; mode =2, 前馈速度 (r/min)
5	mode	角度控制模式选择, 电机支持三种角度控制模式, mode = 0: 多个电机轨迹跟踪模式, 用于一般绕轴运动, 特别适合多个轨迹点输入, 角度输入带宽参数需设置为指令发送频率的一半。 mode = 1: 多个电机梯形轨迹模式, 此时speed为多个电机中的最快速度 (r/min) , param为目标加速度 ((r/min)/s) 。 mode = 2: 前馈控制模式, 这种模式下的speed和torque分别为前馈控制量.前馈控制在原有PID控制基础上加入速度和扭矩前馈, 提高系统的响应特性和减少静态误差。
6	n	id_list 和 angle_list 的长度 (Python函数库中没有该参数)
异常		如果id_list和angle_list长度不一致时会显示Parameter errors in set_angles()!

例如, 控制 1、2、3 号电机以梯形轨迹模式, 分别相对转动 45°、90°、135°, 转速为 10r/min, 角加速度为 20r/min/s。其代码如下:

```
XX.step_angles([1,2,3], [45,90,135], 10, 20, 1)
```

# XX 代表用户定义的对象名称, [1,2,3]代表电机 ID 号列表, [45,90,135]代表电机目标相对转动角度列表, 10 代表转速, 20 代表角加速度, 1 代表电机运动模式。

上述例子对应的 ROS 命令为 `rostopic pub -1 step_angles DrEmpower_can/arg_step_angles "id_list: [0]`

`angle_list: [0]`

`speed: 0.0`

`param: 0.0`

`mode: 0"`, 填入对应参数后敲击回车便可, 如图 2-4。(图 2-4 中代码比较长, 但一般情况下只需键值 `step_angles` 后再空一格便可敲击 Tab 键由 ROS 系统将后续参数补足; 命令中的 -1 为消息发布频率, 可以不写)

```
dr-18000@dr18000-virtual-machine:~$ rostopic pub -1 step_angles DrEmpower_can/arg_step_angles "id_list: [1, 2, 3]
angle_list: [45, 90, 135]
speed: 10.0
param: 20.0
mode: 10"
```

图 2-4 使用 ROS 命令对多个电机进行相对角度控制

## 2.5 单个电机速度控制

单个电机速度控制函数 `set_speed( )`，控制指定电机编号的电机按照指定的速度连续整周转动，其原型及参数解释如表 2-9 和表 2-10。

表2-9 单个电机速度控制函数原型说明

函数原型说明	
函数名	set_speed
函数原型	set_speed(id_num=0,speed=10,param=0, mode=1)
功能描述	设置电机用不同模式以一定速度运行
输入参数	id_num、speed、param、mode
返回值	True、False

表2-10 单个电机速度控制函数参数解释说明

序号	参数	解释
1	id_num	需要设置的电机ID编号，如果不知道当前电机ID，可以用0广播，如果总线上有多个电机，则多个电机都会执行该操作
2	speed	电机速度（r/min）
3	param	mode=1时为前馈扭矩（Nm）； mode!=1,为目标加速度（(r/min)/s）；
4	mode	mode=1，速度前馈控制模式，电机将目标速度直接设为speed mode!=1，速度爬升控制模式，电机将按照目标加速度<axis>.controller.config.vel_ramp_rate变化到speed。
备注		在速度爬升模式下，如果目标加速度<axis>.controller.config.vel_ramp_rate设置为0，则电机速度将保持当前值不变

例如，控制 1 号电机以速度前馈控制模式，转速为 10r/min 转动，代码如下：

```
XX.set_speed(1,10,0,1)
```

# XX 代表用户定义的对象名称，1 代表 1 号电机，10 代表速度 10r/min，1 表示模式电机运动模式

上述例子对应的 ROS 命令为 `rostopic pub -1 set_speed DrEmpower_can/arg_set_speed "id: 0`

```
speed: 0.0
```

```
param: 0.0
```

```
mode: 0"
```

填入对应参数后敲击回车便可，如图 2-5。（图 2-5 中代码比较长，但一般情况下只需键值 `set_speed` 后再空一格便可敲击 Tab 键由 ROS 系统将后续参数补足；命令中的 -1 为消息发布频率，可以不写）



```
dr-18000@dr18000-virtual-machine:~$ rostopic pub -1 set_speed DrEmpower_can/arg_
set_speed "id: 1
speed: 10.0
param: 0.0
mode: 1"
```

图 2-5 使用 ROS 命令对单个电机进行速度控制

## 2.6 多个电机速度控制

多个电机速度控制函数 `set_speeds ( )`，控制指定多个电机编号的电机按照指定的速度连续整周转动，其原型及参数解释如表 2-11 和表 2-12。

表2-11 多个电机速度控制函数原型说明

函数原型说明	
函数名	<code>set_speeds</code>
函数原型	<code>set_speeds(id_list=[1, 2, 3], speed_list=[10.0, 20.0, 30.0], param=0, mode=1, n)</code>
功能描述	多个电机速度控制
输入参数	<code>id_list</code> 、 <code>speed_list</code> 、 <code>param</code> 、 <code>mode</code>
返回值	<code>True</code> 、 <code>False</code>

表2-12 多个电机速度控制函数参数解释说明

序号	参数	解释
1	<code>id_list</code>	电机编号组成的列表
2	<code>speed_list</code>	电机目标速度 (r/min) 组成的列表
3	<code>param</code>	<code>mode=1</code> , 前馈扭矩 (Nm); <code>mode!=1</code> ,或目标加速度 ((r/min)/s)
4	<code>mode</code>	控制模式选择 <code>mode=1</code> , 速度前馈控制模式, 电机将目标速度直接设为 <code>speed</code> <code>mode!=1</code> ,速度爬升控制模式, 电机将按照目标加速度 <code>axis0.controller.config_vel_ramp_rate</code> 变化到 <code>speed</code> 。
5	<code>n</code>	<code>id_list</code> 和 <code>speed_list</code> 的长度 (Python函数库中没有该参数)

例如，控制 1、2、3 号电机以速度前馈模式，分别以 10r/min、20r/min、30 r/min 的速度转动，前馈扭矩为 10r/min。其代码如下：

```
XX.set_speeds([1,2,3], [10,20,30], 10, 1)
```

# XX 代表用户定义的对象名称，[1,2,3]代表电机 ID 号列表，[10,20,30]代表电机目标转速列表，10 代表前馈扭矩，1 代表电机控制模式。

上述例子对应的 ROS 命令为 `rostopic pub -1 set_speeds DrEmpower_can/arg_set_speeds "id_list: [0]`

```
speed_list: [0]
```

```
param: 0.0
```

```
mode: 0
```

`none: 0"`，填入对应参数后敲击回车便可，如图 2-6。（图 2-6 中代码比较长，但一般情况下只需键值 `set_speeds` 后再空一格便可敲击 Tab 键由 ROS 系统将后

续参数补足；命令中的-1 为消息发布频率，可以不写；none 需使用非负整数）

```
dr-18000@dr18000-virtual-machine:~$ rostopic pub -1 set_speeds DrEmpower_can/arg
_set_speeds "id_list: [1, 2, 3]
speed_list: [10, 20, 30]
param: 10.0
mode: 1
none: 0"
```

图 2-6 使用 ROS 命令对多个电机进行速度控制

## 2.7 单个电机力矩控制

单个电机力矩（电流）闭环控制函数 `set_torque()`，控制指定电机编号的电机输出指定的扭矩（Nm），其原型及参数解释如表 2-13 和表 2-14。

表2-13 力矩控制函数原型说明

函数原型说明	
函数名	<code>set_torque</code>
函数原型	<code>set_torque(id_num=0,torque=0.1,param=0, mode=1)</code>
功能描述	设置电机以不同模式按照指定扭矩运行
输入参数	<code>id_num</code> 、 <code>torque</code> 、 <code>param</code> 、 <code>mode</code>
返回值	<code>True</code> 、 <code>False</code>

