

1.Parameters of servo motor

Stroke: 500mm

Speed: 3000 laps/min

Load: 700N

Lap: 50; When the motor runs for 1 laps, the electric actuator will extend by 10mm

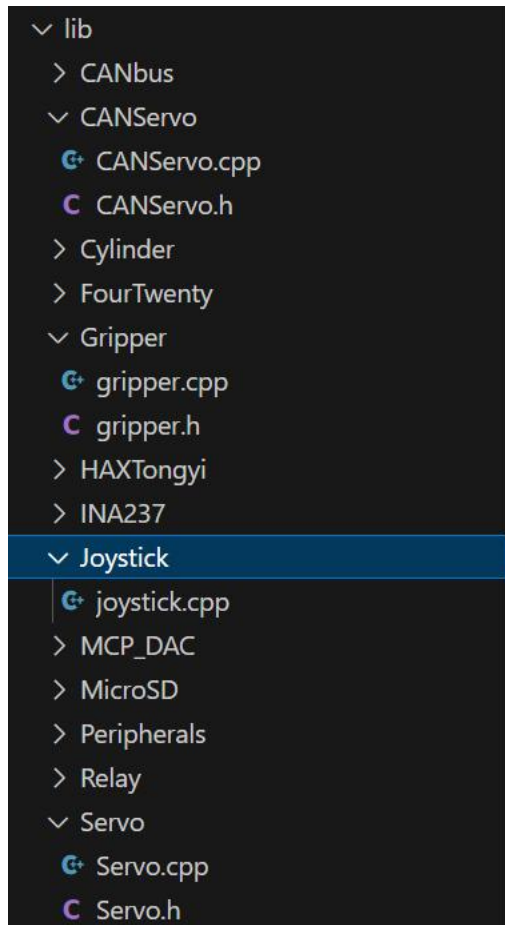
Communication frequency: 250k ; Because the communication frequency of the joystick is 250k, and the joystick and the motor use the same bus, the frequencies of the two must be kept consistent.

In order to change the communication frequency, prepare a RS485 to USB module in advance.



actuator

2.Description of lib



CANServo.h: It was written by Han Long, but I didn't use it.

Gripper.h: I wrote the code from line 89 to the end. `class Gripper` is used for the old machine.

HAXTongyi: It is an example provided by the manufacturer.

