

ServoSDK Manual

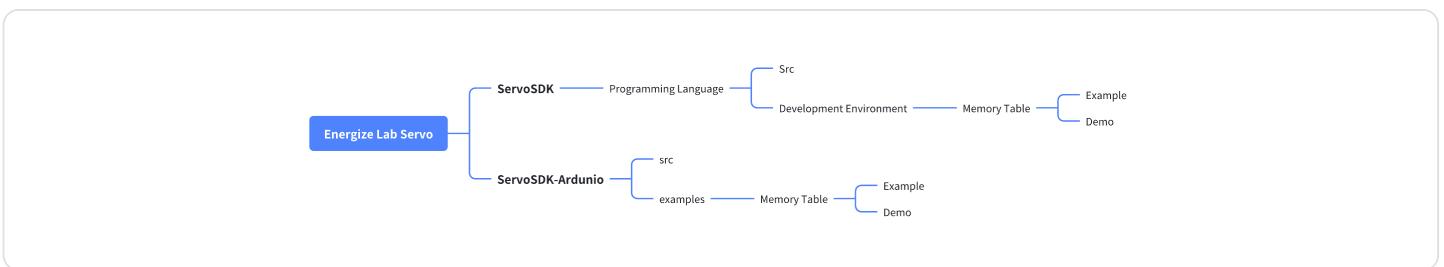
1. Introduction

Welcome to ServoSDK, an open-source software development kit designed specifically for the Energize Lab Servo. This SDK provides multi-language, multi-platform servo usage examples and motion control demos, aimed at helping users easily and quickly master the use of various models, including the EM-2030. It is strongly recommended to refer to the corresponding memory table parameter for understanding and verification.

- Supported Protocols: Energize Lab Servo Communication Protocol
- Supported Memory Tables: Primary Memory Table
- Supported Servos: EM Series, EH-3030
- Supported Programming Languages:
 - Arduino: Includes source code, with compatible examples and demos across multiple platforms.
 - C: Includes source code, with examples and demos based on Windows, STC89C52, and STM32F103C8T6.
 - C++: Includes source code, with examples and demos based on Windows.
 - MicroPython: Includes source code, with examples and demos based on ESP32.
 - Python: Includes source code, with examples and demos based on Windows.

2. Repository

2.1 File Structure



- Src/src: Source code that includes instruction generation and parsing for the entire memory table of servo.

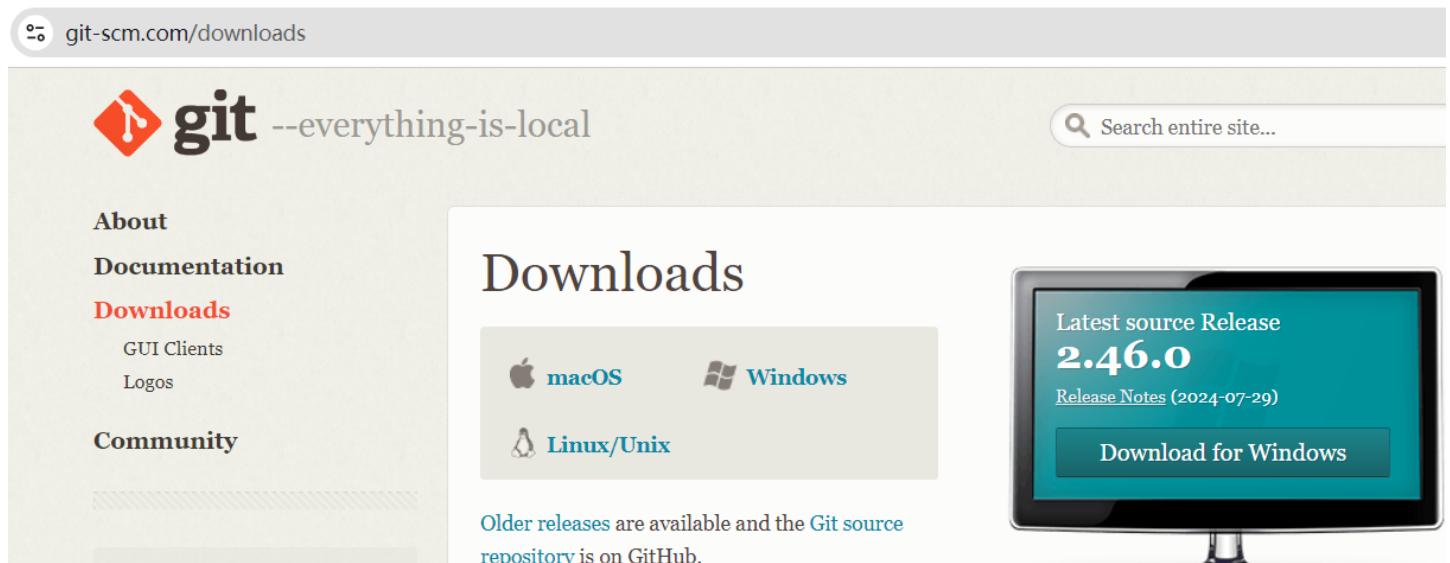
- Example: Demonstrate the use of Ping instruction, read data instruction, write data instruction, sync write instruction, parameter reset instruction, factory reset instruction, and reboot instruction.
- Demo: Demonstrate the motion of the servo in different control modes.

2.2 Download Repository

- Method 1: Download using the Git command. This offers the most features and is suitable for developers and users who need to update code frequently.
- Method 2: Download using a browser.
 - Open with GitHub Desktop. A graphical interface, suitable for users who are not familiar with the command line.
 - Download ZIP. The simplest option, suitable for users who only need to obtain the code once.

2.2.1 Use Git Command

Step 1: Install Git. Visit <https://git-scm.com/downloads> and select the appropriate version of Git to download and install based on your computer's operating system.

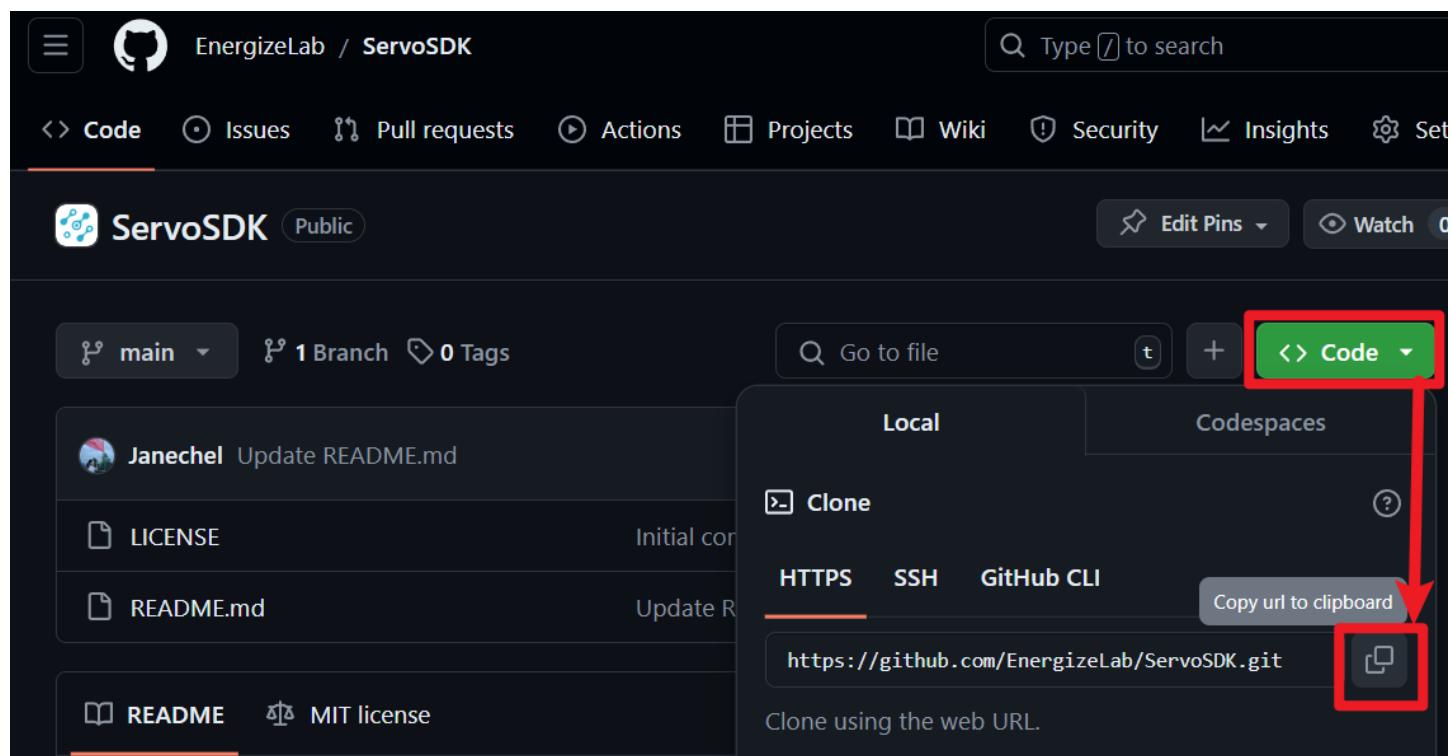


Step 2: Open the Terminal. Open the terminal on your computer and use the 'cd' command to enter the local folder directory where you want to download the repository. For example, type 'cd D:\EnergizeLab' and press Enter.

```
Windows PowerShell x + v
Windows PowerShell
版权所有 (C) Microsoft Corporation。保留所有权利。
安装最新的 PowerShell, 了解新功能和改进! https://aka.ms/PSWindows
PS C:\Users\Janechel> cd D:\EnergizeLab ←
PS D:\EnergizeLab> |
```

Taking Windows 11 as an example

Step 3: Clone the Repository. After entering the local directory, type 'git clone' followed by the target repository's 'URL'. For example, type 'git clone <https://github.com/EnergizeLab/ServoSDK.git>' and press Enter.

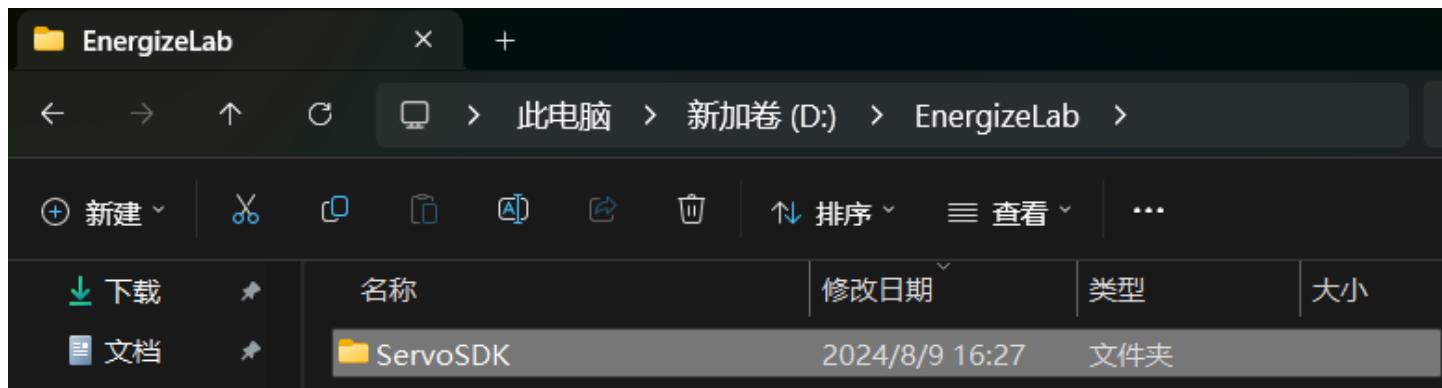


Method to obtain the target repository's 'URL'

```
Windows PowerShell x + v
Windows PowerShell
版权所有 (C) Microsoft Corporation。保留所有权利。
安装最新的 PowerShell, 了解新功能和改进! https://aka.ms/PSWindows
PS C:\Users\Janechel> cd D:\EnergizeLab
PS D:\EnergizeLab> git clone https://github.com/EnergizeLab/ServoSDK.git ←
Cloning into 'ServoSDK'...
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 7 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (7/7), done.
PS D:\EnergizeLab> |
```

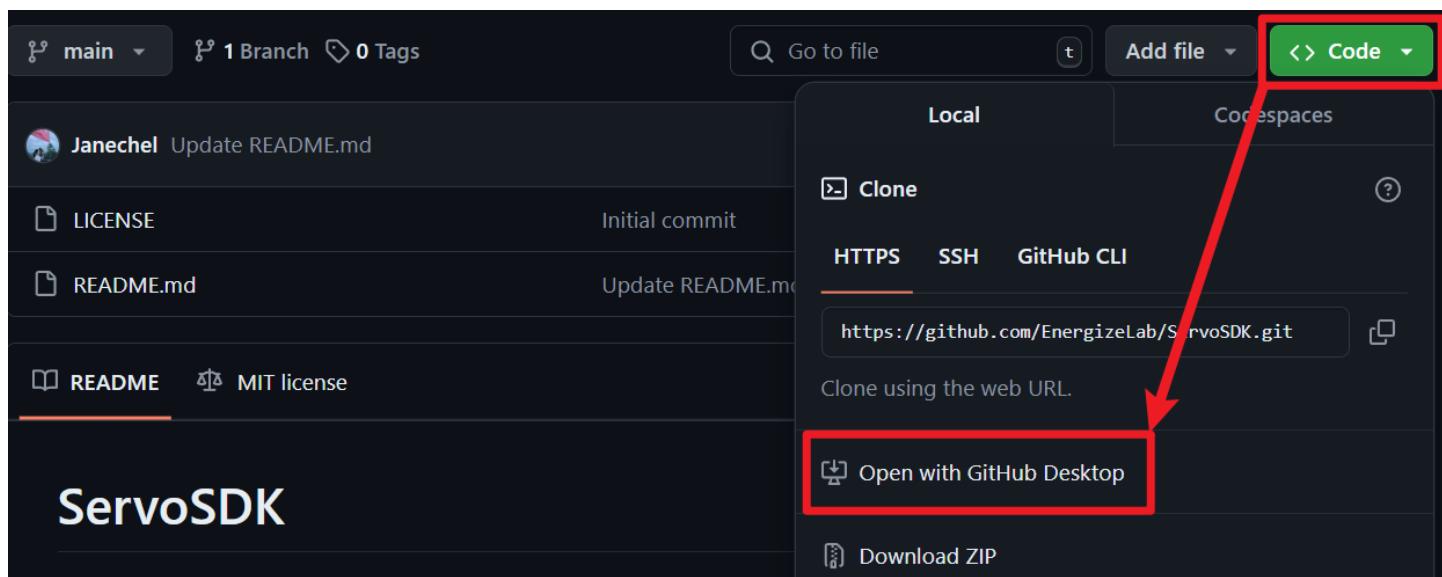
Taking Windows 11 as an example

After the download is complete, you can view the repository in the local folder.

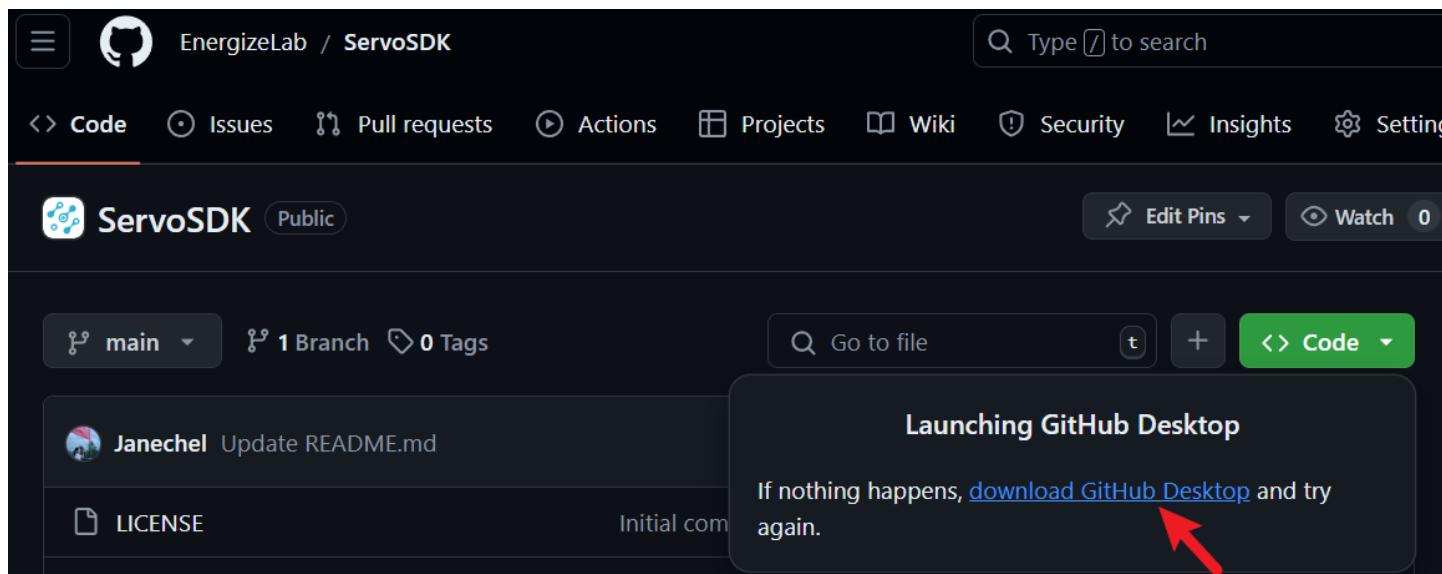


2.2.2 Open with GitHub Desktop

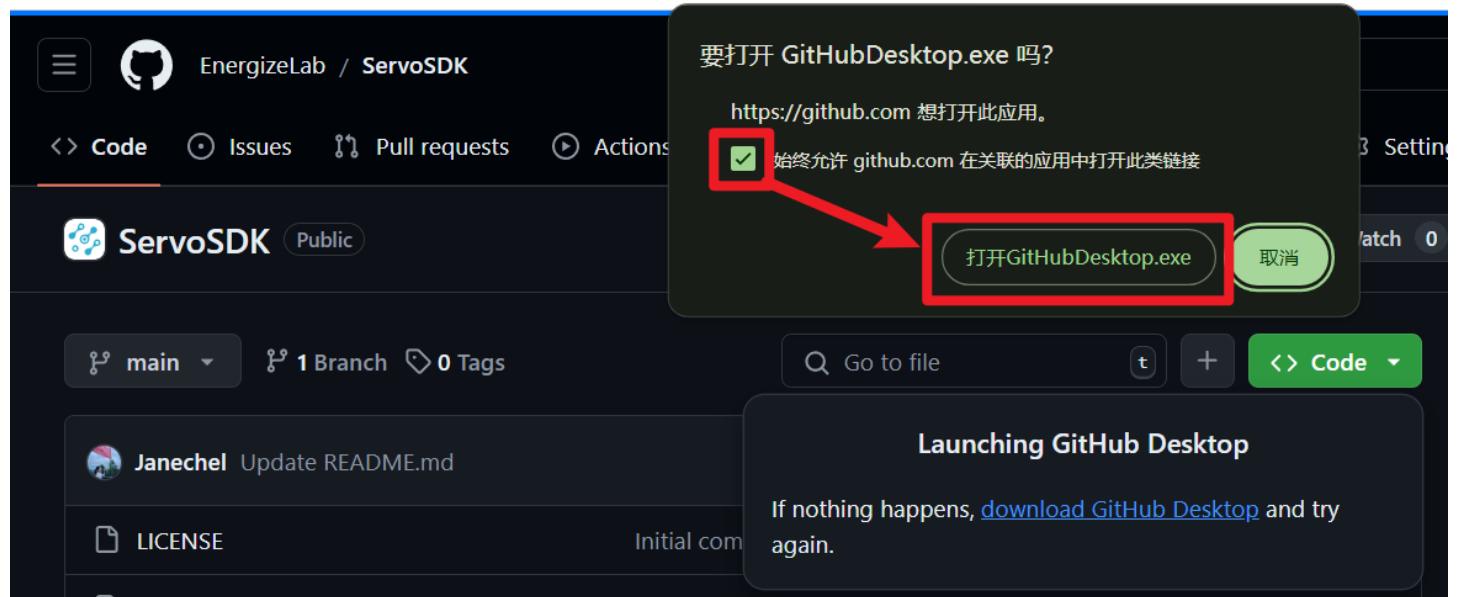
Step 1: Visit <https://github.com/EnergizeLab/ServoSDK>, click 'Code,' and select 'Open with GitHub Desktop.'



Step 2: Install GitHub Desktop. Click the hyperlink and follow the prompts to complete the installation.

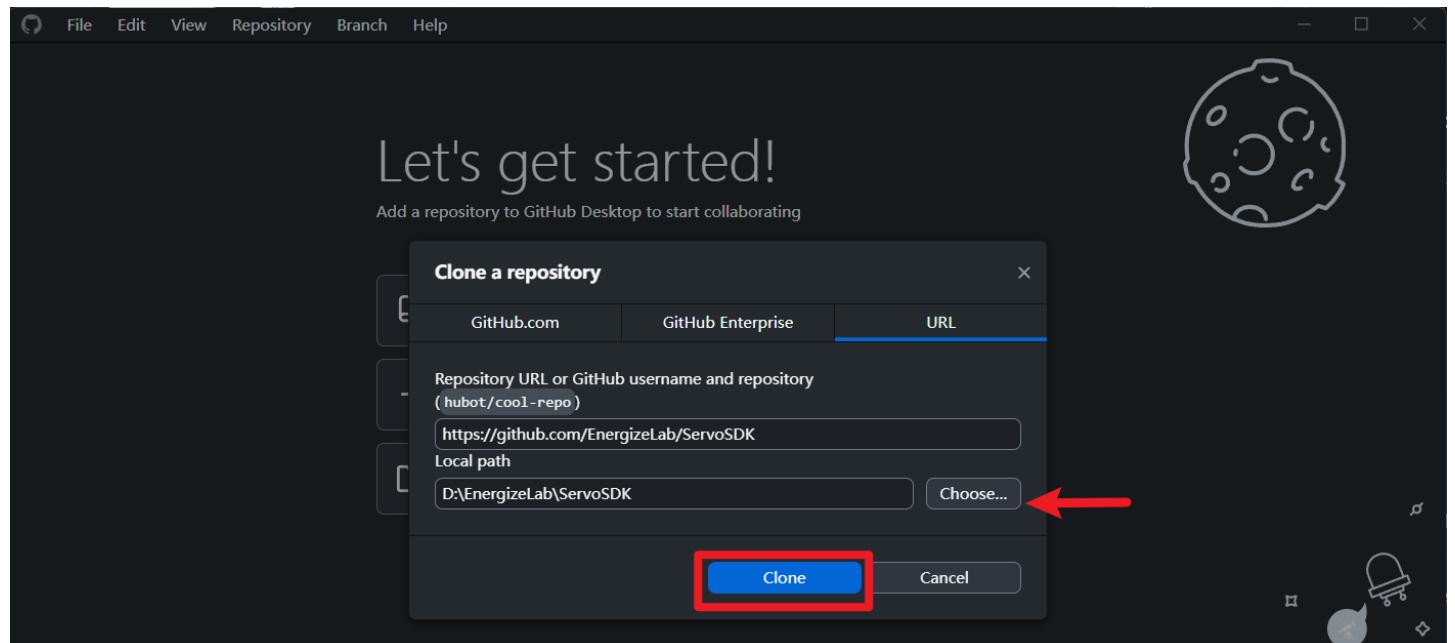


Step 3: Open with GitHub Desktop. Again, select and check "Always allow... to open this type of link," then click to open.



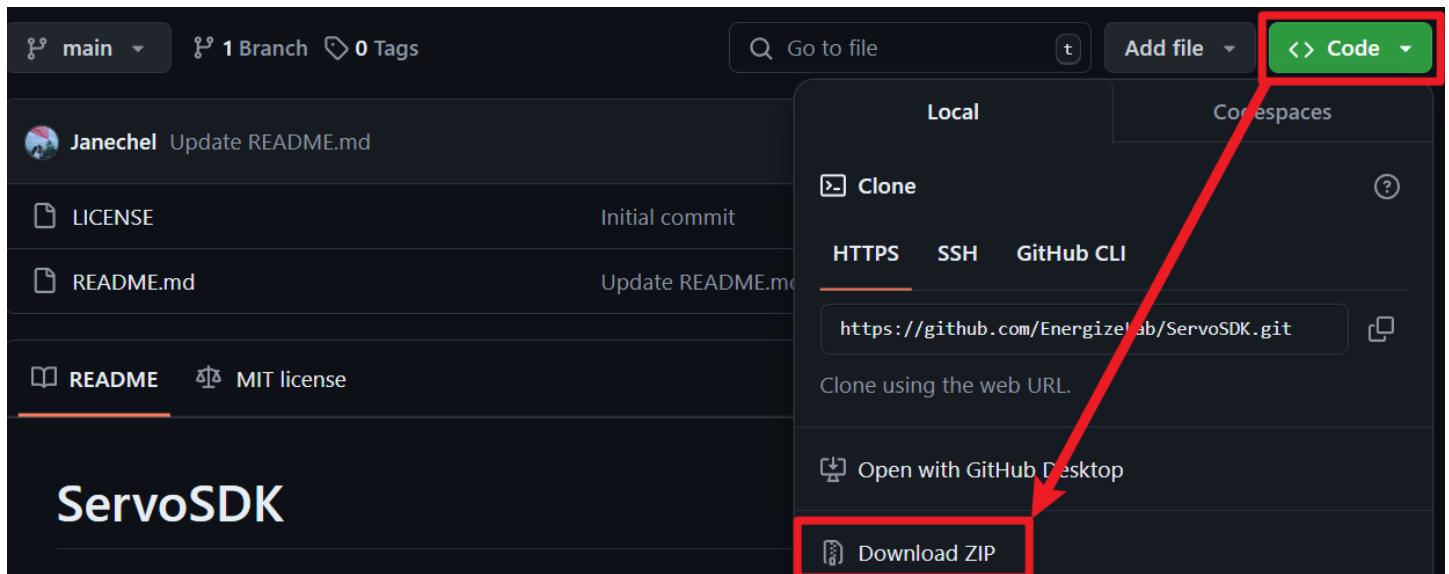
Taking Windows 11 as an example

Step 4: Clone the Repository. Choose the local folder where you want to download the repository and click 'Clone.'

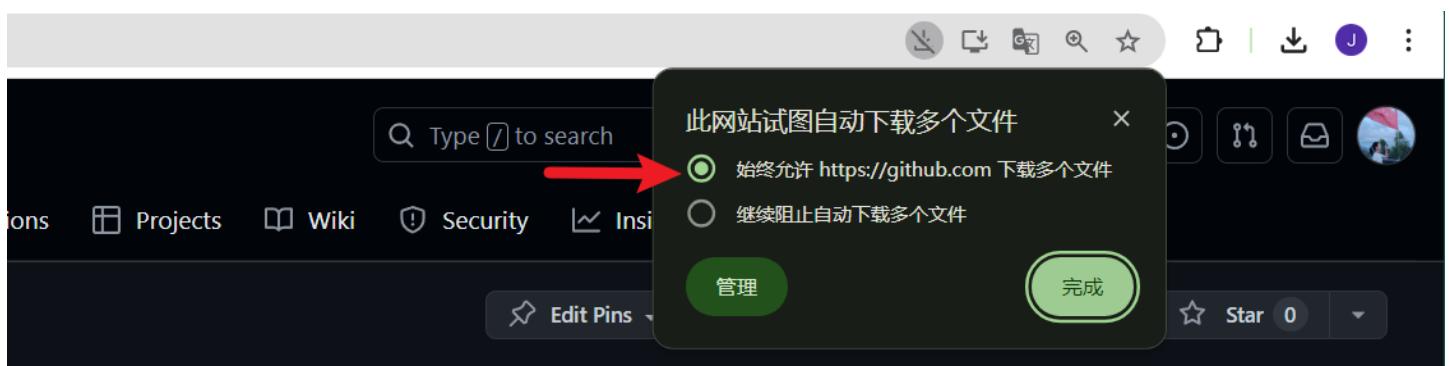


2.2.3 Download ZIP

Step 1: Visit <https://github.com/EnergizeLab/ServoSDK>, click 'Code', and select 'Download ZIP'.

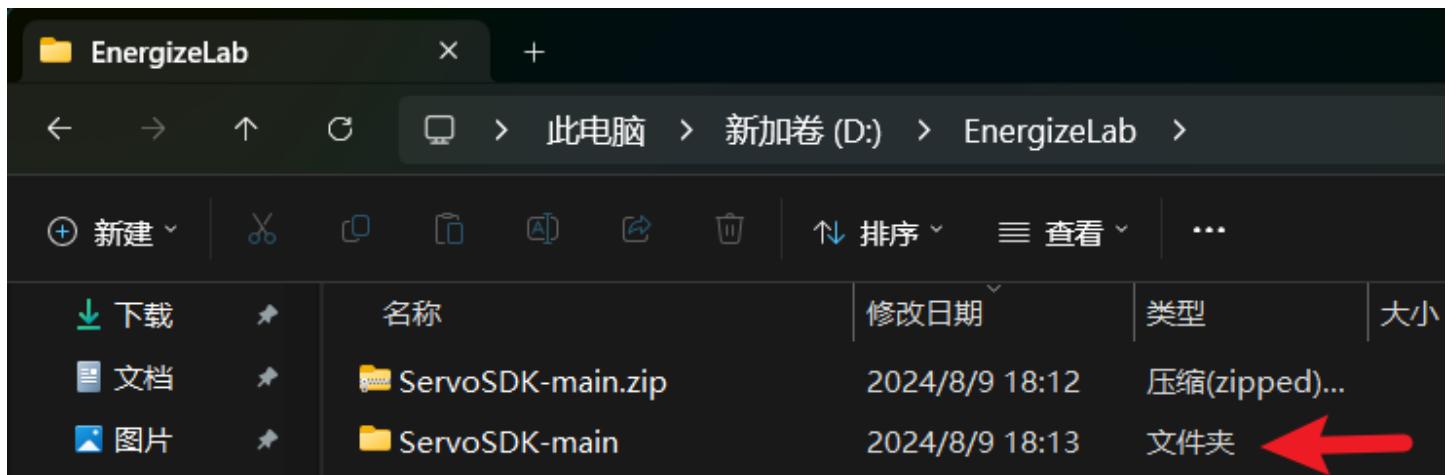


Step 2: Allow multiple file downloads. Check 'Always allow... to download multiple files' and click 'Done'.



Step 3: Clone the repository. Select the local folder where you want to download the repository and click "Save."

Step 4: Extract. After the download is complete, you need to extract the files.



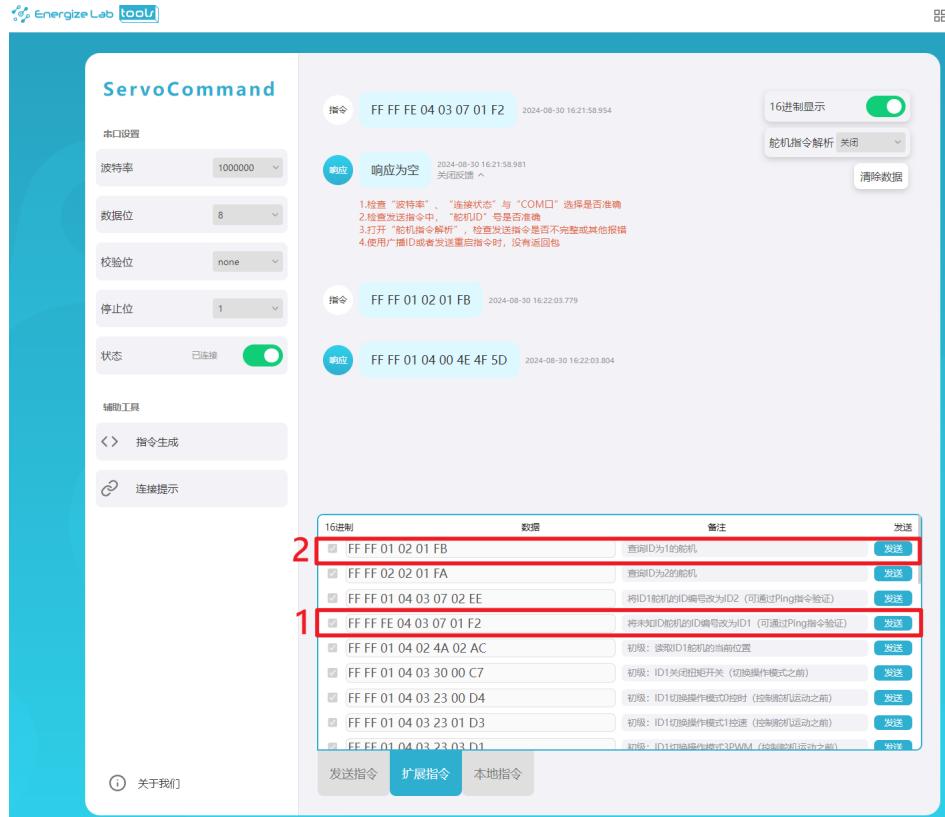
3. Servo

3.1 Preparation

To run the ServoSDK Example and Demo, you need to change the following parameters of the servo:

- ID = 1 (for testing Sync Write instruction, use 1 and 2 respectively)
- Baud Rate = 1 Mbps (the corresponding value in the memory table is 7, which usually doesn't need to be changed)

To change the servo parameters, please use ServoStudio or [ServoCommand](#). Refer to the corresponding user manual for details.