表2-14 力矩控制函数参数解释说明

序号	参数	解释
1	<code>id_num</code>	需要设置的电机ID编号。如不知道电机ID，可以用0广播，如总线上有多个电机，则多个电机都会执行该操作
2	<code>torque</code>	电机输出（Nm）
3	<code>param</code>	<code>mode=1</code> 时该参数无意义； <code>mode!=1</code> 时为扭矩上升速度 <code>&lt;axis&gt;.controller.config.torque_ramp_rate</code> （Nm/s）
4	<code>mode</code>	<code>mode=1</code> ，扭矩直接控制模式，电机将目标扭矩直接设为 <code>torque</code> 。 <code>mode!=1</code> ，扭矩爬升控制模式，电机将按照扭矩上升速率 <code>&lt;axis&gt;.controller.config.torque_ramp_rate</code> （Nm/s）变化到 <code>torque</code> 。
备注		如果电机转速超过您设置的 <code>vel_limit</code> ，电机输出的力矩将会减小。可以设置： <code>&lt;axis&gt;.controller.config.enable_current_mode_vel_limit = False</code> 来禁止力矩减小。 另外在扭矩爬升控制模式下，如果电机扭矩上升速率 <code>&lt;axis&gt;.controller.config.torque_ramp_rate</code> 为0，则电机扭矩将在当前值保持不变。

例如，控制 1 号电机以扭矩直接控制模式，以扭矩 0.1Nm/s 运动，代码如下：

```
XX.set_torque(1, 0.1, 0, 1)
```

# XX 代表用户定义的对象名称，第一个 1 代表 1 号电机，0.1 代表扭矩 0.1 Nm/s，第二个 1 代表电机运动模式 1。

上述例子对应的 ROS 命令为 `rostopic pub -1 set_torque DrEmpower_can/arg_set_torque "id: 0`



torque: 0.0

param: 0.0

mode: 0", 填入对应参数后敲击回车便可, 如图 2-7。(图 2-7 中代码比较长, 但一般情况下只需键值 set\_torque 后再空一格便可敲击 Tab 键由 ROS 系统将后续参数补足; 命令中的-1 为消息发布频率, 可以不写)

```
dr-18000@dr18000-virtual-machine:~$ rostopic pub -1 set_torque DrEmpower_can/arg
_set_torque "id: 1
torque: 0.1
param: 0.0
mode: 1"
```

图 2-7 使用 ROS 命令对单个电机进行力矩控制

## 2.8 多个电机力矩控制

多个电机力矩控制函数 set\_torques ( ), 同时控制多个电机编号的电机目标扭矩 (Nm), 其原型及参数解释如表 2-15 和表 2-16。

表2-15 多个电机扭矩控制函数原型说明

函数原型说明	
函数名	set_torques
函数原型	set_torques(id_list=[1, 2, 3], torque_list=[3.0, 4.0, 5.0], param=0, mode=1, n)
功能描述	多个电机扭矩控制
输入参数	id_list、torque_list、param、mode
返回值	True、False

表2-16 多个电机扭矩控制函数参数解释说明

序号	参数	解释
1	id_list	电机编号组成的列表
2	torque_list	电机目标扭矩 (Nm)组成的列表
3	param	mode=1,改参数无意义; mode!=1,扭矩上升速度axis0.controller.config.torque_ramp_rate (Nm/s)
4	mode	控制模式选择 mode=1, 扭矩直接控制模式, 电机将目标扭矩直接设为torque mode!=1,扭矩爬升控制模式, 电机将按照扭矩上升速率axis0.controller.config.torque_ramp_rate (Nm/s) 变化到torque。
5	n	id_list 和 torque_list 的长度 (Python函数库中没有该参数)
备注		如果电机转速超过您设置的 vel_limit , 电机输出的力矩将会减小。 可以设置 axis0.controller.config.enable_current_mode_vel_limit = False 来禁止力矩减小。 另外在扭矩爬升控制模式下, 如电机扭矩上升速率axis0.controller.config.torque_ramp_rate为0, 则电机扭矩将在当前值保持不变。

例如, 控制 1、2、3 号电机以扭矩爬升控制模式, 分别输出 3Nm、4Nm、5 Nm 的扭矩, 扭矩上升速度为 10Nm/s。其代码如下:

XX. set\_torques ([1,2,3], [3,4,5], 10, 0)

# XX 代表用户定义的对象名称, [1,2,3]代表电机 ID 号列表, [3,4,5]代表电机目标输出扭矩列表, 10 代表扭矩上升速度, 0 代表电机控制模式。

上述例子对应的 ROS 命令为 `rostopic pub -1 set_torques DrEmpower_can/arg_set_torques "id_list: [0]`

`torque_list: [0]`

`param: 0.0`

`mode: 0`

`none: 0"`, 填入对应参数后敲击回车便可, 如图 2-8。(图 2-8 中代码比较长, 但一般情况下只需键值 `set_torques` 后再空一格便可敲击 Tab 键由 ROS 系统将后续参数补足; 命令中的 -1 为消息发布频率, 可以不写; none 需使用非负整数)

```
dr-18000@dr18000-virtual-machine:~$ rostopic pub -1 set_torques DrEmpower_can/arg_set_torques "id_list: [1, 2, 3]
torque_list: [3, 4, 5]
param: 10.0
mode: 0
none: 0"
```

图 2-8 使用 ROS 命令对多个电机进行力矩控制

## 2.9 单个电机阻抗控制

多个电机阻抗控制函数 `impedance_control ( )`, 控制指定电机编号的电机目标位置和速度, 其原型及参数解释如表 2-17 和表 2-18。

表2-17 单个电机阻抗控制函数原型说明

函数原型说明	
函数名	<code>impedance_control</code>
函数原型	<code>impedance_control(id_num=0, pos=0, vel=0, tff=0, kp=0, kd=0)</code>
功能描述	单个电机阻抗控制
输入参数	<code>id_num</code> 、 <code>pos</code> 、 <code>vel</code> 、 <code>tff</code> 、 <code>kp</code> 、 <code>kd</code>

表2-18 多个电机扭矩控制函数参数解释说明

序号	参数	解释
1	<code>id_num</code>	需要设置的电机ID编号,如果不知道当前电机ID, 可以用0广播, 如果总线上有多个电机, 则多个电机都会执行该操作。
2	<code>pos</code>	电机目标角度 (度)
3	<code>vel</code>	电机目标速度 (r/min)
4	<code>tff</code>	前馈扭矩 (Nm)
5	<code>kp</code>	刚度系数(Nm/rad)
6	<code>kd</code>	阻尼系数(Nm/rad/s)
备注		阻抗控制为MIT开源方案中的控制模式, 其目标输出扭矩计算公式如下: $\text{torque} = kp * (\text{pos} - \text{pos}_-) + t\_ff + kd * (\text{vel} - \text{vel}_-)$ 其中 <code>pos_-</code> 和 <code>vel_-</code> 分别为输出轴当前实际位置 (degree) 和当前实际速度 (r/min), <code>kp</code> 和 <code>kd</code> 为刚度系数和阻尼系数, 系数比例与MIT等效;

例如，控制 1 号电机转动到 180 度，目标速度和前馈扭矩设置为 0，刚度系数为 0.5，阻尼系数为 0.2：

```
XX.impedance_control(1, 180, 0, 0, 0.5, 0.2)
```

# XX 代表用户定义的对象名称，1 代表电机 ID 号，180 代表电机转动的角度，第一个 0 代表电机目标速度，第二个 0 代表前馈扭矩，0.5 代表刚度系数，0.2 代表阻尼系数。

## 2.10 急停

急停函数 `estop()`，控制电机紧急停止。电机急停后将切换到 IDLE 待机模式，电机卸载并生成 `ERROR_ESTOP_REQUESTED` 错误标志，不再响应 `set_angle/speed/torque` 指令。如果要恢复正常控制模式，需要首先用 `clear_error` 清除错误标志后，然后用 `set_mode` 函数将模式设置为 2（闭环控制模式），其原型及参数解释如表 2-19 和表 2-20。

表2-19 急停函数原型说明

函数原型说明	
函数名	<code>estop</code>
函数原型	<code>estop(id_num=0)</code>
功能描述	急停
输入参数	<code>id_num</code>
返回值	无

表2-20 急停函数参数解释说明

序号	参数	解释
1	<code>id_num</code>	需要急停的电机ID编号,如果不知道当前电机ID, 可以用0广播, 如果总线上有多个电机, 则多个电机都会执行该操作。

