



Overview

[Specification](#)[Feature](#)[How to Use the Hardware](#)[How to Use the Software](#)[ESP32 Flash Download Tool](#)[Compile Arduino IDE](#)[Install Arduino IDE](#)[Download the Program &](#)[Install Dependency](#)[Libraries](#)[Upload Demo](#)[Product Usage Tutorial](#)[Secondary Development
Tutorial](#)

Resource

[3D Module](#)[2D Drawing](#)[Document](#)[Demo](#)[Software](#)[Open Source Structure](#)

FAQ

Support

[To Top](#)

ST3215 Servo

Overview

As a high-torque programmable serial bus servo, it is equipped with a 360° high-precision magnetic encoder, which can realize 360-degree absolute angle control. With the program to control, any angles can be set as the middle of the position, and it can also be switched to the continuously rotating motor or stepper motor mode. The built-in acceleration start-stop function makes the action softer. There are two interfaces on each servo, which can be used in series. In theory, 253 bus servos can be controlled at the same time, and each servo can obtain its current angle, load, voltage, mode, and so on. It is suitable for robot projects, such as robotic arms, hexapod walkers, humanoid robots, wheeled robots and so on, which require real-time feedback for closed-loop control.

At the same time, we provide an open-source 12-DOF robot dog model for this servo. You can download the model and project files of the open-source structure in [#Resource](#).



TTL bus servo interface

Specification

- Input Voltage: 6-12.6V
- Mechanical Limited Angle: No Limit
- Rotating Angle: 360° (servo mode angle control) / motor mode continuous rotation
- Baudrate: 1Mbps
- Gear: high-precision metal gear
- Idling Speed: 0.222sec/60° (45RPM) @12V
- Pos Sensor Resolution: 360°/4096
- ID Range: 0-253
- Feedback: Position, Load, Speed, Input Voltage
- Idling Current: 180 mA
- Locked-rotor Current: 2.7 A
- Dimension: 45.22mm x 35mm x 24.72mm

Feature

- Up to 253 servos can be connected in series at the same time (requires full power), and also can obtain feedback information from each servo.
- 360°magnetic encoder, larger angle control range.
- High precision, the angle control accuracy reaches 360°/4096.
- Any angles can be set as the middle of the position, which is convenient for assembly.
- Acceleration start and stop function, the movement effect is softer.
- Compact structure design and the product appearance are more beautiful.
- 6-12.6V wide-voltage input, it can directly supply power with the 2s or 3s lithium battery.
- High torque, up to 30kg.cm@12V.
- The working mode can be set by the program: servo angle control/motor model can be rotated continuously.

How to Use the Hardware

- You can use our [Servo Driver](#) to control the bus servo.
- If you want to use the bus servo in your project, you can refer to the following schematic diagram: [Bus servo drive circuit schematic](#).

How to Use the Software

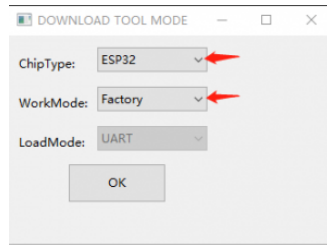
ESP32 Flash Download Tool

Generally, we control the ST series servos by compiling and uploading the demo through the Arduino IDE. However, compiling through Arduino requires installing various dependency libraries before use. Therefore, we provide an ESP32 download tool. With this tool, users can download the demo to the driver board without needing to download other dependency libraries or the Arduino IDE software.