Servo: function of controlling servo motor

Joystick.cpp: `controlCylinders()` is used for testing

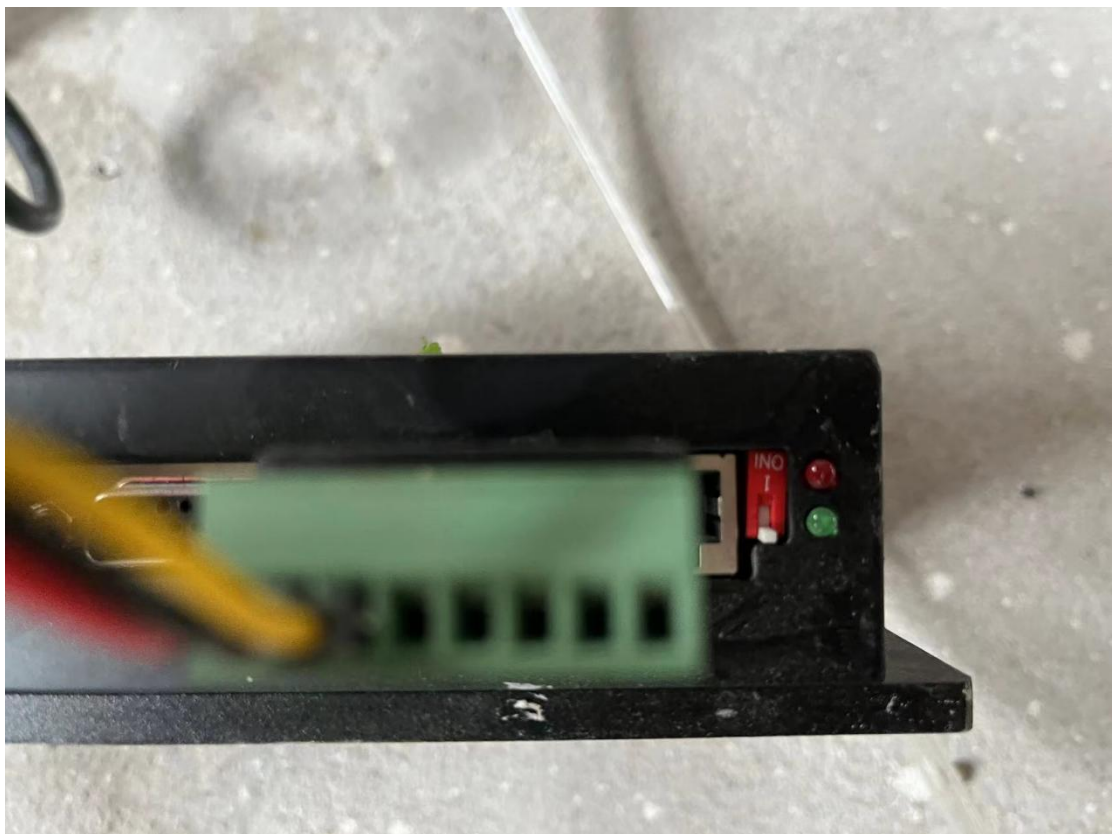
3.Process of controlling the motor

Step1: Turn on the motor power and then turn on the EE box power.

Note: The controller of the motor can only receive the NMT command from the PCB after it is started, and can only start CAN communication after successfully receiving the command.

Step2: Wait

Note: use the current code, after about twenty seconds, the motor will be enabled, and after another ten seconds, you can control the motor using the joystick.



The red LED is an alarm light. When it flashes, it indicates that the motor or controller is running in error. The current solution is to power off and restart.

The green LED is the controller operation indicator. When it is always on, it indicates that the motor is already in the enabled state.

Step3: control

6.8.6 Default PDO Mapping Parameter in Location Mode

The default configuration PDO mapping parameter of the driver is shown in the following table:

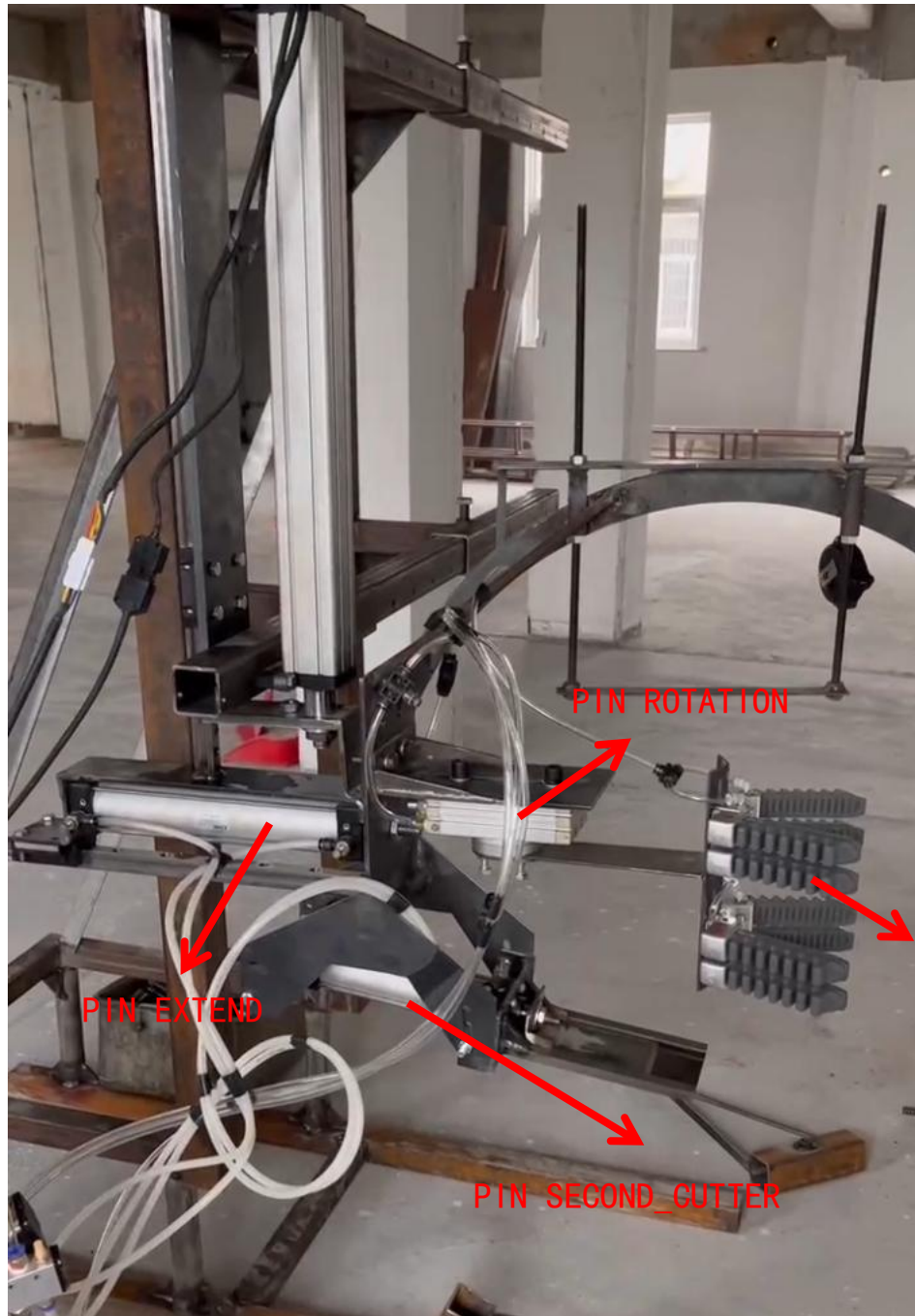
CANopen_Protocol_Software_V4_32												
1. CANopen 远程设备 2. PDO 映射 3. 服务数据对象 4. CANopen 配置 5. CANopen I/O 映射 6. 状态 7. 信息												
8. 选择接收 PDO (RPDO)												
10	名称	11 索引	12 子索引	13 位长度								
	<input checked="" type="checkbox"/> 1. receive PDO parameter	16#1400										
	Controlword	16#6040	16#00	16								
	<input checked="" type="checkbox"/> 2. receive PDO parameter	16#1401										
	Modes_of_operation	16#6060	16#00	8								
	<input checked="" type="checkbox"/> 3. receive PDO parameter	16#1402										
	Target_velocity	16#60FF	16#00	32								
	Target_torque	16#6071	16#00	16								
	<input checked="" type="checkbox"/> 4. receive PDO parameter	16#1403										
	Target_Position	16#607A	16#00	32								
	Profile_velocity	16#6081	16#00	32								
9. 选择发送 PDO (TPDO)												
10	名称	11 索引	12 子索引	13 位长度								
	<input checked="" type="checkbox"/> 1. transmit PDO parameter	16#1800										
	Statusword	16#6041	16#00	16								
	<input checked="" type="checkbox"/> 2. transmit PDO parameter	16#1801										
	Error_code	16#603F	16#00	16								
	<input checked="" type="checkbox"/> 3. transmit PDO parameter	16#1802										
	Velocity_actual_value	16#606C	16#00	32								
	Current_actual_value	16#6078	16#00	16								
	<input checked="" type="checkbox"/> 4. transmit PDO parameter	16#1803										
	Position_control_Cylinder_Num	16#60FB	16#02	32								
	Position_actual_value	16#6064	16#00	32								