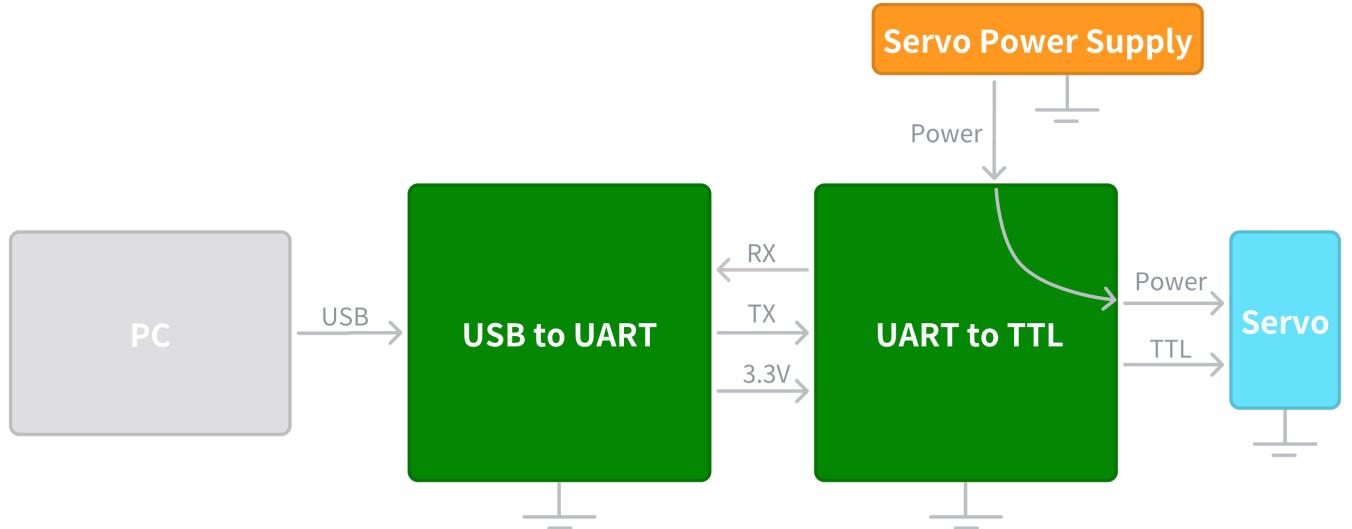


Taking modifying a single servo ID = 1 as an example

3.2 Communication Circuits

3.2.1 Servo and PC Communication

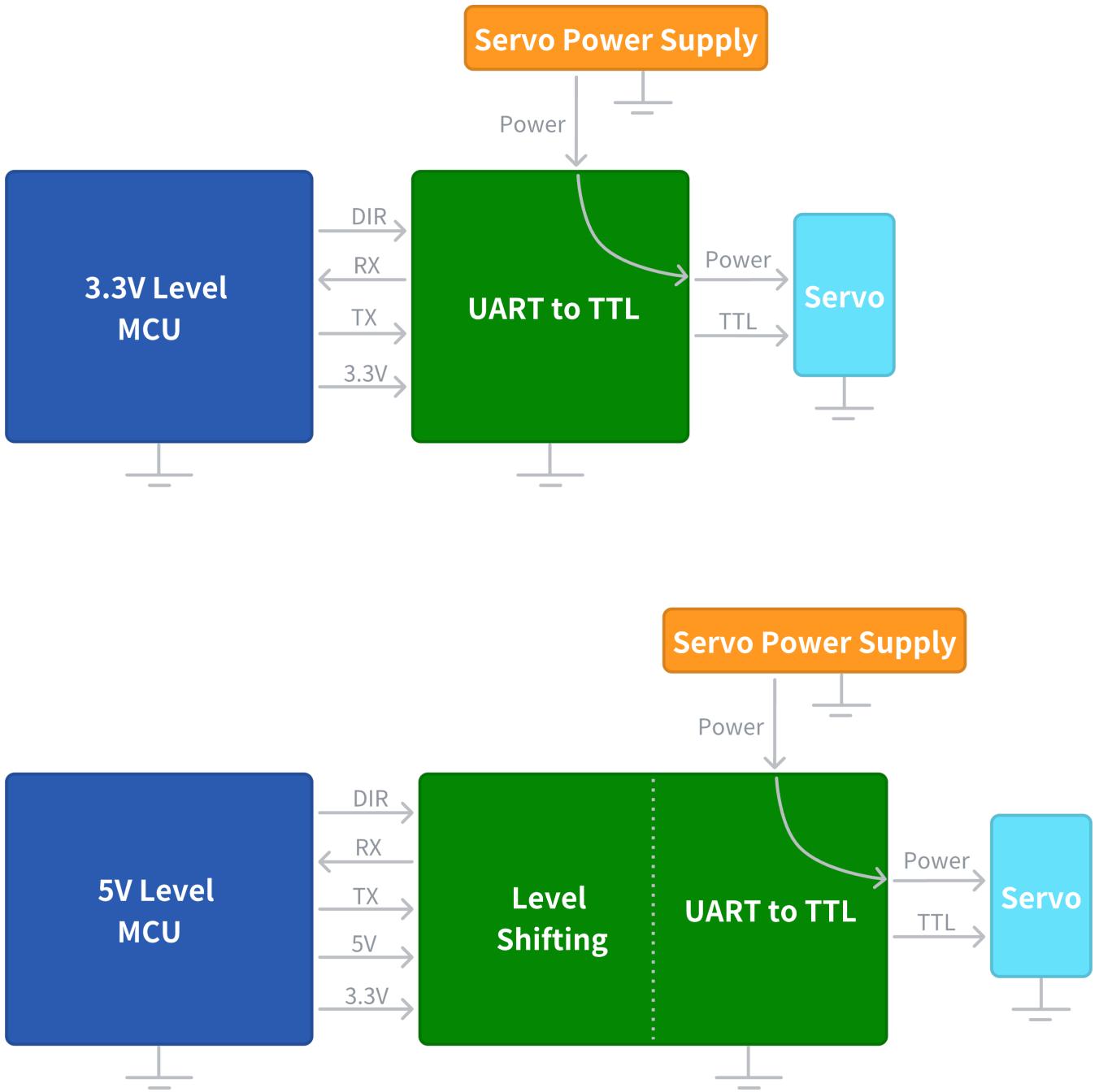
To enable communication between the Energize Lab Servo and a PC, a UART to TTL module and a USB to UART module is required, as illustrated in the diagram below:



Servo and PC Connection Diagram

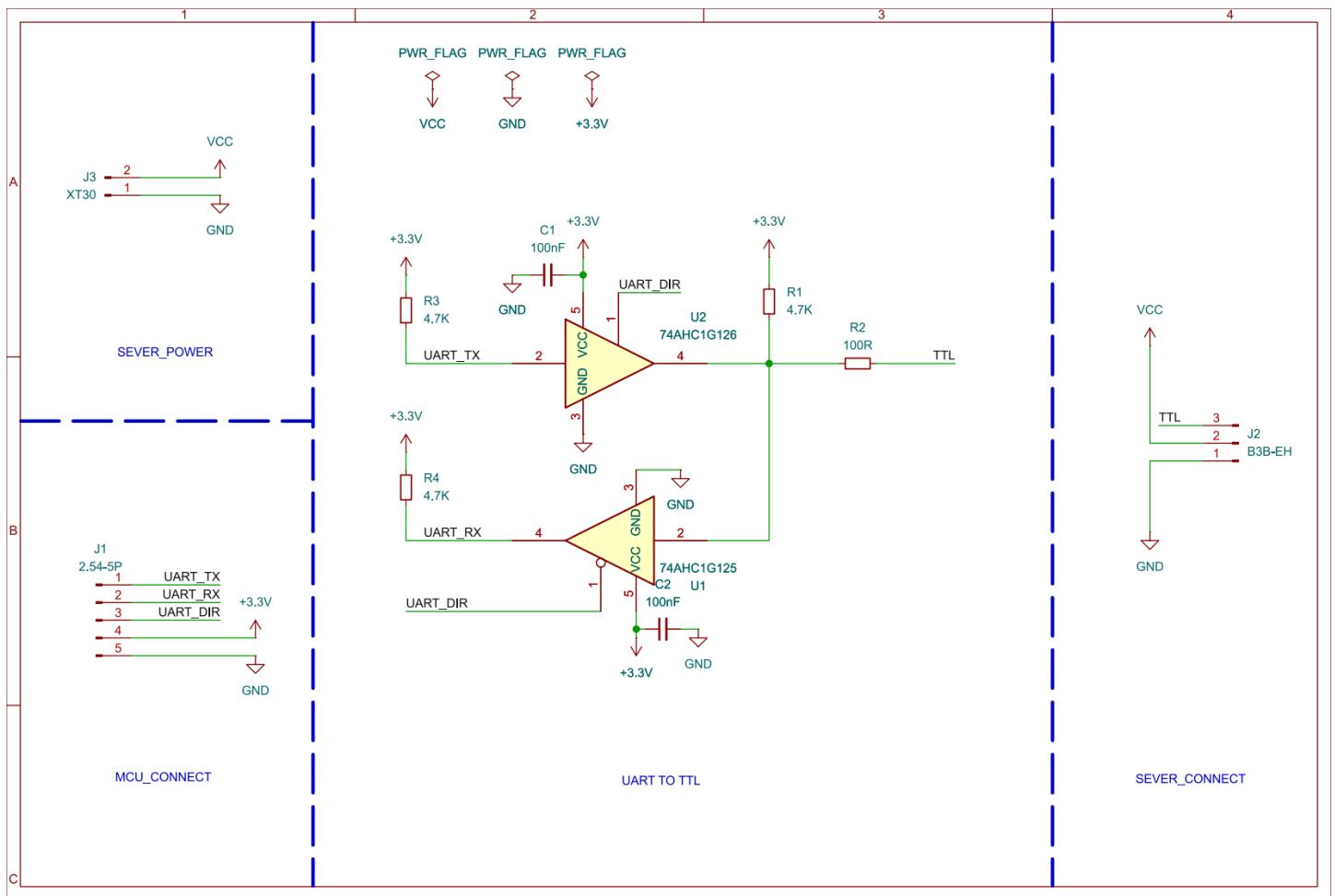
3.2.2 Servo and MCU Communication

To enable communication between the Energize Lab Servo and the MCU, a UART to TTL module is needed; if the MCU operates at 5V, a level shifting module is also necessary, as illustrated in the diagram below:

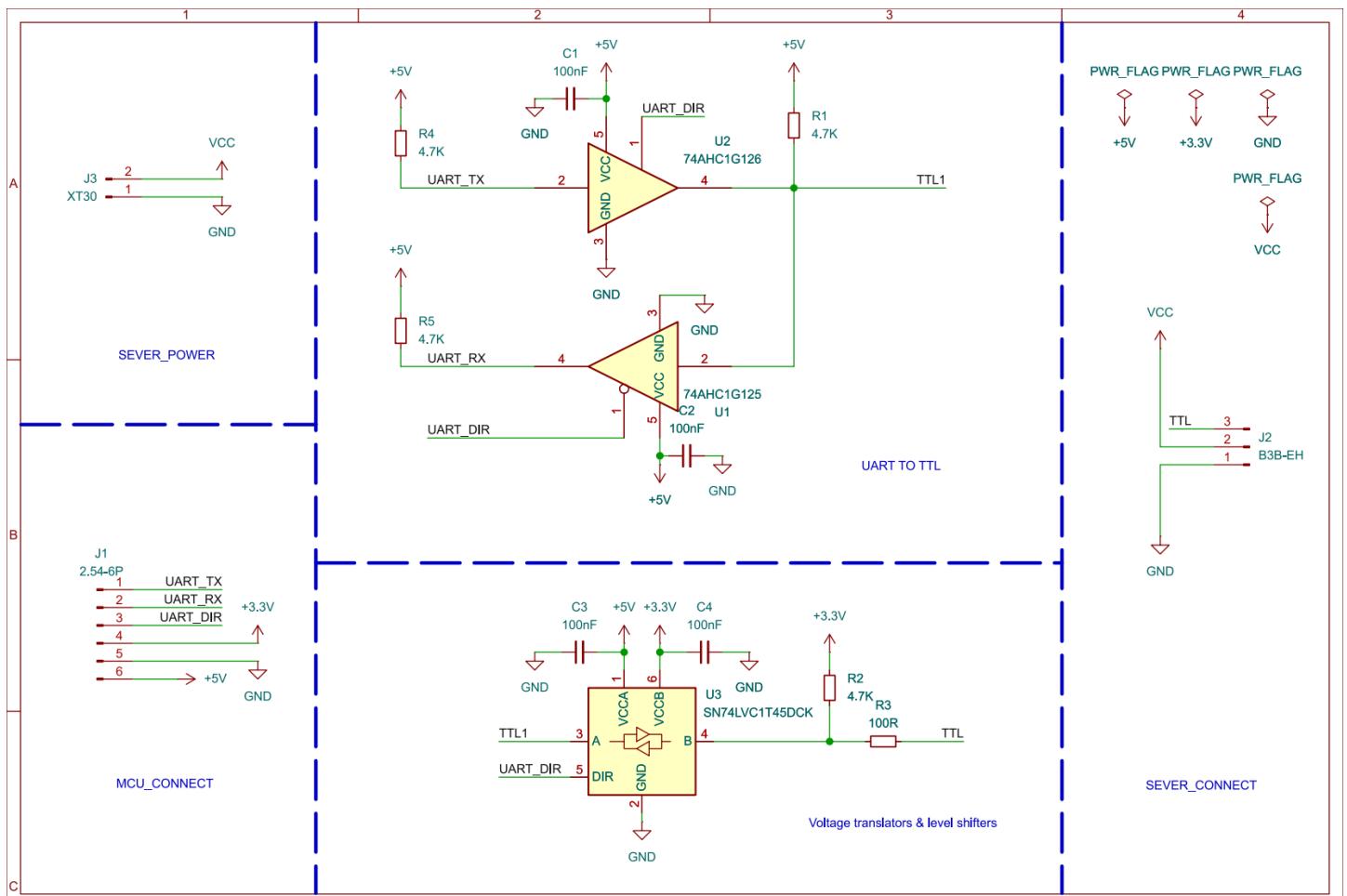


Servo and MCU Connection Diagram

UART to TTL Circuit Diagram



UART to TTL Module Circuit Diagram



Level Shifting & UART to TTL Module Circuit Diagram

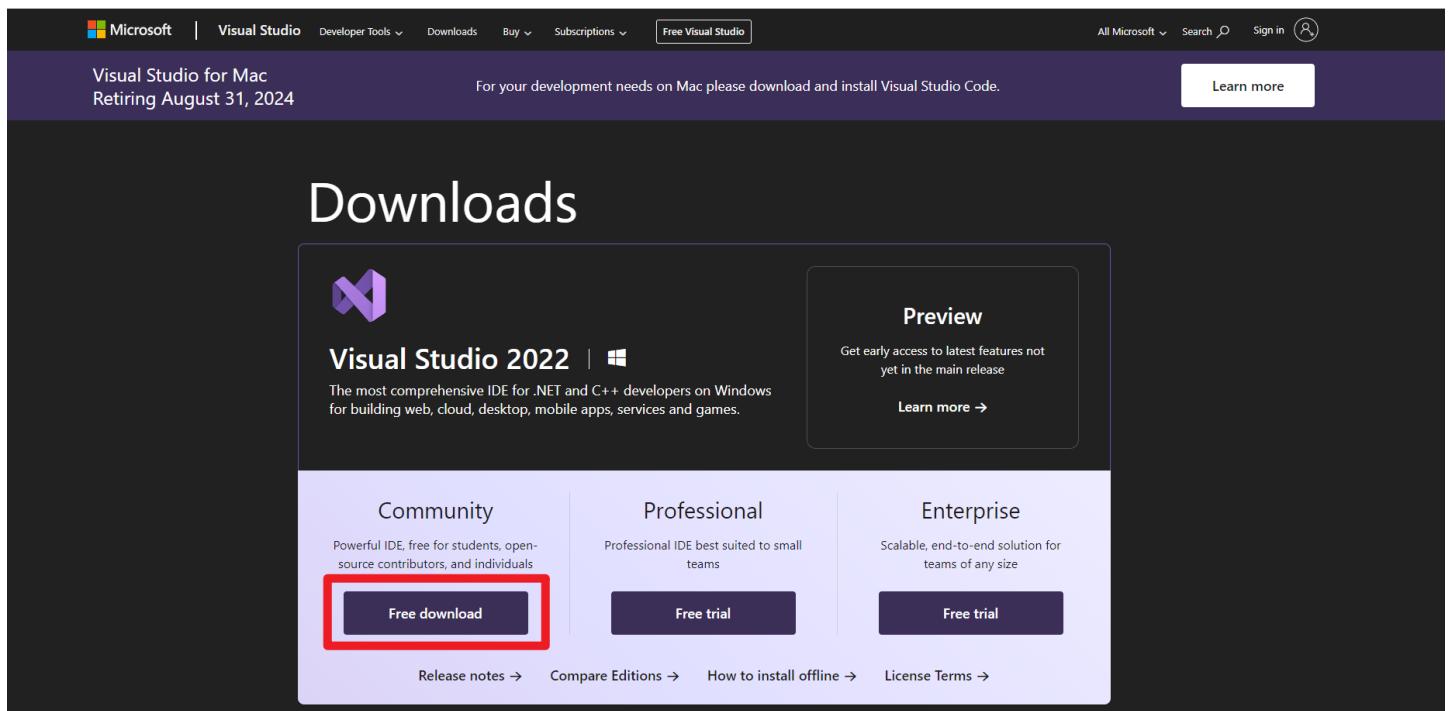
4. C

4.1 Windows

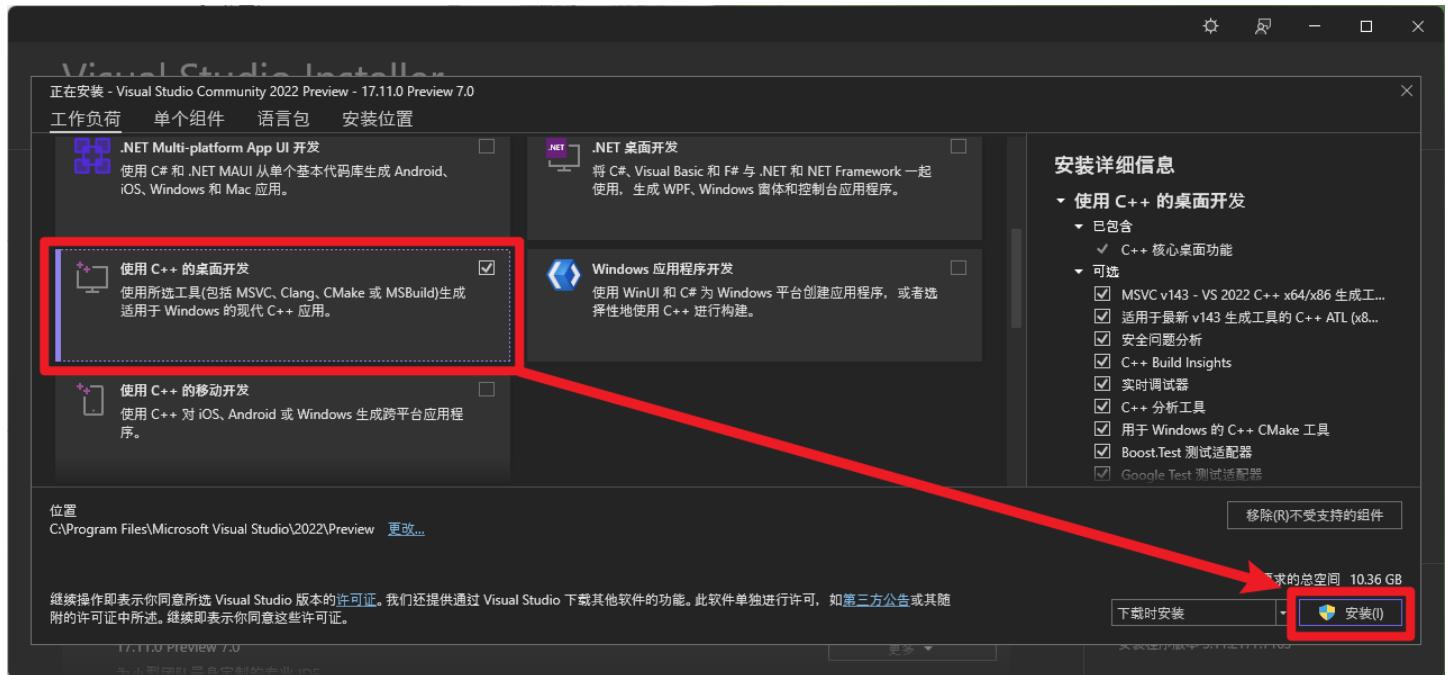
4.1.1 Environment Setup

Set up a C programming environment using the IDE, taking Visual Studio 2022 as an example.

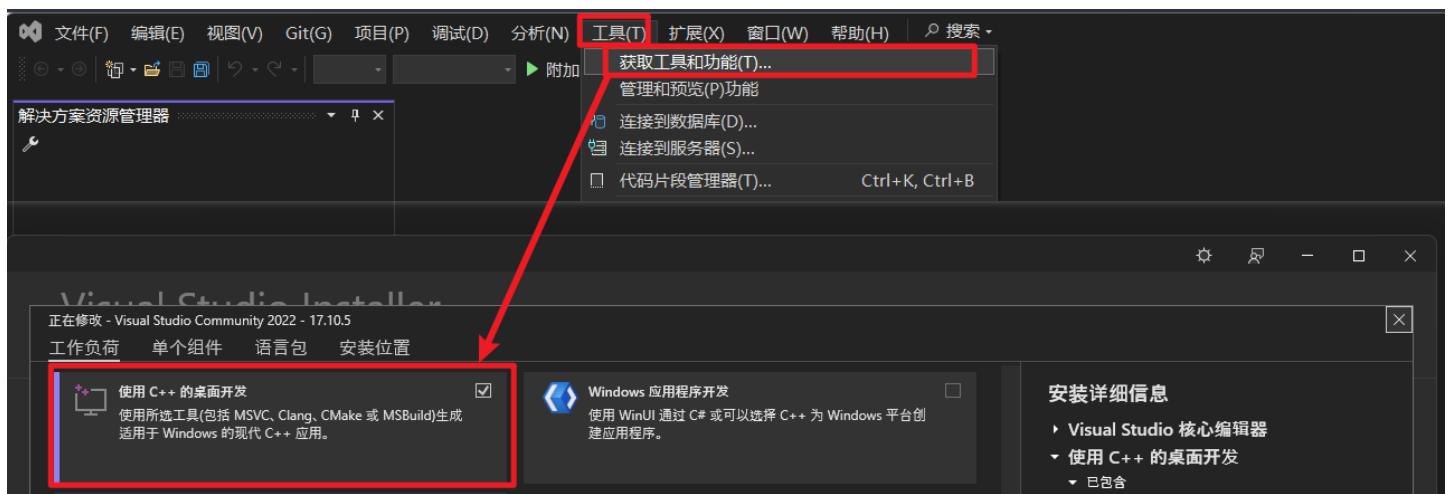
Step 1: Install Visual Studio 2022. Visit <https://visualstudio.microsoft.com/downloads/> and select the appropriate version to download and install based on your computer's operating system.



Step 2: Install C desktop development. C is included under C++, so select 'Desktop development with C++' in the workload options.

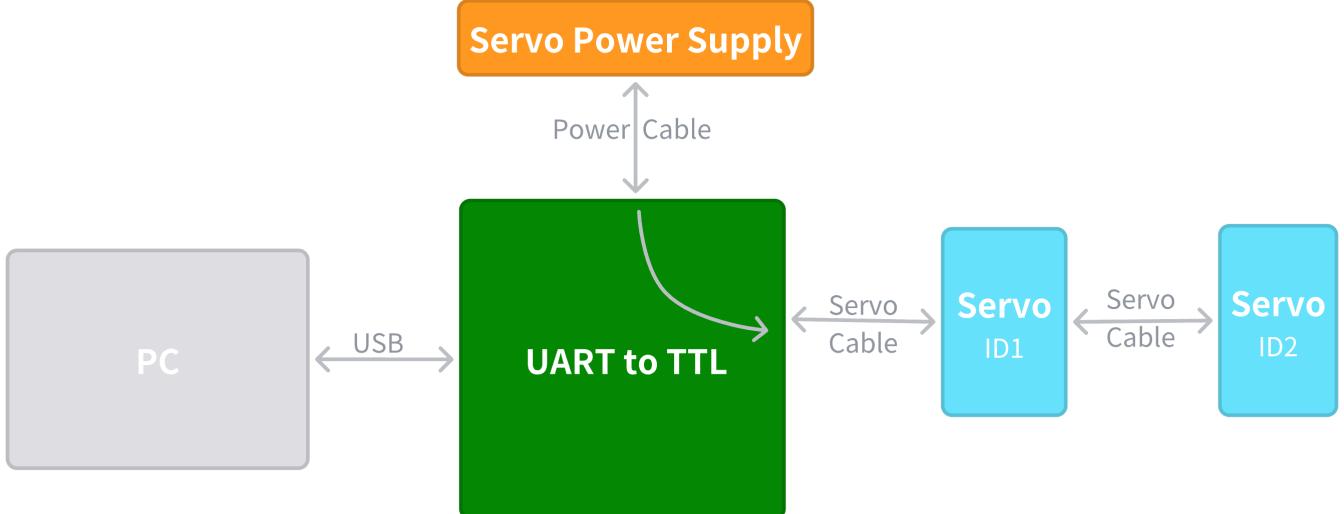


When installing the software, check 'Desktop development with C++' directly in the workload.



If not checked, you can also install it through 'Tools - Get tools and features'!

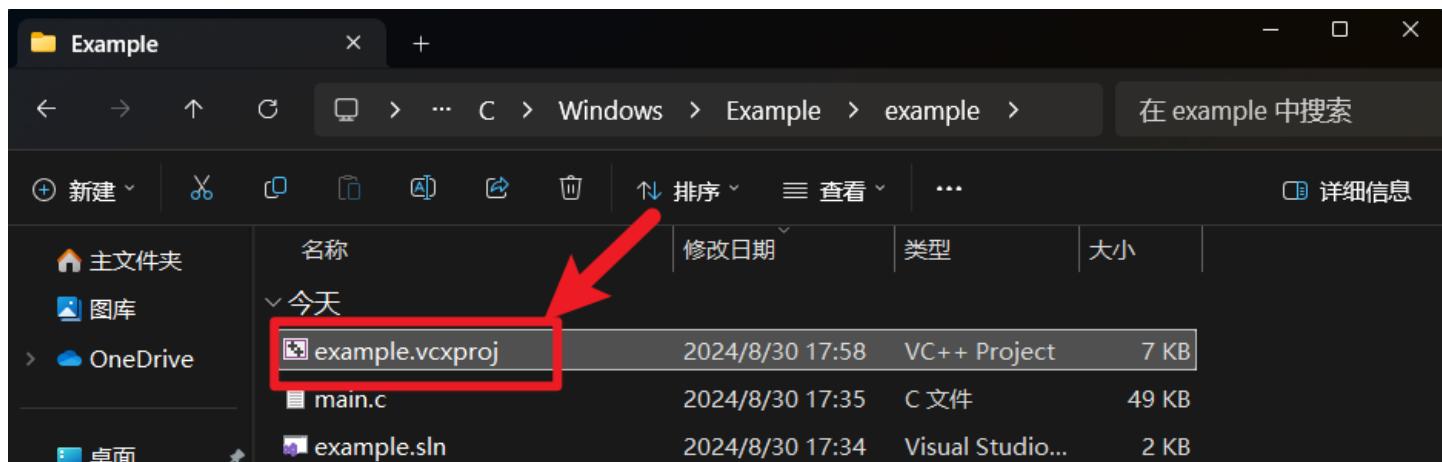
4.1.2 Hardware Connection



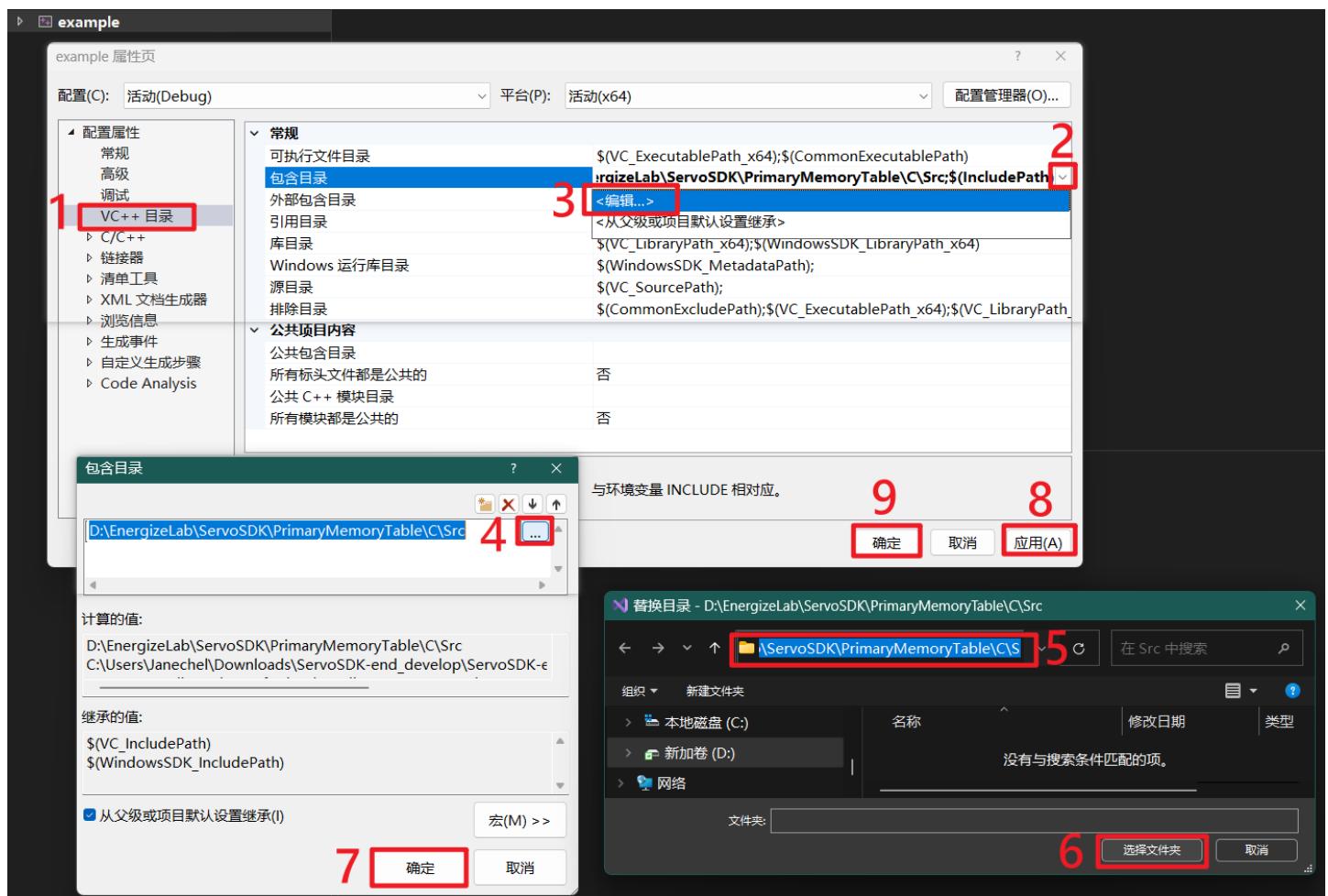
PC and Servo Connection Diagram

4.1.3 Run the Example

Step 1: Open the example.vcxproj file. Taking EM-2030 as an example, double-click to open the example.vcxproj located in ServoSDK\Windows\PrimaryMemoryTable\Example\example.



Step 2: Replace the include directory with the local Src path. Navigate through 'Debug -> Example debug properties -> VC++ Directories -> Include Directories dropdown -> Edit -> More -> Find local Src -> Select folder -> Confirm -> Apply -> OK'.



To avoid issues with missing .h files, it is recommended to set the Configuration (C) and Platform (P) to 'All Configurations' and 'All Platforms'

Step 3: Change the actual USBLINK port number. In line 121 of the main.c file, change '\\\\.\\COM18' to the actual USBLINK port number.