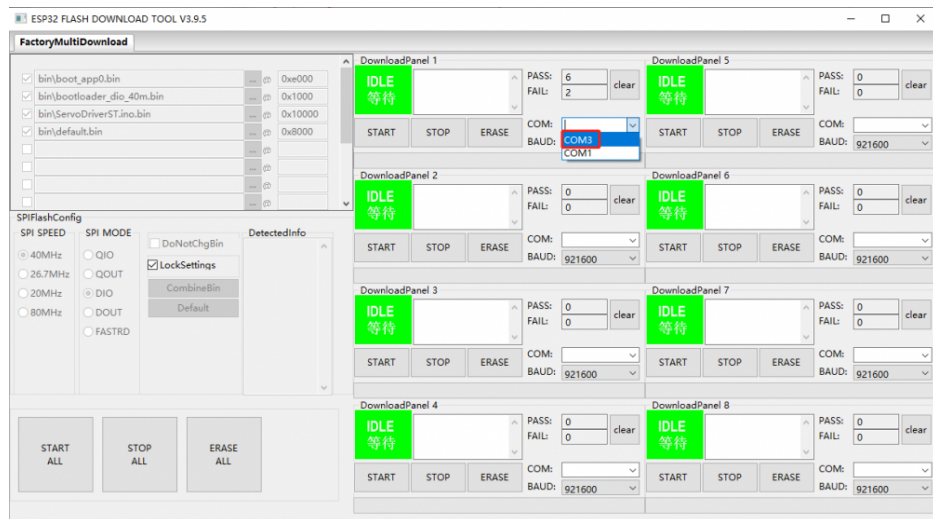
- Click [here](#) to download ESP32 Flash Download Tool. Once downloaded, extract the files and double-click on the "flash_download_tool_3.9.5.exe" program. After opening, two windows will appear. The UI interface of the download tool is the

one you need to operate, while the other window serves as a terminal to display the working status of the download tool.

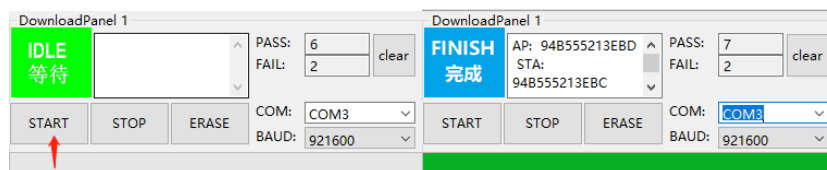
- In the "DOWNLOAD TOOL MODE" interface, select Chip Type as ESP32 and WorkMode as Factory. When using Factory to call the binary file, it will use a relative path, eliminating the need for users to manually input the binary file path. After making these selections, click OK.



- In this software interface, keep the "LockSettings" option checked. On the right side, you can simultaneously upload demos to 8 driver boards. Next, connect the servo driver board to the computer using a USB cable. Click on "COM" and select the newly appeared COM port (in my case, it's COM3). BAUD is used to set the download speed, higher values result in faster speeds, with the ESP32 supporting speeds up to 921600.



- After selecting, click START to begin uploading the demo. Once the upload is complete, "IDLE" will change to "FINISH". After completing this process, you can disconnect the USB connection between the servo driver board and the computer. Then, connect the servos to the servo driver board, provide power to the driver board, and start controlling the ST series servos.



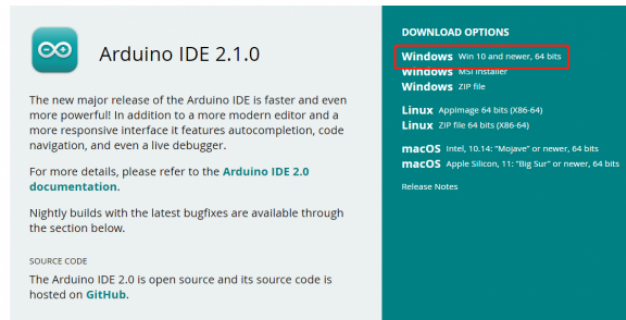
Compile Arduino IDE

- We provide ST3215-compatible servo demo for [Servo Driver](#), you can use it at [Servo Driver](#) to download [open-source demo \(Arduino\)](#) to control the ST series servos, the following steps:

Install Arduino IDE

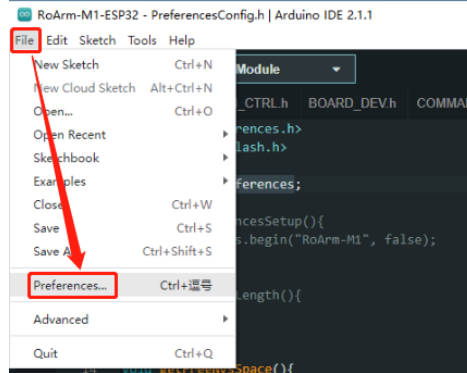
- You can directly click this [link](#) to download the Arduino IDE from [Arduino.cc](#).