Use PDO; position Mode

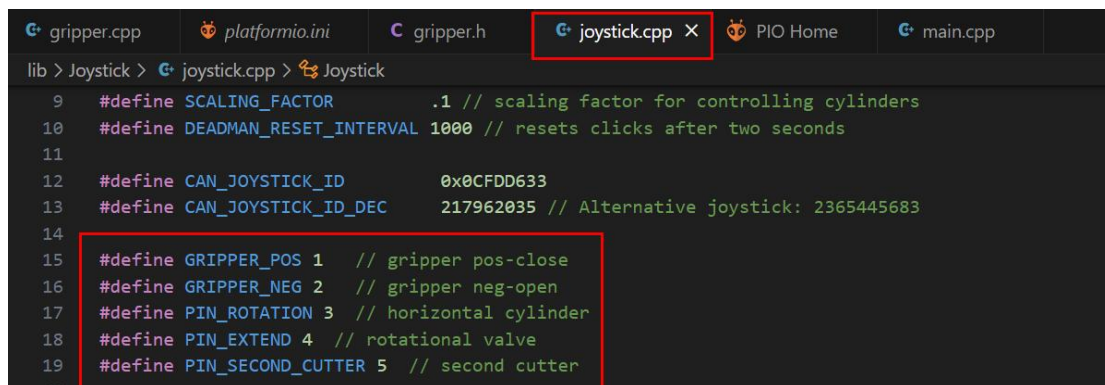
Step4: After operating the actuator to 0 position(Stroke: 500mm), turn off the power

Note: The motor currently used is a relative value motor.