Note: In C/C++, since Microsoft's predefined standard devices only include 'COM1-COM9', if the user connects to a hardware port 'COM10 or above', a specific format must be used, such as

'\\\\.\\\\COM10'. Otherwise, after starting debugging, you will encounter 'Fail to open port'.

```
1 //Taking COM6 as an example
2 HANDLE hSerial = CreateFile("COM6", GENERIC_READ | GENERIC_WRITE, 0, NULL,
    OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
```

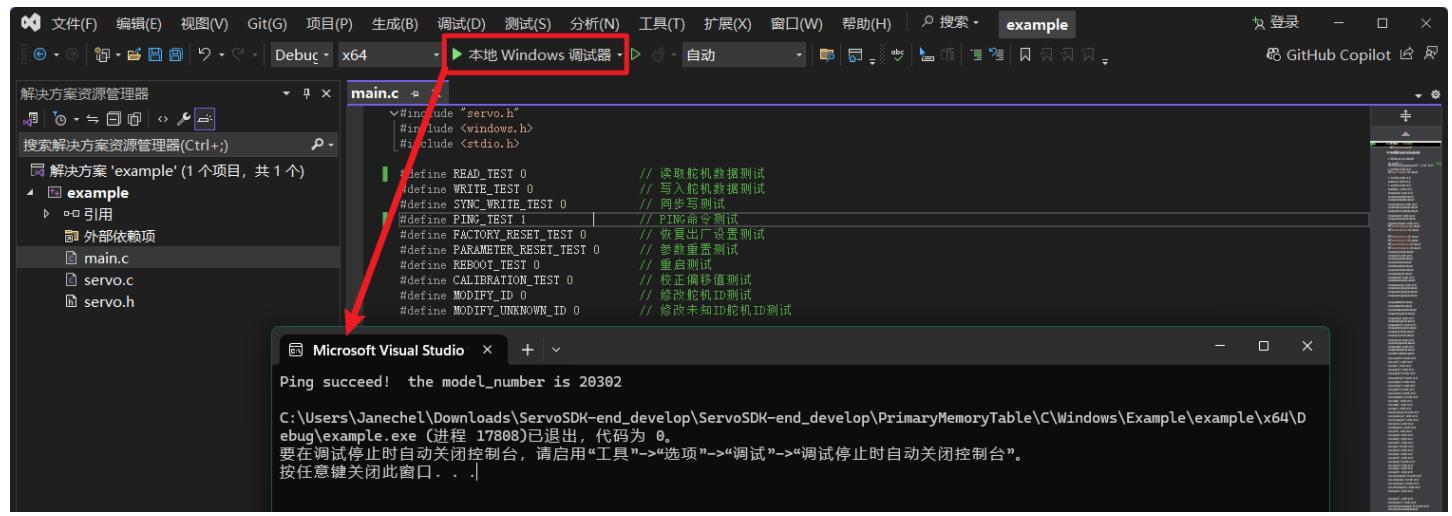
Step 4: Change the module value you want to test to 1. In lines 5-14 of the main.c file, change the module value from '0' to '1' for the module you want to test.

```
1 //Taking test the Ping module as an example
2 #define READ_TEST 0                      //Read Servo Data Test
3 #define WRITE_TEST 0                     //Write Servo Data Test
4 #define SYNC_WRITE_TEST 0                //Sync Write Test
5 #define PING_TEST 1                      //PING Instruction Test
6 #define FACTORY_RESET_TEST 0            //Factory Reset Test
7 #define PARAMETER_RESET_TEST 0        //Parameter Reset Test
8 #define REBOOT_TEST 0                  //Reboot Test
9 #define CALIBRATION_TEST 0            //Calibration Test
10 #define MODIFY_ID 0                  //Change Known Servo ID Test
11 #define MODIFY_UNKNOWN_ID 0          //Change Unknown Servo ID Test
```

Step 5: Change the maximum number of servos for sync write and the print output enable as needed. In lines 4-5 of the PrimaryServo.h file, MAX_SERVERS indicates the maximum supported number of sync write servos. If the actual number of servos exceeds 20, change it accordingly; it should be greater than or equal to the actual number of servos. If print output is not needed, set PRINTF_ENABLE to '0'.

```
1 //Taking 1 servo and print enable as an example (no change is required)
2 #define MAX_SERVERS 20                 //Maximum number of servos for sync write.
3 #define PRINTF_ENABLE 1                //Print output enable.
```

Step 6: Run the program. Click the 'Local Windows Debugger' button in VS to run the program.



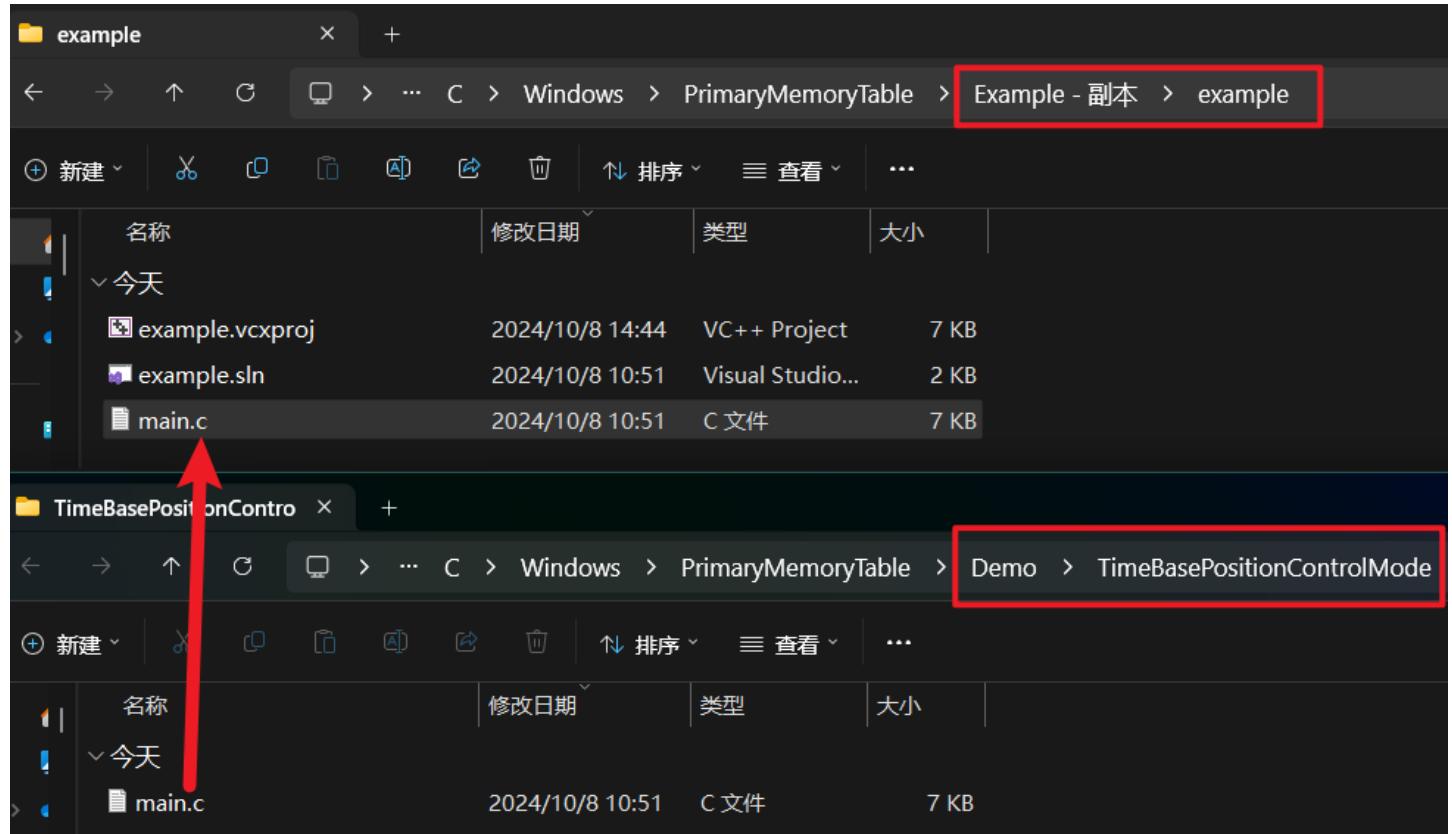
Taking test the Ping module as an example

4.1.4 Run the Demo

In the same programming language, the servo operation libraries (Src) across different platforms are identical. To run the Demo, one simply needs to replace the main file in the platform's Example with a main file from the Demo that corresponds to a specific motion control.

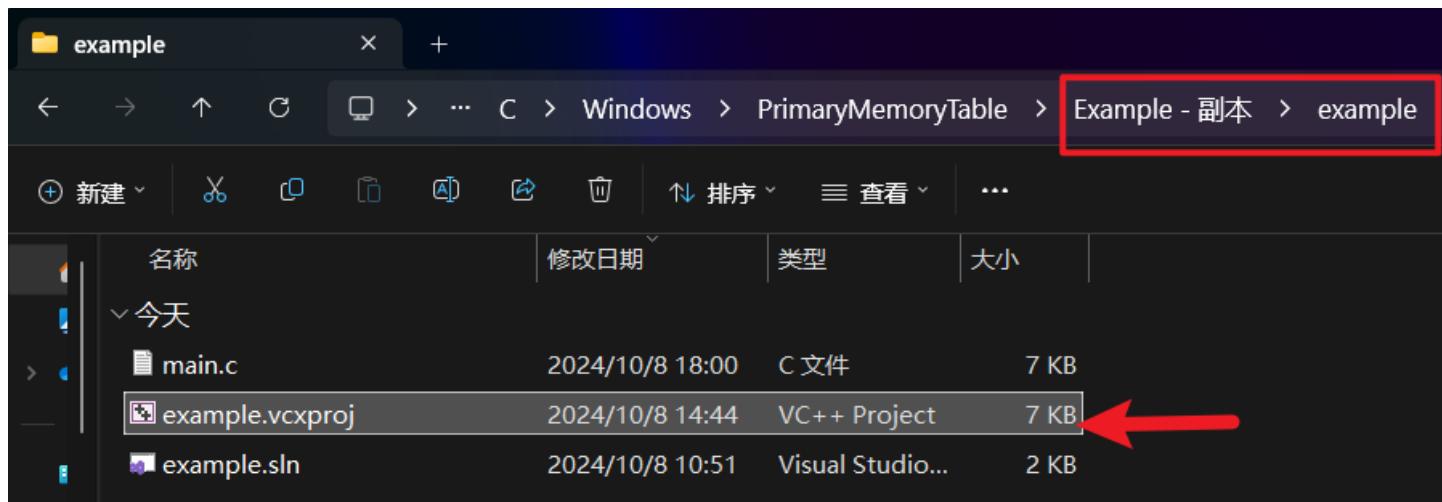
For instance, taking the C language, Windows platform, and the EM-2030 time base position control Demo as an example.

Step 1: Create a copy of the Example and replace the main file. Replace the main file in the copy with the main file from a specific motion control Demo.



delete the main file in the copy and copy the main file from the Demo into the copy

Step 2: Open the project file in the copy.



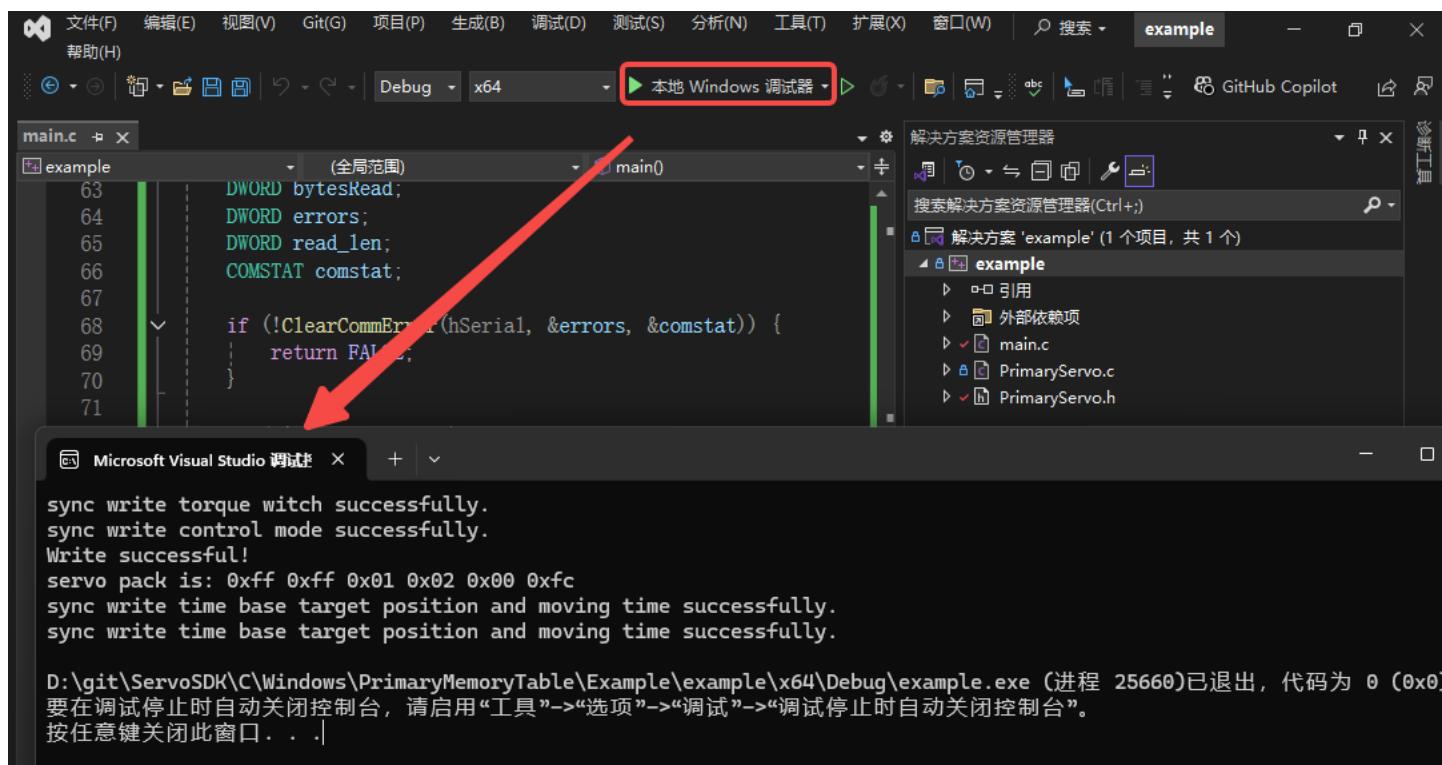
Taking open the `example.vcxproj` file as an example

Step 3: Replace the included directories with the local Src. Refer to *section 4.1.3 Run the Example*; if it has already been replaced, you can skip this step.

Step 4: Change the actual USBLINK port number. In the `main.c` file, change the code on line 105 from `"\.\.\COM16"` to the actual USBLINK port number.

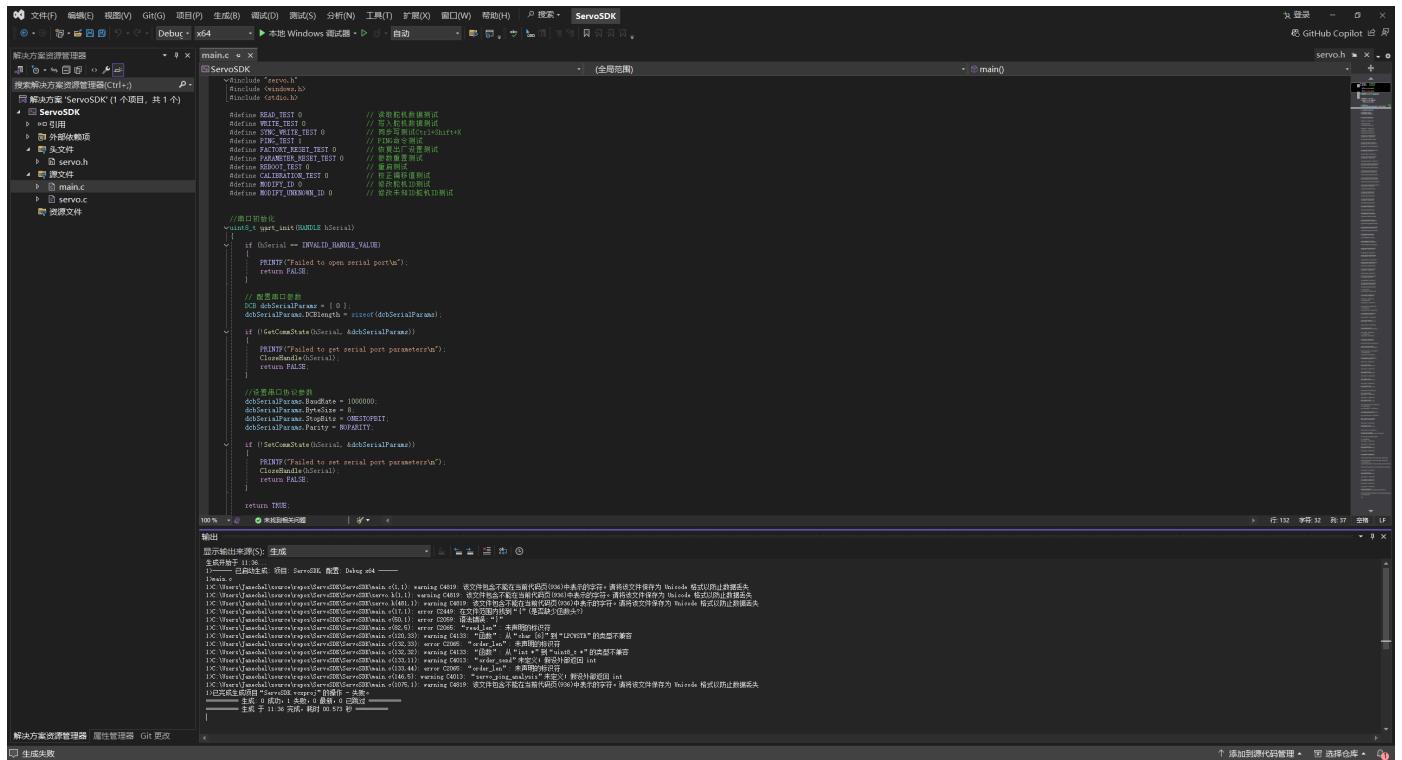
```
1 //Taking COM8 as an example
2 HANDLE hSerial = CreateFile("COM8", GENERIC_READ | GENERIC_WRITE, 0,
    NULL,OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
```

Step 5: Run the program.



4.1.5 FAQ

- If users are debugging the program by selecting "Create New Project," directly copying main.c, PrimaryServo.c, and PrimaryServo.h from C\Windows\Example into their newly created project folder, and then adding these three files to the project using "Add Existing Item," they might encounter errors such as "This file contains characters that cannot be represented in the current code page."



```
#include "main.h"
#include "servo.h"
#include "servo.h"
#include "config.h"

#define SERVO_TEST_0           // 测试舵机角度
#define WRITE_TEST_0           // 写入舵机角度测试
#define SYNC_WRITE_TEST_0      // 同步写入测试(5ms Shift+X)
#define FACTORY_RESET_TEST_0   // 重置出厂设置测试
#define SERVO_STOP_TEST_0      // 停止舵机测试
#define SERVO_CALIBRATION_TEST_0 // 校准舵机测试
#define WIGGLE_UNKOWN_ID_0     // 滑动未知ID舵机ID测试

// 口信号初始化
void _uart_init(HMODULE hSerial)
{
    if (hSerial == INVALID_HANDLE_VALUE)
    {
        PRINTF("Failed to open serial port\\%m");
        return FALSE;
    }

    // 配置串口参数
    DCB dbSerialParams = { 0 };
    dbSerialParams.DCBlength = sizeof(dbSerialParams);

    if (!GetCommState(hSerial, &dbSerialParams))
    {
        PRINF("Failed to get serial port parameter\\m");
        CloseHandle(hSerial);
        return FALSE;
    }

    // 配置串口参数
    dbSerialParams.BaudRate = 1000000;
    dbSerialParams.ByteSize = 8;
    dbSerialParams.StopBits = ONESTOPBIT;
    dbSerialParams.Parity = NOPARITY;

    if (!SetCommState(hSerial, &dbSerialParams))
    {
        PRINF("Failed to set serial port parameter\\m");
        CloseHandle(hSerial);
        return FALSE;
    }

    return TRUE;
}

// 读出数据
int readData(HANDLE hSerial)
{
    char buffer[100];
    int bytesRead = 0;
    int totalBytes = 0;
    int count = 0;
    while (bytesRead > 0)
    {
        bytesRead = ReadFile(hSerial, buffer + totalBytes, 1, &bytesRead, NULL);
        if (bytesRead > 0)
        {
            totalBytes += bytesRead;
            count++;
            PRINF("Read %d bytes: %s\\n", bytesRead, buffer + totalBytes);
        }
        else
        {
            PRINF("Read failed: %m\\n");
            break;
        }
    }
    return count;
}
```

- The correct approach is to use "Add New Item," select the appropriate file type, rename the files, and confirm their addition. Next, the corresponding code from the three files in the SDK should be copied into the newly created files. Once the code modifications are complete, click "Start Debugging." If a pop-up appears stating "Some Unicode characters in this file cannot be saved in the current code page. Would you like to save this file with Unicode encoding to maintain your data?", click "Yes." At this point, the execution should proceed normally.

The screenshot shows the Microsoft Visual Studio IDE interface. In the center, there is a code editor window displaying C code for a servo driver. A warning dialog box is overlaid on the code, asking if the user wants to save changes to the file 'servo.c' because it contains Unicode characters. The dialog has buttons for 'Yes(Y)', 'No(N)', 'Save(S)', and 'Cancel'. At the bottom of the screen, there is a command-line window showing build output and errors related to the servo driver.

```

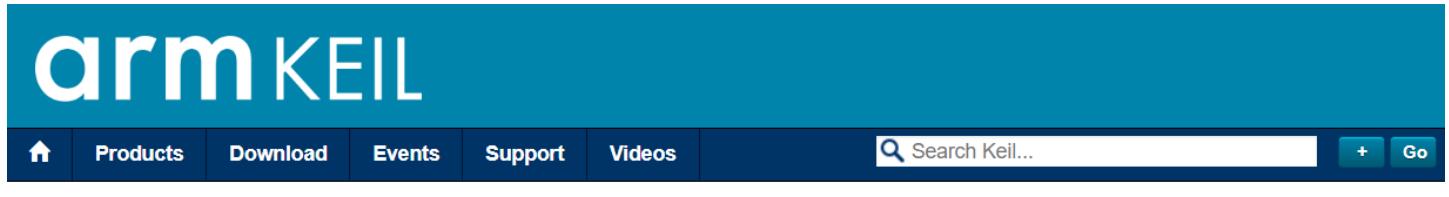
//设置各个电机的相对时间和位置差运动时间
    ret = order_receive(hSerial, pack);
    if (ret == FALSE)
        return FALSE;
    Sleep(100);
    ret = order_send(hSerial, order_write_target_addr, order_write_time_base, target_pos, order_buffer, order_len);
    if (ret == FALSE)
        return FALSE;
    Sleep(1);
    ret = order_receive(hSerial, pack);
    if (ret == FALSE)
        return FALSE;
    Sleep(100);
    //设置各电机的相对时间和位置差运动时间
    servo_gyro_write_time_base(target_pos_and_moving_time);
    ret = order_send(hSerial, order_buf[0], order_len);
    if (ret == FALSE)
        return FALSE;
    Sleep(1);
    ret = order_receive(hSerial, pack);
    if (ret == FALSE)
        return FALSE;
    Sleep(1000);
    if (ret == FALSE)
        return FALSE;
    //关闭串口
    CloseHandle(hSerial);
    return 0;
}

```

4.2 STM32 Standard Library

4.2.1 Environment Setup

Step 1: Install Keil5. Visit <http://www.keil.com/download/product/>, select 'MDK-ARM', and follow the prompts to download and install.



Download Products

Select a product from the list below to download the latest version.



MDK-Arm
 Version 5.40 (May 2024)
 Development environment for Cortex and Arm devices.



C51

Version 9.61 (December 2022)
 Development tools for all 8051 devices.



C251

Version 5.60 (May 2018)
 Development tools for all 80251 devices.



C166

Version 7.57 (May 2018)
 Development tools for C166, XC166, & XC2000 MCUs.

Product Information

Software & Hardware Products

Arm Development Tools

C166 Development Tools

C51 Development Tools

C251 Development Tools

Debug Adapters

Evaluation Boards

Product Brochures

Newsletters

Device Database®

Device List

Compliance Testing

ISO/ANSI Compliance

Validation and Verification

Distributors

Overview

Home / Product Downloads

MDK-ARM

MDK-ARM Version 5.40

Version 5.40

- Review the [hardware requirements](#) before installing this software.
- Note the [limitations of the evaluation tools](#).
- [Further installation instructions for MDK5](#)

(MD5:5e41b1488de2d622dbf2ce046d4583b9)

To install the MDK-ARM Software...

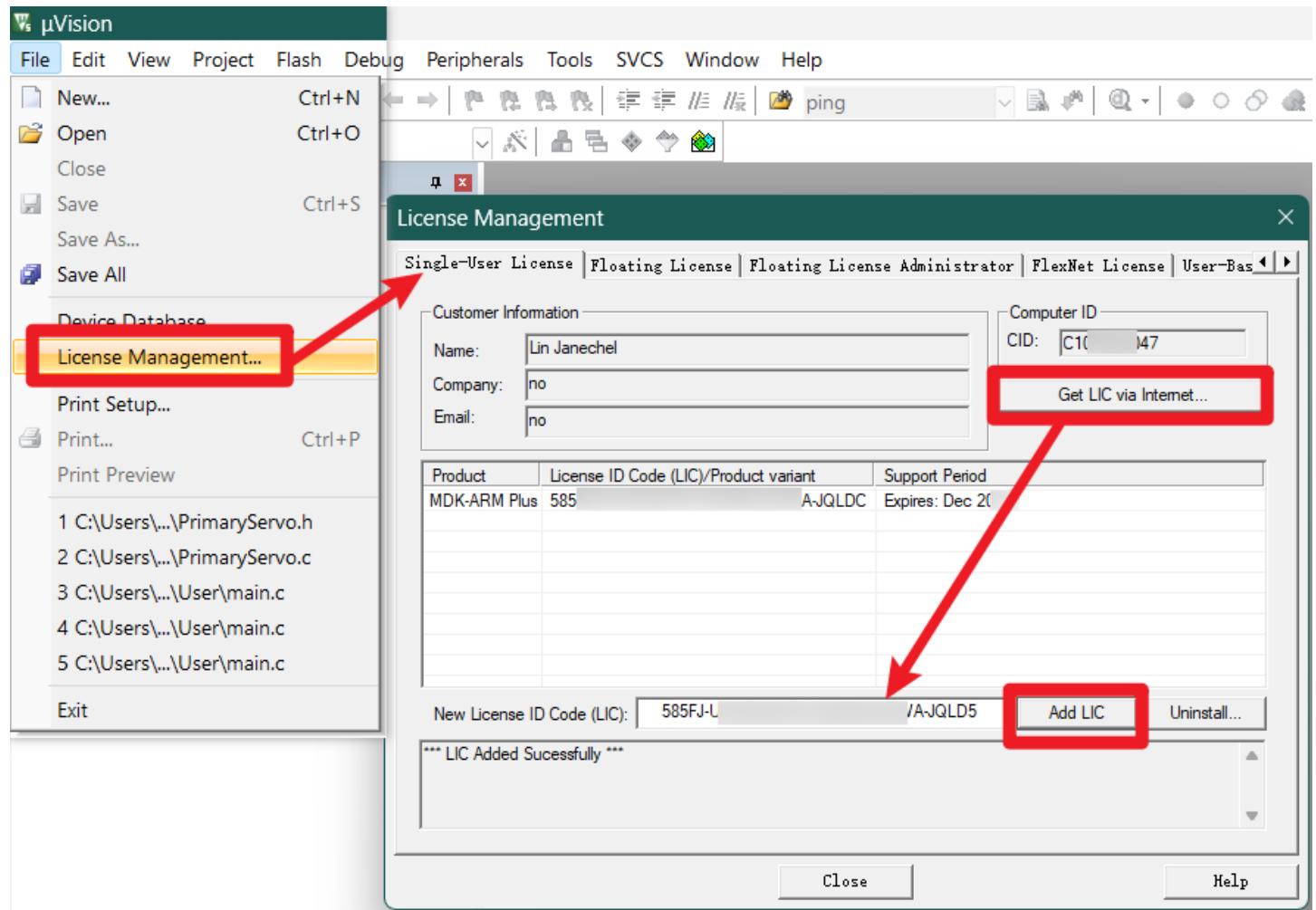
- Right-click on **MDK540.EXE** and save it to your computer.
- PDF files may be opened with Acrobat Reader.
- ZIP files may be opened with PKZIP or WINZIP.

MDK540.EXE (868,978K)

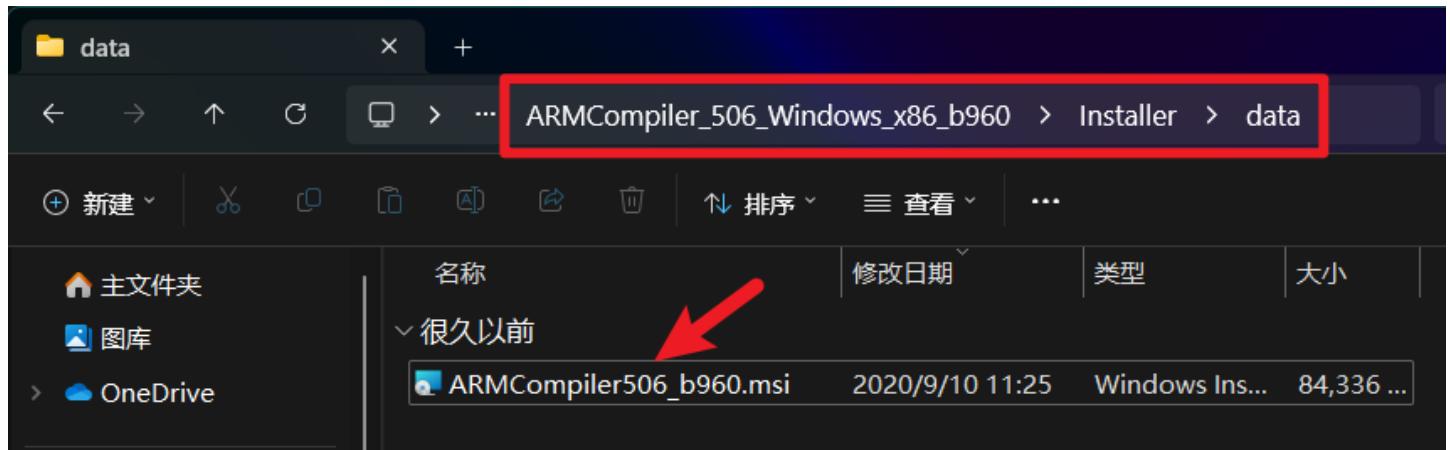
Monday, May 27, 2024

Note: To avoid issues, it is recommended not to select a folder that requires administrator privileges (e.g., Program Files) for the installation path

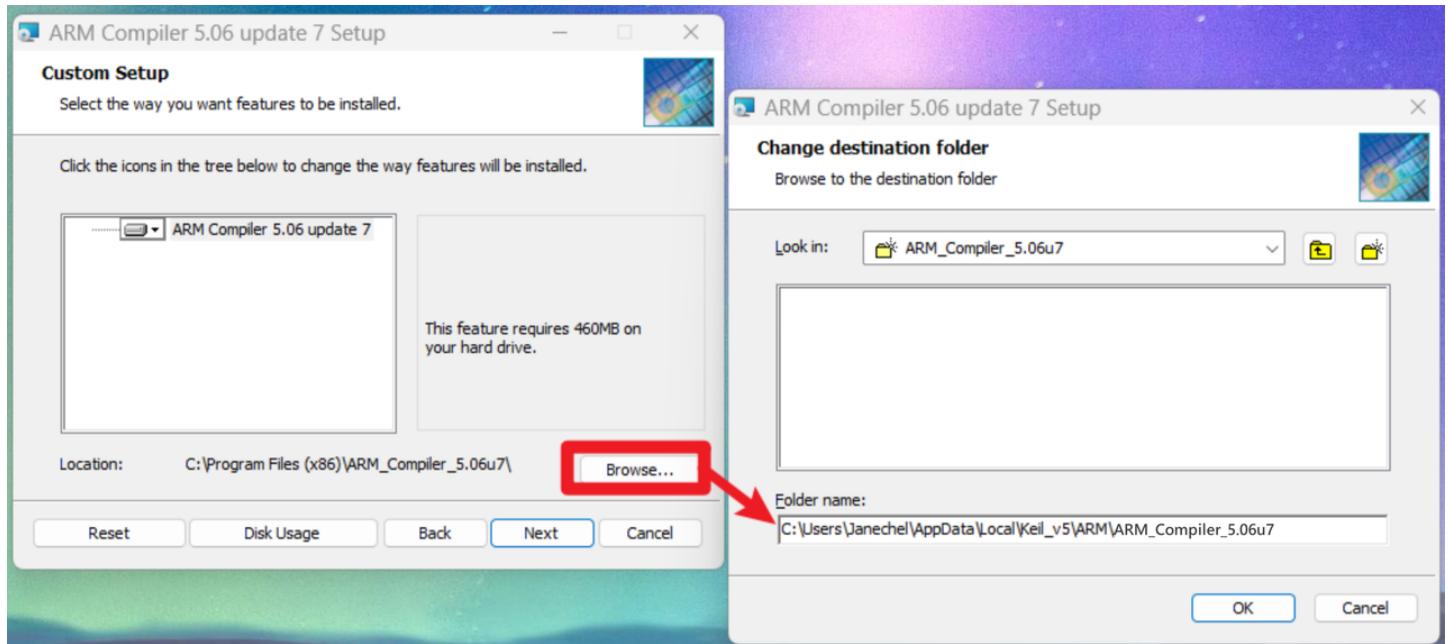
Step 2: Register Keil5 MDK. Obtain the license key through official channels, then click on File -> License Management -> Single-User, enter the license key, and click Add LIC to complete the registration.