- Download the latest Arduino IDE 2.1.0 package from [Arduino website](#). The official IDE supports different operating systems, just download it according to your operating system. The one I downloaded here is for Windows. (If you have already installed it, just skip to the second step) The installation process is very simple, just keep clicking next step.



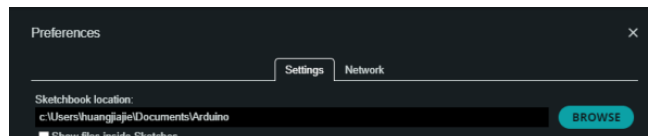
Note: It will prompt you to install the driver during installation, you can click to install.

1. Open Arduino IDE, click the File on the left corner and choose "Preferences".



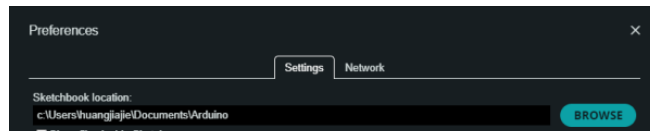
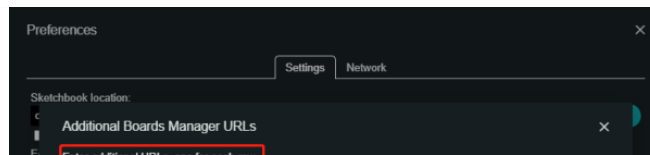
2. Add the following link in the additional development board manager URL, then click OK.

```
https://dl.espressif.com/dl/package_esp32_index.json
```



Note: If you need to add more than one development board URL, it is not necessary to delete the URL of the ESP32 development board support, you can directly add the other URL to another line, the default display is comma separated URL. For example, if you need to add the URL of the ESP8266 development board, add it directly to another line, it will be displayed as follows:

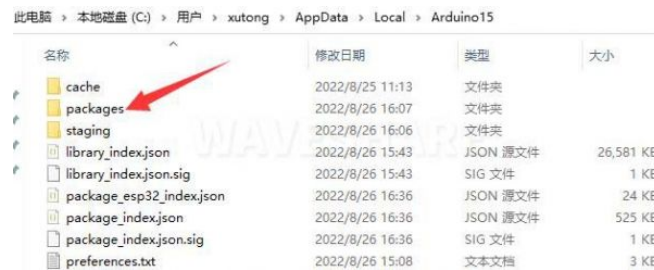
```
https://dl.espressif.com/dl/package_esp32_index.json , http://arduino.esp8266.com/stable/package_esp8266com_index.json
```



- Download the [package file](#), and copy the unzipped packages files to the following path:

C:\Users\username\AppData\Local\Arduino15

The "username" here would vary according to your computer's username, copy the unzipped packages file to the Arduino15 folder.



Download the Program & Install Dependency Libraries

- Please install the following libraries through Manage Libraries:
 - Adafruit SSD1306
 - Adafruit NeoPixel
- Click Tools > Manage Libraries to open the library manager.

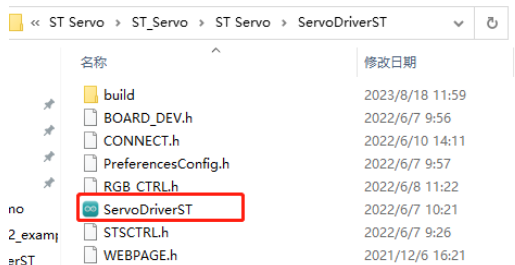
- Enter the libraries that need to be installed through the library manager in the search box, and install:

- Adafruit SSD1306
- Adafruit NeoPixel
- Click "Update", and if there is no "Update", please click "Install" for the newest version.