例如，控制 1 号电机急停，代码如下：

```
XX.estop(1)
```

# XX 代表用户定义的对象名称，1 表示 1 号电机急停。

上述例子对应的 ROS 命令为 `rostopic pub -1 estop DrEmpower_can/arg_estop "id: 0"`，填入对应参数后敲击回车便可，如图 2-9。（图 2-9 中代码比较长，但一般情况下只需键值 `estop` 后再空一格便可敲击 Tab 键由 ROS 系统将后续参数补足；命令中的 -1 为消息发布频率，可以不写）

```
dr-18000@dr18000-virtual-machine:~$ rostopic pub -1 estop DrEmpower_can/arg_estop "id: 1"
```

图 2-9 使用 ROS 命令对 1 号电机进行急停控制

## 第三章 参数设置功能

### 3.1 设置 ID 号

设置电机 ID 号函数 `set_id( )`，改变电机 ID 号（掉电保存），如果更改电机号后，需要首先用 `clear_error` 清除错误标志后，然后用 `set_mode` 函数将模式设置为 2（闭环控制模式）其原型及参数解释如表 3-1 和表 3-2。

表3-1 设置 ID 号函数原型说明

函数原型说明	
函数名	<code>set_id</code>
函数原型	<code>set_id(id_num=0, new_id=0)</code>
功能描述	设置电机 ID 号函数
输入参数	<code>id_num</code> 、 <code>new_id</code>
返回值	无

表3-2 设置 ID 号函数参数解释说明

序号	参数	解释
1	<code>id_num</code>	需要重新设置编号的电机编号,如果不知道当前电机编号，可以用0广播，但是这时总线上只能连一个电机，否则多个电机会被设置成相同编号
2	<code>new_id</code>	新电机编号，电机ID号范围为1~63

例如，设置 1 号电机为 2，代码如下：

```
XX. set_id(1, 2)
```

# XX 代表用户定义的对象名称，1 代表 1 号电机，2 代表将 ID 设置为 2。

上述例子对应的 ROS 命令为 `rostopic pub -1 set_id DrEmpower_can/arg_set_id "id_old: 0`

`id_new: 0"`，填入对应参数后敲击回车便可，如图 3-1。（图 3-1 中代码比较长，但一般情况下只需键值 `set_id` 后再空一格便可敲击 Tab 键由 ROS 系统将后续参数补足；命令中的 -1 为消息发布频率，可以不写）

```
dr-18000@dr18000-virtual-machine:~$ rostopic pub -1 set_id DrEmpower_can/arg_set_id "id_old: 1
id_new: 2"
```

图 3-1 使用 ROS 命令对指定电机进行急停控制

### 3.2 设置 uart 串口波特率(M)

设置电机串口波特率函数 `set_uart_baud_rate( )`，设置 UART 串口波特率（掉电保存），其原型及参数解释如表 3-3 和表 3-4。

表3-3 设置 uart 波特率函数原型说明

函数原型说明	
函数名	<code>set_uart_baud_rate</code>

函数原型说明	
函数原型	set_uart_baud_rate(id_num=0,baud_rate=115200)
功能描述	设置 UART 串口波特率
输入参数	id_num、 baud_rate
返回值	无

表3-4 设置 uart 波特率函数参数解释说明

序号	参数	解释
1	id_num	需要重新设置波特率的电机编号，如果不知道当前电机编号，可以用0广播
2	baud_rate	uart串口波特率，支持9600、19200、57600、115200中任意一种，修改成功后需手动将主控UART波特率也修改为相同值
备注		这个串口波特率只对UART总线（TX/RX）接口有效，USB接口中的串口为虚拟串口，波特率不用设置，可以自动适应上位机的波特率。

例如，将 1 号电机串口波特率设置为 57600，代码如下：

```
XX.set_uart_baud_rate(1, 57600)
```

# XX 代表用户定义的对象名称，1 代表 1 号电机，57600 代表波特率。

### 3.3 设置 can 波特率

设置电机 CAN 波特率函数 set\_can\_baud\_rate( )，设置 CAN 波特率（掉电保存），其原型及参数解释如表 3-5 和表 3-6。

表3-5 设置 can 波特率函数原型说明

函数原型说明	
函数名	set_can_baud_rate
函数原型	set_can_baud_rate(id_num=0,baud_rate=500000)
功能描述	设置 can 波特率
输入参数	id_num、 baud_rate
返回值	无

表3-6 设置 can 波特率函数参数解释说明

序号	参数	解释
1	id_num	需要重新设置波特率的电机编号，如果不知道当前电机编号，可以用0广播
2	baud_rate	CAN波特率，支持125k、250k、500k、1M中任意一种，修改成功后需手动将主控CAN波特率也修改为相同值

例如，将 1 号电机 CAN 波特率设置为 250000，代码如下：

```
XX.set_can_baud_rate(1,250000)
```

# XX 代表用户定义的对象名称，1 代表 1 号电机，250000 代表 CAN 波特率。

上述例子对应的 ROS 命令为 rostopic pub -1 set\_uart\_baud\_rate DrEmpower\_can/arg\_set\_uart\_baud\_rate "id: 0

baud\_rate: 0", 填入对应参数后敲击回车便可, 如图 3-2。(图 3-2 中代码比较长, 但一般情况下只需键值 set\_uart\_baud\_rate 后再空一格便可敲击 Tab 键由 ROS 系统将后续参数补足; 命令中的 -1 为消息发布频率, 可以不写)

```
dr-18000@dr18000-virtual-machine:~$ rostopic pub -1 set_uart_baud_rate DrEmpower
_can/arg_set_uart_baud_rate "id: 1
baud_rate: 250000"
```

图 3-2 使用 ROS 命令设置 1 号电机的 CAN 通信波特率

### 3.4 设置模式

设置电机模式函数 set\_mode()。设置电机进入不同的控制模式, 其原型及参数解释如表 3-7 和表 3-8。

表3-7 设置模式函数原型说明

函数原型说明	
函数名	set_mode
函数原型	set_mode(id_num=0, mode=2)
功能描述	设置电机模式
输入参数	id_num、mode
返回值	无

表3-8 设置模式函数参数解释说明

序号	参数	解释
1	id_num	需要设置的电机ID编号, 如果不知道当前电机ID, 可以用0广播, 如果总线上有多个电机, 则多个电机都会执行该操作
2	mode	<p>mode = 1: IDLE待机模式, 电机将关掉PWM输出, 电机卸载。</p> <p>mode = 2: 闭环控制模式, set_angle, set_speed, set_torque函数必须在闭环控制模式下才能进行控制。(电机上电后的默认模式)。</p> <p>mode = 3: 电机参数校准模式, 这种模式下会运行电机校准程序, 更新电机参数, 校准后电机将进入IDLE待机模式。</p> <p>mode = 4: 电机编码器校准模式, 这种模式下会运行编码器校准程序, 具体操作为控制电机匀速正转反转各一周, 记录下角度偏差, 并更新相关参数, 校准后电机将进入IDLE待机模式。</p>
备注		模式3和模式4是用来校准电机和编码器参数, 出厂前已完成校准, 正常情况下不要使用

例如, 将 1 号电机设置为模式 2, 代码如下:

```
XX.set_mode(1,2)
```

# XX 代表用户定义的对象名称, 1 代表 1 号电机, 2 代表电机闭环控制模式。

上述例子对应的 ROS 命令为 rostopic pub -1 set\_mode DrEmpower\_can/arg\_set\_mode "id: 0

mode: 0", 填入对应参数后敲击回车便可, 如图 3-3。(图 3-3 中代码比较长, 但一般情况下只需键值 set\_mode 后再空一格便可敲击 Tab 键由 ROS 系统将

后续参数补足；命令中的-1 为消息发布频率，可以不写）

```
dr-18000@dr18000-virtual-machine:~$ rostopic pub -1 set_mode DrEmpower_can/arg_set_mode "id: 1
mode: 2"
```

图 3-3 使用 ROS 命令设置 1 号电机设置为闭环控制模式

### 3.5 设置零点

设置电机零点位置函数 `set_zero_position()`，设置当前位置为电机输出轴零点，设置完后当前位置为 0 度，其原型及参数解释如表 3-9 和表 3-10。

表3-9 设置零点函数原型说明

函数原型说明	
函数名	<code>set_zero_position</code>
函数原型	<code>set_zero_position(id_num=0)</code>
功能描述	设置当前位置为零点位置
输入参数	<code>id_num</code>
返回值	无

表3-10 设置零点函数参数解释说明

序号	参数	解释
1	<code>id_num</code>	需要设置的电机ID编号，如果不知道当前电机ID，可以用0广播，如果总线上有多个电机，则多个电机都会执行该操作