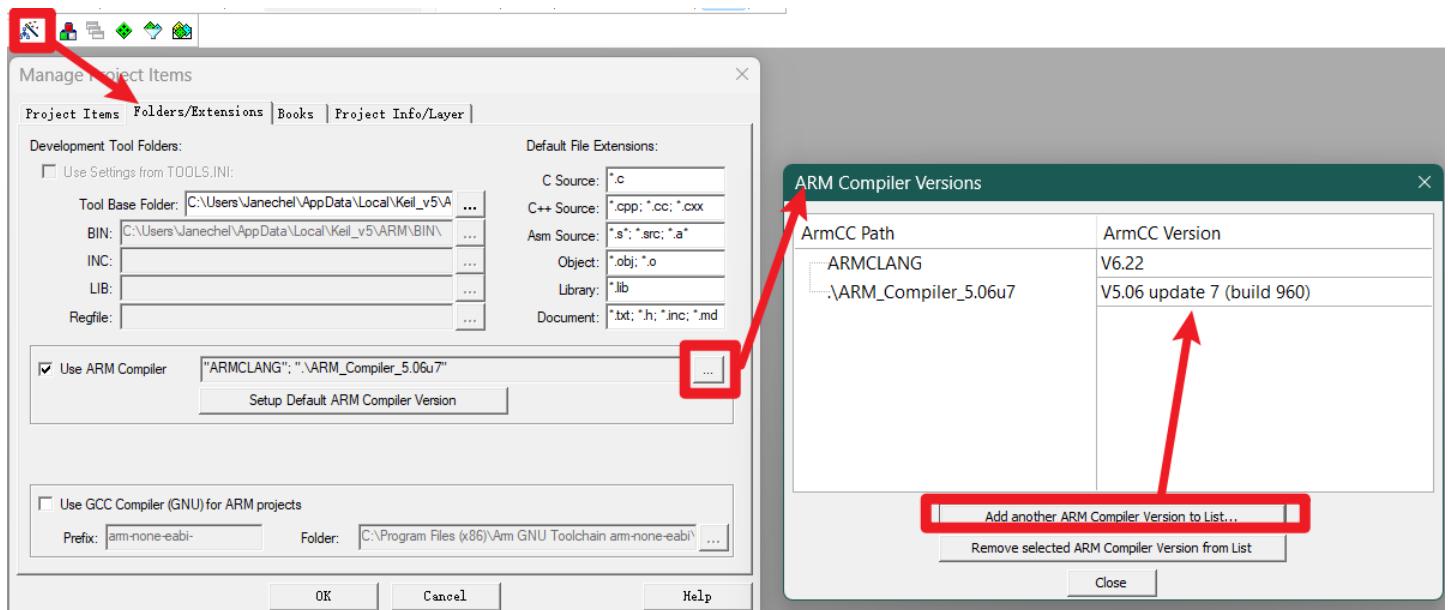
Step 3: Install Compiler Version 5. Visit <https://developer.arm.com/downloads/view/ACOMP5> and follow the instructions to download and install it.



Extract the installation package and run the .msi file.

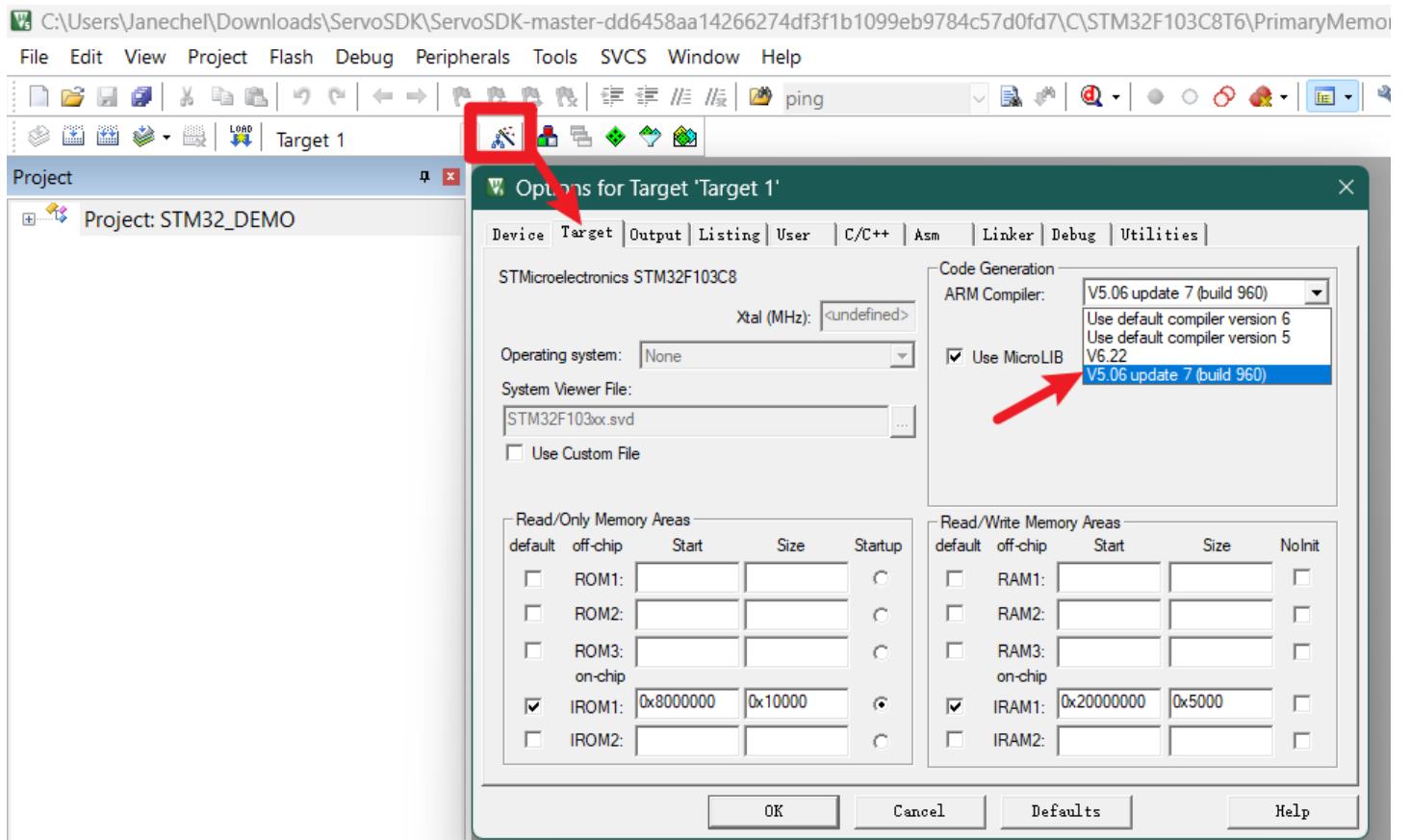


Locate the Keil_v5 installation directory using the desktop shortcut, and replace it with the previously noted path.



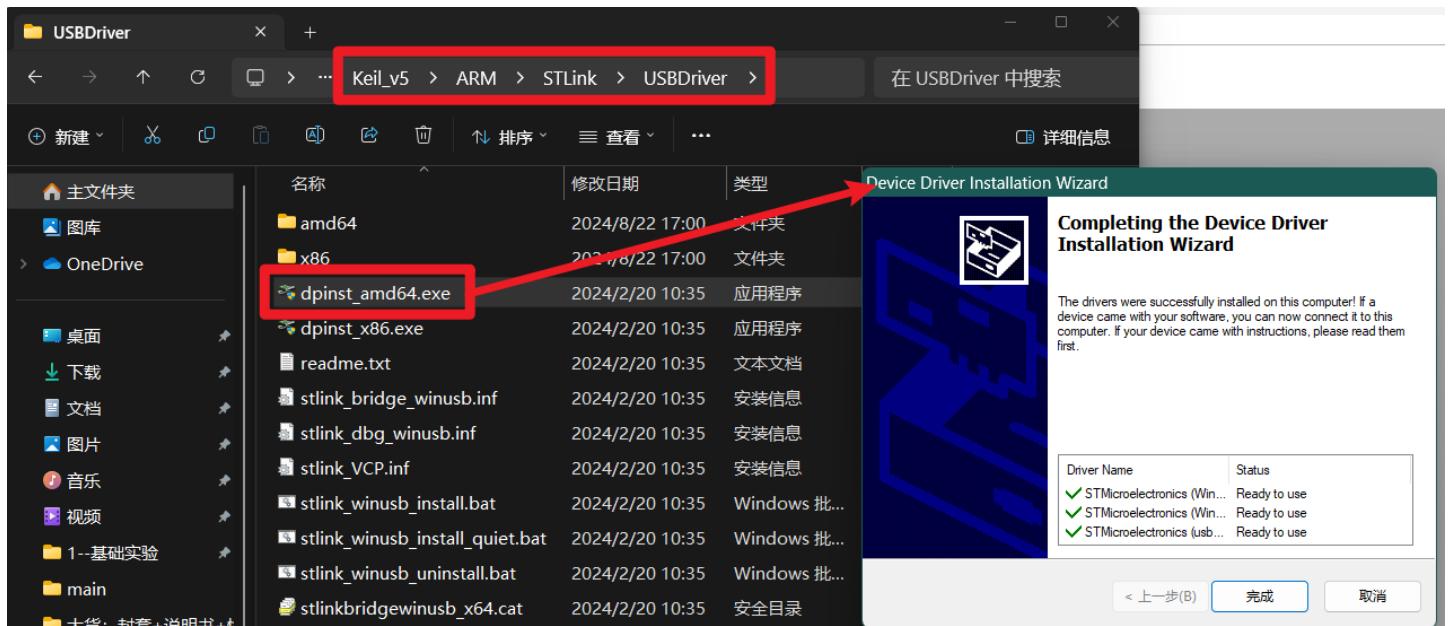
Open Keil5 to configure the compiler.

Step 4: Select the V.506 update 7 compiler. Open Keil5 and navigate to Options for Target -> Target -> ARM Compiler -> V.506 update 7 (build 960), then click OK to confirm the changes.



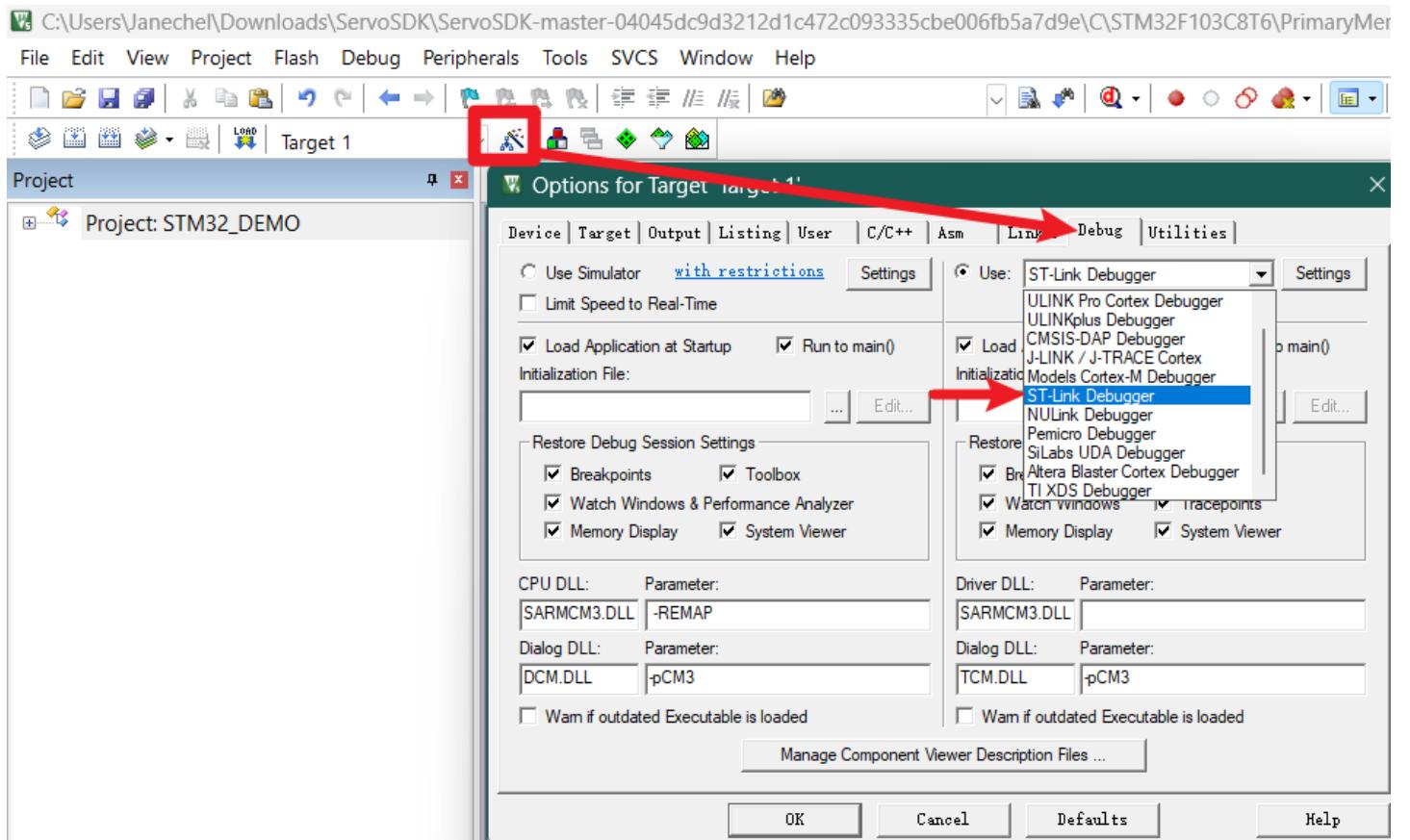
Select V.506 update 7(build 960)

Step 5: Install the ST-LINK driver. Open the local folder Keil5 -> ARM -> STLink -> USBDriver and run dpinst_amd64.exe, then follow the prompts to complete the installation.



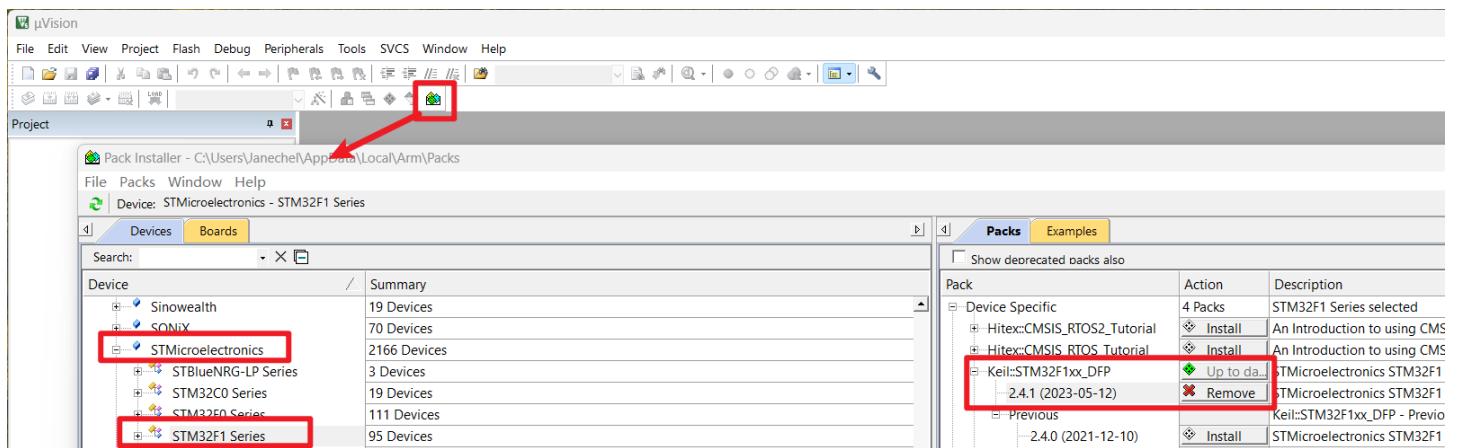
Note: If using a different debugger, the operation and selection are similar.

Step 6: Select the ST-Link Debugger. Open Keil5 and navigate to Options for Target -> Debug -> Use -> ST-Link Debugger, then click OK to confirm the changes.

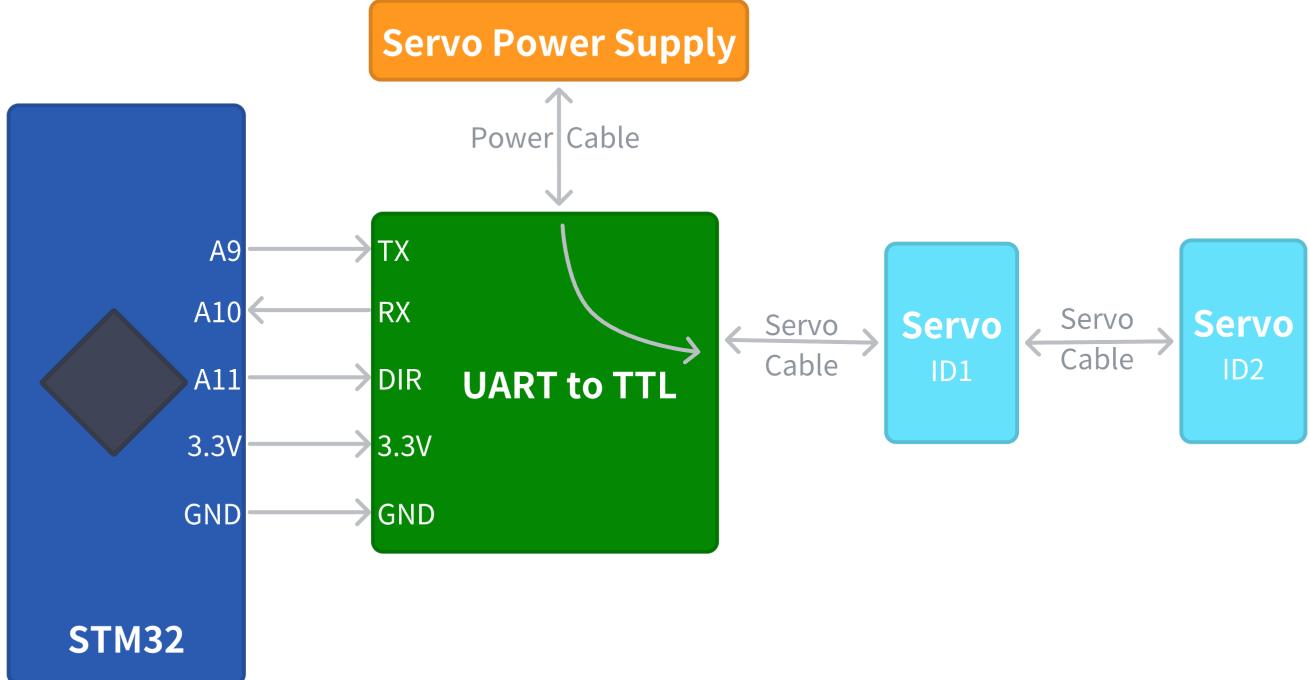


Select ST-Link Debugger

Step 7: Install the STM32F1xx_DFP chip package. Open the Keil5 chip packages, then select STMicroelectronics -> STM32F1 Series -> Keil::STM32F1xx_DFP -> 2.4.1, and click Install.



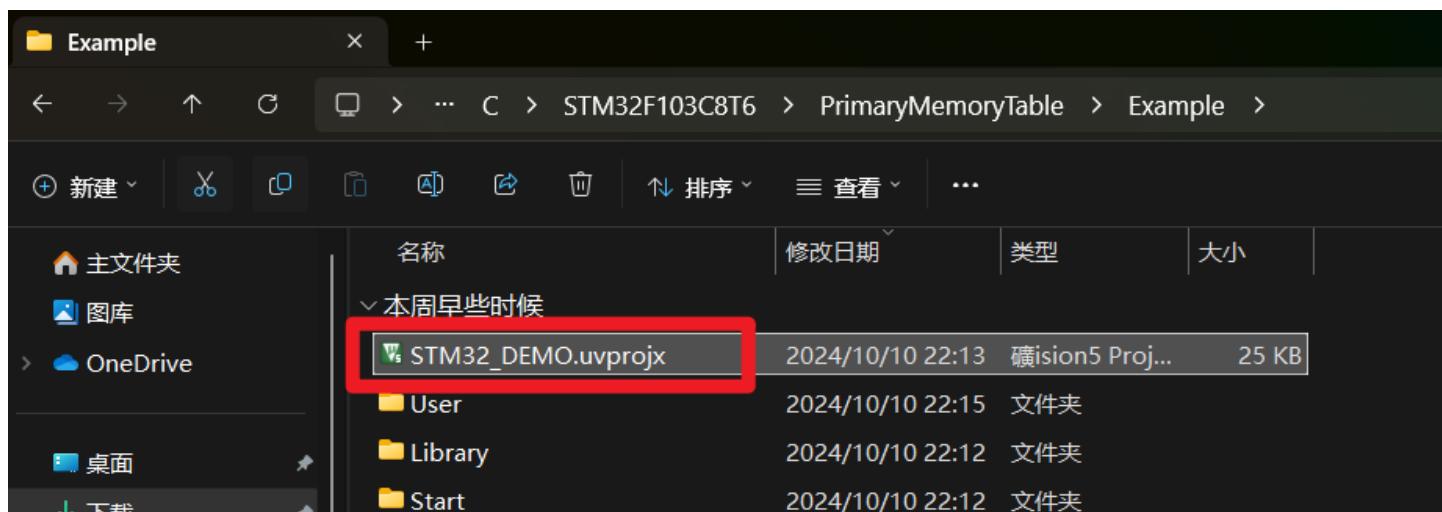
4.2.2 Hardware Connection



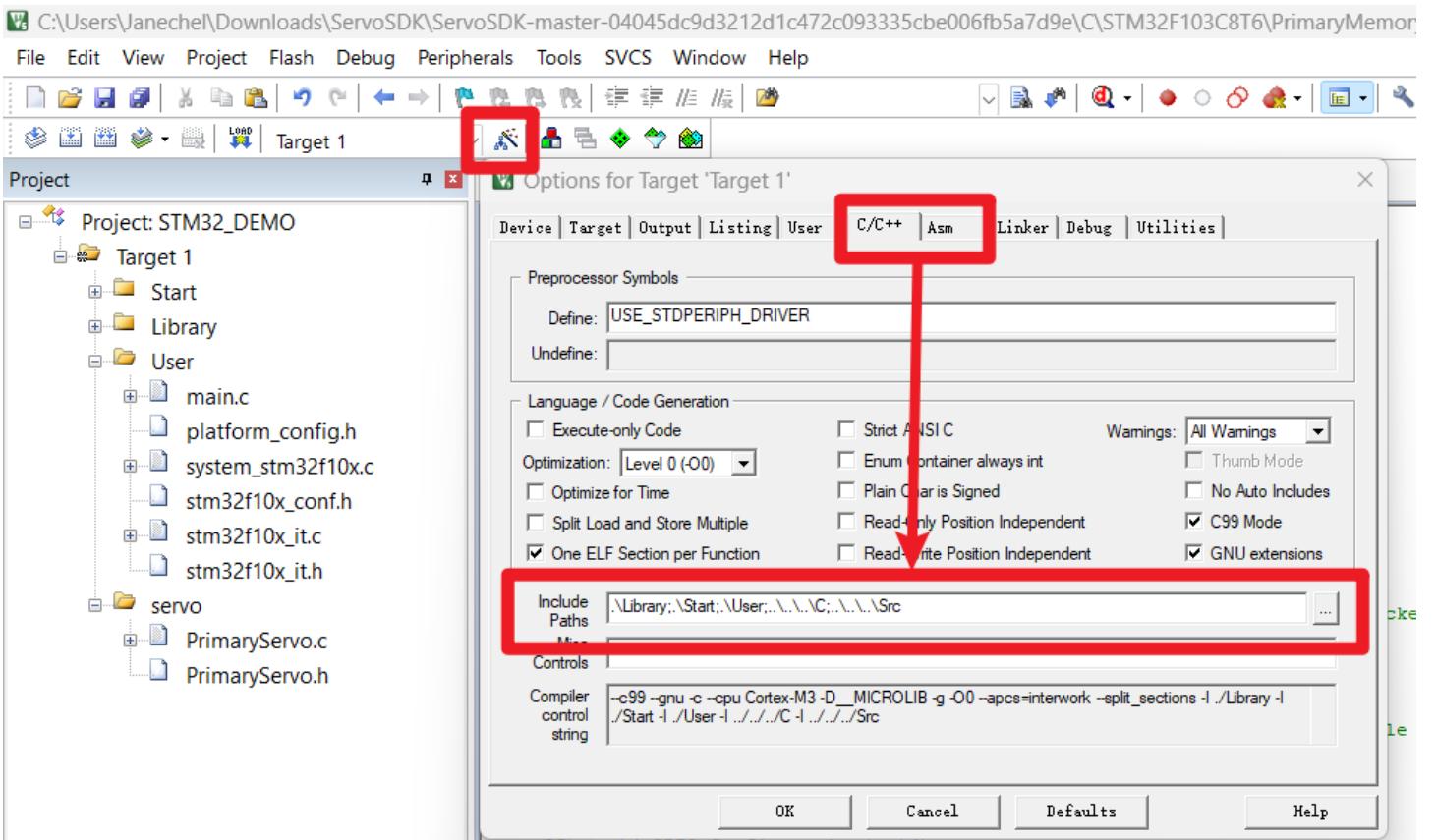
STM32 and Servo Connection Diagram

4.2.3 Run the Example

Step 1: Open the STM32_DEMO.uvprojx file. Take EM-2030 as an example. double-click to open the STM32_DEMO.vcxproj located in
ServoSDK\C\STM32F103C8T6\PrimaryMemoryTable\Example.



Step 2: Replace the C/C++ and ASM including directories with the local Src path.



Step 3: Change the module value you want to test to 1. In the main.c file, change the module value from "0" to "1" in lines 7-16.

```

1 //Taking test the Ping module as an example
2 #define READ_TEST 0                      //Read Servo Data Test
3 #define WRITE_TEST 0                     //Write Servo Data Test
4 #define SYNC_WRITE_TEST 0                //Sync Write Test
5 #define PING_TEST 1                     //PING Instruction Test
6 #define FACTORY_RESET_TEST 0            //Factory Reset Test
7 #define PARAMETER_RESET_TEST 0          //Parameter Reset Test
8 #define REBOOT_TEST 0                  //Reboot Test
9 #define CALIBRATION_TEST 0             //Calibration Test
10 #define MODIFY_ID_TEST 0              //Change Known Servo ID Test
11 #define MODIFY_UNKNOWN_ID_TEST 0       //Change Unknown Servo ID Test

```

Step 4: Change the maximum number of servos for sync write and enable printing output as needed. In the PrimaryServo.h file, lines 4-5, MAX_SERVERS represents the upper limit of the maximum supported sync write servos. If the actual number of servos exceeds 20, adjust this value accordingly; it should be set to at least the actual number of servos. If printing output is not needed, change PRINTF_ENABLE to "0".

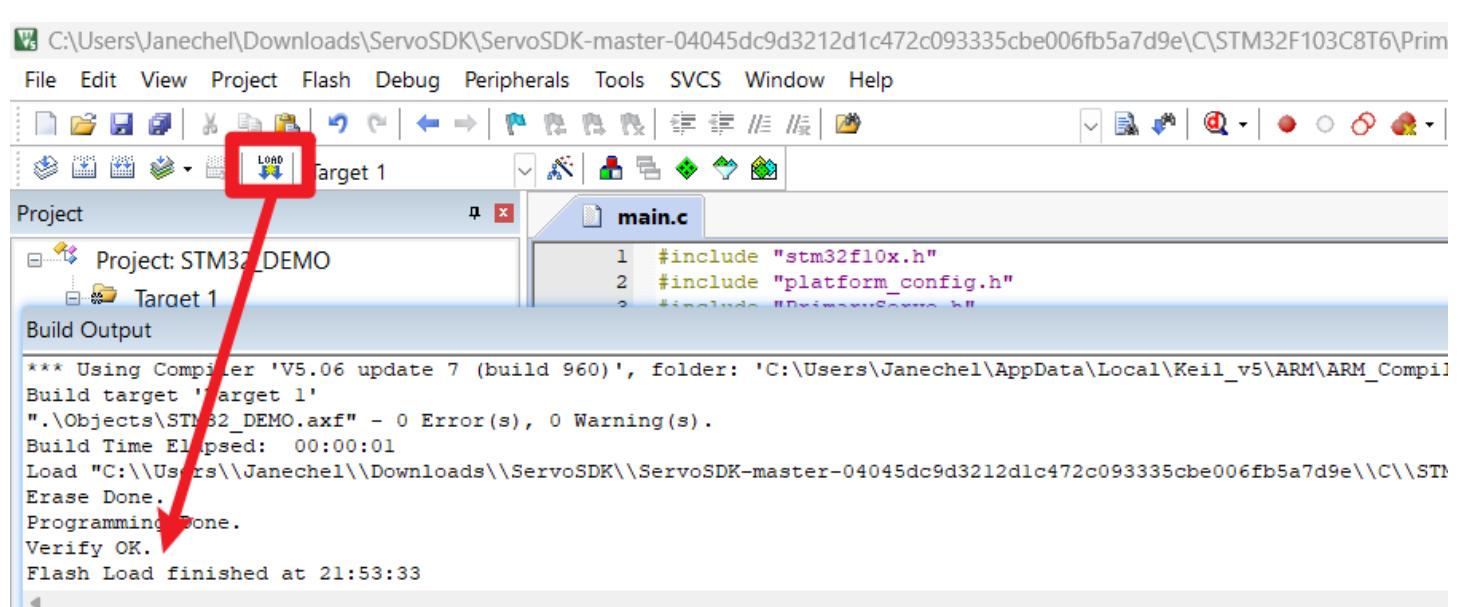
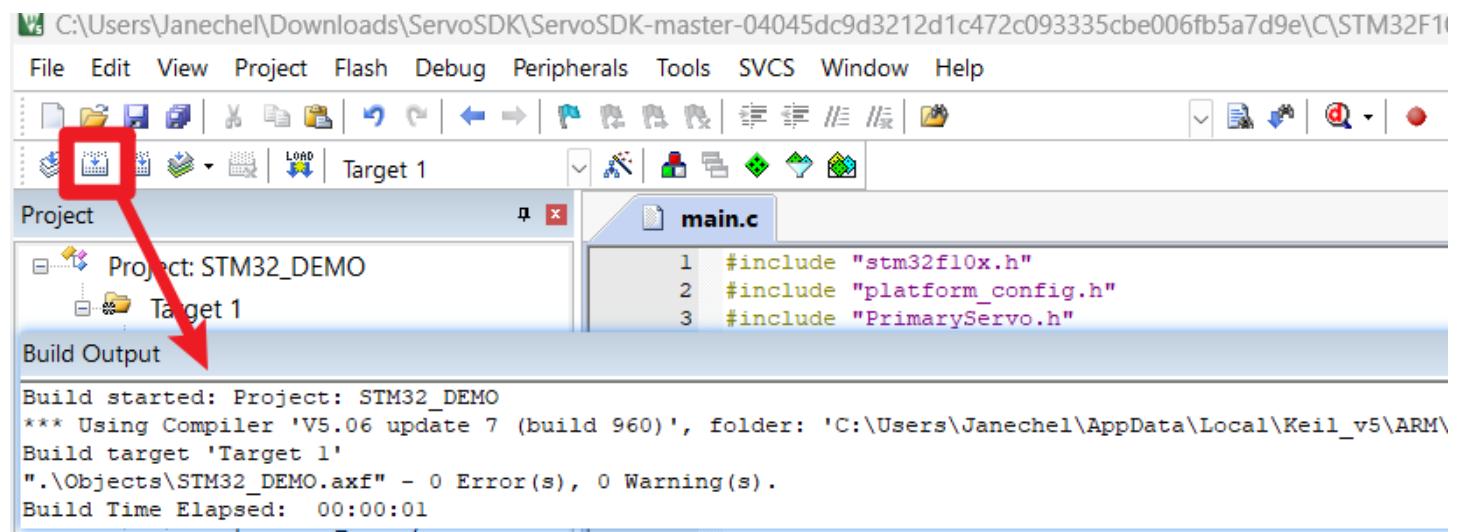
```

1 //Taking one servo and print enable as an example (no changes needed).
2 #define MAX_SERVERS 20                  //Maximum number of servos for sync write.