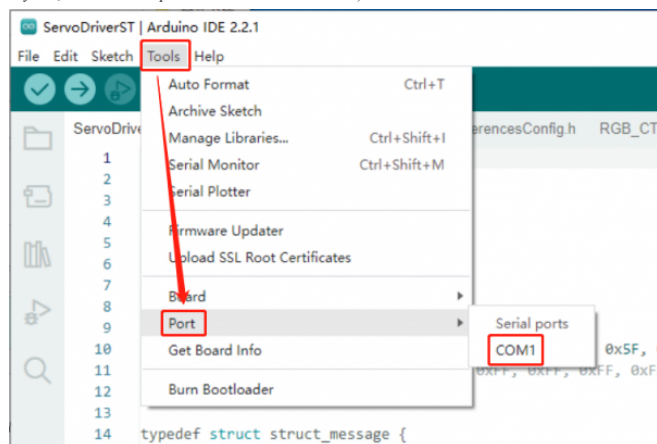
- Download the bus servo library and the Arduino program for the servo driver board:
- [ServoDriverST Open Source Program \(Arduino\)](#)
- After downloading, unzip it, and you can get two file folders:
 - "ServoDriverST": control ST series servo demo
 - "SCServo": control servo library
- Copy the SCServo folder to \Documents\Arduino\libraries after decompression, and install them for controlling the servo.

Upload Demo

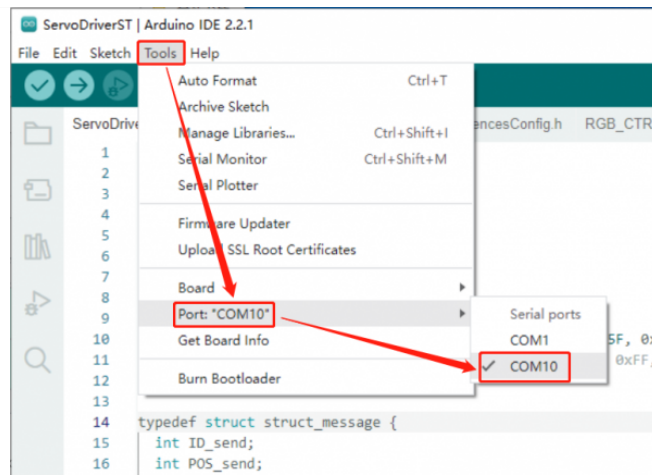
1. Double-click on ServoDriverST.ino:



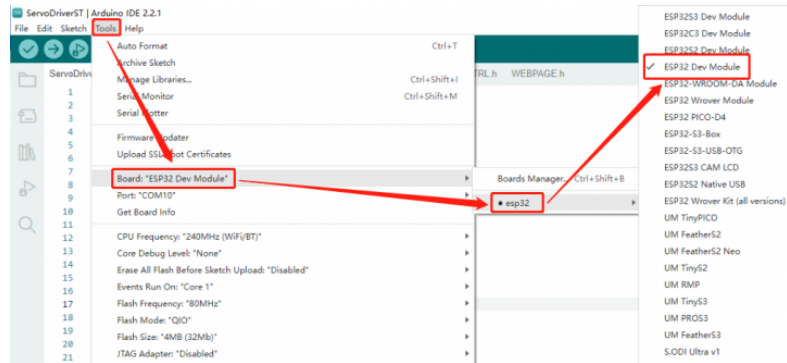
2. Click on "Tools" -> "Port", please remember the COM port on your PC, and do not click on this COM port (COM1 is the port on my PC, different computers show different COMs).