例如，将 1 号电机当前位置设为零点，代码如下：

```
XX. set_zero_position(1)
```

# XX 代表用户定义的对象名称，1 代表 1 号电机。

上述例子对应的 ROS 命令为 `rostopic pub -1 set_zero_position DrEmpower_can/arg_set_zero_position "id: 0"`，填入对应参数后敲击回车便可，如图 3-4。（图 3-4 中代码比较长，但一般情况下只需键值 `set_zero_position` 后再空一格便可敲击 Tab 键由 ROS 系统将后续参数补足；命令中的-1 为消息发布频率，可以不写）

```
dr-18000@dr18000-virtual-machine:~$ rostopic pub -1 set_zero_position DrEmpower_can/arg_set_zero_position "id: 1"
```

图 3-4 使用 ROS 命令设置 1 号电机当前位置为零点位置

### 3.6 设置 GPIO 控制接口模式(M)

设置电机预留的两个引脚模式，支持的模式有 UART 串口和 Step/Dir 脉冲，通过调用 `set_GPIO_mode()`函数进行切换并设置基本参数，其原型及参数解释如表 3-11 和表 3-12。

表3-11 设置 GPIO 控制接口模式函数原型说明



函数原型说明	
函数名	set_GPIO_mode
函数原型	set_GPIO_mode(id_num=0, mode='tx_rx', param=115200)
功能描述	设置 GPIO 控制接口模式
输入参数	id_num、mode、param
返回值	无

表3-12 设置 GPIO 控制接口模式函数参数解释说明

序号	参数	解释
1	id_num	需要修改的电机ID编号,如果不知道当前电机ID, 可以用0广播, 如果总线上有多个电机, 则多个电机都会执行该操作
2	mode	Python: 'tx_rx'表示选择uart串口模式; 'step_dir'表示选择Step/Dir接口控制模式。 C/C++: mode == 0 表示选择uart串口模式; mode == 1表示选择Step/Dir接口控制模式。
3	param	当选择uart串口模式时, param表示串口的波特率, 支持9600、19200、57600、115200其中一种; 当选择Step/Dir接口控制模式时, param表示电机内部转子转一圈对应的脉冲数, 支持1-1024(必须是整数)。

例如, 设置 1 号电机 GPIO 控制接口模式为 Step/Dir 接口控制模式, 电机内部转子转一圈对应的脉冲数为 1024, 代码如下:

```
XX.set_GPIO_mode(1, 'step_dir',1024)
```

# XX 代表用户定义的对象名称, 1 代表 1 号电机, 'step\_dir'代表电机 GPIO 控制接口模式为 Step/Dir 接口控制模式, 1024 代表电机内部转子转一圈对应的脉冲数。

用户也可以使用上位机更改电机的 GPIO 控制接口模式, 当电机为 Step/Dir 接口控制模式时, 用户可以按照引脚定义图把电机的 Step/Dir 接口以及 GND 与单片机的 GPIO 以及 GND 进行连接。Step/Dir 接口控制模式可以使用高低电频来控制电机转动, Dir 接口控制电机转动方向, Step 接口控制步长。

**注意:**

GPIO 控制接口模式为 Step/Dir 接口控制模式时, 脉冲数 param 表示电机内部转子转动一周的脉冲数, 若需要电机外部转盘转动一周则需要 (脉冲数 X 电机减速比)

例如: 设置电机内部转子转一圈对应的脉冲数为 1024, 则电机外部转盘转一圈所需的脉冲数为 1024X50=51200。所以若需要电机外部转盘转动一圈, 需要单片机与 Step 接口相连的 GPIO 口发送 51200 个高低电频。

### 3.7 设置电机软件限位极限位置

设置电机软件限位极限位置 set\_angle\_range( ), 设置电机预输出轴软件限位极限位置值, 设置成功后电机在位置、速度及扭矩控制模式电机输出轴将被限制



在[angle\_min, angle\_max]范围内，其原型及参数解释如表 3-13 和表 3-14。

表3-13 设置电机软件限位极限位置函数原型说明

函数原型说明	
函数名	set_angle_range
函数原型	set_angle_range(id_num=0, angle_min=-360, angle_max=360)
功能描述	设置电机软件限位极限位置
输入参数	id_num、angle_min、angle_max
返回值	True、False

表3-14 设置电机软件限位极限位置函数参数解释说明

序号	参数	解释
1	id_num	需要修改的电机ID编号，如果不知道当前电机ID，可以用0广播，如果总线上有多个电机，则多个电机都会执行该操作
2	angle_min	软件限位最小角度（该参数与axis0.output_shaft.circular_setpoint_min对应）
3	angle_max	软件限位最大角度（该参数与axis0.output_shaft.circular_setpoint_max对应）

例如，设置 1 号电机软件限位最小角度为 0，软件限位最大角度为 120°，代码如下：

```
XX. set_angle_range (1, 0, 120)
```

# XX 代表用户定义的对象名称，1 代表 1 号电机，0 代表软件限位最小角度，120 代表软件限位最大角度。

**注意：**

当前输出轴位置必须在[angle\_min, angle\_max]范围内，否则将设置失败

### 3.8 设置梯形轨迹模式下速度曲线类型

设置梯形轨迹模式速度曲线类型 set\_traj\_mode()，可以选择梯形速度轨迹和 S 形速度轨迹（两种曲线效果如下图所示，适用于位置控制-梯形轨迹模式），其原型及参数解释如

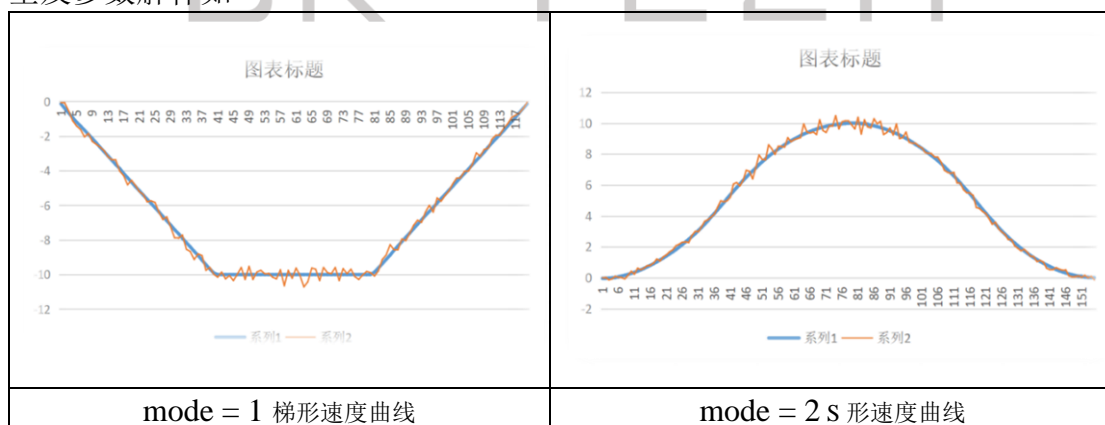


表3-15 设置梯形轨迹模式函数原型说明

函数原型说明	
函数名	set_traj_mode
函数原型	set_traj_mode(id_num=0, mode=1)

功能描述	设置梯形轨迹模式 set_traj_mode()
输入参数	id_num、mode
返回值	无

表3-16 设置梯形轨迹模式函数参数解释说明

序号	参数	解释
1	id_num	需要修改的电机ID编号, 如果不知道当前电机ID, 可以用0广播, 如果总线上有多个电机, 则多个电机都会执行该操作
2	mode	mode: 1表示梯形轨迹模式, 2表示S形轨迹模式 (指的是速度曲线形状, 适用于位置控制-梯形轨迹模式)

例如, 将 1 号电机的梯形轨迹模式修改为模式 2-S 型轨迹模式代码如下:

```
XX. set_traj_mode (1,2)
```

# XX 代表用户定义的对象名称, 1 代表 1 号电机, 2 代表将速度曲线修改为 S 型轨迹

### 3.9 设置指定参数

修改电机属性函数 write\_property(), 这里的属性参数为电机控制参数, 其原型及参数解释如表 3-17 和表 3-18。

表3-17 设置指定参数函数原型说明

函数原型说明	
函数名	write_property
函数原型	write_property(id_num=0,property="", value=0)
功能描述	修改电机属性参数
输入参数	id_num、property、value
返回值	无