```

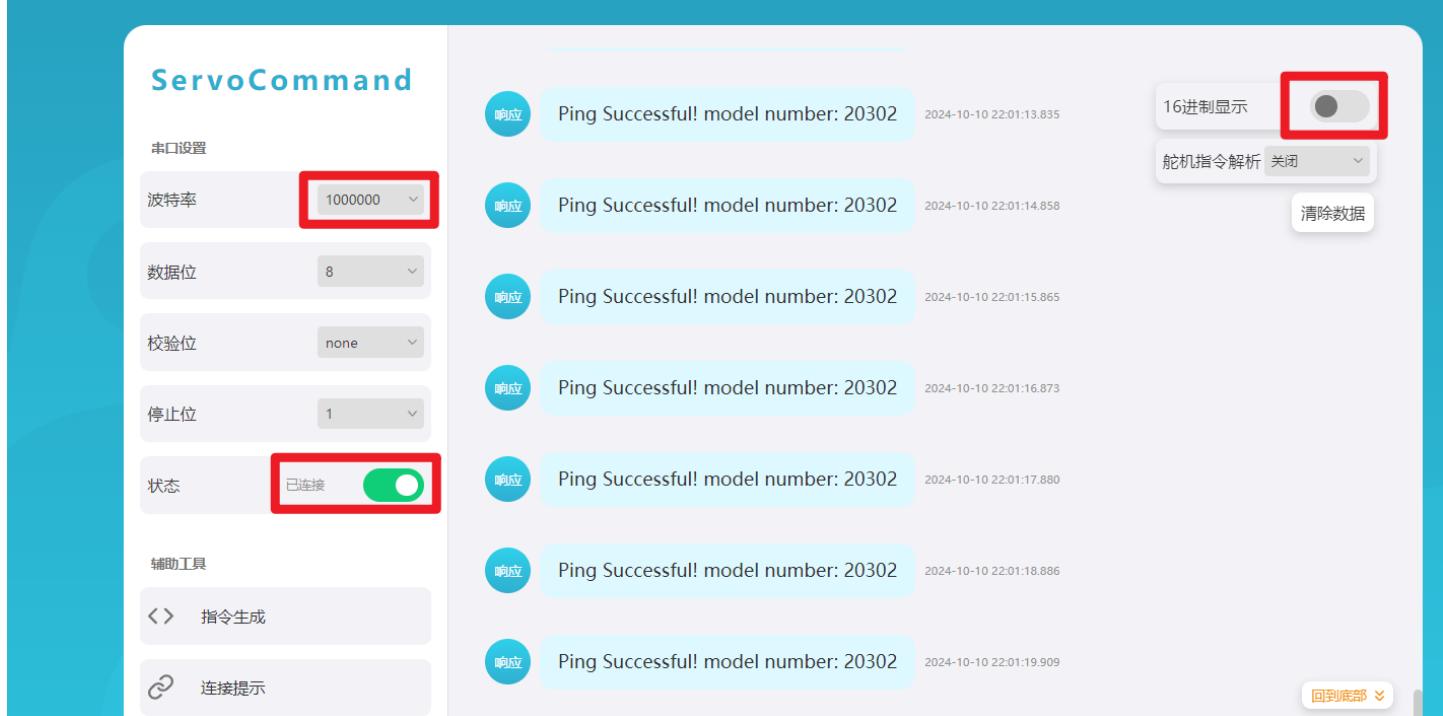
```
3 #define PRINTF_ENABLE 1 //Print output enable.
```

Step 5: Compile and download. Click Build to compile the project; once there are no errors, click Download to transfer to the microcontroller.



Download finished

Step 6: Run the program. Short press the Reset button on the microcontroller to restart the program. You can check the printed information using ServoCommand.

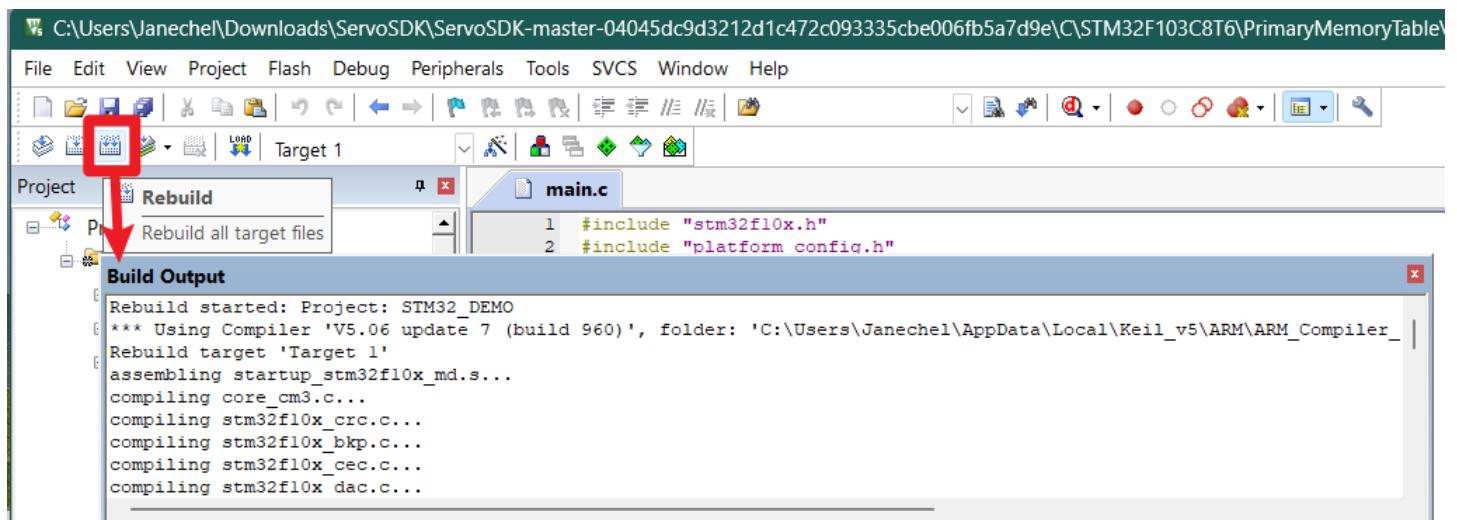


Taking test the Ping module as an example

4.2.4 Run the Demo

For details, refer to *section 4.1.4 Run the Demo*.

Note: After replacing the main file, it is recommended to click 'Rebuild' to recompile the project before downloading and running the program.



4.3 STM32 HAL Library

4.3.1 Environment Setup

Step 1: Install and configure Keil5 MDK. For details, refer to *section 4.2.1 Environment Setup*.

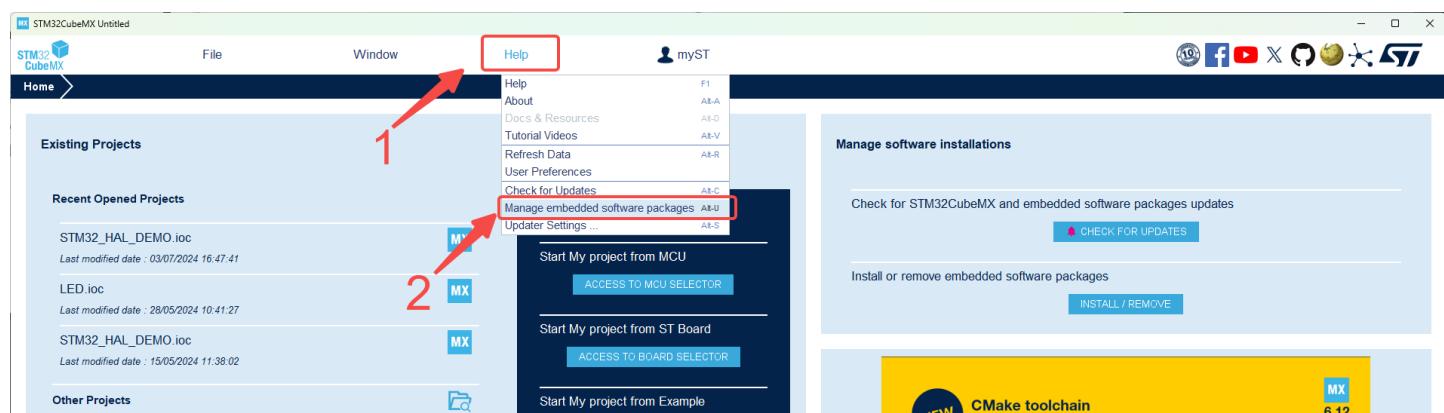
Step 2: Install STM32CubeMX. Visit <https://www.st.com/zh/development-tools/stm32cubemx.html#get-software>, select 'STM32CubeMX-Win', and follow the prompts to

download and install.

获取软件

产品型号	一般描述	最新版本	下载	所有版本
+ STM32CubeMX-Lin	STM32Cube init code generator for Linux	6.12.0	获取最新版本	选择版本
+ STM32CubeMX-Mac	STM32Cube init code generator for macOS	6.12.0	获取最新版本	选择版本
+ STM32CubeMX-Win	STM32Cube init code generator for Windows	6.12.0	获取最新版本	选择版本

Step 3: Install the STM32F1 chip package. Open STM32CubeMX, then navigate to Help -> Manage Embedded Software Packages -> STM32F1 -> Latest Version, and click Install.



MX Embedded Software Packages Manager

STM32Cube MCU Packages and embedded software packs releases

Releases Information was last refreshed less than one hour ago.

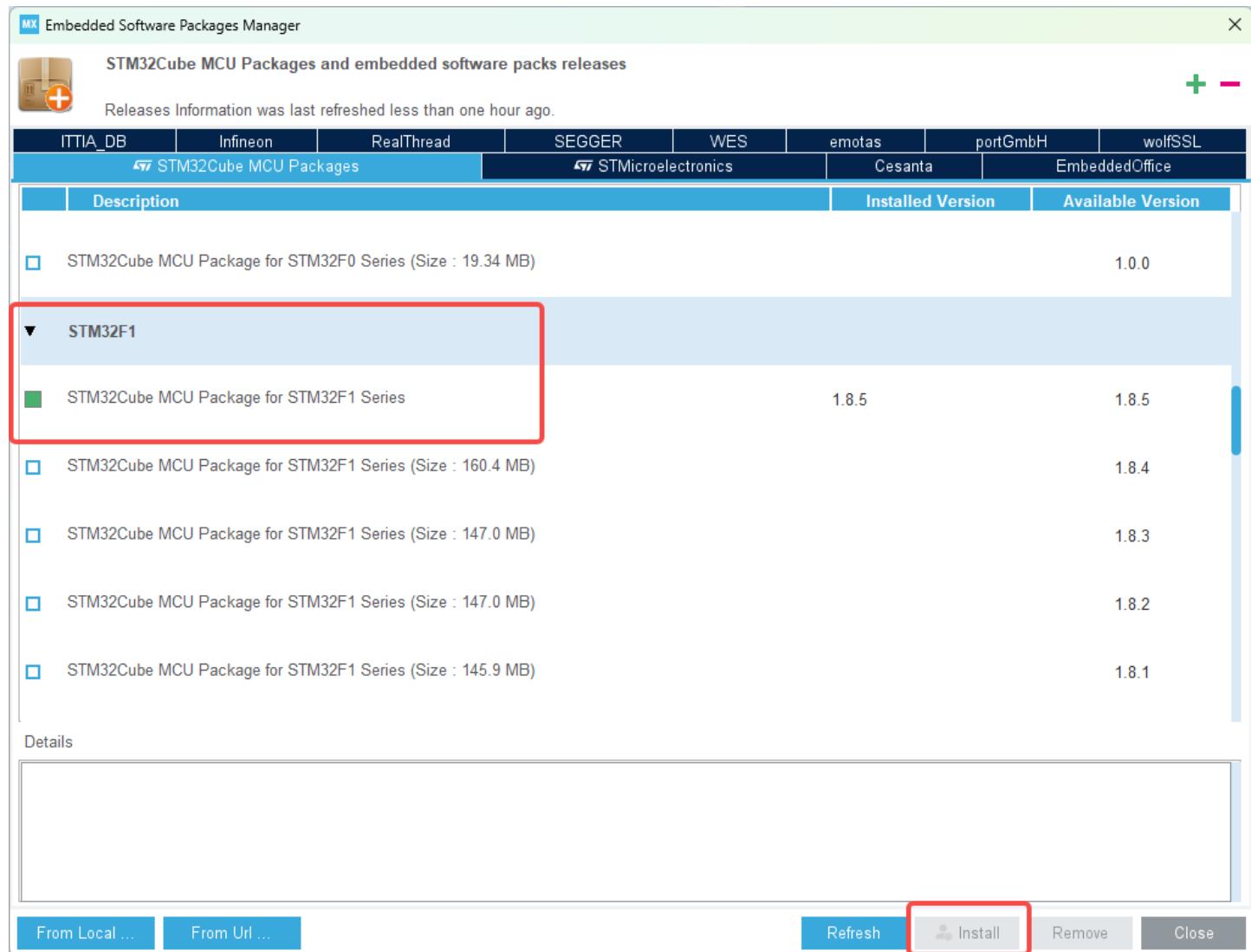
+ -

ITIA_DB	Infineon	RealThread	SEGGER	WES	emotas	portGmbH	wolfSSL
STM32Cube MCU Packages	STMicroelectronics	Cesanta	EmbeddedOffice				

Description	Installed Version	Available Version
<input type="checkbox"/> STM32Cube MCU Package for STM32F0 Series (Size : 19.34 MB)		1.0.0
STM32F1		
<input checked="" type="checkbox"/> STM32Cube MCU Package for STM32F1 Series	1.8.5	1.8.5
<input type="checkbox"/> STM32Cube MCU Package for STM32F1 Series (Size : 160.4 MB)		1.8.4
<input type="checkbox"/> STM32Cube MCU Package for STM32F1 Series (Size : 147.0 MB)		1.8.3
<input type="checkbox"/> STM32Cube MCU Package for STM32F1 Series (Size : 147.0 MB)		1.8.2
<input type="checkbox"/> STM32Cube MCU Package for STM32F1 Series (Size : 145.9 MB)		1.8.1

Details

From Local ... From Url ... Refresh **Install** Remove Close



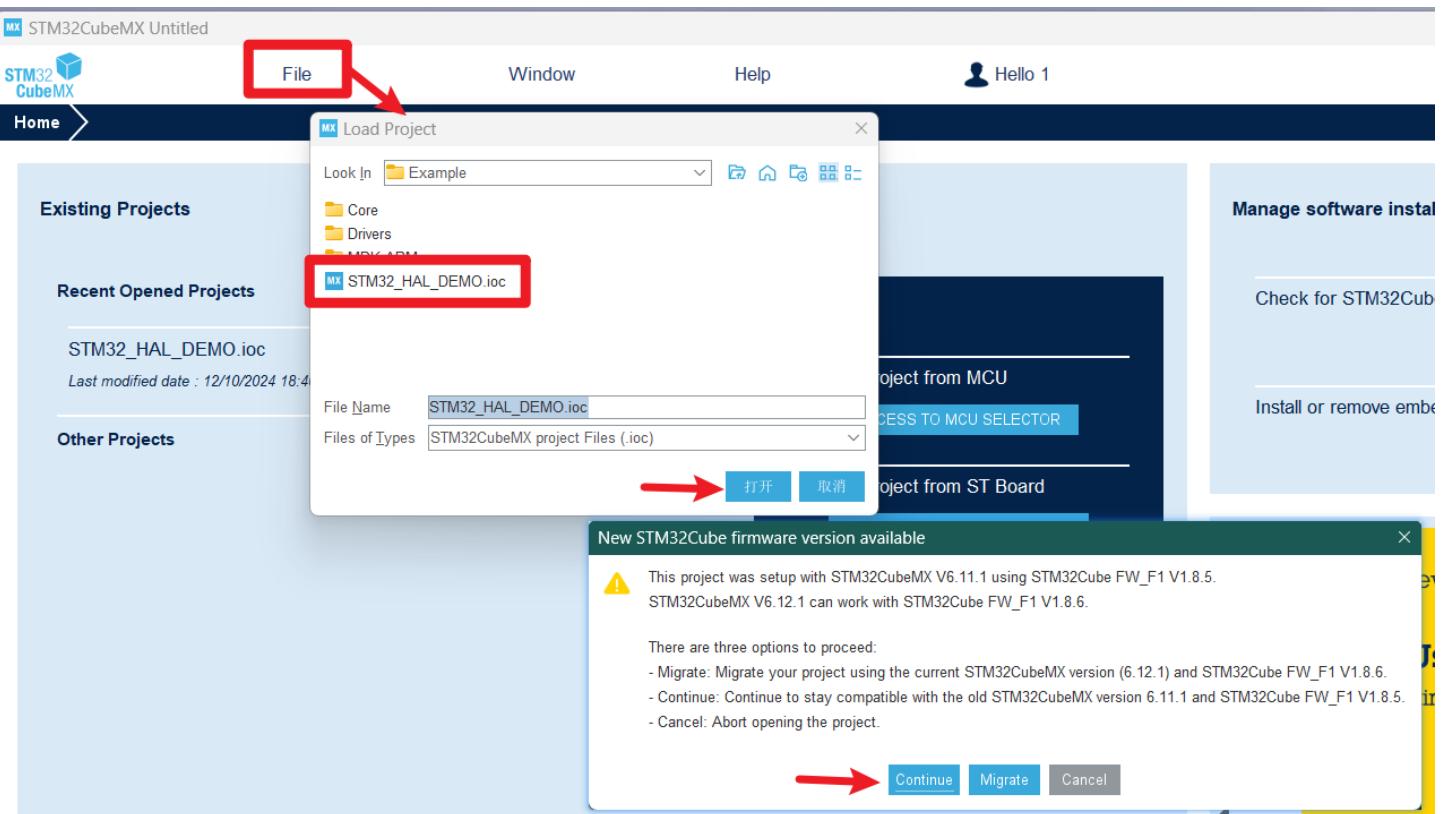
4.3.2 Hardware Connection

For details, refer to *section 4.2.2 Hardware Connection* for details.

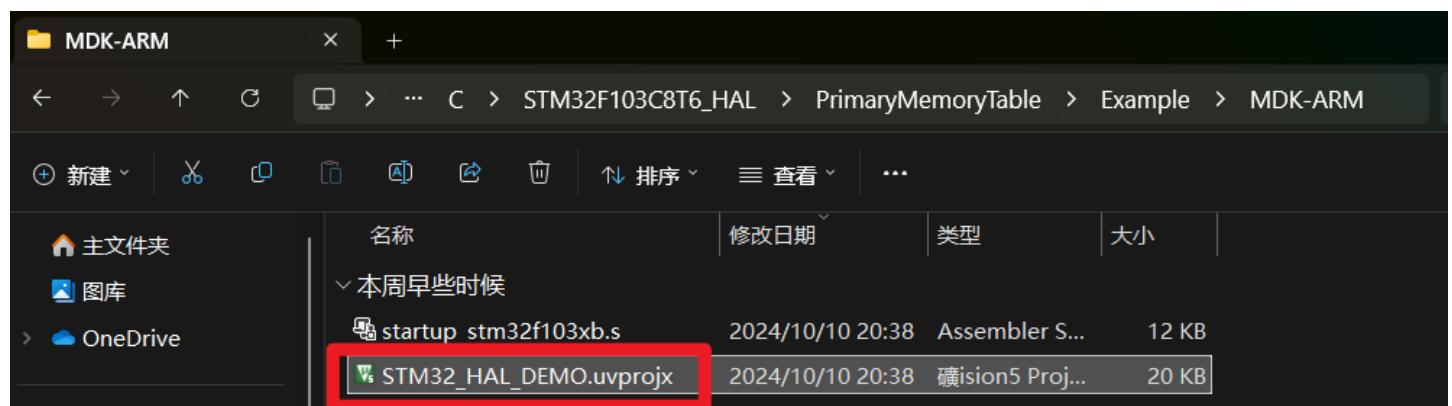
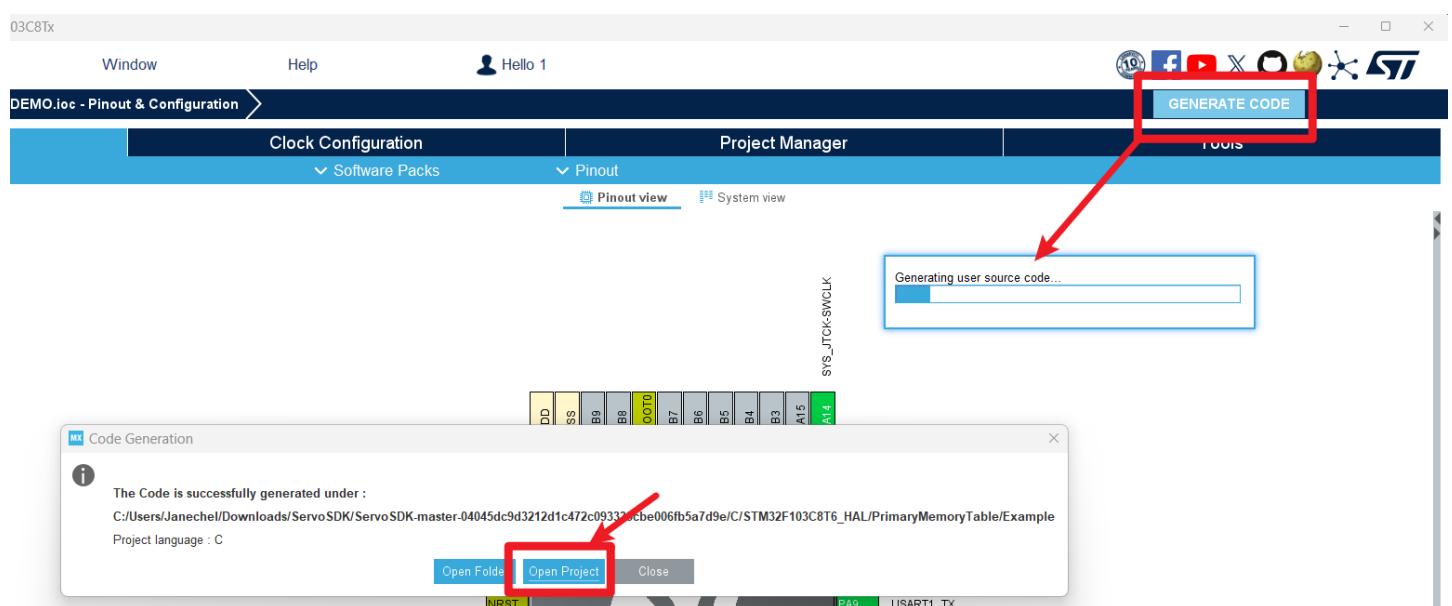
4.3.3 Run the Example

Note: When using this project for the first time, it is recommended to open the project file with STM32CubeMX to avoid encountering the "Unknown compiler" error after compilation.

Step 1: Open the STM32_HAL_DEMO.ioc file. Launch STM32CubeMX and select File -> Load Project -> STM32_HAL_DEMO.ioc -> Open -> Continue -> GENERATE CODE -> Open Project.

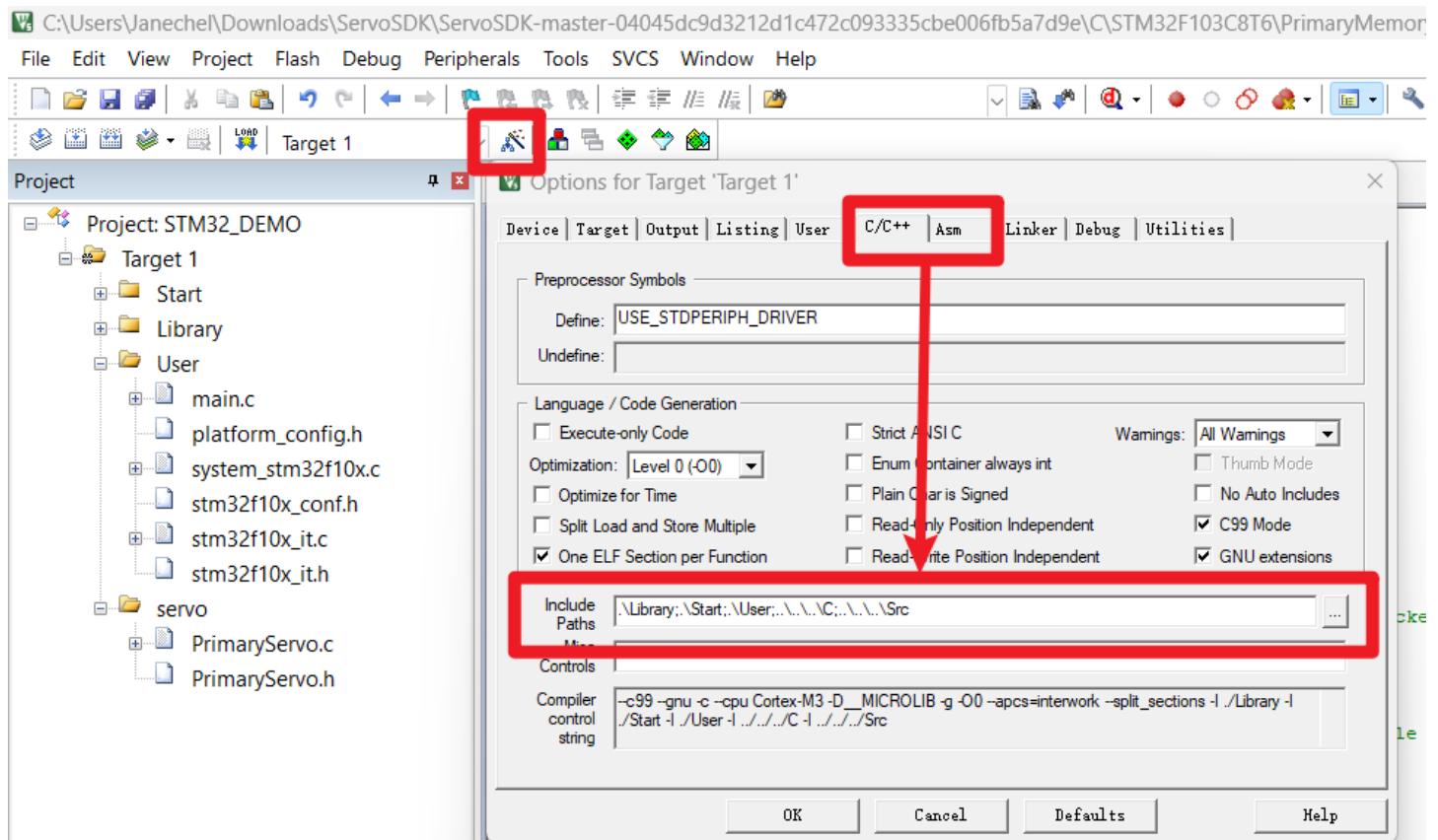


Taking EM-2030 as an example, STM32_HAL_DEMO.ioc in
C:\STM32F103C8T6_HAL\PrimaryMemoryTable\Example\MDK-ARM



STM32_HAL_DEMO.uvprojx file can be directly opened, if this is not the first time it has been accessed.

Step 2: Replace the C/C++ and ASM including directories with the local Src path.



Step 3: Change the module value you want to test to 1. In the main.c file, modify the module value from "0" to "1" in lines 61-70.

```
1 //Taking test the Ping module as an example
2 #define READ_TEST 0                      //Read Servo Data Test
3 #define WRITE_TEST 0                     //Write Servo Data Test
4 #define SYNC_WRITE_TEST 0                //Sync Write Test
5 #define PING_TEST 1                     //PING Instruction Test
6 #define FACTORY_RESET_TEST 0            //Factory Reset Test
7 #define PARAMETER_RESET_TEST 0          //Parameter Reset Test
8 #define REBOOT_TEST 0                  //Reboot Test
9 #define CALIBRATION_TEST 0             //Calibration Test
10 #define MODIFY_ID 0                   //Change Known Servo ID Test
11 #define MODIFY_UNKNOWN_ID 0           //Change Unknown Servo ID Test
```

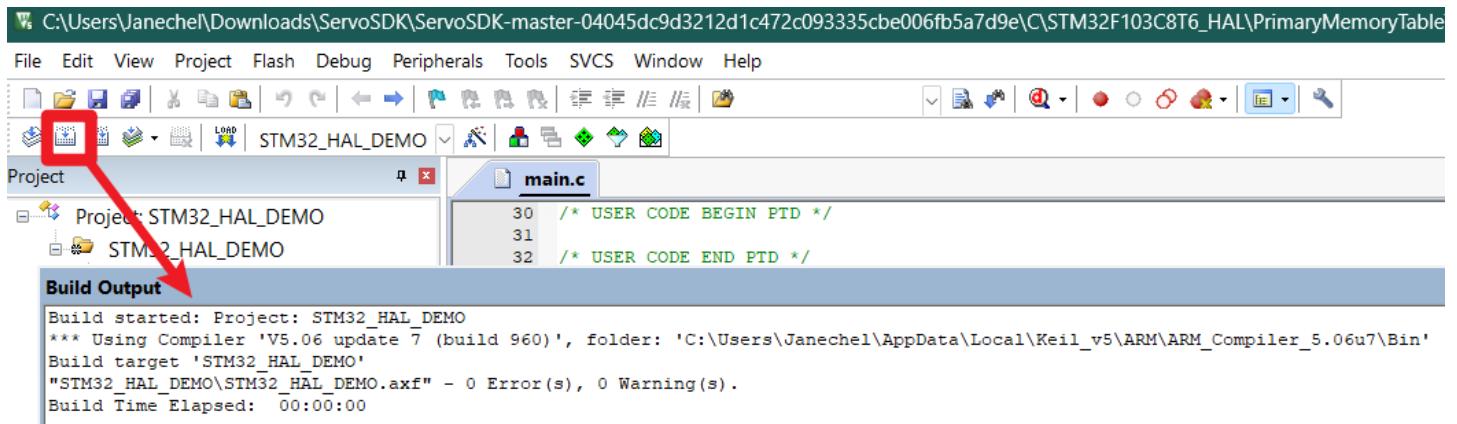
Step 4: Change the maximum number of servos for sync write and enable printing output as needed. In the PrimaryServo.h file, lines 4-5, MAX_SERVERS represents the upper limit of the maximum supported sync write servos. If the actual number of servos exceeds 20, adjust this value accordingly; it should be set to at least the actual number of servos. If printing output is not needed, change PRINTF_ENABLE to "0".

```

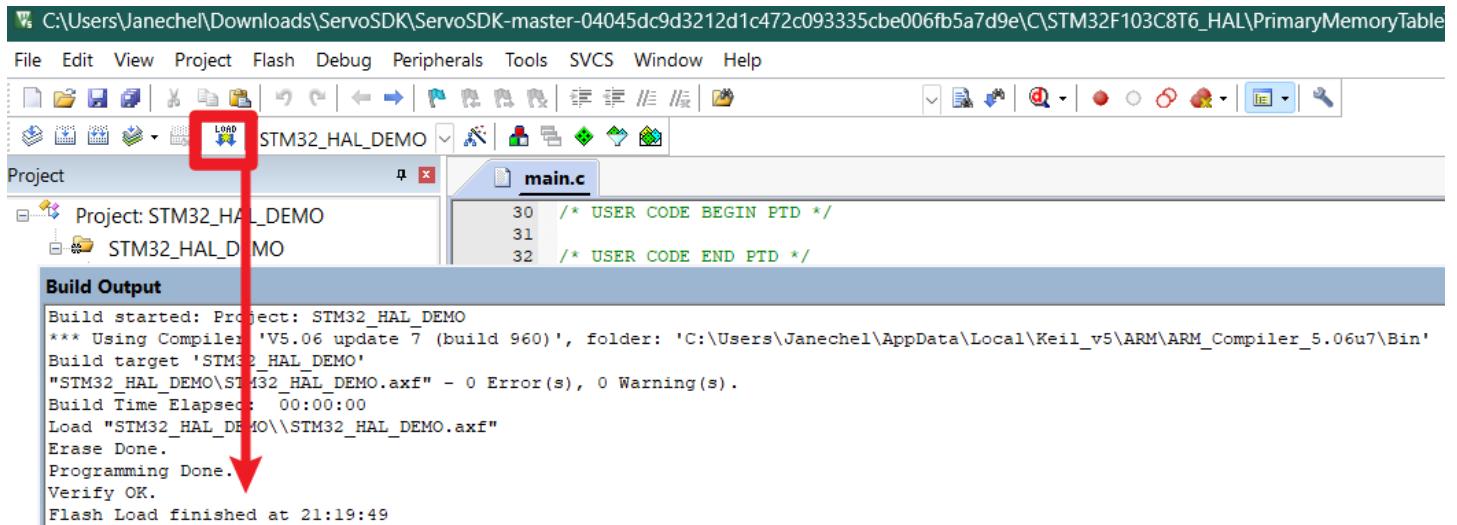
1 //Taking one servo and print enable as an example (no changes needed).
2 #define MAX_SERVERS 20           //Maximum number of servos for sync write.
3 #define PRINTF_ENABLE 1          //Print output enable.

```

Step 5: Compile and download. Click Build to compile the project; once there are no errors, click Download to transfer to the microcontroller.