4. code

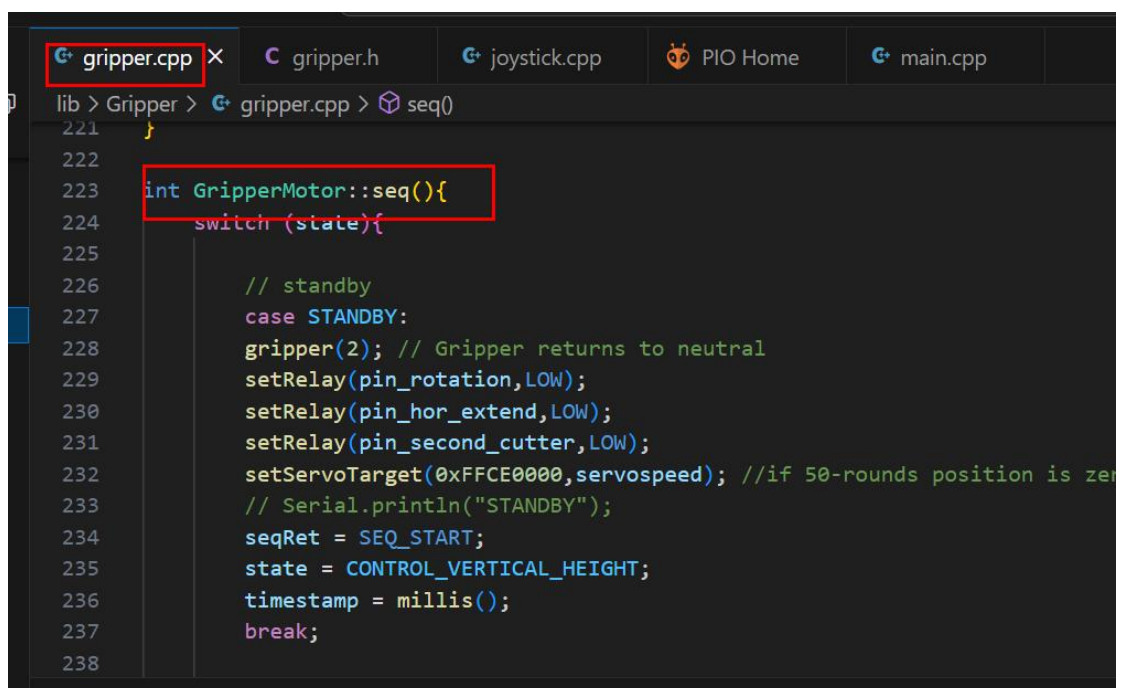


4.1 definition of pin



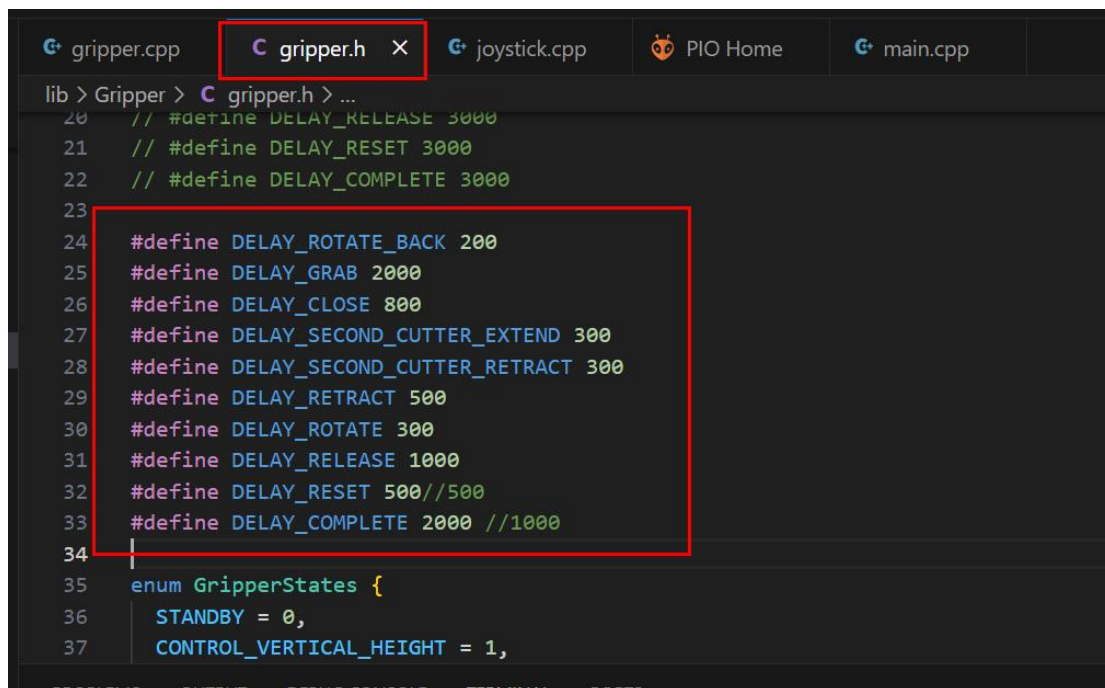
```
9  #define SCALING_FACTOR .1 // scaling factor for controlling cylinders
10 #define DEADMAN_RESET_INTERVAL 1000 // resets clicks after two seconds
11
12 #define CAN_JOYSTICK_ID 0xCFDD633
13 #define CAN_JOYSTICK_ID_DEC 217962035 // Alternative joystick: 2365445683
14
15 #define GRIPPER_POS 1 // gripper pos-close
16 #define GRIPPER_NEG 2 // gripper neg-open
17 #define PIN_ROTATION 3 // horizontal cylinder
18 #define PIN_EXTEND 4 // rotational valve
19 #define PIN_SECOND_CUTTER 5 // second cutter
```