3. Connect the serial bus servo with the PC with a USB cable, click on "Tools" -> "Port", and then click on the new COM port (here is COM10):



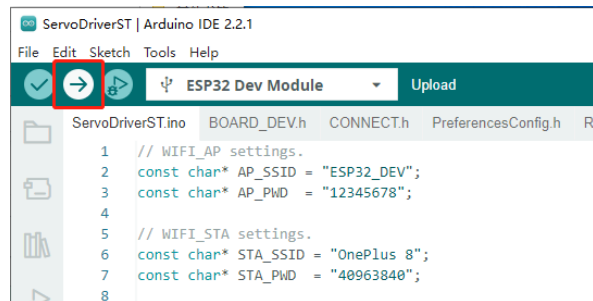
4. In Arduino IDE, click on "Tools" -> "Board:ESP32 Dev Module" -> "esp32" -> "ESP32 Dev Module".



5. Click on "Tools", and the other settings as shown below: (it is recommended to set "Huge APP" for "Partition Scheme", and "PSARM" must be enabled).



6. After setting, click to upload, and the demo will be uploaded to the device.



After uploading, "Leaving... Hard resetting via RTS pin..." indicates the successful uploading.

Product Usage Tutorial

- The following tutorial uses [servo driver](#) by default to explain how to use the servo, if you need to use it in your own project, you can refer to the sample program and the [bus servo drive circuit schematic](#) for secondary development.
- The factory default ID of the servo is 1. Among the servos connected to the same servo drive board, the same ID can only correspond to one servo, and there cannot be more than one servo with the same ID in the one control circuit. When you set the ID for the servo, try to ensure that only one servo whose ID is to be changed is connected to the driver board. After the ID is changed, it will be permanently saved in the servo and will not be lost even if the power is off.
- First of all, you need to set an independent ID for each servo. When setting an ID for one servo, the driver board should not be connected to other servos.
- If you use the ST3215 servo, after connecting it to the driver board, you need to provide 6-12.6V (12V is recommended) via the DC port (5.5*2.1mm) on the board, and the port will directly power the servo. If you use a large number of servos, you need to provide enough currents for the port.
- After the driver board is powered on, a WiFi hotspot will be created by default. The default hotspot name is ESP32_DEV, and its password is 12345678. Please connect to this hotspot with your phone.

- After connecting to the hotspot issued by the servo driver board, it is recommended to use Google Chrome to access the address 192.168.4.1. Noted that the phone will fail to verify the server with ping after connecting the WiFi, so it may automatically switch to the other known WiFi. In this case, you should connect ESP32_DEV again, and the phone will not connect to other WiFi automatically after the reconnection is successful.
- The driver board will automatically scan the servo ID (0-20) when it is powered on (in order to save the boot time). If your servo ID is more than 20, you must change the MAX_ID value in ServoDriverST.ino and upload it to the driver board again.