表3-18 设置指定参数函数参数解释说明

序号	参数	解释
1	id_num	需要修改的电机ID编号, 如果不知道当前电机ID, 可以用0广播, 如果总线上有多个电机, 则多个电机都会执行该操作
2	property	需要读取的属性参数名称, 例如"vbus_voltage", "axis0.config.can_node_id"等, 具体参数名称见附录表格中的参数名及其地址
3	value	对应参数的目标值

例如, 设置 1 号电机最大限制速度为 30, 代码如下:

```
XX. write_property (1,"axis0.controller.config.vel_limit",30)
```

# XX 代表用户定义的对象名称, 1 代表 1 号电机, axis0.controller.config.vel\_limit 代表电机最大速度限制参数名, 30 代表将其设置为 30r/min。

上述例子对应的 ROS 命令为 rostopic pub -1 write\_property DrEmpower\_can/arg\_write\_property "id: 0

```
property: "
```

value: 0", 填入对应参数后敲击回车便可, 如图 3-5。(图 3-5 中代码比较长, 但一般情况下只需键值 write\_property 后再空一格便可敲击 Tab 键由 ROS 系

系统将后续参数补足；命令中的-1 为消息发布频率，可以不写）

```
dr-18000@dr18000-virtual-machine:~$ rostopic pub -1 write_property DrEmpower_can  
/arg_write_property "id: 1  
property: 'axis0.controller.config.vel_limit'  
value: 30"
```

图 3-5 使用 ROS 命令设置 1 号电机的最大速度限制为 30r/min



## 第四章 参数回读功能

## 4.1 获取 ID 号

读取电机 ID 号函数 `get_id( )`，其原型及参数解释如表 4-1 和表 4-2。

表4-1 获取 ID 号函数原型说明

函数原型说明	
函数名	<code>get_id</code>
函数原型	<code>get_id(id_num=0)</code>
功能描述	获取电机 ID 号函数
输入参数	<code>id_num</code>
返回值	无

表4-2 获取 ID 号函数参数解释说明

序号	参数	解释
1	<code>id_num</code>	需要读取的电机编号，如果不知道当前电机编号，可以用0广播，但是这时总线上只能连一个电机，否则将报错

例如，获取当前电机 ID 号，代码如下：

```
XX.get_id(0)
```

# XX 代表用户定义的对象名称，将会返回当前电机的 ID 号。

上述例子对应的 ROS 命令为 `rostopic pub -1 get_id DrEmpower_can/arg_get_id "id: 0"`，填入对应参数后敲击回车便可，如图 4-1。（图 4-1 中代码比较长，但一般情况下只需键值 `get_id` 后再空一格便可敲击 Tab 键由 ROS 系统将后续参数补足；命令中的 -1 为消息发布频率，可以不写）

```
dr-18000@dr18000-virtual-machine:~$ rostopic pub -1 get_id DrEmpower_can/arg_get_id "id: 0"
```

图 4-1 使用 ROS 命令读取当前电机 ID 号

## 4.2 读取位置和速度

读取电机的当前位置和速度读取电机当前位置和速度列表函数 `get_state( )`，单位分别为转（r）和转每分钟(r/min)，其原型及参数解释如表 4-3 和表 4-4。

在最新版本库函数和电机固件中，该函数已改为**实时状态快速读取接口**（`address=0x00`），电机的实时状态会自动保存在库函数全局变量 `motor_state` 数组中，同时保持了该函数的返回值不变，仍是当前位置和当前速度列表。

（必须保证全局变量 `MOTOR_NUM` 大于或等于最大的电机 ID 号）

表4-3 读取位置和速度函数原型说明

函数原型说明	
函数名	<code>get_state</code>

函数原型	get_state(id_num=0)
功能描述	读取电机当前角度及速度
输入参数	id_num
返回值	Python: [pos, vel] : 位置和速度列表 C/C++: struct servo_state 储存位置和速度的结构体

表4-4 读取位置和速度函数参数解释说明

序号	参数	解释
1	id_num	需要读取的电机编号, 如果不知道当前电机编号, 可以用0广播, 但是这时总线上只能连一个电机, 否则将报错

例如, 获取 1 号电机的速度和位置, 代码如下:

```
XX.get_state(1)
```

# XX 代表用户定义的对象名称, 将会返回 1 号电机位置和速度组成的列表。

上述例子对应的 ROS 命令为 `rostopic pub -1 get_state DrEmpower_can/arg_get_state "id: 0"`, 填入对应参数后敲击回车便可, 如图 4-2。(图 4-2 中代码比较长, 但一般情况下只需键值 `get_state` 后再空一格便可敲击 Tab 键由 ROS 系统将后续参数补足; 命令中的 -1 为消息发布频率, 可以不写)

```
dr-18000@dr18000-virtual-machine:~$ rostopic pub -1 get_state DrEmpower_can/arg_get_state "id: 1"
```

图 4-2 使用 ROS 命令读取 1 号电机角度和转速

### 4.3 读取电压和电流

读取电机的当前位置和速度函数 `get_volcur()`, 读取电机当前电压和 q 轴电流列表, 单位分别为伏 (V) 和安 (A), 其原型及参数解释如表 4-5 和表 4-6。

表4-5 读取电压和电流函数原型说明

函数原型说明	
函数名	get_volcur
函数原型	get_volcur (id_num=0)
功能描述	读取电机当前电压及电流
输入参数	id_num
返回值	[vol, cur] : 电压和电流列表 C/C++: struct servo_volcur 储存电压和电流的结构体

表4-6 读取电压和电流函数参数解释说明

序号	参数	解释
1	id_num	需要读取的电机编号, 如果不知道当前电机编号, 可以用0广播, 但是这时总线上只能连一个电机, 否则将报错

例如, 读取 1 号电机的电流和电压, 代码如下:

```
XX.get_volcur(1)
```

# XX 代表用户定义的对象名称,将会返 1 号电机由电压和电流组成的列表。

上述例子对应的 ROS 命令为 `rostopic pub -1 get_volcur DrEmpower_can/arg_get_volcur "id: 0"`, 填入对应参数后敲击回车便可, 如图 4-3。(图 4-3 中代码比较长, 但一般情况下只需键值 `get_volcur` 后再空一格便可敲击 Tab 键由 ROS 系统将后续参数补足; 命令中的 -1 为消息发布频率, 可以不写)

```
dr-18000@dr18000-virtual-machine:~$ rostopic pub -1 get_volcur DrEmpower_can/arg_get_volcur "id: 1"
```

图 4-3 使用 ROS 命令读取 1 号电机角度和转速

#### 4.4 读取 GPIO 控制接口模式(M)

读取电机预留的两个引脚当前模式及参数函数 `get_GPIO_mode()`, 支持的模式有 UART 串口和 Step/Dir 接口, 其原型及参数解释如表 4-7 和表 4-8。

表4-7 读取 GPIO 控制接口模式函数原型说明

函数原型说明	
函数名	<code>get_GPIO_mode</code>
函数原型	Python: <code>get_GPIO_mode(id_num=0)</code> C/C++: <code>get_GPIO_mode(uint8_t id_num, uint8_t *enable_uart, uint8_t *enable_step_dir, uint32_t *n)</code>
功能描述	读取电机当前模式及参数
输入参数	<code>id_num</code>
返回值	无

表4-8 读取 GPIO 控制接口模式函数参数解释说明

序号	参数	解释
1	<code>id_num</code>	需要读取的电机编号,如果不知道当前电机编号, 可以用0广播, 但是这时总线上只能连一个电机, 否则将报错
2	<code>enable_uart</code>	是否为 uart 模式 ( <a href="#">Python函数库中没有该参数</a> )
3	<code>enable_step_dir</code>	是否为 step dir 模式 ( <a href="#">Python函数库中没有该参数</a> )
4	<code>n</code>	串口波特率或脉冲数/每圈 ( <a href="#">Python函数库中没有该参数</a> )

例如, 读取 1 号电机的当前模式及参数, 代码如下:

```
XX. get_GPIO_mode(1)
```

# XX 代表用户定义的对象名称, 将会显示 1 号电机当前模式及参数。

#### 4.5 读取指定参数

读取电机属性参数函数 `read_property()`, 读取电机属性参数, 这里的属性参数包括电机状态量及电机控制参数, 其原型及参数解释如表 4-9 和表 4-10。

