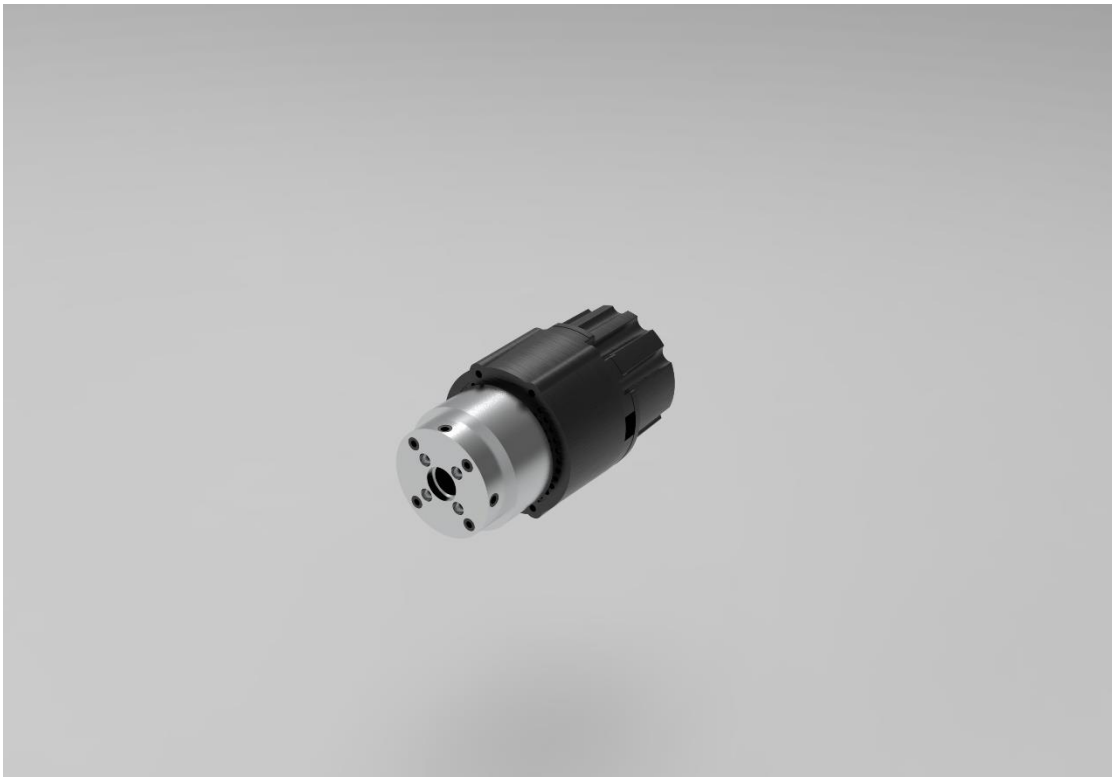
Robodyno Pro 可编程电机
技术文档
V1.0



系列型号

PJ-MP470/12

PJ-MP470/44

版本历史

编号	版本号	修订人	修订内容	修订日期
1	V1.0	赵嵩阳	文档发布	2021/12/10

目录

技术参数

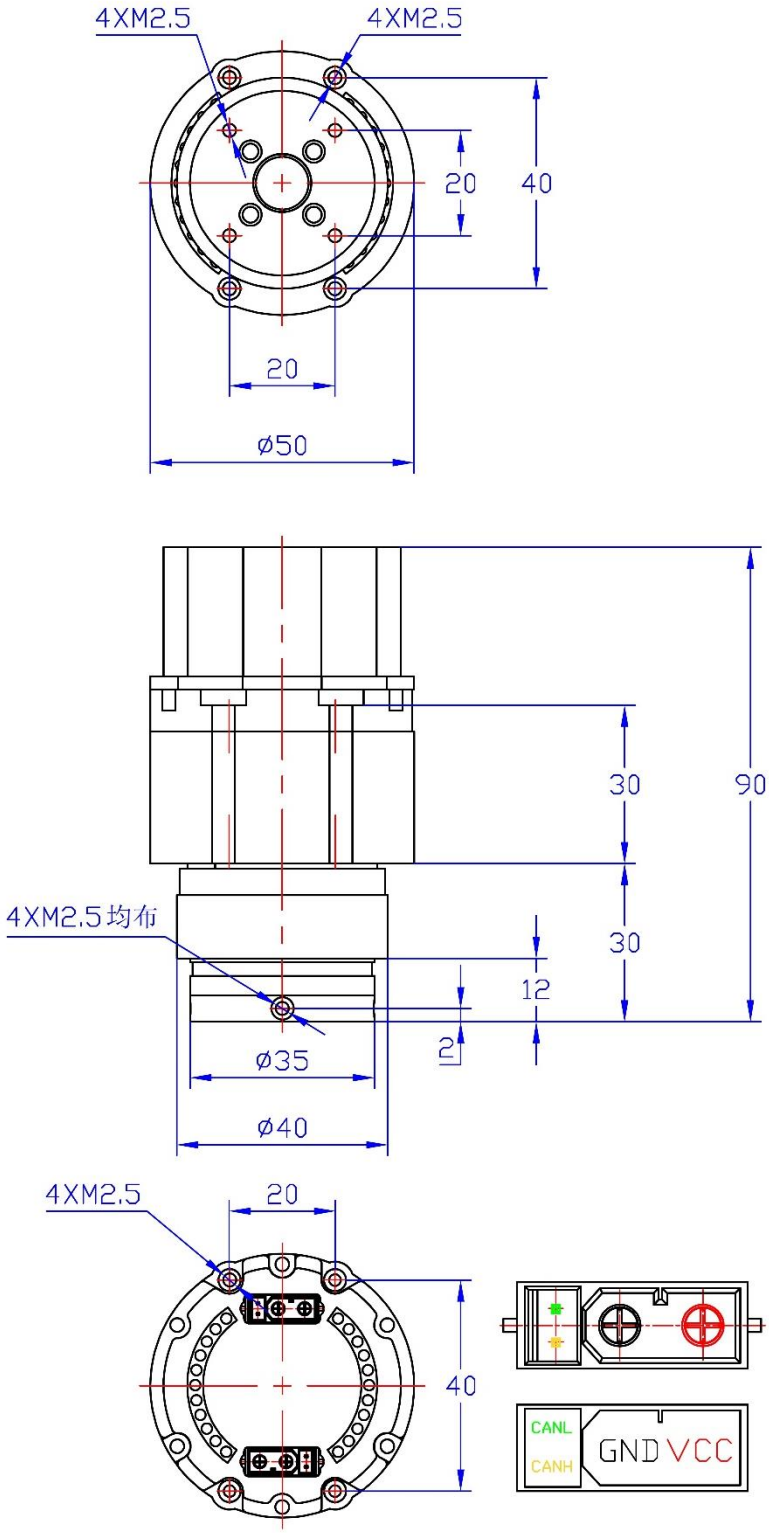
PJ-MP470/12

基本参数	电机类型	无刷伺服电机
	输入电压(V)	11.1~25.2
	减速比	12.45:1
	环境温度(°C)	-20~50
	旋转角度	多圈连续旋转
	通信方式	CAN 2.0
	通信速率	默认 1Mbit/s，可调
	编码器类型	绝对位置
	编码器分辨率	14bit (16384/r)
机械参数	直径(mm)	50
	长度(mm)	90
	重量(g)	360
	背隙(arcmin)	15
工作参数	额定功率(W)	100
	额定电压(V)	12.6
	额定母线电流(A)	2
	额定转速(rpm)	144
	额定扭矩(Nm)	1
	最大母线电流(A)	4.5
	最大相电流(A)	15
	峰值转速(rpm)	300
	峰值扭矩(Nm)	3.3
	工作温度(°C)	≤80

PJ-MP470/44

基本参数	电机类型	无刷伺服电机
	输入电压(V)	11.1 ~ 25.2
	减速比	44 : 1
	环境温度(°C)	-20 ~ 50
	旋转角度	多圈连续旋转
	通信方式	CAN 2.0
	通信速率	默认 1Mbit/s，可调
	编码器类型	绝对位置
	编码器分辨率	14bit (16384/r)
机械参数	直径(mm)	50
	长度(mm)	90
	重量(g)	360
	背隙(arcmin)	15
工作参数	额定功率(W)	100
	额定电压(V)	12.6
	额定母线电流(A)	2
	额定转速(rpm)	40
	额定扭矩(Nm)	3.5
	最大母线电流(A)	4.5
	最大相电流(A)	15
	峰值转速(rpm)	85
	峰值扭矩(Nm)	8
	工作温度(°C)	≤80