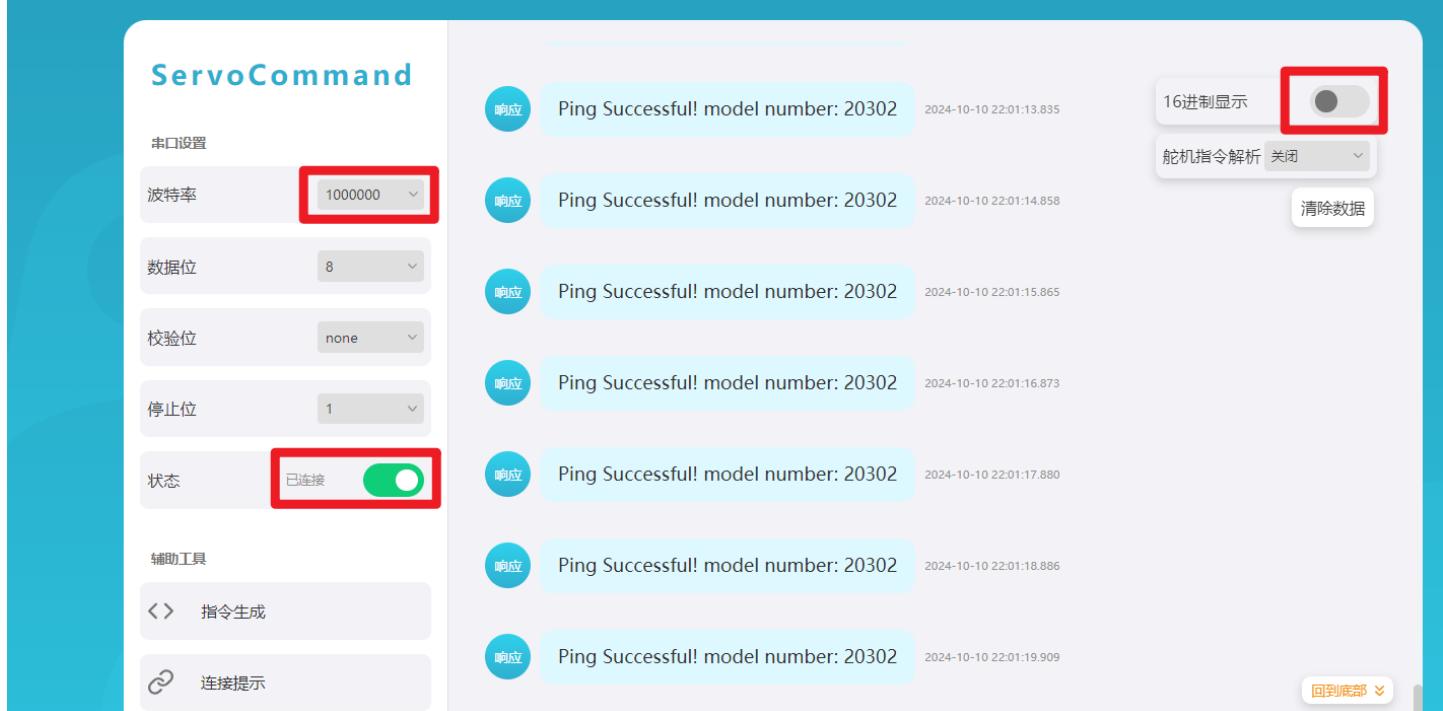


Compilation completed without errors



Download finished

Step 6: Run the program. Short press the Reset button on the microcontroller to restart the program. You can check the printed information using ServoCommand.

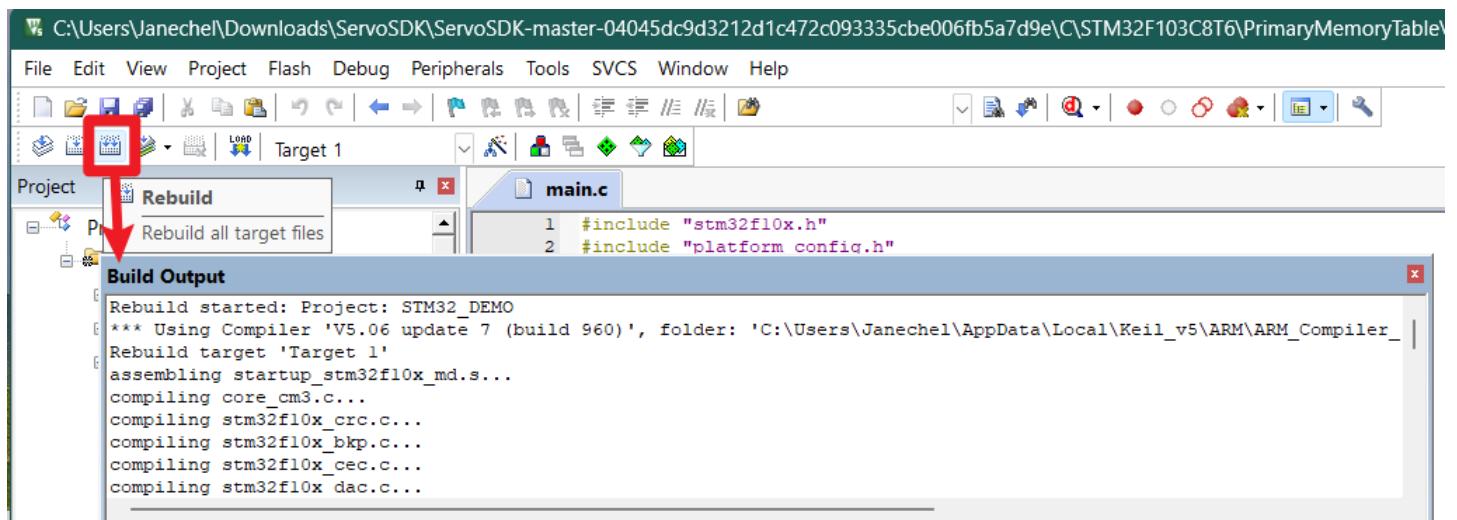


Taking test the Ping module as an example

4.3.4 Run the Demo

For details, refer to *section 4.1.4 Run the Demo*.

Note: After replacing the main file, it is recommended to click 'Rebuild' to recompile the project before downloading and running the program.



4.4 STC89C52

4.4.1 Environment Setup

Step 1: Install Keil5 C51. Visit <http://www.keil.com/download/product/>, select 'C51', and follow the prompts to download and install.

Download Products

Select a product from the list below to download the latest version.



MDK-Arm

Version 5.40 (May 2024)

Development environment for Cortex and Arm devices.



C51

Version 9.61 (December 2022)

Development tools for all 8051 devices.

Product Information

Software & Hardware Products

Arm Development Tools

C166 Development Tools

C51 Development Tools

C251 Development Tools

Debug Adapters

Evaluation Boards

Product Brochures

Newsletters

Device Database®

Device List

Compliance Testing

ISO/ANSI Compliance

Validation and Verification

Distributors

Home / Product Downloads

C51

Development tools for Classic and Extended 8051 Microcontrollers

Version 9.61

The Keil C51 Evaluation Kit allows you to create programs for all 8051 derivatives.

- Review the [hardware requirements](#) before installing this software.
- Note the [limitations of the evaluation tools](#).

MD5:8f0de26cd2613974defd3dd9aedc36ee

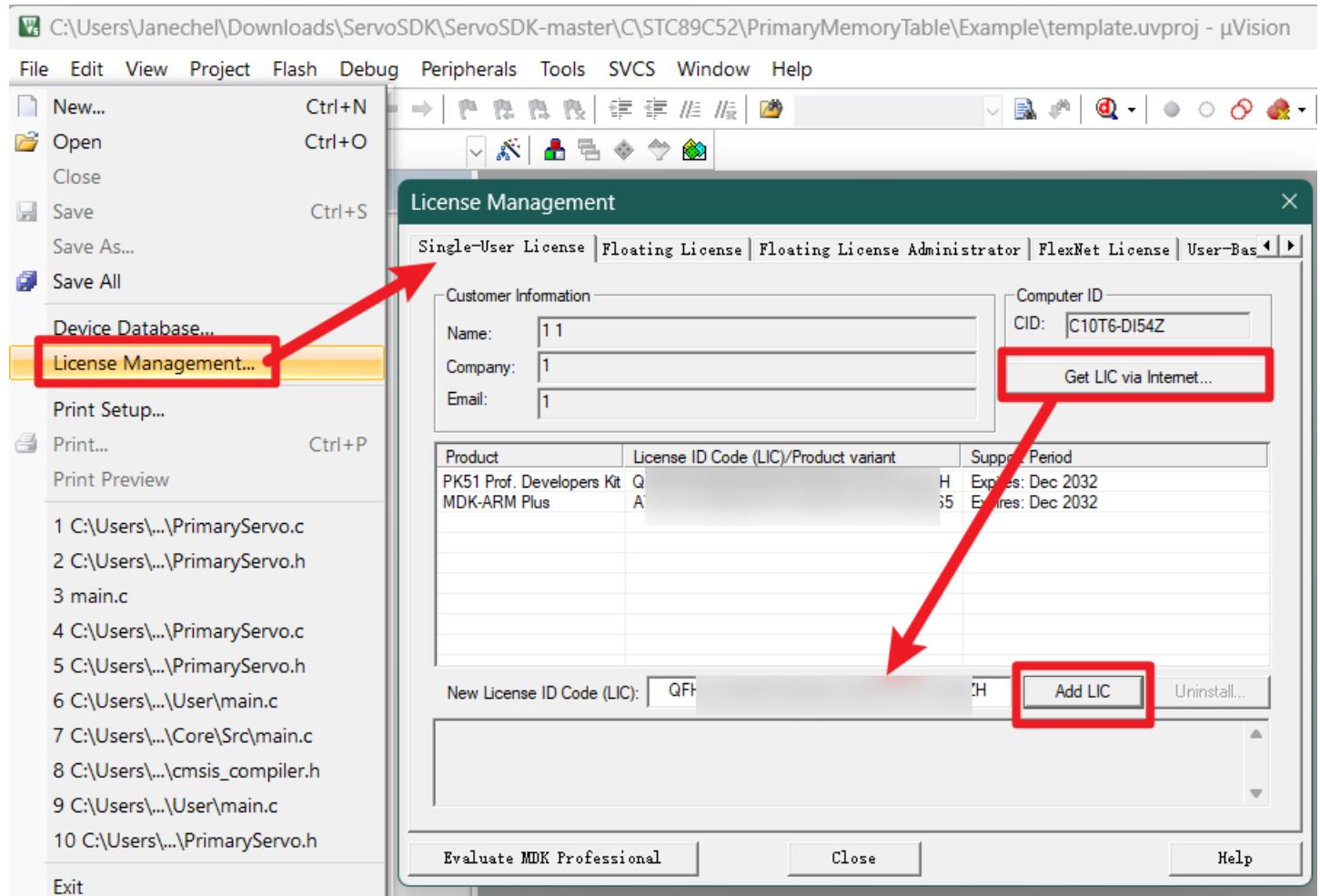
To install the C51 Software...

- Right-click on **C51V961.EXE** and save it to your computer.
- PDF files may be opened with Acrobat Reader.
- ZIP files may be opened with PKZIP or WINZIP.

C51V961.EXE (107,294K)

Monday, December 19, 2022

Step 2: Register Keil5 C51. Obtain the license key through official channels, then click File -> License Management -> Single-User, enter the license key, and click Add LIC to complete the registration.



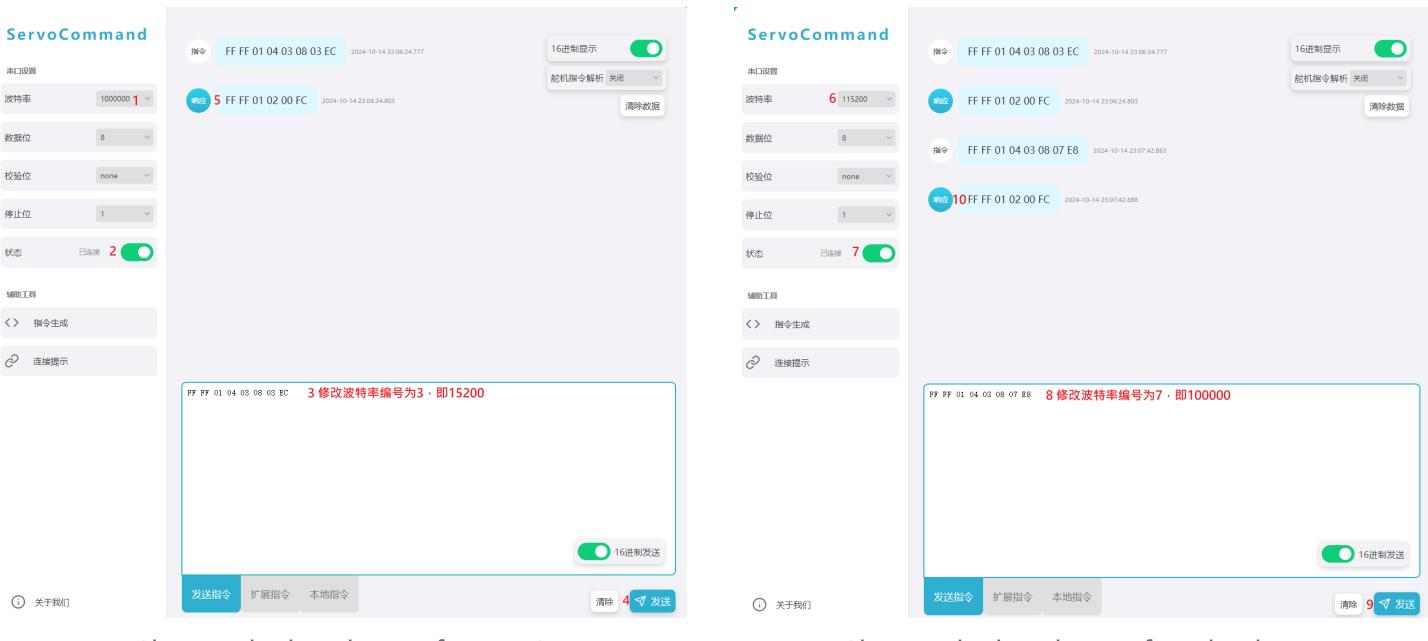
Step 3: Install the download tool. For example, to install PZ-ISP.exe, visit <http://www.prechin.cn/gongsixinwen/208.html>, click on the link for the 51 Microcontroller Series - PZ - 2&3&4 (A2&A3&A4) Development Board, and then select 5 - Development Tools -> 3 - Programming Software -> PZ-ISP -> PZ-ISP.exe.

[返回上一级 | ... > 普中-2&普中-... > 5--开发工具 > 3-程序下载软件 > PZ-ISP \(推...](#)

文件名

 PZ-ISP.exe

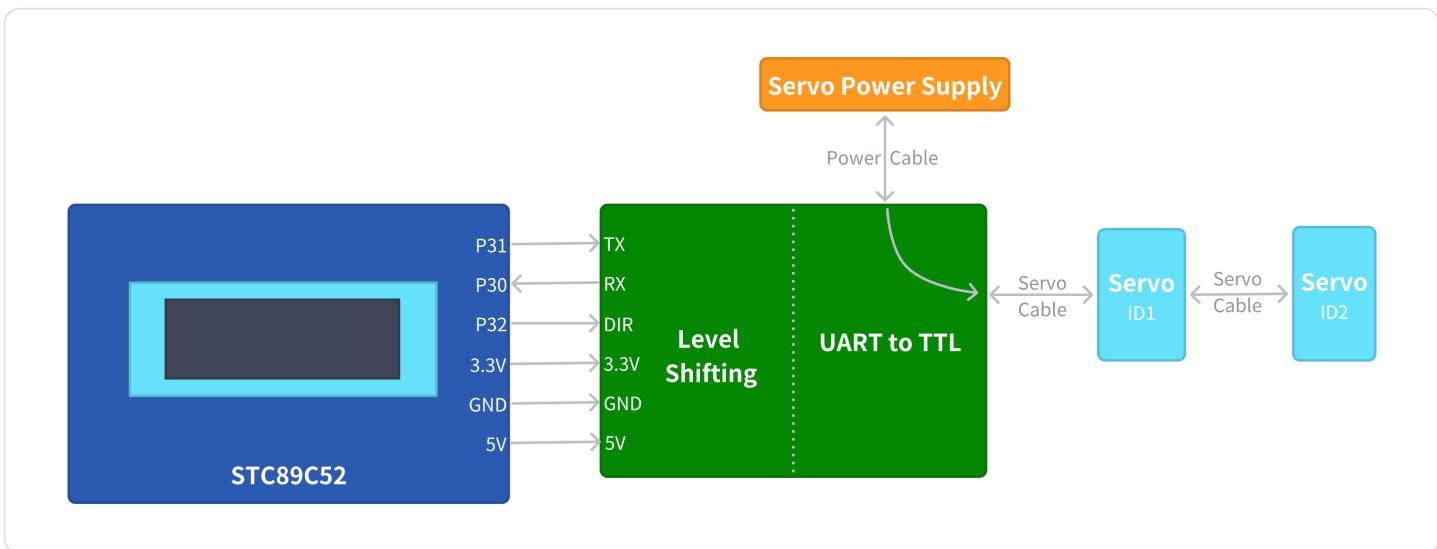
Step 4: Change the servo baud rate. Since the microcontroller hardware oscillator frequency used for this test is low, the highest baud rate that can be provided is 115200. Therefore, before testing, the baud rate for servos ID1 and ID2 needs to be changed to 3 (which corresponds to 115200). You can use ServoStudio or [ServoCommand](#) to make this change; please refer to the respective user manual for details.



Change the baud rate of ID1 to 3

Change the baud rate of ID1 back to 7

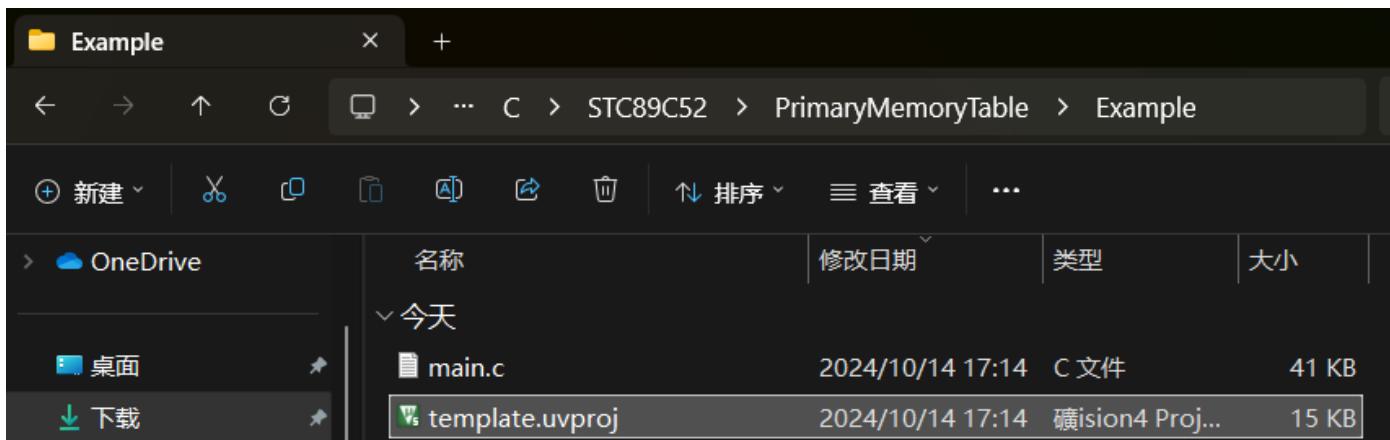
4.4.2 Hardware Connection



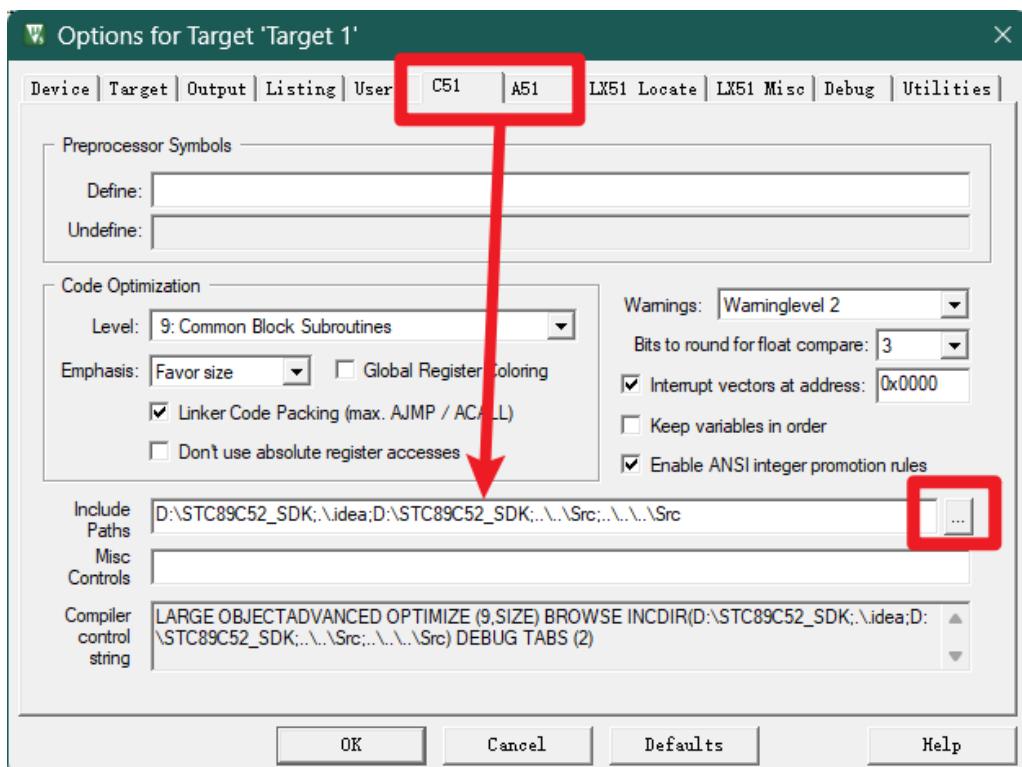
STC89C52 and Servo Connection Diagram

4.4.3 Run the Example

Step 1: Open the template.uvproj file. Taking EM-2030 as an example, double-click to open the template.uvproj located in ServoSDK\{C\}\STC89C52\PrimaryMemoryTable\Example.



Step 2: Replace the C51 and A51 including directories with the local Src path.



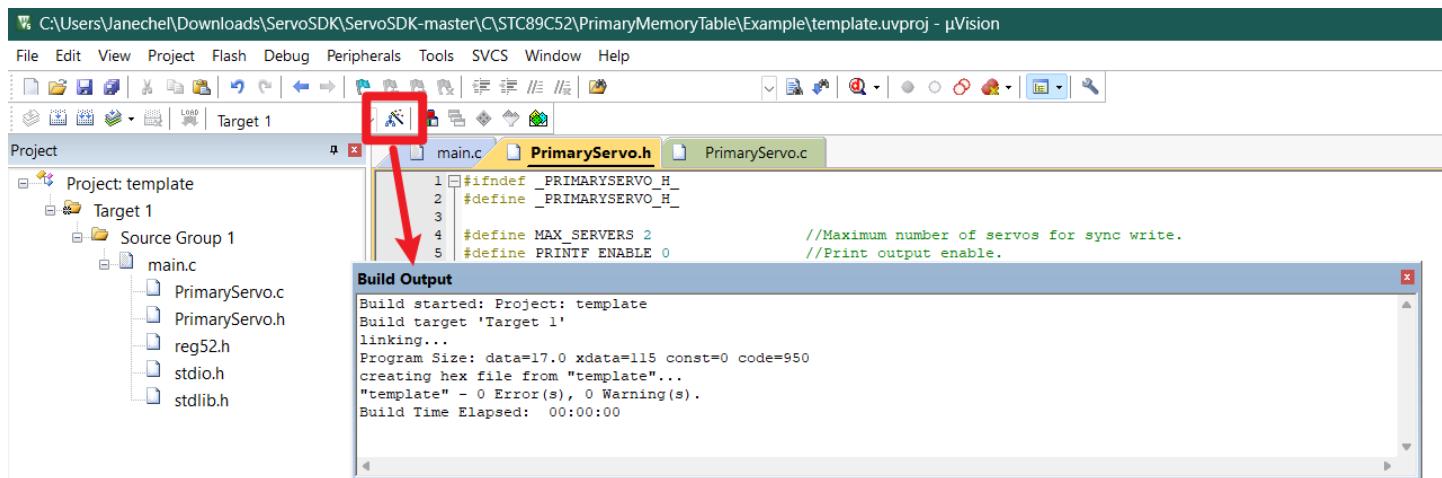
Step 3: Change the module value you want to test to 1. In the main.c file, modify the module value from "0" to "1" in lines 4-13.

```
1 //Taking test the SYNC WRITE module as an example
2 #define READ_TEST 0                      //Read Servo Data Test
3 #define WRITE_TEST 0                     //Write Servo Data Test
4 #define SYNC_WRITE_TEST 1                //Sync Write Test
5 #define PING_TEST 0                      //PING Instruction Test
6 #define FACTORY_RESET_TEST 0            //Factory Reset Test
7 #define PARAMETER_RESET_TEST 0          //Parameter Reset Test
8 #define REBOOT_TEST 0                   //Reboot Test
9 #define CALIBRATION_TEST 0             //Calibration Test
10 #define MODIFY_ID 0                    //Change Known Servo ID Test
11 #define MODIFY_UNKNOWN_ID 0           //Change Unknown Servo ID Test
```

Step 4: Change the servo output and quantity configuration. Due to the hardware limitations of the STC89C52 test board, change the MAX_SERVERS in PrimaryServo.h (lines 4-5) to 2, and set PRINTF_ENABLE to 0 (disable).

```
1 //Taking one servo and print disabled as an example (no changes needed).
2 #define MAX_SERVERS 2                  //Maximum number of servos for sync write.
3 #define PRINTF_ENABLE 0                 //Print output enable.
```

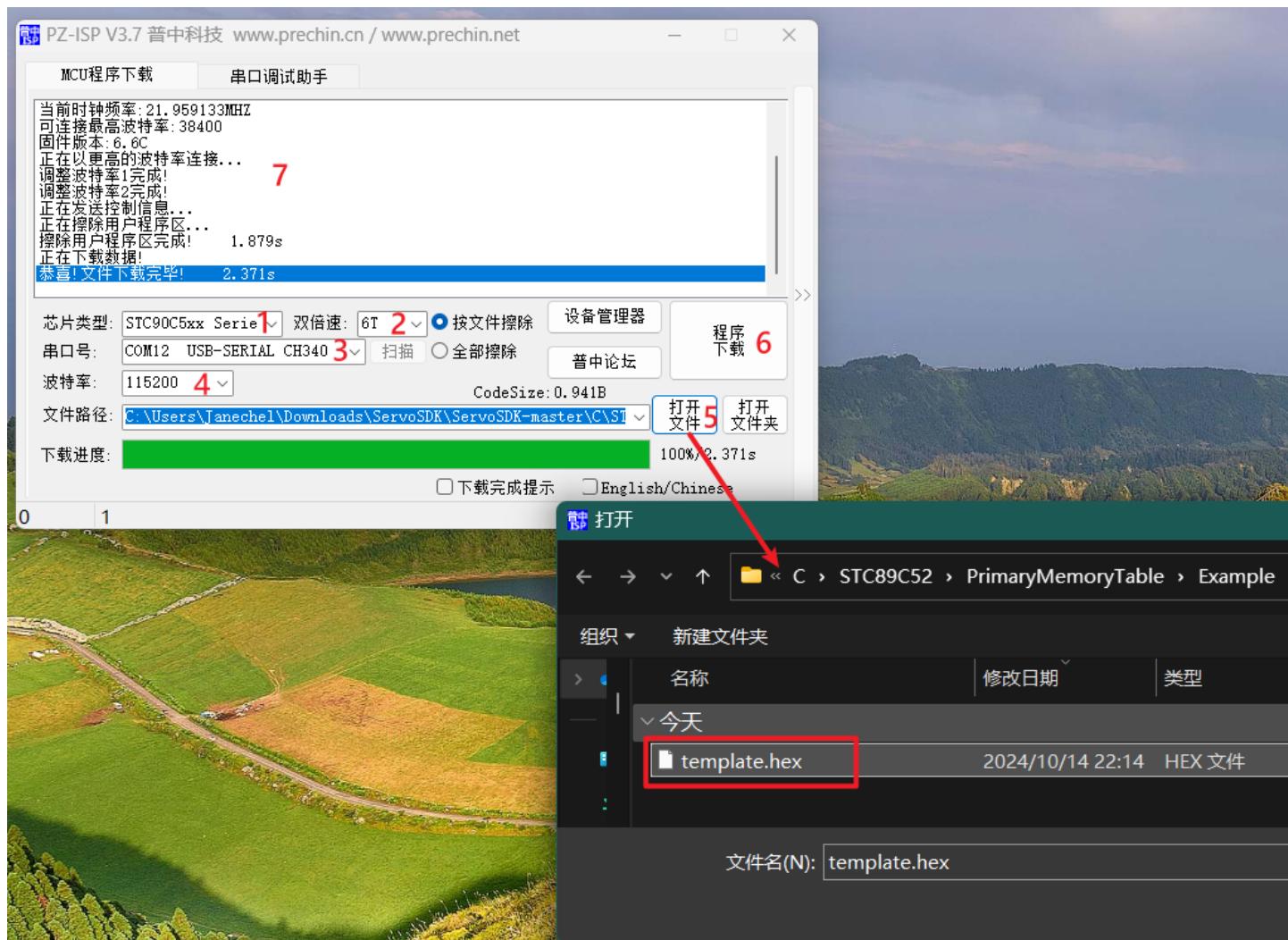
Step 5: Compile the project. Click Build to compile, ensuring there are no errors.



Compilation completed without errors

Step 6: Download the template.hex file. For example, with PZ-ISP.exe, use double-speed 6T for programming.

Note: To avoid write timeout issues, disconnect the wire connected to pin P31 before clicking "Program Download." After the file download is complete, reconnect the wire to pin P31.



Step 7: Run the program. Short press the Reset button on the microcontroller to restart the program. Since printing is disabled, you can only verify successful operation by observing the

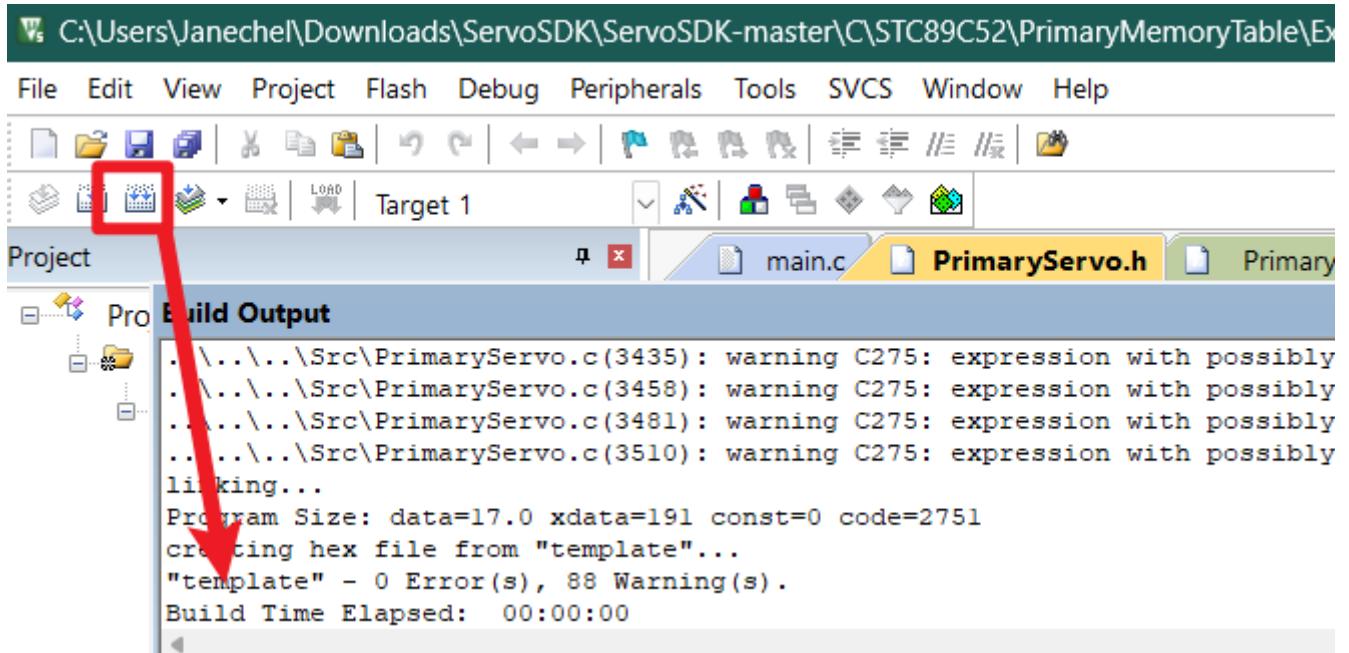
servo's movement.

Note: After testing, remember to change the baud rate back to 7 (which corresponds to 1,000,000) to avoid affecting servo testing on other platforms.

4.4.4 Run the Demo

For details, refer to *section 4.1.4 Run the Demo*.

Note: After replacing the main file, click 'Rebuild' to recompile the project before downloading and running the program.



4.4.5 FAQ

- Printing output enabled causing the microcontroller to hang
 - Since the 51 microcontroller only has one serial port for communication with the servo and no extra port for printing output, PRNTF_ENABLE needs to be set to 0 to disable it.
- PZ-ISP not set to double-speed 6T
 - Due to hardware limitations, this 51 microcontroller cannot reach a baud rate of 1,000,000; therefore, double-speed 6T is enabled to achieve the maximum baud rate of 115,200.
- PZ-ISP download failure
 - Try disconnecting the wire from pin P31 before downloading again.