4.2 Sequence



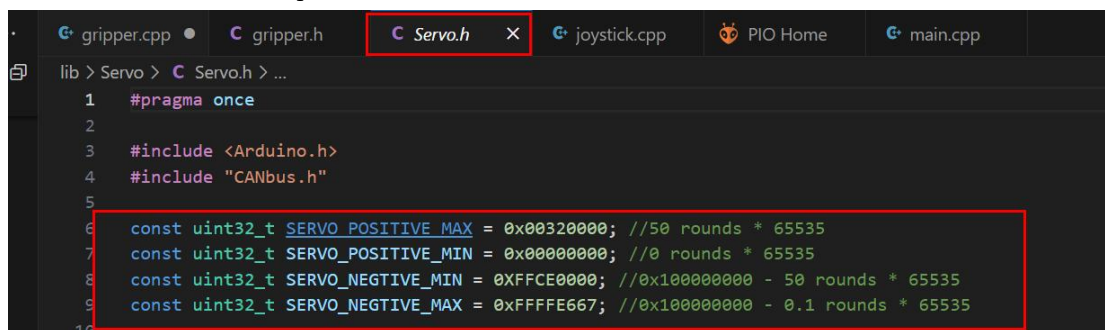
```
221 }
222
223 int GripperMotor::seq(){
224     switch (state){
225
226         // standby
227         case STANDBY:
228             gripper(2); // Gripper returns to neutral
229             setRelay(pin_rotation, LOW);
230             setRelay(pin_hor_extend, LOW);
231             setRelay(pin_second_cutter, LOW);
232             setServoTarget(0xFFCE0000, servospeed); //if 50-rounds position is zero
233             // Serial.println("STANDBY");
234             seqRet = SEQ_START;
235             state = CONTROL_VERTICAL_HEIGHT;
236             timestamp = millis();
237             break;
238     }
```


4.3 Time control of sequence



```
lib > Gripper > C gripper.h > ...
20 // #define DELAY_RELEASE 3000
21 // #define DELAY_RESET 3000
22 // #define DELAY_COMPLETE 3000
23
24 #define DELAY_ROTATE_BACK 200
25 #define DELAY_GRAB 2000
26 #define DELAY_CLOSE 800
27 #define DELAY_SECOND_CUTTER_EXTEND 300
28 #define DELAY_SECOND_CUTTER_RETRACT 300
29 #define DELAY_RETRACT 500
30 #define DELAY_ROTATE 300
31 #define DELAY_RELEASE 1000
32 #define DELAY_RESET 500//500
33 #define DELAY_COMPLETE 2000 //1000
34
35 enum GripperStates {
36     STANDBY = 0,
37     CONTROL_VERTICAL_HEIGHT = 1,
```

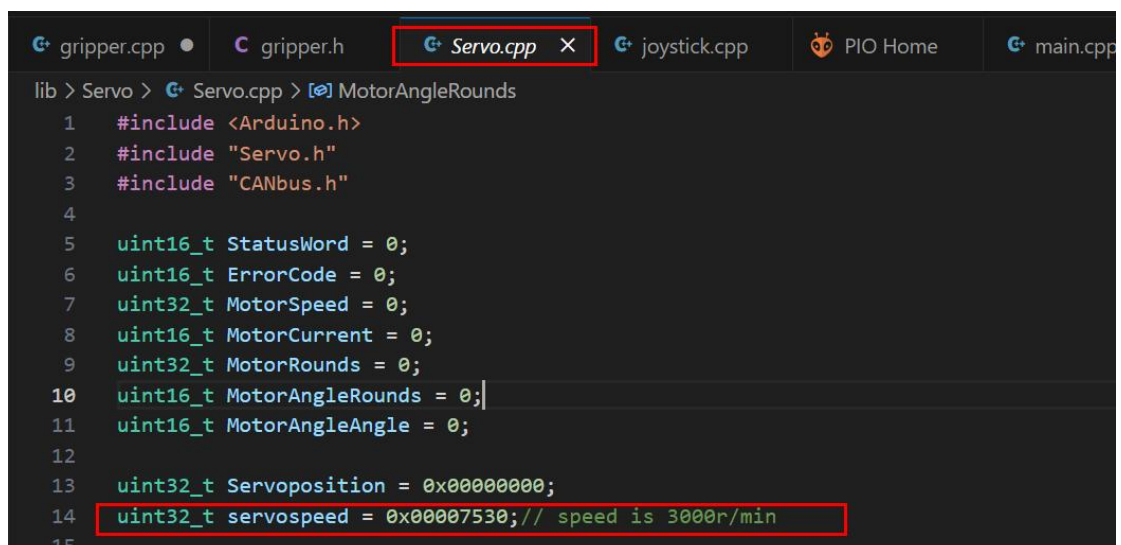
4.4 Set the limit position of the motor



```
lib > Servo > C Servo.h > ...
1 #pragma once
2
3 #include <Arduino.h>
4 #include "CANbus.h"
5
6 const uint32_t SERVO_POSITIVE_MAX = 0x00320000; //50 rounds * 65535
7 const uint32_t SERVO_POSITIVE_MIN = 0x00000000; //0 rounds * 65535
8 const uint32_t SERVO_NEGTIVE_MIN = 0xFFCE0000; //0x100000000 - 50 rounds * 65535
9 const uint32_t SERVO_NEGTIVE_MAX = 0xFFFFE667; //0x100000000 - 0.1 rounds * 65535
10
```