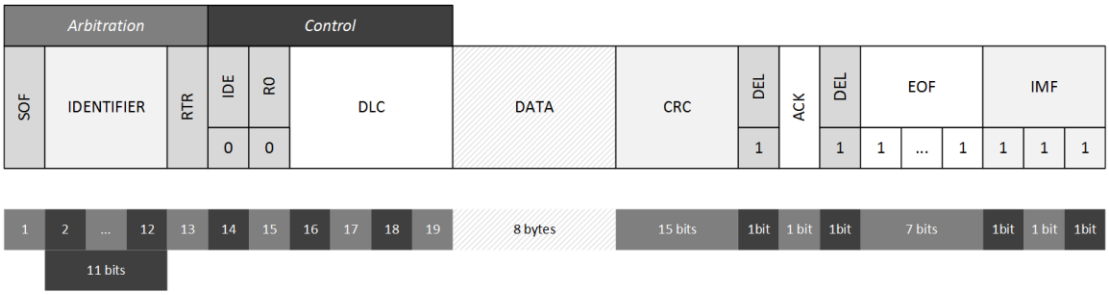
机械尺寸及通信接口


4XM2.5
4XM2.5
20
40
20
ø50
30
90
30
4XM2.5均布
12
2
ø35
ø40
4XM2.5
20
40
CANL
CANH
GND
VCC

CAN 总线通信协议

基本说明

Robodyno Pro 可编程电机通信方式遵循 CAN 2.0 规范，使用 11 位标识符的标准格式，默认通信速率 1Mbit/s，通信速率 250kbit/s – 1Mbit/s 可调，数据帧格式如图：



- IDENTIFIER：数据帧 ID 标识符
- RTR：远程请求帧标识符
- DLC：DATA 数据长度（字节数）
- DATA：数据段，0-8 字节
- CRC：循环冗余校验

Robodyno Pro 可编程电机通信协议将协议中的 11 位 ID 标识符分为 6 位设备 ID（最大 0x3F）与 5 位功能标识码（最大 0x1F），即：

IDENTIFIER(11bit) = device_id << 5 | function_code

例如 ID 为 0x10 的可编程电机，心跳包（功能码 0x02）的 CAN 数据帧 ID 为 0x202。

通信协议中的数据帧存在 3 种不同的发送方式，对应 3 种发送者：

1. 设备主动发送：如设备定期发送的心跳包。
2. 主机发送指令：如重启指令。
3. 主机请求数据：如查询电机反馈信息。主机会在功能 ID 对应的地址上发送远程请求帧（RTR），设备接收到请求指令后会在相同地址上返回对应的数据内容。

协议内容

功能 ID	功能名	发送者	数据长度	数据段								数据①	数据②	数据③
				0	1	2	3	4	5	6	7			
0x01	查询 API 版本	请求	8	①		②		③				主版本号	副版本号	设备类型
0x02	心跳包	设备	8	①								心跳包		
0x03	紧急停止	主机	0											
0x04	重启	主机	0											
0x05	清除错误	主机	0											

功能 ID	功能名	发送者	数据长度	数据段								数据①	数据②	数据③
				0	1	2	3	4	5	6	7			
0x06	保存设置	主机	0											
0x07	配置 CAN 总线	主机	8	①		②		③				CAN_ID	比特率	心跳间隔
0x08	设置运行状态	主机	4	①								电机状态		
0x09	查询硬件状态	请求	8	①				②				总线电压	内部温度	
0x0A	查询反馈数据	请求	8	①				②		③		位置	速度	力矩
0x0B	查询控制模式	请求	2-8	①		②						控制模式	模式参数	
0x0C	设置控制模式	主机	2-8	①		②						控制模式	模式参数	
0x0D	查询 PID	请求	8	①				②		③		位置环 P	速度环 P	速度环 I
0x0E	设置 PID	主机	8	①				②		③		位置环 P	速度环 P	速度环 I
0x0F	查询电机限值	请求	8	①				②				电压上限	电流上限	
0x10	设置电机限值	主机	8	①				②				电压上限	电流上限	
0x11	设置位置	主机	8	①				②		③				
0x12	设置速度	主机	8	①				②						
0x13	设置力矩	主机	4	①										
0x14	恢复出厂设置	主机	0											

注：数据段存储均采用小端序（Little-Endian）。

Python API 参考

1. 初始化电机对象

```
from protobot.can_bus import Robot
from protobot.can_bus.nodes import MotorFactory
robot = Robot()
motor = robot.add_device('motor0', MotorFactory(), 0x15,
reduction=12.45)
```

参数：

- name: 电机名
- factory: 电机对象工厂类
- id: 电机设备 ID（出厂 0x10）
- reduction: 电机减速比

2. 电机状态

```
motor.status()
```

最近一次心跳包的内容

返回值：

- 电机状态 dict

- state: 工作模式（1-空闲，8-使能）
- error: 错误码（1-电压不足，14-急停）
- motor_err: 电机相关错误码
- encoder_err: 编码器相关错误码
- controller_err: 控制器相关错误码
- control_mode: 控制模式（1-力矩控制，2-速度控制，3-位置控制）
- input_mode: 输入模式（1-直接值，2-带加速度，3-带滤波，5-梯形轨迹）
- timestamp: 时间戳，上一次收到心跳包的时间

3. 获取 API 版本

```
get_api_version(timeout = 0)
```

参数：
- timeout: 请求超时时间(s)，0 代表无超时时间

返回值：
- API 版本 dict
- device_uuid: 设备 uuid
- main_version: 主版本号
- sub_version: 副版本号