5. C++

5.1 Windows

5.1.1 Development Environment Setup

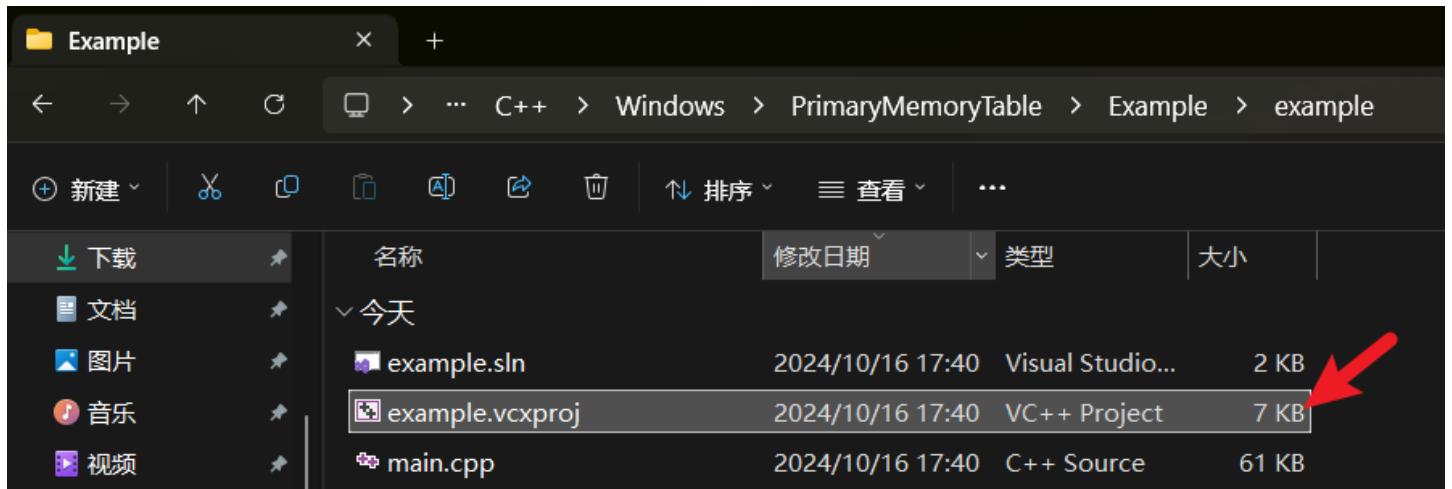
For details, refer to section 4.1.1 *Environment Setup*.

5.1.2 Hardware Connection

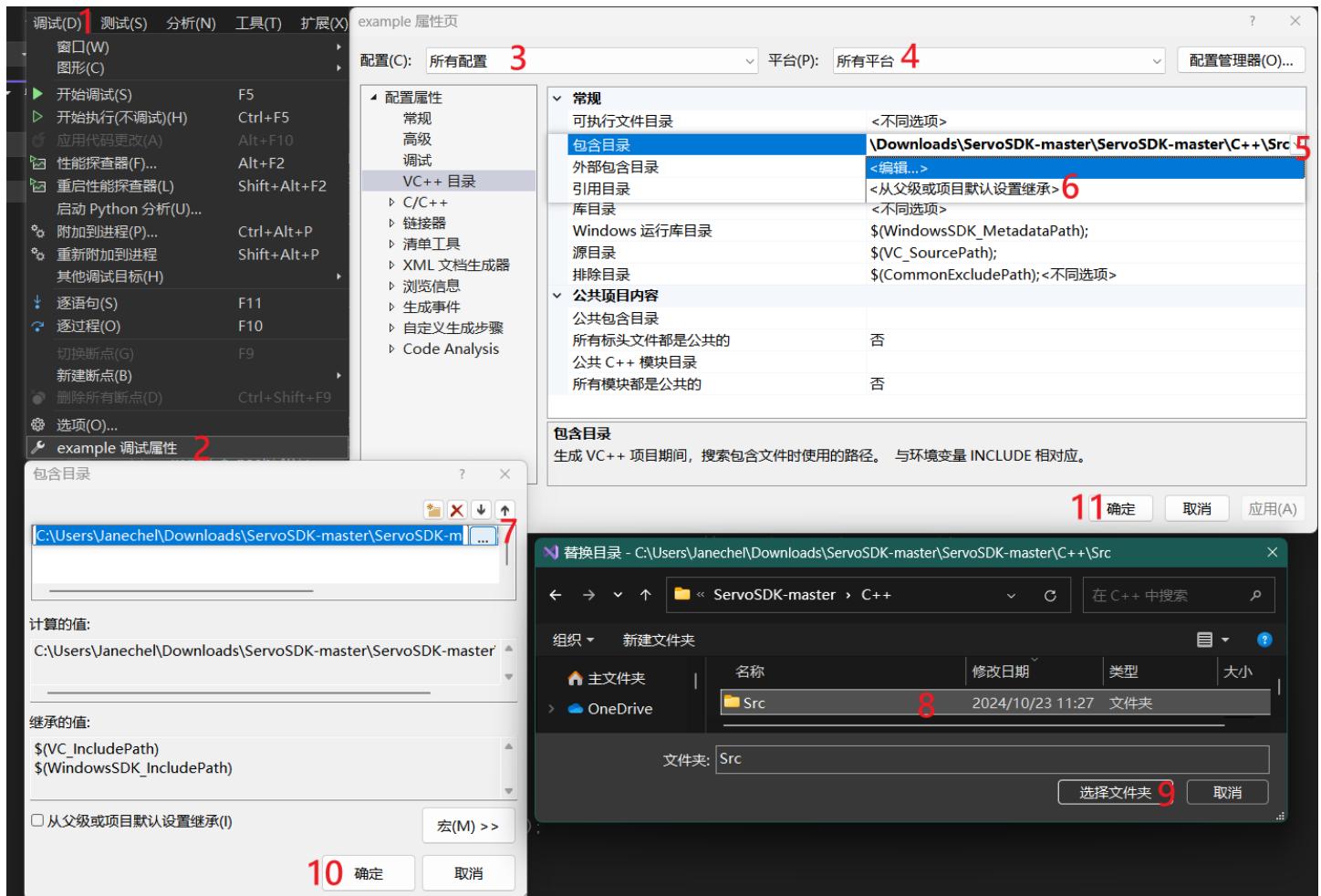
For details, refer to section 4.1.2 *Hardware Connection*.

5.1.3 Run the Example

Step 1: Open the example.vcxproj file. Taking EM-2030 as an example, double-click to open the example.vcxproj located in ServoSDK\C++\Windows\PrimaryMemoryTable\Example\example.



Step 2: Replace the include directory with the local src path. Navigate through 'Debug -> Example debug properties -> All Configurations -> All Platforms -> VC++ Directories -> Include Directories dropdown -> Edit -> More -> Find local Src -> Select folder -> Confirm -> Apply -> OK'.



To avoid issues with missing .h files, it is recommended to set the Configuration (C) and Platform (P) to 'All Configurations' and 'All Platforms'

Step 3: Modify the actual USBLINK port number. In the main.c file, change the code on line 35 from '12' to the actual USBLINK port number.

```

1 //Taking COM9 as an example
2 if (serialPort.Open(9, 1000000))

```

Step 4: Change the module value you want to test to 1. In the main.cpp file, change the module value from "0" to "1" in lines 4-13.

```

1 //Taking test the Ping module as an example
2 #define READ_TEST 0 //Read Servo Data Test
3 #define WRITE_TEST 0 //Write Servo Data Test
4 #define SYNC_WRITE_TEST 0 //Sync Write Test
5 #define PING_TEST 1 //PING Instruction Test
6 #define FACTORY_RESET_TEST 0 //Factory Reset Test
7 #define PARAMETER_RESET_TEST 0 //Parameter Reset Test
8 #define REBOOT_TEST 0 //Reboot Test
9 #define CALIBRATION_TEST 0 //Calibration Test

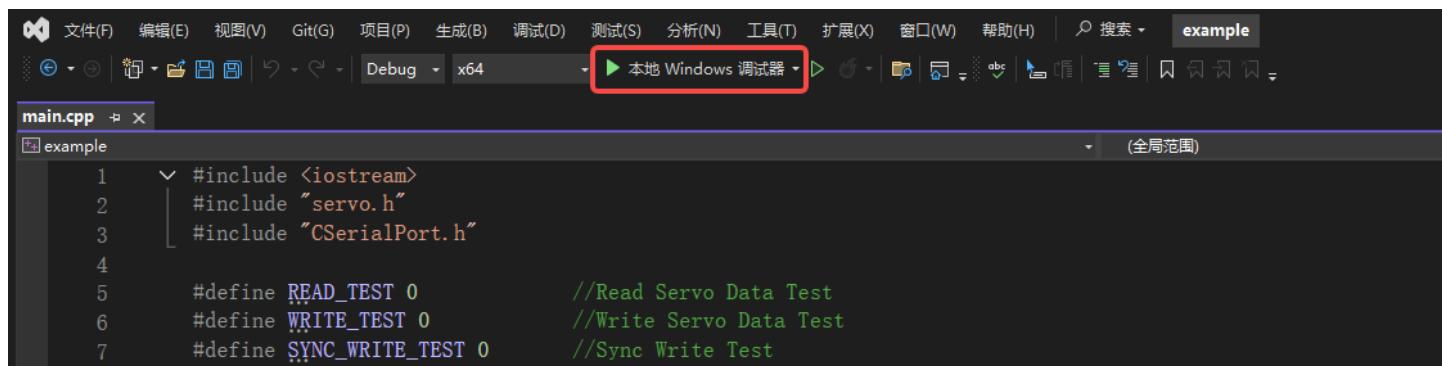
```

```
10 #define MODIFY_ID 0           //Change Known Servo ID Test
11 #define MODIFY_UNKNOWN_ID 0    //Change Unknown Servo ID Test
```

Step 5: Change the printing output enable as needed. In the PrimaryServo.h file, change line 6. If printing output is not needed, set PRINTF_ENABLE to "0".

```
1 //Taking print enable as an example (no changes needed).
2 #define PRINTF_ENABLE 1          //Print output enable.
```

Step 6: Run the program. Click the "Local Windows Debugger" button in Visual Studio to run the program.



5.1.4 Run the Demo

For details, refer to *section 4.1.4 Run the Demo*.

6. Python

6.1 Windows

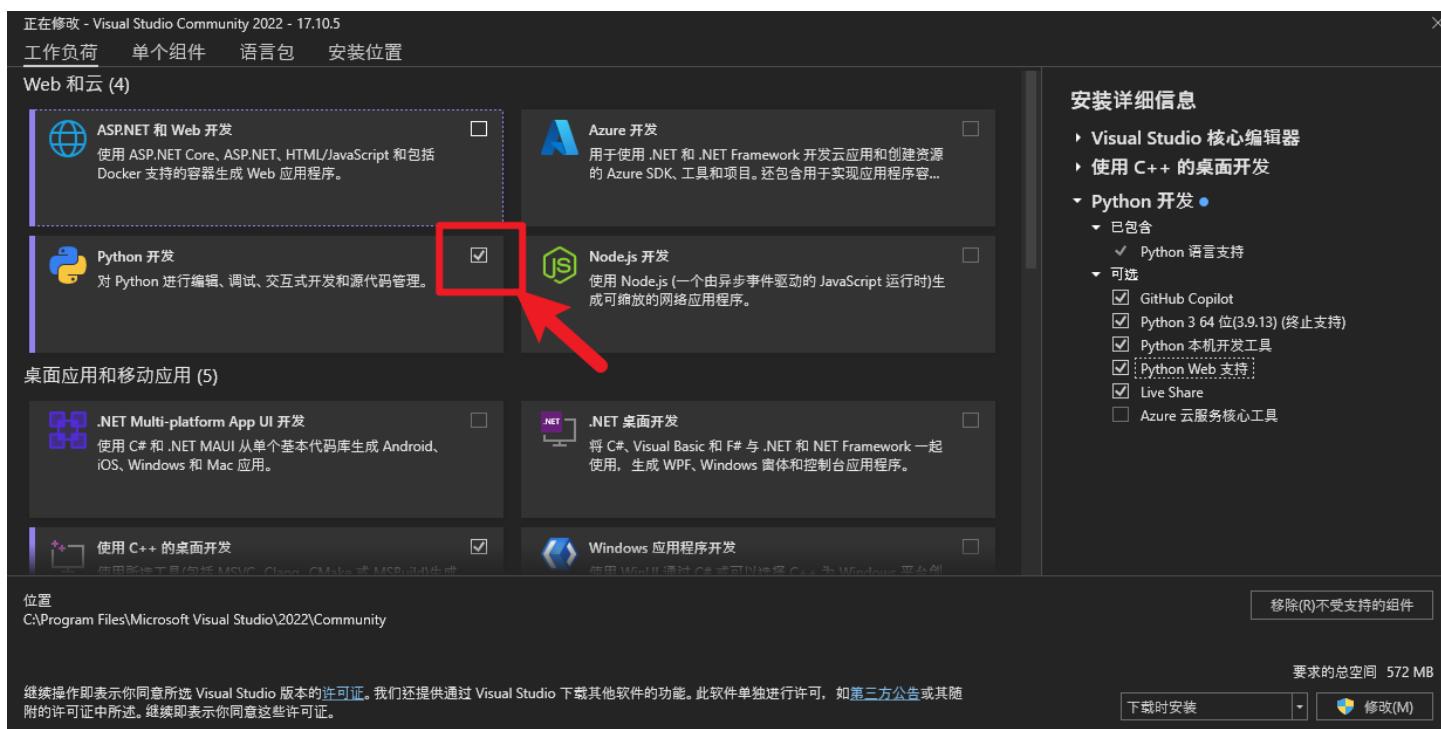
6.1.1 Environment Setup

Set up a Python programming environment using an IDE, taking Visual Studio 2022 as an example.

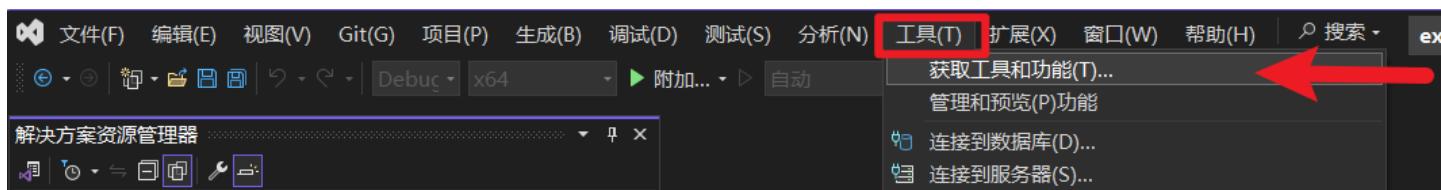
Step 1: Install Visual Studio 2022. Visit <https://visualstudio.microsoft.com/downloads/> and select the appropriate version for your operating system to download and install.

The screenshot shows the Microsoft Visual Studio Downloads page. At the top, there's a banner for Visual Studio for Mac, which is retiring on August 31, 2024. Below it, the main heading is 'Downloads'. A large box highlights the 'Community' edition of Visual Studio 2022. A red box surrounds the 'Free download' button. To the right, there's a 'Preview' section for the latest features.

Step 2: Install Python development.

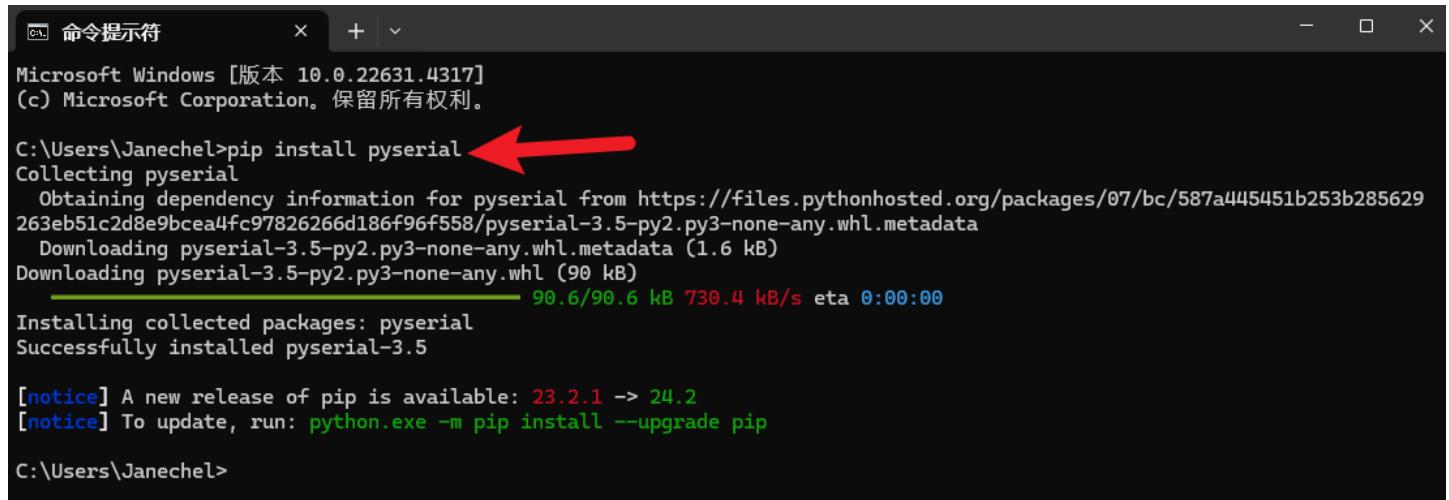


When installing the software, simply check the 'Python development' workload.



If you haven't checked it, you can also install it through the 'Menu Bar - Tools - Get Tools and Features.'

Install the pyserial module. In the desktop search box, enter the cmd command to open the Command Prompt. Then, type pip install pyserial and press Enter to install the module.



```
C:\Users\Janechel>pip install pyserial
Collecting pyserial
  Obtaining dependency information for pyserial from https://files.pythonhosted.org/packages/07/bc/587a445451b253b285629263eb51c2d8e9bcea4fc97826266d186f96f558/pyserial-3.5-py2.py3-none-any.whl.metadata
    Downloading pyserial-3.5-py2.py3-none-any.whl.metadata (1.6 kB)
  Downloading pyserial-3.5-py2.py3-none-any.whl (90 kB)  90.6/90.6 kB 730.4 kB/s eta 0:00:00
Installing collected packages: pyserial
Successfully installed pyserial-3.5

[notice] A new release of pip is available: 23.2.1 -> 24.2
[notice] To update, run: python.exe -m pip install --upgrade pip
C:\Users\Janechel>
```

6.1.2 Hardware Connection

For details, refer to section *4.1.2 Hardware Connection*.

6.1.3 Run the Example

Step 1: Open the example.pyproj file. Taking EM-2030 as an example, double-click to open the example.pyproj located in ServoSDK\Python\Windows\PrimaryMemoryTable\Example\example.

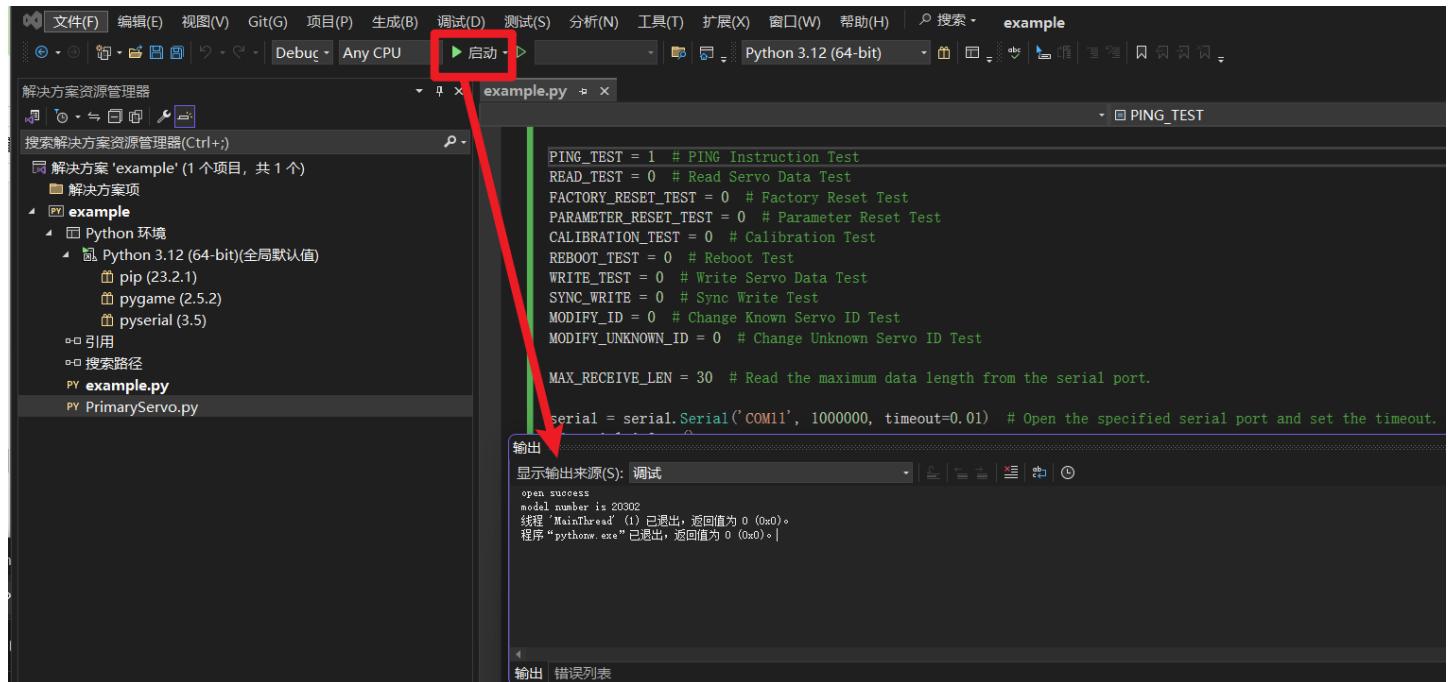
Step 2: Change the actual USBLINK port number. In the example.py file, change the code on line 18 from "18" to the actual USBLINK port number.

```
1 #Taking COM9 as an example.
2 serial = serial.Serial('COM11', 1000000, timeout=0.01) # Open the specified
   serial port and set the timeout.
```

Step 3: Change the module value you want to test to 1. In the example.py file, change the module value from "0" to "1" in lines 5-14.

```
1 #Taking test the Ping module as an example
2 PING_TEST = 1 # PING Instruction Test
3 READ_TEST = 0 # Read Servo Data Test
4 FACTORY_RESET_TEST = 0 # Factory Reset Test
5 PARAMETER_RESET_TEST = 0 # Parameter Reset Test
6 CALIBRATION_TEST = 0 # Calibration Test
7 REBOOT_TEST = 0 # Reboot Test
8 WRITE_TEST = 0 # Write Servo Data Test
9 SYNC_WRITE = 0 # Sync Write Test
10 MODIFY_ID = 0 # Change Known Servo ID Test
11 MODIFY_UNKNOWN_ID = 0 # Change Unknown Servo ID Test
```

Step 4: Run the program. Click the "Start" button in VS to run the program.



Taking test the Ping module as an example

6.1.4 Run the Demo

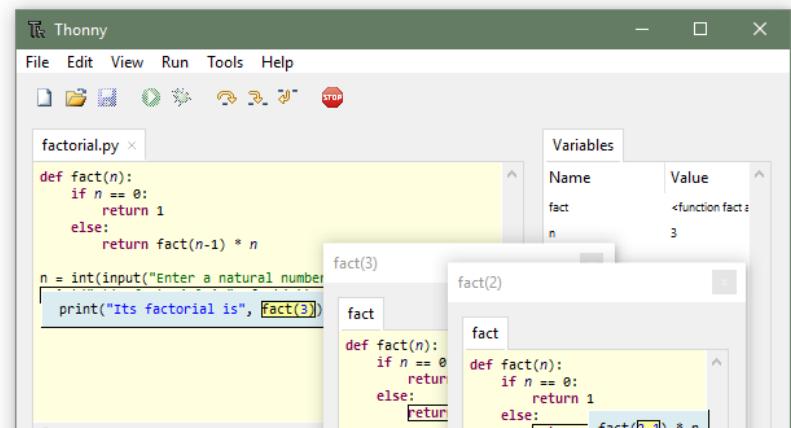
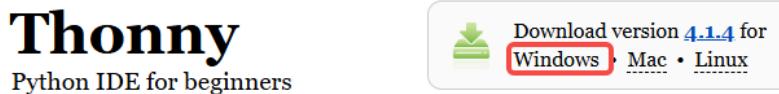
For details, refer to *section 4.1.4 Run the Demo*.

7. MicroPython

7.1 ESP32

7.1.1 Environment Setup

Step 1: Install Thonny. Visit <https://thonny.org/> and select the appropriate version for your operating system to download and install Thonny.



Taking Windows as an example

Step 2: Download the MicroPython firmware. Visit https://micropython.org/download/ESP32_GENERIC/ and download the latest firmware.

Firmware

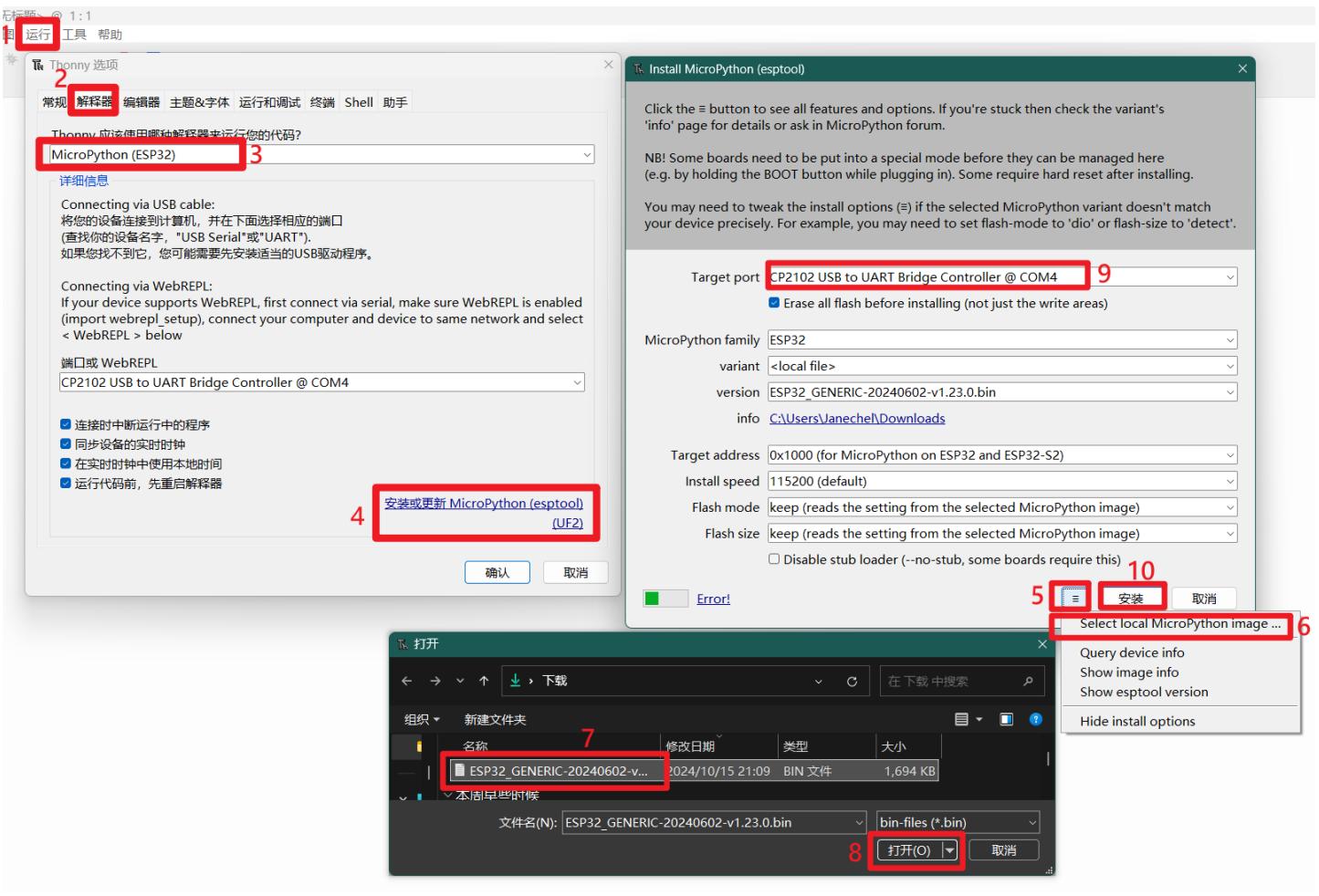
Releases

- v1.23.0 (2024-06-02) .bin / [.app-bin] / [.elf] / [.map] / [Release notes] (latest)
- v1.22.2 (2024-02-22) .bin / [.app-bin] / [.elf] / [.map] / [Release notes]
- v1.22.1 (2024-01-05) .bin / [.app-bin] / [.elf] / [.map] / [Release notes]
- v1.22.0 (2023-12-27) .bin / [.app-bin] / [.elf] / [.map] / [Release notes]
- v1.21.0 (2023-10-05) .bin / [.app-bin] / [.elf] / [.map] / [Release notes]
- v1.20.0 (2023-04-26) .bin / [.elf] / [.map] / [Release notes]
- v1.19.1 (2022-06-18) .bin / [.elf] / [.map] / [Release notes]
- v1.18 (2022-01-17) .bin / [.elf] / [.map] / [Release notes]
- v1.17 (2021-09-02) .bin / [.elf] / [.map] / [Release notes]
- v1.16 (2021-06-23) .bin / [.elf] / [.map] / [Release notes]
- v1.15 (2021-04-18) .bin / [.elf] / [.map] / [Release notes]
- v1.14 (2021-02-02) .bin / [.elf] / [.map] / [Release notes]
- v1.13 (2020-09-02) .bin / [.elf] / [.map] / [Release notes]
- v1.12 (2019-12-20) .bin / [.elf] / [.map] / [Release notes]

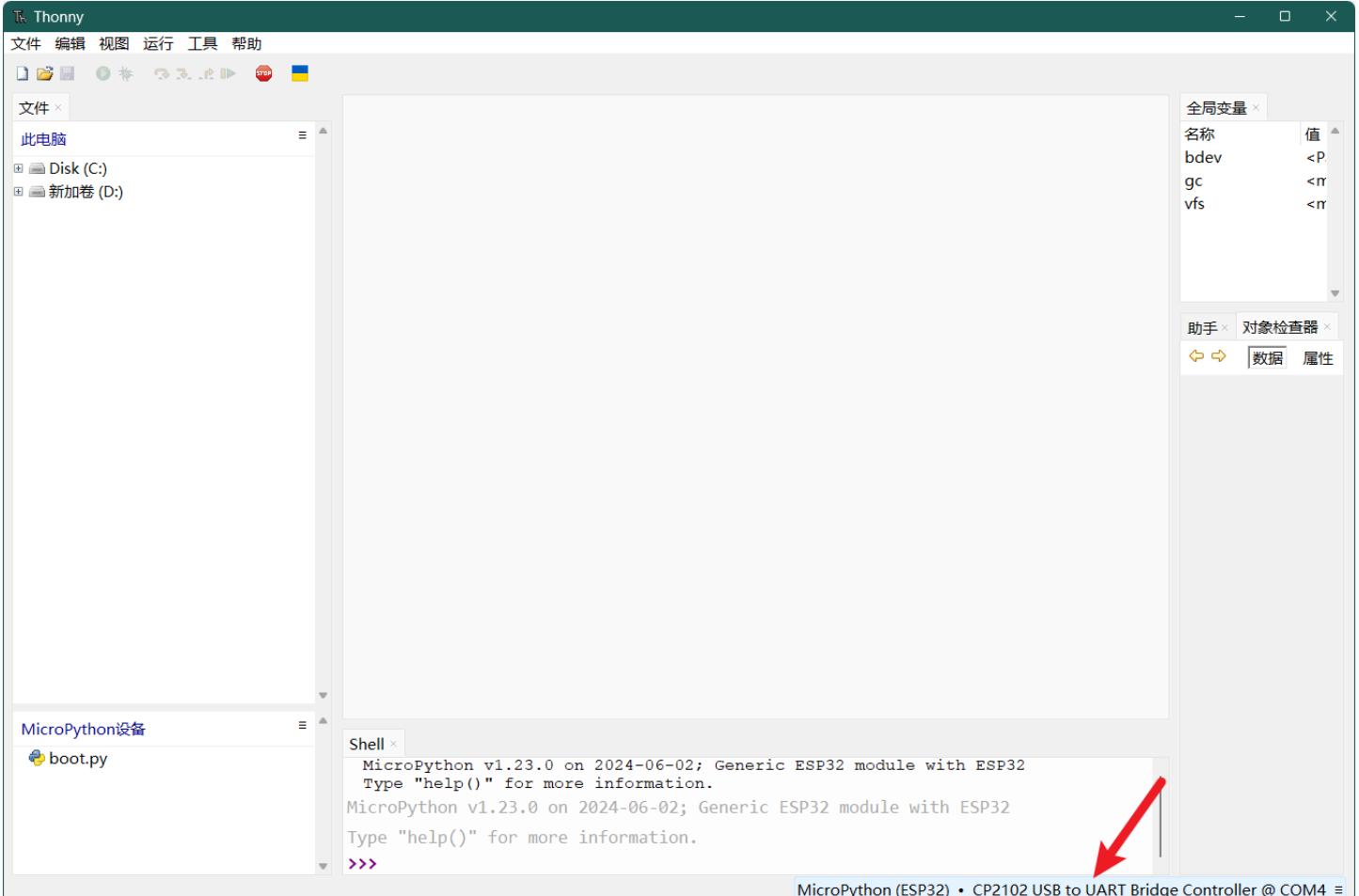
Step 3: Connect the device. Use a USB cable to connect the ESP32 to the PC, which will automatically detect and install the driver. If the device is not recognized, you will need to manually download the CP210x driver and update it.