表4-9 读取指定参数函数原型说明

函数原型说明
--------

函数名	read_property
函数原型	read_property(id_num=0,property="")
功能描述	读取电机属性参数
输入参数	id_num、 property
返回值	Value: 返回对应属性参数的值

表4-10 读取指定参数函数参数解释说明

序号	参数	解释
1	id_num	需要读取的电机编号,如果不知道当前电机编号,可以用0广播,但是这时总线上只能连一个电机,否则将报错
2	property	需要读取的属性参数名称,例如"vbus_voltage", "axis0.config.can_node_id"等,具体参数名称见附录表格中的参数名及其地址

例如,读取1号电机电压值

XX. read\_property (1,"vbus\_voltage")

# XX 代表用户定义的对象名称,1为电机ID号,vbus\_voltage为总线电压值参数名

上述例子对应的ROS命令为 rostopic pub -1 read\_property DrEmpower\_can /arg\_read\_property "id: 0

property: """,填入对应参数后敲击回车便可,如图4-4。(图4-4中代码比较长,但一般情况下只需键值 read\_property 后再空一格便可敲击 Tab 键由ROS系统将后续参数补足;命令中的-1为消息发布频率,可以不写)

```
dr-18000@dr18000-virtual-machine:~$ rostopic pub -1 read_property DrEmpower_can/
arg_read_property "id: 1
property: 'vbus_voltage'"
```

图4-4 使用ROS命令读取1号电机总线电压值



## 第五章 辅助功能

## 5.1 清除错误标志

清除错误标志函数 `clear_error()`，一旦电机运行过程中出现任何错误，电机将进入 IDLE 模式，如果要恢复正常控制模式，需要首先用 `clear_error` 清除错误标志后,然后用 `set_mode` 函数将模式设置为 2（闭环控制模式），其原型及参数解释如表 5-1 和表 5-2。

表5-1 清除错误标志函数原型说明

函数原型说明	
函数名	<code>clear_error</code>
函数原型	<code>clear_error(id_num=0)</code>
功能描述	清除电机错误标志
输入参数	<code>id_num</code>
返回值	无

表5-2 清除错误标志函数参数解释说明

序号	参数	解释
1	<code>id_num</code>	需要清除错误标志的电机ID编号，如果不知道当前电机ID，可以用0广播，如果总线上有多个电机，则多个电机都会执行该操作

例如，清除 1 号电机的错误标志，代码如下：

```
XX.clear_error(1)
```

# XX 代表用户定义的对象名称，将会清除 1 号电机的所有错误标志

上述例子对应的 ROS 命令为 `rostopic pub -1 clear_error DrEmpower_can/arg_clear_error "id: 0"`，填入对应参数后敲击回车便可，如图 5-1。（图 5-1 中代码比较长，但一般情况下只需键值 `clear_error` 后再空一格便可敲击 Tab 键由 ROS 系统将后续参数补足；命令中的 -1 为消息发布频率，可以不写）

```
dr-18000@dr18000-virtual-machine:~$ rostopic pub -1 clear_error DrEmpower_can/arg_clear_error "id: 1"
```

图 5-1 使用 ROS 命令清除 1 号电机错误

## 5.2 显示错误标志

打印电机错误编号函数 `dump_error()`，读取电机错误信息编码，如果错误编码为 0，表示无异常。如果错误编码不为 0，则表示存在故障。其原型及参数解释如表 5-3 和表 5-4。

表5-3 显示错误标志函数原型说明

函数原型说明	
函数名	<code>dump_error</code>



函数原型说明	
函数原型	Python: dump_error(id_num=0) C/C++: dump_error(uint8_t id_num, char *error)
功能描述	显示电机错误标志
输入参数	id_num
返回值	无

表5-4 显示错误标志函数参数解释说明

序号	参数	解释
1	id_num	需要读取的电机编号，如果不知道当前电机编号，可以用0广播，但是这时总线上只能连一个电机，否则将报错
2	error	错误信息 (Python函数库中没有该参数)

例如，打印 1 号电机的错误标志，代码如下：

```
XX.dump_error(1)
```

# XX 代表用户定义的对象名称，会打印出 1 号电机的错误信息

上述例子对应的 ROS 命令为 `rostopic pub -1 dump_error DrEmpower_can/arg_dump_error "id: 0"`，填入对应参数后敲击回车便可，如图 5-2。（图 5-2 中代码比较长，但一般情况下只需键值 `dump_error` 后再空一格便可敲击 Tab 键由 ROS 系统将后续参数补足；命令中的 -1 为消息发布频率，可以不写）

```
dr-18000@dr18000-virtual-machine:~$ rostopic pub -1 dump_error DrEmpower_can/arg_dump_error "id: 1"
```

图 5-2 使用 ROS 命令打印 1 号电机错误标志

### 5.3 保存电机配置

保存电机配置函数 `save_config()`，正常情况下，通过 `write_property` 修改的属性电机上电重启之后，会恢复为修改前的值，如果想永久保存，则需要用 `save_config` 函数将相关参数保存到 flash 中，掉电不丢失。其原型及参数解释如表 5-5 和表 5-6。

表5-5保存电机配置函数原型说明

函数原型说明	
函数名	save_config
函数原型	save_config(id_num=0)
功能描述	保存电机配置
输入参数	id_num
返回值	无

表5-6保存电机配置函数参数解释说明

序号	参数	解释
1	id_num	需要读取的电机编号，如果不知道当前电机编号，可以用0广播，但是这时总线上只能连一个电机，否则将报错

## 5.4 电机重启

电机重启函数 `reboot()`，使电机软件重启，效果与重新上电类似。其原型及参数解释如和。

表5-7电机重启函数原型说明

函数原型说明	
函数名	<code>reboot</code>
函数原型	<code>reboot(id_num=0)</code>
功能描述	电机重启
输入参数	<code>id_num</code>
返回值	无

表5-8电机重启函数参数解释说明

序号	参数	解释
1	<code>id_num</code>	需要读取的电机编号,如果不知道当前电机编号,可以用0广播,但是这时总线上只能连一个电机,否则将报错

## 5.5 单个电机等待函数(等待单个电机运动到目标角度)

单个电机等待函数 `position_done()`，延时等待直到给定电机到达目标位置(只对角度位置控制指令有效)，其原型及参数解释如下。

表5-9单个电机等待函数原型说明

函数原型说明	
函数名	<code>position_done</code>
函数原型	<code>position_done(id_num=0)</code>
功能描述	单个电机等待
输入参数	<code>id_num</code>
返回值	无

表5-10 单个电机等待函数参数解释说明

序号	参数	解释
1	<code>id_num</code>	需要读取的电机编号,如果不知道当前电机编号,可以用0广播,但是这时总线上只能连一个电机,否则将报错

## 5.5 多个电机等待函数(等待多个电机运动都到目标角度)

多个电机等待函数 `positions_done()`，程序等待（阻塞）直到所有电机都到达目标位置(只对角度控制指令有效)，其原型及参数解释如下。

表5-11 多个电机等待函数原型说明

函数原型说明	
函数名	<code>positions_done</code>
函数原型	<code>positions_done(id_list=[1, 2, 3],n)</code>
功能描述	多个电机等待函数
输入参数	<code>id_list</code>
返回值	无

表5-12 多个电机等待函数参数解释说明

序号	参数	解释
1	id_list	电机编号组成的列表
2	n	id_list 和 torque_list 的长度 (Python函数库中没有该参数)