- If the servo is connected after the driver board is powered on, you need to click **Start Searching** on the mobile browser page to scan again.
- **Active ID** is the currently selected servo ID number, **ID to Set** is the new ID number to be set, adjust the value of **ID to Set** through **ID to Set+** and **ID to Set-**, and press the **Set New ID** button to set the servo ID **Active ID** as **ID to Set**.
- When the IDs of all servos are set, you can connect them all. If you do not restart the device, you need to click **Start Searching** to scan all servos.
- **ID :** shows all ID numbers of servos currently connected to the servo driver board.
- **Active ID** The currently selected servo ID number, and the subsequent operations are all controlling this **Active ID** servo.
- You can select the **Active ID** servo via the **ID to Set+** and **ID to Set-** buttons.
- **Middle** key, you can turn the servo to the middle position, the position range of the servo is 0-4095, and the middle position is 2047.
- **Stop** key, the default program will not let the servo move all the time. If the servo cannot be stopped after the secondary development, you can press the **Stop** key to halt the servo movement.
- **Release** key, press it and the servo will close torque lock (Torque Off), then you can turn the servo by hand.
- **Torque** key, after pressing the servo, the servo will turn on the torque lock (Torque On), at this time, the servo will maintain the specified position with force.
- **Position+** key, after pressing the servo, the servo will start to rotate clockwise. When the servo is in the servo mode, it will not continue to rotate after turning to the 4095 position.
- **Position-** key, after pressing the servo, the servo will start to rotate counterclockwise. When the servo is in the servo mode, it will not continue to rotate after turning to the 0 position.
- **Speed+** and **Speed-** are used to set the speed of the servo. The maximum setting for ST series servos is about 3073. The speed is the number of steps per second, 50 steps/sec≈0.732RPM.
- **Set Middle Position** key, no matter what position the current servo is in, when this key is pressed, the servo will set the current position to the middle position (2047).
- **Set Servo Mode** key, set the servo to servo mode, 360° absolute angle control can be performed in the servo mode, the setting will be permanently saved and will not be lost even if the power is turned off.
- **Set Motor Mode** key, set the servo to stepper motor mode, the servo can rotate continuously 30,000 steps. If you press it continuously, the motor will keep rotating. This mode can control the relative angle within ±7 circles and the number of circles will not be saved when power is off, but the stepper motor mode settings will be permanently saved even if power is off.
- **Start Serial Forwarding** button, set the servo driver board to serial forwarding mode, you can directly control the servo and get the feedback of the servo with the typeC interface on the board. This function is used to debug the servo.
- **Normal** key, set the servo driver board to normal mode, in this mode, it will not send or receive any information via ESP-NOW.
- **Leader** key, set the servo driver board as the host, in this mode, the driver board continuously sends the ID, position, and speed of the current **Active ID** servo to the driver board of the controlled slave through the ESP-NOW protocol. After the servo driver board is turned on, the **MAC:** on the first line of the screen is the MAC address of this development board, which is unique, such as MAC: 08 3A F2 93 5F A8. The premise of ESP-NOW communication is to get the MAC address of the slave, please write down the address, and fill it in the broadcastAddress[] = {0x08, 0x3A, 0xF2, 0x93, 0x5F, 0xA8} of ServoDriverST.ino, then upload it to the driver of the host and the function is enabled.

- **Follower** key, set the servo drive board as the slave, after changing the MAC address of the slave in the program of the host, it can act under the remote control of the host.
- **RainbowON** and **RainbowOFF** keys are used to turn on and off the rainbow effect of RGB lights.

Secondary Development Tutorial

- Servo initialization:

Each program for controlling the servo needs to initialize servo before it can be used.

```
#include <SCServo.h>
SMS_STS st;
void setup(){
  Serial1.begin(1000000); //Initialize the serial port, if you use ESP32 and other devices, you can also choose a custom serial port
  // Serial1.begin(1000000, SERIAL_8N1, RX, TX); // custom serial port
  st.pSerial = &Serial1;
  while(!Serial1) {}
}
```

- Change servo ID:

In the servos connected in series, each ID corresponds to only one servo, otherwise the information fed back by the servos cannot be obtained normally. When changing the servo ID, please try to ensure that the driver board is connected to only one servo, and the ID will be permanently saved in the servo.

```
#include <SCServo.h>
SMS_STS st;
int ID_ChangeFrom = 1; // Change the original servo ID, and the factory default is 1
int ID_ChangeTo = 2; // new ID
void setup(){
  Serial1.begin(1000000);
  st.pSerial = &Serial1;
  while(!Serial1) {}

  st.unLockEeprom(ID_ChangeFrom); //Unlock EPROM-SAFE
  st.writeByte(ID_ChangeFrom, SMS_STS_ID, ID_ChangeTo); //Change ID
  st.LockEeprom(ID_ChangeTo); // EPROM-SAFE is locked
}

void loop(){
}
```

- Ping:

Used to test whether a servo is connected normally.

```
#include <SCServo.h>
SMS_STS st;
int TEST_ID = 3; // Servo ID to test
void setup()
{
  Serial.begin(115200);
  Serial1.begin(1000000, SERIAL_8N1, RX, TX); // custom serial port
  st.pSerial = &Serial1;
  while(!Serial1) {}
}

void loop()
{
  int ID = st.Ping(TEST_ID); //Ping the servo with the given ID, and return -1 if it fails
  if(ID!=-1){
    Serial.print("Servo ID:");
    Serial.println(ID, DEC);
    delay(100);
  }else{
    Serial.println("Ping servo ID error!");
    delay(2000);
  }
}
```