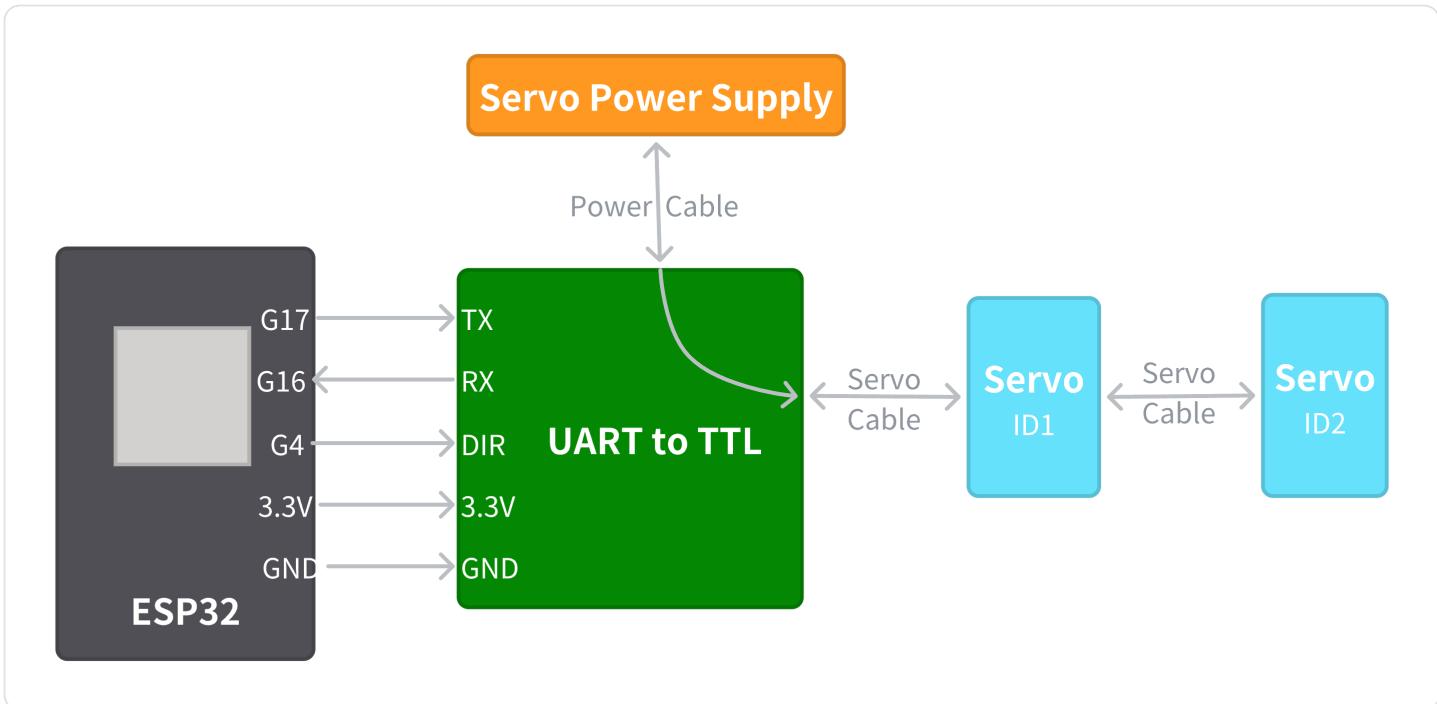
Step 4: Upload the MicroPython firmware. Open Thonny and navigate to Run -> Configure Interpreter -> MicroPython (ESP32) -> Install or Update MicroPython -> More -> Select Local MicroPython Image. Locate the downloaded .bin file, open it, select the target port, and click Install.



Step 5: Select the configurator.



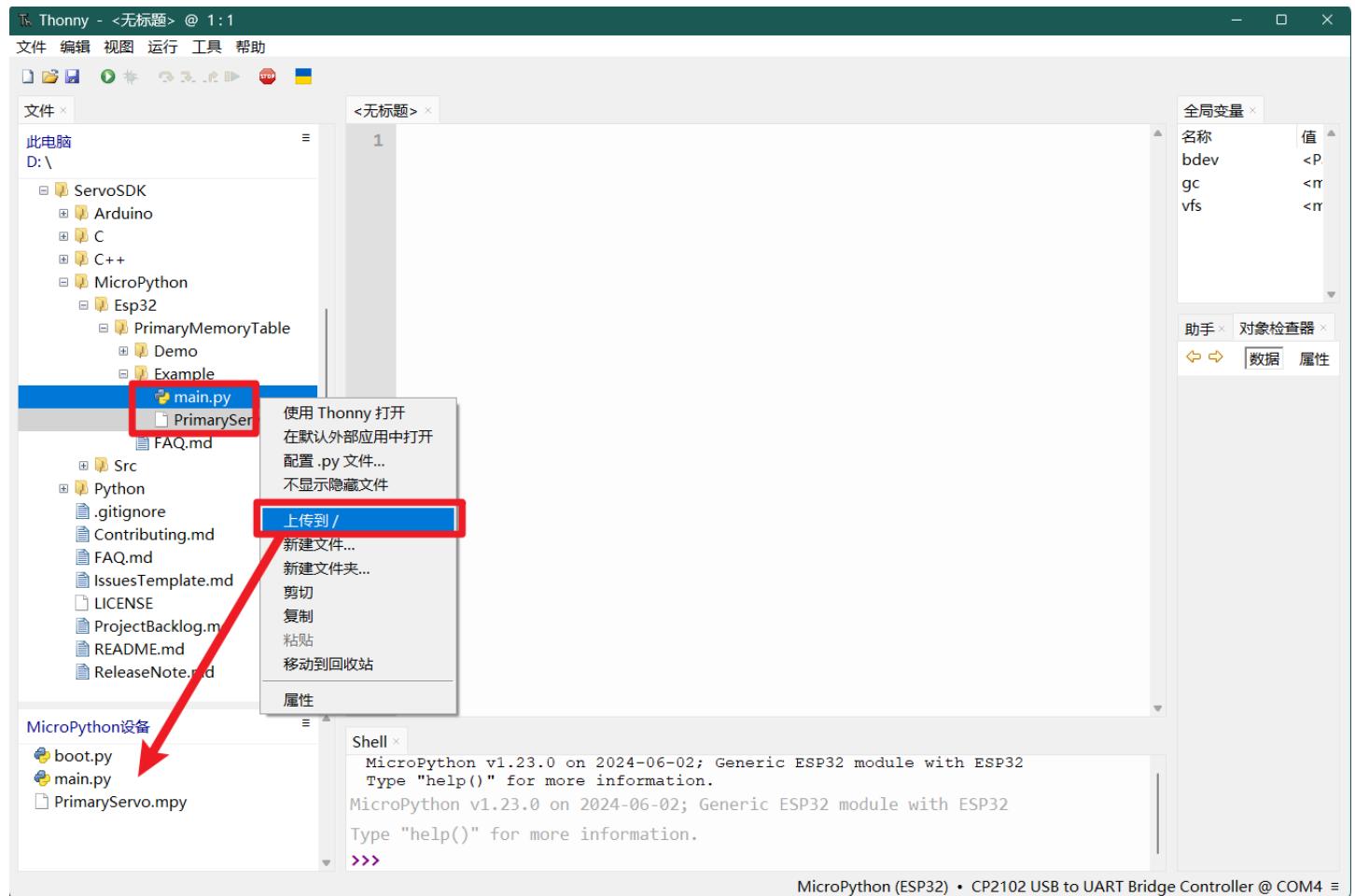
7.1.2 Hardware Connection



ESP32 and Servo Connection Diagram

7.1.3 Run the Example

Step 1: Upload main.py and PrimaryServo.mpy to the ESP32 device. Taking EM-2030 as an example, open Thonny, in the file window, locate main.py and PrimaryServo.mpy under ServoSDK\MicroPython\ESP32\PrimaryMemoryTable\Example, and upload them to the MicroPython device.



If the MicroPython device does not have a boot.py file, please reinstall the MicroPython firmware.

Step 2: Change the module value you want to test to 1. In the main.py file on the MicroPython device, modify lines 6-15 to change the functionality module value from "0" to "1."

```

1 #Taking test the Ping module as an example
2 PING_TEST = 1 # PING Instruction Test
3 READ_TEST = 0 # Read Servo Data Test
4 FACTORY_RESET_TEST = 0 # Factory Reset Test
5 PARAMETER_RESET_TEST = 0 # Parameter Reset Test
6 CALIBRATION_TEST = 0 # Calibration Test
7 REBOOT_TEST = 0 # Reboot Test
8 WRITE_TEST = 0 # Sync Write Test
9 SYNC_WRITE = 0 # Sync Write Test
10 MODIFY_ID = 0 # Change Known Servo ID Test
11 MODIFY_UNKNOWN_ID = 0 # Change Unknown Servo ID Test

```

Step 3: Run the current script. If you want to retain the settings after power loss, you need to edit the main file on the device.

Note: Due to the large size of the library files, importing them at once may fail because of memory fragmentation. To address this, convert the PrimaryServo.py file into a

PrimaryServo.mpy file using bytecode freezing. This file will not affect the normal import of modules.

The screenshot shows the Thonny IDE interface. The left sidebar displays a file tree with the following structure:

- D:\
- ServoSDK
- Arduino
- C
- C++
- MicroPython
- Esp32
- PrimaryMemoryTable
- Demo
- Example
- main.py (highlighted with a red box)
- PrimaryServo.mpy
- FAQ.md
- Src
- Python
- .gitignore
- Contributing.md
- FAQ.md
- IssuesTemplate.md
- LICENSE
- ProjectBacklog.md
- README.md
- ReleaseNote.md

The main editor window shows the content of main.py:

```
1 import micropython
2 from machine import Pin, UART
3 import time
4 from PrimaryServo import *
5
6 PING_TEST = 1 # PING Instruction Test
7 READ_TEST = 0 # Read Servo Data Test
8 FACTORY_RESET_TEST = 0 # Factory Reset Test
9 PARAMETER_RESET_TEST = 0 | # Parameter Reset Test
10 CALIBRATION_TEST = 0 # Calibration Test
11 REBOOT_TEST = 1 # Reboot Test
12 WRITE_TEST = 0 # Sync Write Test
13 SYNC_WRITE = 0 # Sync Write Test
14 MODIFY_ID = 0 # Change Known Servo ID Test
15 MODIFY_UNKNOWN_ID = 0 # Change Unknown Servo ID Test
16
17 ret = 0 # Status Flag
18 output_buffer = bytearray(40) # Store Generated Instructions
19 output_buffer_len = [0] # Instruction Length
20 receive_data = bytearray(40) # Store the received status packet
21 receive_data_len = 0 # Length of received data.
22 analysis_data = [0] # Data parsed from the status packet
23 position = [0] # Present position of the servo
24 current = [0] # Present current of the servo
25 write_buffer = bytearray(20) # Write data to the memory table
```

The bottom shell tab shows the command:

```
>>> %Run -c $EDITOR_CONTENT
```

The output is:

```
MPY: soft reboot
model number is 20302
```

A red arrow points to the 'model number is 20302' line in the shell output.

7.1.4 Run the Demo

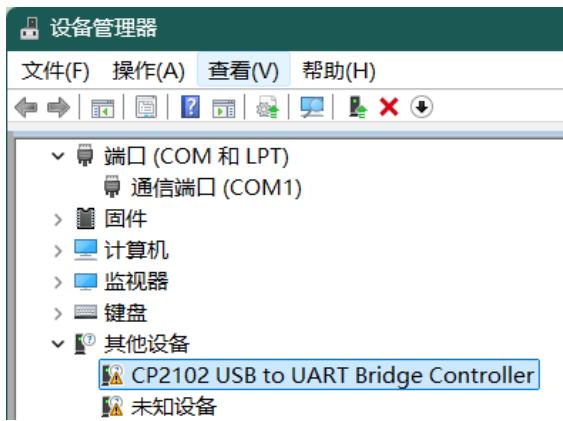
For details, please refer to *section 4.1.4 Demo*.

7.1.5 FAQ

- An index out of bounds error occurs in the get_check() function for the byte array.

```
Traceback (most recent call last):
  File "<stdin>", line 29, in <module>
  File "servo.py", line 2698, in servo set torque switch analysis
  File "servo.py", line 310, in servo unpack
  File "servo.py", line 224, in get check
IndexError: bytearray index out of range
```

- The servo may not be powered on; be mindful of supplying power to the servo.
- Exclamation mark appears next to the device in device manager after connecting ESP32



- Step 1: Download the driver. Visit <https://cn.silabs.com/developers/usb-to-uart-bridge-vcp-drivers?tab=downloads> and select the appropriate version for your operating system to download and install.
- Step 2: Right-click on the device -> Update Driver -> Browse my computer for drivers -> Browse to the path where you downloaded the driver -> Next -> Search automatically for updated driver software.

8. Arduino

8.1 Environment Setup

Step 1: Install Arduino. Visit <https://www.arduino.cc/en/software>, and select the appropriate version for your operating system to download and install.

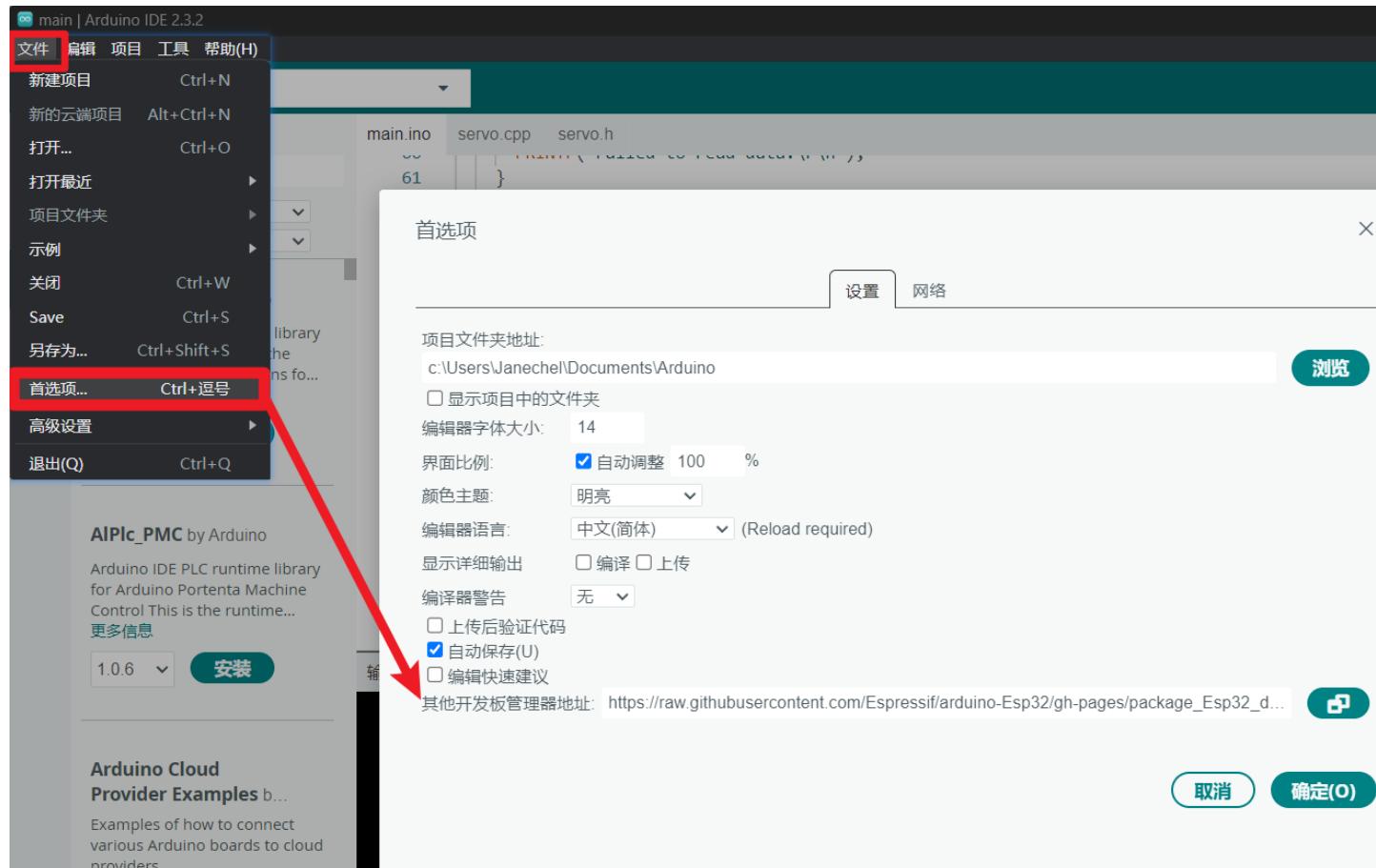
Downloads

The screenshot shows the Arduino IDE 2.3.2 download page. On the left, there is a logo of two interlocking circles and the text 'Arduino IDE 2.3.2'. Below this, a paragraph describes the new features of the release. A link to the 'Arduino IDE 2.0 documentation' is provided. Further down, a section for 'Nightly builds' is mentioned. At the bottom, a link to the 'SOURCE CODE' on GitHub is shown. On the right, a 'DOWNLOAD OPTIONS' section lists download links for various platforms: Windows (Win 10 and newer, 64 bits), Windows (MSI installer), Windows (ZIP file), Linux (AppImage 64 bits (X86-64)), Linux (ZIP file 64 bits (X86-64)), macOS (Intel, 10.15: "Catalina" or newer, 64 bits), and macOS (Apple Silicon, 11: "Big Sur" or newer, 64 bits). A 'Release Notes' link is also present.

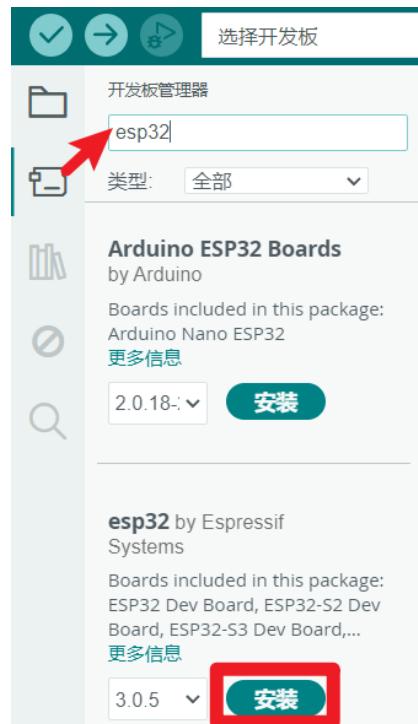
Taking Windows as an example

Step 2: Add the ESP32 development board. Open Arduino, then select File -> Preferences. In the Additional Board Manager URLs field, paste the following link:

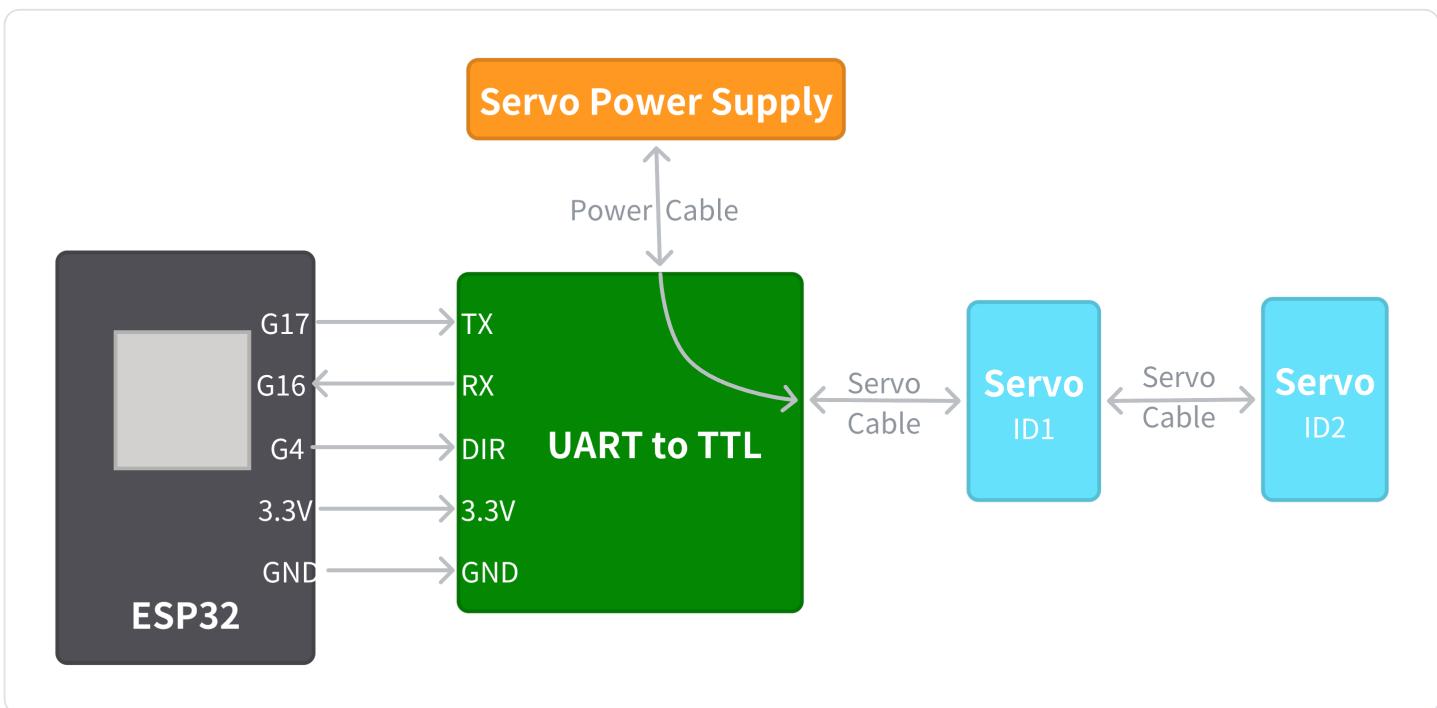
```
1 https://raw.githubusercontent.com/Espressif/arduino-Esp32/gh-pages/package_Esp32_dev_index.json
```



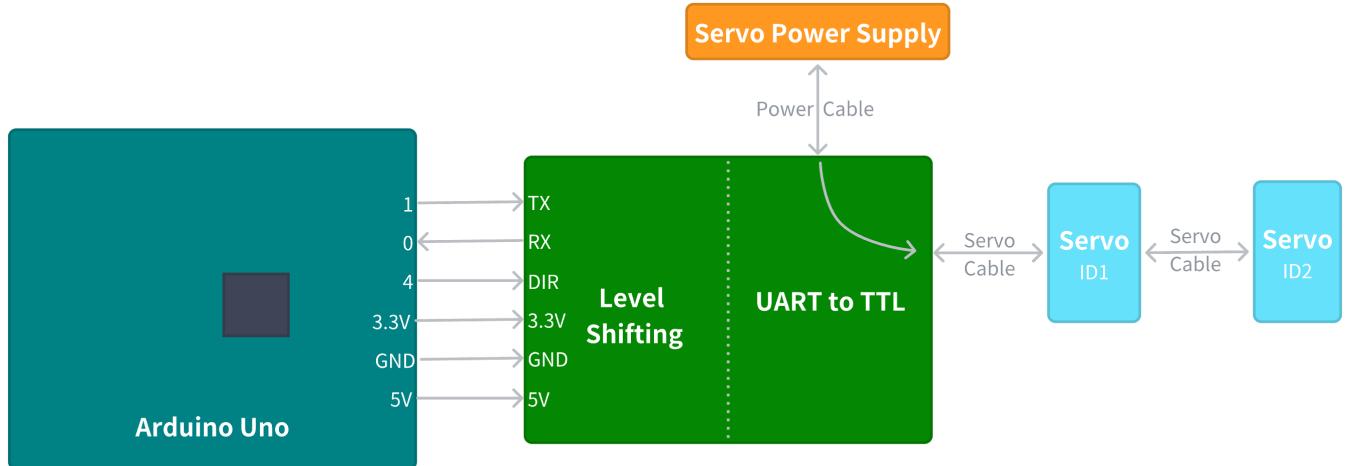
Step 3: Install the ESP32 development board. Open Arduino, click on the Boards Manager, enter 'esp32', select the latest version, and complete the installation.



8.2 Hardware Connection



ESP32 and Servo Connection Diagram



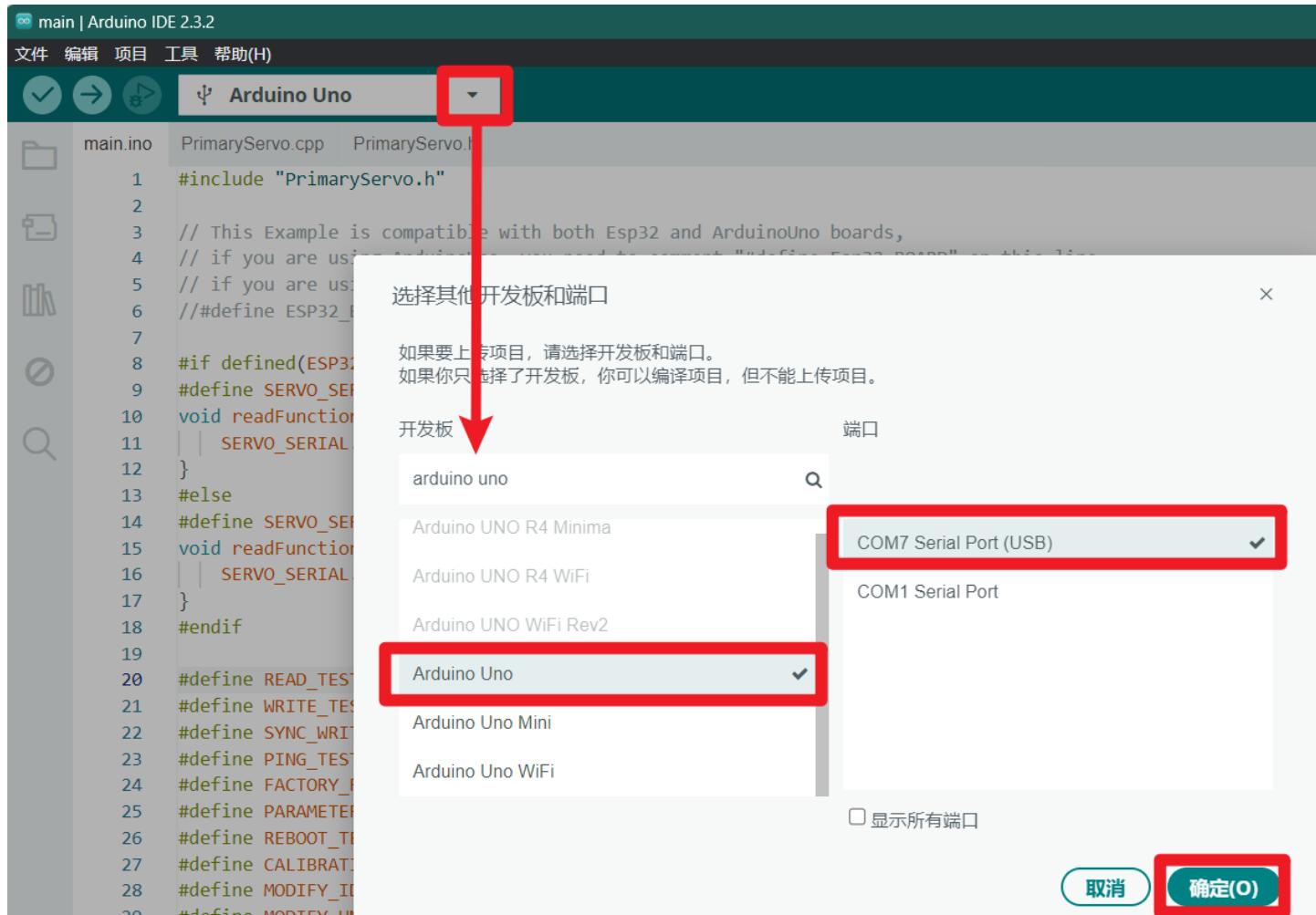
Arduino Uno and Servo Connection Diagram

8.3 Run the Example

Step 1: Download the servo control library. Open the Arduino software and download the Energize Lab Servo library. Click on "..." and select the corresponding example to open.



Step 2: Select the serial port and board type. Choose the appropriate serial port and board type based on the development board you are using.



Taking Arduino Uno as an example

Step 3: Comment out Esp32_BOARD based on the board selection. This example is compatible with both Esp32 and ArduinoUno boards. If using ArduinoUno, comment out line 6 of main.ino, "#define Esp32_BOARD". If using Esp32, no changes are needed.

```
1 // This Example is compatible with both Esp32 and ArduinoUno boards,
2 // if you are using ArduinoUno, you need to comment "#define Esp32_BOARD" on
3 // this line,
4
5 //Taking Arduino Uno as an example
6 //#define ESP32_BOARD
7
8 #if defined(ESP32_BOARD)
9 #define SERVO_SERIAL Serial2
10 void readFunction(uint8_t* pack, uint8_t pack_len) {
11     SERVO_SERIAL.read(pack, pack_len);
12 }
13 #else
14 #define SERVO_SERIAL Serial
15 void readFunction(uint8_t* pack, uint8_t pack_len) {
16     SERVO_SERIAL.readBytes(pack, pack_len);
```

```
17 }  
18 #endif
```

Step 4: Change the module value you want to test to 1. In lines 17-26 of main.ino, change the module value "0" to "1" for the module you want to test.

Note: It is recommended to test only one module at a time to avoid upload errors.

```
1 //Taking test the Ping module as an example  
2 #define READ_TEST 0           //Read Servo Data Test  
3 #define WRITE_TEST 0          //Write Servo Data Test  
4 #define SYNC_WRITE_TEST 0     //Sync Write Test  
5 #define PING_TEST 1           //PING Instruction Test  
6 #define FACTORY_RESET_TEST 0  //Factory Reset Test  
7 #define PARAMETER_RESET_TEST 0 //Parameter Reset Test  
8 #define REBOOT_TEST 0          //Reboot Test  
9 #define CALIBRATION_TEST 0    //Calibration Test  
10 #define MODIFY_ID 0           //Change Known Servo ID Test  
11 #define MODIFY_UNKNOWN_ID 0   //Change Unknown Servo ID Test
```

Step 5: Change the maximum number of servos for sync write and enable printing output as needed. In lines 6-8 of the PrimaryServo.h file, `MAX_SERVOS` defines the maximum number of synchronized servos supported, with a default of writing data for 6 servos simultaneously.

`Serial` is the output serial port for printing actual information, which is the same for both ESP32 and Arduino, and does not need to be modified. If you do not need to print output results, change the value of `PRINTF_ENABLE` to `0`.

Note: Due to memory limitations of different boards, the maximum number of synchronized servos supported may vary. For example, Arduino Uno supports up to 6 servos for synchronized writing in the provided examples, while ESP32 can support more

```
1 //Taking one servo and print enable as an example (no changes needed).  
2 #define ServoPrintf Serial      //A serial port for output information.  
3 #define PRINTF_ENABLE 1          //Print output enable.  
4 #define MAX_SERVERS 6            //Maximum number of servos for sync write.
```

Step 6: Compile and upload. To avoid upload failures, unplug the RX and TX pin wires before clicking 'Upload'.

Step 7: Check the information. After compilation is complete, reconnect the RX and TX wires, open the Serial Monitor, and adjust the baud rate to view the complete debug information.

```
14 }
15 #endif
16
17 #define READ_TEST 0           //Read Servo Data Test
18 #define WRITE_TEST 0          //Write Servo Data Test
19 #define SYNC_WRITE_TEST 0     //Sync Write Test
20 #define PING_TEST 1            //PING Instruction Test
21 #define FACTORY_RESET_TEST 0   //Factory Reset Test
22 #define PARAMETER_RESET_TEST 0 //Parameter Reset Test
23 #define REBOOT_TEST 0          //Reboot Test
24 #define CALIBRATION_TEST 0     //Calibration Test
25 #define MODIFY_ID 0             //Change Known Servo ID Test
26 #define MODIFY_UNKNOWN_ID 0     //Change Unknown Servo ID Test
27
28 uint8_t ret;                  //Change Unknown Servo ID Test
29 uint8_t order_buffer[40];      //Store Generated Instructions
30 uint8_t order_len;            //Instruction Length
31 uint8_t pack[40];             //Store the received status packet
32 uint8_t pack_len;             //Response packet length.
33 uint32_t analysis_data;       //Data parsed from the status packet
34 uint16_t position;            //Present position of the servo
```

输出 gdb-server 串口监视器 X

消息 (按回车将消息发送到“COM4”上的“ESP32 Dev Module”)

Ping Successful!

model number: 20302

波特率 115200

The baud rate for ESP32 is 115200, while for Arduino Uno it is 1000000

8.4 Run the Demo

For details, refer to *section 4.1.4 Run the Demo*.

Note: If you are using Arduino Uno, you need to comment out line 6 of the .ino file, which is `//#define ESP32_BOARD.`

8.5 FAQ

- Due to the memory limitations of Uno, it may be necessary to appropriately reduce the maximum number of sync write servos.
- Arduino library files are in read-only status within the Arduino software; if editing is needed, locate the local library files, open them, and then edit.



The screenshot shows the Arduino IDE interface. In the center, there's a code editor window titled "Arduino Uno". The code in the editor is:

```
1 #include "PrimaryServo.h"
2
3 // This Ex
4 // if you
5 // if you
6 //#define
7
```

A tooltip is displayed over the line "#include \"PrimaryServo.h\"". The tooltip contains the text "PrimaryServo.h" in red, which is highlighted by a red arrow pointing from the text in the tooltip to the corresponding line in the code editor. Below the tooltip, the full path to the file is shown: "C:\Users\Janechel\AppData\Local\Temp\arduino-language-server2015543858\build\sketch\PrimaryServo.h".

To open the .h file: Move the mouse to the file location, copy the path from the popup window, and open it in the file manager.