4.5 Set the speed of the motor

Generally set the motor to run at the fastest speed.



```
lib > Servo > C Servo.cpp > [O] MotorAngleRounds
1 #include <Arduino.h>
2 #include "Servo.h"
3 #include "CANbus.h"
4
5 uint16_t StatusWord = 0;
6 uint16_t ErrorCode = 0;
7 uint32_t MotorSpeed = 0;
8 uint16_t MotorCurrent = 0;
9 uint32_t MotorRounds = 0;
10 uint16_t MotorAngleRounds = 0;
11 uint16_t MotorAngleAngle = 0;
12
13 uint32_t Servoposition = 0x00000000;
14 uint32_t servospeed = 0x00007530; // speed is 3000r/min
15
```

4.6 Feedback value of motor

```
gripper.cpp • gripper.h Servo.cpp X joystick.cpp PIO Home main.cpp
lib > Servo > Servo.cpp > can_heartbeat_publish(HBSTATE)
125 void can_change_servo_mode() {
157 #endif
158 }
159 }
160
161 void CAN_TPDO_CALLBACK() {
162     twai_message_t can_receive_msg;
163     int16_t cmd_val;
164
165     int ret = twai_receive(&can_receive_msg, 0);
166     if (ret == ESP_OK) {
167         // #if SERIAL_DEBUG_SEND_FLG or SERIAL_DEBUG_AUTO_FLG or SERIAL_DEBUG_MANU_FLG or \
168         // SERIAL_GENERAL
169         Serial.print("tpdo read 0x");
170         Serial.print(can_receive_msg.identifier, HEX);
171         Serial.print(", [");
172         for (uint8_t idx = 0; idx < 8; idx++) {
173             Serial.print("0x");
```

	CAN 标识符	字节数	字节 1	字节 2	字节 3	字节 4	字节 5	字节 6	字节 7	字节 8
TPDO1	0x180+Node-ID	2	00	00						
			CANOPEN 控制状态反馈, SDO 地址: 0x6041							
			禁止时间: 20 (2ms);事件时间: 50ms							
TPDO2	0x280+Node-ID	2	00	00						
			错误代码反馈, SDO 地址: 0x603F							
			禁止时间: 20 (2ms);事件时间: 50ms							
TPDO3	0x380+Node-ID	6	00	00	00	00	00	00		
			电机速度反馈, 0x606C				电机电流反馈, 0x6078			
			禁止时间: 200 (20ms);事件时间: 50ms							
TPDO4	0x480+Node-ID	8	00	00	00	00	00	00	00	00
			电机圈数反馈, 0x60FB 02				电机实时角度反馈, 0x6064 (低 16 位角度+高 16 位圈数)			
			禁止时间: 200 (20ms);事件时间: 50ms							