4. 电机软急停

```
estop()
```

5. 电机重启

```
reboot()
```

6. 清除错误

```
clear_errors()
```

7. 保存设置

```
save_configuration()
```

设置参数后默认不会保存，直到调用此函数

8. 设置电机 CAN_ID

```
set_can_id(id, bitrate, rate)
```

参数：
- id：电机新 CAN_ID (0x01~0x3F)
- bitrate: CAN 速率（0-250kbit/s, 1–500kbit/s, 2–1Mbit/s）

- rate: 心跳包发送周期，ms

9. 电机使能

```
enable()
```

10. 电机失能

```
disable()
```

11. 电机校准

```
calibrate()
```

校准后需保存参数

12. 读取总线电压

```
get_vbus(timeout)
```

参数：

- timeout

返回值：

- 总线电压值(V)

13. 读取电机温度

```
get_temperature(timeout)
```

参数：

- timeout

返回值：

- 电机温度(℃)

14. 读取电机状态参数

```
get_status(timeout)
```

参数：

- timeout

返回值：

- 电机状态参数(位置 rad, 速度 rad/s, 力矩 Nm)

15. 读取电机位置

```
get_pos(timeout)
```

参数：

- timeout

返回值：

- 位置(rad)

16. 读取电机速度

```
get_vel(timeout)
```

参数：

- timeout

返回值：

- 速度(rad/s)

17. 读取电机力矩

```
get_torque(timeout)
```

参数：

- timeout

返回值：

- 力矩(Nm)

18. 读取电机控制模式

```
get_controller_modes(timeout)
```

参数：

- timeout

返回值：

- 控制模式(control_mode,input_mode)
  - control_mode: 控制模式（1-力矩控制，2-速度控制，3-位置控制）
  - input_mode: 输入模式（1-直接值，2-带加速度，3-带滤波，5-梯形轨迹)

19. 读取加速度输入模式参数

```
get_ramp_mode_ramp(timeout)
```

参数:

- timeout

返回值：

- ramp: 加速度(rad/s^2)

20. 读取滤波输入模式参数

```
get_filter_mode_bandwidth(timeout)
```

参数：

- timeout

返回值：

- bandwidth: 滤波带宽 / 输入频率(Hz)

21. 读取轨迹模式参数

```
get_traj_mode_params(timeout)
```

参数:

- timeout

返回值：

- 轨迹参数(最大速度, 加速度, 减速度)

22. 读取电机 PID 参数

```
get_controller_pid(timeout)
```

参数：

- timeout

返回值：

- 电机 PID(位置环 P, 速度环 P, 速度环 I)

23. 设置电机 PID 参数

```
set_controller_pid(pos_p, vel_p, vel_i)
```

参数:

- pos_p: 位置环比例系数
- vel_p: 速度环比例系数
- vel_i: 速度环积分系数

24. 读取电机速度电流限制

```
get_limits(timeout)
```

参数:

- timeout

返回值：

- (输入端最大速度(r/s), 最大电流(A))

25. 读取电机速度限制

```
get_vel_limit(timeout)
```

参数:

- timeout

返回值:

- 输出端最大速度(rad/s)

26. 读取电机电流限制

```
get_current_limit(timeout)
```

参数:

- timeout

返回值:

- 最大电流(A)

27. 设置电机速度电流限制

```
set_limits(vel_limit, torque_limit)
```

参数:

- vel_limit: 输入端最大速度(r/s)
- torque_limit: 最大电流(A)

28. 设置电机速度限制

```
set_vel_limit(vel_limit)
```

参数:

- vel_limit: 输出端最大速度(rad/s)

29. 进入直接位置模式

```
position_mode()
```

直接 PID 控制位置

30. 进入滤波位置模式

```
position_filter_mode(bandwidth)
```

参数:

- bandwidth: 滤波带宽 / 控制频率(Hz)

31. 进入轨迹位置模式

```
position_traj_mode(max_vel, accel, decel)
```

参数:

- max_vel: 最高速度
- accel: 加速度
- decel: 减速度

32. 设置位置

```
set_pos(position)
```

参数:

- position: 目标位置(rad)

33. 进入直接速度模式

```
velocity_mode()
```

速度 PID 控制

34. 进入匀加减速速度模式

```
velocity_ramp_mode(ramp)
```

参数:

- ramp: 加速度(rad/s^2)

35. 设置速度

```
set_vel(velocity)
```

参数:

- velocity: 目标速度(rad/s)

36. 进入力矩控制模式

```
torque_mode()
```

37. 设置力矩

```
set_torque(torque)
```

参数:

- torque: 目标力矩(Nm)