## 附录：常用参数及其地址表

常用参数及其地址表

属性名称	中文解释	Address	读写性（未注明即 可读可写）	是否 可修改	单位	类型	适用 系列
vbus_voltage	总线电压	1	只读		V	float32	
ibus	总线电流	2	只读		A	float32	
serial_number	硬件序列号	3~4	只读			uint64	
hw_version	硬件版本号	5	只读			Uint32	
fw_version	固件版本号	6	只读			Uint32	
config.uart_baudrate	UART 波特率	10001		可修改		uint32	M
config.max_regen_current	最大再生电流	10002				float32	
config.dc_bus_undervoltage_trip_level	设置低电压报警阈值	10003			V	float32	
config.dc_bus_overvoltage_trip_level	设置过压报警阈值	10004			V	float32	
config.dc_max_positive_current	供电设备能够输出的最大电流	10005			A	float32	
config.dc_max_negative_current	供电设备能够反向吸收的最大电 流	10006			A	float32	
config.brake_resistance	刹车电阻阻值	10007			Ω	float32	M
config.enable_uart	UART 串口使能	10008		可修改		bool	M
can.error	CAN 错误代码	20001	只读			uint32	
can.config.baud_rate	CAN 通信波特率	21001		可修改		uint32	
can.config.protocol	CAN 通信协议选择	21002				uint32	
can.config.timeout_ms	CAN 节点保护超时时间	21003		可修改		uint32	
can.config.heartbeat_ms	CAN 节点心跳发送周期	21003		可修改		uint32	
axis0.error	错误代码	30001	只读			uint32	
axis0.current_state	当前状态	30002	只读			uint32	
axis0.requested_state	命令 axis0 进入某个状态	30003				uint32	
axis0.config.can_node_id	can 总线节点 ID(电机 ID)	31001		可修改		uint32	
axis0.config.startup_motor_calibration	开机电机校准使能	31002				bool	M
axis0.config.startup_encoder_index_search	开机自动寻找零点使能	31003		可修改		bool	G
axis0.config.startup_encoder_offset_calibration	开机编码器校准使能	31004				bool	M
axis0.config.startup_closed_loop_control	开机自动进入闭环控制模式	31005				bool	M
axis0.config.enable_step_dir	Step/Dir 脉冲方向控制使能	31006		可修改		bool	M
axis0.config.turns_per_step	一个脉冲对应的电机转子圈数	31007			turn/step	float	M
axis0.config.calibration_lockin.current	编码器校准时电机电流	31101			A	float32	

属性名称	中文解释	Address	读写性（未注明即可读可写）	是否可修改	单位	类型	适用系列
axis0.config.calibration_lockin.ramp_time	编码器校准时电机电流上升到给定值所需时间	31102			s	float32	M
axis0.config.calibration_lockin.ramp_distance	编码器校准时电机转子转动角度	31103			rad	float32	M
axis0.config.calibration_lockin.accel	编码器校准时爬升加速度	31104			rad/s^2	float32	M
axis0.config.calibration_lockin.vel	编码器校准时转子转速	31105			rad/s	float32	M
axis0.config.extra_setting.enable_circular_setpoint_limit	是否启用软件限位	31202		可修改		bool	
axis0.config.extra_setting.circular_setpoint_min	软件限位电机转子最小极限位置	31203			turn	float32	
axis0.config.extra_setting.circular_setpoint_max	软件限位电机转子最小极限位置	31204			turn	float32	
axis0.config.extra_setting.zero_offset_counts	编码器零点偏差	31205			count	float32	
axis0.config.extra_setting.stored_turns_counts	保留位	31206				int32	
axis0.config.extra_setting.enable_multi_circle	是否多圈计数功能	31207		可修改		bool	
axis0.config.extra_setting.gear_ratio	电机减速比	31208				float32	
axis0.config.extra_setting.stall_current_limit	堵转保护电流阈值	31209		可修改	A	float32	
axis0.config.extra_setting.stall_time_limit	堵转保护时间阈值	31210		可修改	s	float32	
axis0.config.extra_setting.enable_stall_limit	是否启用堵转保护	31211		可修改		bool	
axis0.config.extra_setting.encoder_offset_calibrated	编码器是否已标定	31213				bool	
axis0.config.extra_setting.enable_reply_state	是否启用控制指令实时状态回读	31214		可修改		bool	
axis0.config.extra_setting.enable_damp_idle	是否启用待机模式阻尼功能	31215		可修改		bool	
axis0.controller.error	控制器错误代码	32001	只读			uint32	
axis0.controller.input_pos	输入的电机目标位置	32002			turn	float32	
axis0.controller.input_vel	输入的电机目标转速	32003			turn/s	float32	
axis0.controller.input_torque	输入的电机输出的力矩大小	32004			Nm	float32	
axis0.controller.pos_setpoint	电机目标位置	32005	只读		turn	float32	
axis0.controller.vel_setpoint	电机目标转速	32006	只读		turn/s	float32	
axis0.controller.torque_setpoint	电机目标电流	32007	只读		Nm	float32	
axis0.controller.trajjectory_done	角度控制电机是否到达目标位置	32008	只读			bool	
axis0.controller.config.enable_gain_scheduling	是否启用增益规划功能	32101				bool	M
axis0.controller.config.enable_vel_limit	是否启用转速限制功能	32102				bool	
axis0.controller.config.enable_current_mode_vel_limit	是否在电流模式下启用速度限制	32103		可修改		bool	
axis0.controller.config.enable_overspeed_error	当转速超过设置限制值时是否停止电机并报错	32104				bool	

属性名称	中文解释	Address	读写性（未注明即可读可写）	是否可修改	单位	类型	适用系列
axis0.controller.config.control_mode	控制模式	32105				unit32	
axis0.controller.config.input_mode	输入模式	32106				unit32	
axis0.controller.config.gain_scheduling_width	速度环增益规划功能启用的位置误差范围	32107			turn	float32	M
axis0.controller.config.pos_gain	位置环增益	32108		可修改	(turn/s)/turn	float32	
axis0.controller.config.vel_gain	速度环增益	32109		可修改	Nm/(turn/s)	float32	
axis0.controller.config.vel_integrator_gain	速度环积分增益	32110		可修改	Nm/(turn/s^2)	float32	
axis0.controller.config.vel_limit	最大转速	32111			turn/s	float32	
axis0.controller.config.vel_limit_tolerance	最大转速波动容忍度	32112				float32	
axis0.controller.config.vel_ramp_rate	转速爬升时得爬升速率	32113			turn/s^2	float32	
axis0.controller.config.torque_ramp_rate	扭矩爬升时得爬升速率	32114			Nm/s	float32	
axis0.controller.config.circular_setpoints	是否启用环形控制模式	32115				bool	M
axis0.controller.config.circular_setpoint_range	环形控制模式下可输入转动范围	32116			turn	float32	M
axis0.controller.config.inertia	电机转动惯量	32117			Nm/(turn/s^2)	float32	
axis0.controller.config.input_filter_bandwidth	轨迹跟踪模式角度输入滤波带宽	32118		可修改	1/s	float32	
axis0.motor.error	电机错误代码	33001				uint32	
axis0.motor.is_calibrated	电机是否已校准	33002	只读			bool	M
axis0.motor.effective_current_lim	实际电流限制值（由于温度升高或设定的最大电流而计算得到）	33003	只读		A	float32	M
axis0.motor.config.pre_calibrated	电机是否已成功校准	33101				bool	M
axis0.motor.config.pole_pairs	电机极对数	33102				int32	
axis0.motor.config.calibration_current	电机校准时的电流大小	33103			A	float32	M
axis0.motor.config.resistance_calib_max_voltage	电机校准自动检测相电阻时最大电压	33104			V	float32	
axis0.motor.config.phase_inductance	电机相电感	33105			H	float32	
axis0.motor.config.phase_resistance	电机相电阻	33106			$\Omega$	float32	
axis0.motor.config.torque_constant	电机（转子）扭矩常数	33107			Nm/A	float32	
axis0.motor.config.direction	电机运行方向	33108				int32	
axis0.motor.config.motor_type	电机类型	33109				uint32	
axis0.motor.config.current_lim	电机最大运行电流	33110			A	float32	