- Write position:

Can be used to control the rotation of individual servos.

```
#include <SCServo.h>
SMS_STS st;
void setup()
{
  Serial1.begin(1000000);
  st.pSerial = &Serial1;
```



```

    while(!Serial1) {}
}

void loop()
{
    st.WritePos(1, 1000, 1500, 50); // Control the servo with ID 1 to rotate to the position of 1000
    at a speed of 1500 and start and stop the acceleration of 50.
    delay(754); //[(P1-P0)/V]*1000+100

    st.WritePos(1, 20, 1500, 50); // Control the servo with ID 1 to rotate to the position 20 at a speed
    of 1500 and start and stop the acceleration at 50.
    delay(754); //[(P1-P0)/V]*1000+100
}

```

- Synchronous write

Can be used to control multiple servos at the same time (turn to different positions and different speeds).

```

#include <SCServo.h>
SMS_STS st;

// the uart used to control servos.
// GPIO 18 - S_RXD, GPIO 19 - S_TXD, as default.
#define S_RXD 18 //Custom IO pins of the serial port. If a custom serial port is not used, replace
Serial1.begin(1000000, SERIAL_8N1, S_RXD, S_TXD); with Serial1.begin(1000000);
#define S_TXD 19

byte ID[2];
s16 Position[2];
u16 Speed[2];
byte ACC[2];

void setup()
{
    Serial1.begin(1000000, SERIAL_8N1, S_RXD, S_TXD);
    st.pSerial = &Serial1;
    delay(1000);
    ID[0] = 1; // Save the IDs of the servos that need to be controlled into the ID[]
    ID[1] = 2; // Save the IDs of the servos that need to be controlled into the ID[]
    Speed[0] = 3400; // Set the servo speed, Speed[0] should correspond to the servo with ID[0]
    Speed[1] = 3400; // Set the servo speed, Speed[1] should correspond to the servo with ID[1]
    ACC[0] = 50; // Set the start/stop acceleration. The smaller the value, the lower the acceleration.
    The maximum value that can be set is 150.
    ACC[1] = 50;
}

void loop()
{
    Position[0] = 3000; // Set the target position for the servo with ID[0] (which has an ID of 1) to
    a range of 0-4095
    Position[1] = 3000; // Set the target position for the servo with ID[0] (which has an ID of 1) to
    a range of 0-4095
    st.SyncWritePosEx(ID, 2, Position, Speed, ACC); //servo(ID1/ID2) speed=3400, acc=50, move to position=3000.
    delay(2000);

    Position[0] = 100;
    Position[1] = 100;
    st.SyncWritePosEx(ID, 2, Position, Speed, ACC); //servo(ID1/ID2) speed=3400, acc=50, move to position=100.
    delay(2000);
}

```

- Get servo information feedback.

```

#define S_RXD 18
#define S_TXD 19
#include <SCServo.h>

SMS_STS sms_sts;

void setup()
{
    Serial1.begin(1000000, SERIAL_8N1, S_RXD, S_TXD);
    Serial.begin(115200);
    sms_sts.pSerial = &Serial1;
    delay(1000);
}

void loop()
{
    int Pos;
    int Speed;
    int Load;
}

```

```

int Voltage;
int Temper;
int Move;
int Current;
if(sms_sts.FeedBack(1)!=-1){
    Pos = sms_sts.ReadPos(-1);
    Speed = sms_sts.ReadSpeed(-1);
    Load = sms_sts.ReadLoad(-1);
    Voltage = sms_sts.ReadVoltage(-1);
    Temper = sms_sts.ReadTemper(-1);
    Move = sms_sts.ReadMove(-1);
    Current = sms_sts.ReadCurrent(-1);
    Serial.print("Position:");
    Serial.println(Pos);
    Serial.print("Speed:");
    Serial.println(Speed);
    Serial.print("Load:");
    Serial.println(Load);
    Serial.print("Voltage:");
    Serial.println(Voltage);
    Serial.print("Temper:");
    Serial.println(Temper);
    Serial.print("Move:");
    Serial.println(Move);
    Serial.print("Current:");
    Serial.println(Current);
    delay(10);
}else{
    Serial.println("FeedBack err");
    delay(500);
}

Pos = sms_sts.ReadPos(1); // Get the position feedback
if(Pos!=-1){
    Serial.print("Servo position:");
    Serial.println(Pos, DEC);
    delay(10);
}else{
    Serial.println("read position err");
    delay(500);
}

Voltage = sms_sts.ReadVoltage(1); //Get the voltage feedback
if(Voltage!=-1){
    Serial.print("Servo Voltage:");
    Serial.println(Voltage, DEC);
    delay(10);
}else{
    Serial.println("read Voltage err");
    delay(500);
}

Temper = sms_sts.ReadTemper(1); //Get the temperature feedback
if(Temper!=-1){
    Serial.print("Servo temperature:");
    Serial.println(Temper, DEC);
    delay(10);
}else{
    Serial.println("read temperature err");
    delay(500);
}

Speed = sms_sts.ReadSpeed(1); //Get the speed feedback
if(Speed!=-1){
    Serial.print("Servo Speed:");
    Serial.println(Speed, DEC);
    delay(10);
}else{
    Serial.println("read Speed err");
    delay(500);
}

Load = sms_sts.ReadLoad(1); //Get the load feedback
if(Load!=-1){
    Serial.print("Servo Load:");
    Serial.println(Load, DEC);
    delay(10);
}else{
    Serial.println("read Load err");
    delay(500);
}

Current = sms_sts.ReadCurrent(1); //Get the current feedback
if(Current!=-1){
    Serial.print("Servo Current:");
    Serial.println(Current, DEC);
}

```

```

        delay(10);
    }else{
        Serial.println("read Current err");
        delay(500);
    }

    Move = sms_sts.ReadMove(1);    //Get the movement feedback
    if (Move!=-1){
        Serial.print("Servo Move:");
        Serial.println(Move, DEC);
        delay(10);
    }else{
        Serial.println("read Move err");
        delay(500);
    }
    Serial.println();
}

```

Resource

3D Module

- [ST3215 Servo STEP Module](#)
- [Servo Driver Step Module](#)

2D Drawing

- [2D Drawing](#)

Document

- [Schematic](#)
- [User Manual](#)
- [Communication Protocol User Manual](#)
- [scservo](#)

Demo

- [Demo](#)

Software

- [Arduino IDE](#)

Open Source Structure

- [12DOF ROBOTIC DOG STEP MODEL](#)
- [12DOF Robotic Dog Fusion360 Project](#)
- [12DOF ROBOTIC DOG Creo5.0 Project](#)

FAQ

Question:Would it be possible please, to get the full register list for the st3215 serial servo?

Answer:

You can check the attached st3215 serial servo memory table.

[memory table](#)

Question:What is the precise connector type of ST3215 Servo?

Answer:

It is 5264/2.54 3P connector.

Question:Can you give me an indication of the life of ST3215 in terms of cycles or hours?

Answer:

There is no certain value of the life, according to the reliability testing, the life of the servo should be more than 100000 times (forward for 0.25s, stop for 0.5s backward for 0.25s, and then stop for 0.5s, take as one time).

Question:Does servo work on half duplex or full duplex signal?

Answer:

It is half duplex signal.

Support

Technical Support

If you need technical support or have any feedback/review, please click the **Submit Now** button to submit a ticket, Our support team will check and reply to you within 1 to 2 working days. Please be patient as we make every effort to help you to resolve the issue.

Working Time: 9 AM - 6 PM GMT+8 (Monday to Friday)

Submit Now