属性名称	中文解释	Address	读写性（未注明即 可读可写）	是否 可修改	单位	类型	适用 系列
axis0.motor.config.current_lim_margin	最大运行电流超出容忍度	33111				float32	M
axis0.motor.config.torque_lim	电机（转子）输出最大扭矩	33112			Nm	float32	
axis0.motor.config.requested_current_range	电流最大采样阈值	33113			A	float32	M
axis0.motor.config.current_control_bandwidth	电流控制环的控制带宽	33114		可修改		float32	
axis0.motor.current_control.Ibus	电机母线电流	33201			A	float32	
axis0.motor.current_control.final_v_alpha	最终输出到 SVM 的 Valpha	33202			V	float32	
axis0.motor.current_control.final_v_beta	最终输出到 SVM 的 Vbeta	33203			V	float32	
axis0.motor.current_control.Id_setpoint	电流环控制输入的目标直轴电流	33204			A	float32	
axis0.motor.current_control.Iq_setpoint	电流环控制输入的目标交轴电流	33205	只读		A	float32	
axis0.motor.current_control.Iq_measured	通过电流采样获取的交轴电流	33206	只读		A	float32	
axis0.motor.current_control.Id_measured	通过电流采样获取的直轴电流	33207	只读		A	float32	
axis0.encoder.error	编码器错误代码	34001	只读			uint32	
axis0.encoder.is_ready	编码器是否准备就绪	34002	只读			bool	M
axis0.encoder.index_found	是否检测到编码器索引信号	34003	只读			bool	M
axis0.encoder.pos_estimate	当前电机（转子）实际位置	34004	只读		turn	float32	
axis0.encoder.pos_circular	当前约束在 cpr 范围内的位置	34005	只读		turn	float32	M
axis0.encoder.vel_estimate	当前电机（转子）实际转速	34006	只读		turn/s	float32	
axis0.encoder.spi_error_rate	spi 通信错误率	34007	只读			float32	
axis0.encoder.battery_voltage	编码器内置电池电压	34008	只读		V	float32	
axis0.encoder.config.mode	编码器类型	34101				mode	M
axis0.encoder.config.use_index	是否使用 Z 轴信号	34102				bool	M
axis0.encoder.config.cpr	编码器转动每圈脉冲数	34103			count	int32	
axis0.encoder.config.pre_calibrated	编码器是否事先已被校准	34104				bool	
axis0.encoder.config.bandwidth	编码器更新带宽	34105				float32	
axis0.encoder.config.abs_spi_cs_gpio_pin	编码器 SPI 片选 GPIO 引脚号	34106				uint32	M
axis0.trap_traj.config.vel_limit	梯形轨迹模式下速度限制	35101			turn/s	float32	
axis0.trap_traj.config.accel_limit	梯形轨迹模式下加速加速度限制	35102			turn/(s^2)	float32	
axis0.trap_traj.config.decel_limit	梯形轨迹模式下减速加速度限制	35103			turn/(s^2)	float32	
axis0.trap_traj.config.traj_mode	梯形轨迹模式下速度轨迹类型	35104		可修改		uint32	
axis0.fet_thermistor.error	电机驱动板过温保护错误标志	36001	只读			uint32	
axis0.fet_thermistor.temperature	电机驱动板当前温度	36002	只读		°	float32	



属性名称	中文解释	Address	读写性（未注明即可读可写）	是否可修改	单位	类型	适用系列
axis0.fet_thermistor.config.enabled	电机驱动板过温保护使能	36101				bool	M
axis0.fet_thermistor.config.temp_limit_lower	电机驱动板过温保护最低温度	36102		可修改	°	float32	
axis0.fet_thermistor.config.temp_limit_upper	电机驱动板过温保护最高温度	36103		可修改	°	float32	
axis0.motor_thermistor.error	电机过温保护错误标志	37001	只读			unit32	
axis0.motor_thermistor.temperature	电机当前温度	37002	只读		°	float32	
axis0.motor_thermistor.config.enabled	电机过温保护使能	37101				bool	M
axis0.motor_thermistor.config.temp_limit_lower	电机过温保护最低温度	37102		可修改	°	float32	
axis0.motor_thermistor.config.temp_limit_upper	电机过温保护最高温度	37103		可修改	°	float32	
axis0.output_shaft.input_pos	电机输出轴输入目标角度	38001			degree	float32	
axis0.output_shaft.input_vel	电机输出轴输入目标速度	38002			r/min	float32	
axis0.output_shaft.input_torque	电机输出轴输入目标扭矩	38003			Nm	float32	
axis0.output_shaft.pos_setpoint	电机输出轴当前目标角度	38004	只读		degree	float32	
axis0.output_shaft.vel_setpoint	电机输出轴当前目标速度	38005	只读		r/min	float32	
axis0.output_shaft.torque_setpoint	电机输出轴当前目标扭矩	38006	只读		Nm	float32	
axis0.output_shaft.pos_estimate	电机输出轴当前实际角度	38007	只读		degree	float32	
axis0.output_shaft.vel_estimate	电机输出轴当前实际速度	38008	只读		r/min	float32	
axis0.output_shaft.torque_estimate	电机输出轴当前实际扭矩	38009	只读		Nm	float32	
axis0.output_shaft.circular_setpoint_min	电机输出轴软件限位最小角度	38010		可修改	degree	float32	
axis0.output_shaft.circular_setpoint_max	电机输出轴软件限位最大角度	38011		可修改	degree	float32	
axis0.output_shaft.vel_limit	电机输出轴最大速度限制	38012		可修改	r/min	float32	
axis0.output_shaft.torque_lim	电机输出轴最大扭矩限制	38013		可修改	Nm	float32	
axis0.output_shaft.trap_traj_vel_limit	梯形轨迹模式下电机输出轴最大速度限制	38014		可修改	r/min	float32	
axis0.output_shaft.trap_traj_accel_limit	梯形轨迹模式下电机输出轴最大加速加速度限制	38015		可修改	r/(min*s)	float32	
axis0.output_shaft.trap_traj_decel_limit	梯形轨迹模式下电机输出轴最大减速加速度限制	38016		可修改	r/(min*s)	float32	
axis0.output_shaft.vel_ramp_rate	电机输出轴速度上升速率	38017		可修改	r/(min*s)	float32	
axis0.output_shaft.torque_ramp_rate	电机输出轴扭矩上升速率	38018		可修改	Nm/s	float32	
axis0.output_shaft.steps_per_turn	电机输出轴每圈对应的脉冲数	38019		可修改	count/r	float32	M
axis0.output_shaft.torque_constant	电机输出轴扭矩常数	38020			Nm/A	float32	
axis0.output_shaft.load_inertia	电机输出轴端负载转动惯量	38021		可修改	Nm/(r/s)	float32	

属性名称	中文解释	Address	读写性（未注明即可读可写）	是否可修改	单位	类型	适用系列
					<sup>^2)</sup>		
axis0.output_shaft.homing_torque_lim	单编码器版本电机上电回零过程扭矩限制（无正负）	38022		可修改	Nm	float32	G
axis0.output_shaft.homing_vel_limit	单编码器版本电机上电回零过程最大速度（正负确定转动方向）	38023		可修改	r/min	float32	G
axis0.output_shaft.homing_pos_setpoint	回零时机械限位位置对应的关节目标角度（默认为0）	38024		可修改	°	float32	G
axis0.output_shaft.homing_move_range	回零过程搜索角度范围（机械限位需要在角度范围内）	38025		可修改	°	float32	G

注：

1. 在是否可修改一栏中，“可修改”表示用户可以根据需要自行修改，其中用蓝色字体标注的部分表示该参数已经封装进库函数中，最好通过的相应函数进行同步修改，例如 axis0.output\_shaft.vel\_ramp\_rate 为 set\_speed(mode=0)中的 param 参数。
2. 最后四项参数 axis0.output\_shaft.homing\_xxxx 仅适用于 G 系列单编码器版本关节电机，用于开机自动回零；
3. 最后一列适用系列 M 表示仅使用与 M 系列关节产品，G 系列仅适用于 G 系列关节产品，空白表示两种系列均适用；

数据类型在实际传输时使用类型数字代号进行传输，对应规则如下：

数据类型	Float32	Uint16	Int16	Uint32	Int32
数字代号	0	1	2	3	4

另外 bool 在实际传输中使用 Uint32 进行传输，对应数字代号为 3；

电机转动惯量（axis0.controller.config.inertia）单位为 Nm/(turn/s<sup>2</sup>)，国际单位应该为 kg\*m<sup>2</sup>。主要是因为电机内部加速度单位为(turn/s<sup>2</sup>)，与电机惯量直接相乘，得到扭矩 Nm；为此在设置该属性时，需要将国际单位转换为 Nm/(turn/s<sup>2</sup>)，转换规则如下：

- a. 加速度国际单位为 rad/s<sup>2</sup>， $M = \alpha * I = \text{rad/s}^2 * \text{kg} * \text{m}^2 = \text{turn} * 2\pi / \text{s}^2 * \text{kg} * \text{m}^2 = \text{Nm}$   
 $\text{Nm}/(\text{turn}/\text{s}^2) = 2\pi * \text{kg} * \text{m}^2$
- b. 电机输出轴的负载转动惯量如果也需要补偿，则需要首先按照上条规则将单位换算成 Nm/(turn/s<sup>2</sup>)；然后再除以减速比的平方；最后将电机转子惯量和负载转动惯量相加，即可得到电机转动惯量（axis0.controller.config.inertia）的设置值；